CybOX™ Version 2.1.1. Part 69: Win Executable File Object

Committee Specification Draft 01 / Public Review Draft 01

20 June 2016

Specification URIs

This version:

http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part69-win-executable-file/cybox-v2.1.1-csprd01-part69-win-executable-file.docx (Authoritative)
http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part69-win-executable-file/cybox-v2.1.1-csprd01-part69-win-executable-file.html
http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part69-win-executable-file/cybox-v2.1.1-csprd01-part69-win-executable-file.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/cti/cybox/v2.1.1/part69-win-executable-file/cybox-v2.1.1-part69-win-executable-file.docx (Authoritative)
http://docs.oasis-open.org/cti/cybox/v2.1.1/part69-win-executable-file/cybox-v2.1.1-part69-win-executable-file.html
http://docs.oasis-open.org/cti/cybox/v2.1.1/part69-win-executable-file/cybox-v2.1.1-part69-win-executable-file.pdf

Technical Committee:

OASIS Cyber Threat Intelligence (CTI) TC

Chair:

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

Editors:

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

Additional artifacts:

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

Related work:

This specification is related to:

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

Abstract:

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

Status:

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

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

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

Citation format:

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

[CybOX-v2.1.1-win-executable-file]

CybOX™ Version 2.1.1. Part 69: Win Executable File Object. Edited by Desiree Beck, Trey Darley, Ivan Kirillov, and Rich Piazza. 20 June 2016. OASIS Committee Specification Draft 01 / Public Review Draft 01. http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part69-win-executable-file/cybox-v2.1.1-csprd01-part69-win-executable-file.html. Latest version: http://docs.oasis-open.org/cti/cybox/v2.1.1/part69-win-executable-file/cybox-v2.1.1-part69-win-executable-file.html.

Notices

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

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

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

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

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

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

[All text is normative unless otherwise labeled]

The Cyber Observable Expression (CybOX^{TM}) 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 Win Executable File Object Version 2.1.1 data model, which is one of eighty-eight CybOX Object data models.

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

1.1 CybOX^{TM} Specification Documents

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

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

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

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

1.2 Document Conventions

The following conventions are used in this document.

1.2.1 Fonts

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

- Capitalization is used for CybOX high level concepts, which are defined in *CybOX Version 2.1.1 Part 1: Overview*.

Examples: Action, Object, Event, Property

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

  Examples: `ActionType`, `cyboxCommon:BaseObjectPropertyType`

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

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

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

1.2.2 UML Package References

Each CybOX data model is captured in a different UML package (e.g., Core package) where the packages together compose the full CybOX UML model. To refer to a particular class of a specific package, we use the format `package_prefix:class`, where `package_prefix` corresponds to the appropriate UML package.

The package_prefix for the Windows Executable File data model is `WinExecutableFileObj`. Note that in this specification document, we do not explicitly specify the package prefix for any classes that originate from the Win Executable File Object data model.

1.2.3 UML Diagrams

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

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

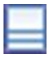
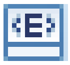
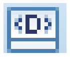
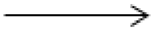
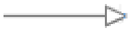
1.2.3.1 Class Properties

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

1.2.3.2 Diagram Icons and Arrow Types

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

Table 1-1. UML diagram icons

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

1.2.4 Property Table Notation

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

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

1.2.5 Property and Class Descriptions

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

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

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

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

1.3 Terminology

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

1.4 Normative References

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

2 Background Information

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

2.1 Cyber Observables

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

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

2.2 Objects

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

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

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

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

3 Data Model

3.1 WindowsExecutableFileObjectType Class

The `WindowsExecutableFileObjectType` class is intended to characterize Windows PE (Portable Executable) files. The UML diagram corresponding to the `WindowsExecutableFileObjectType` class is shown in **Figure 3-1**.

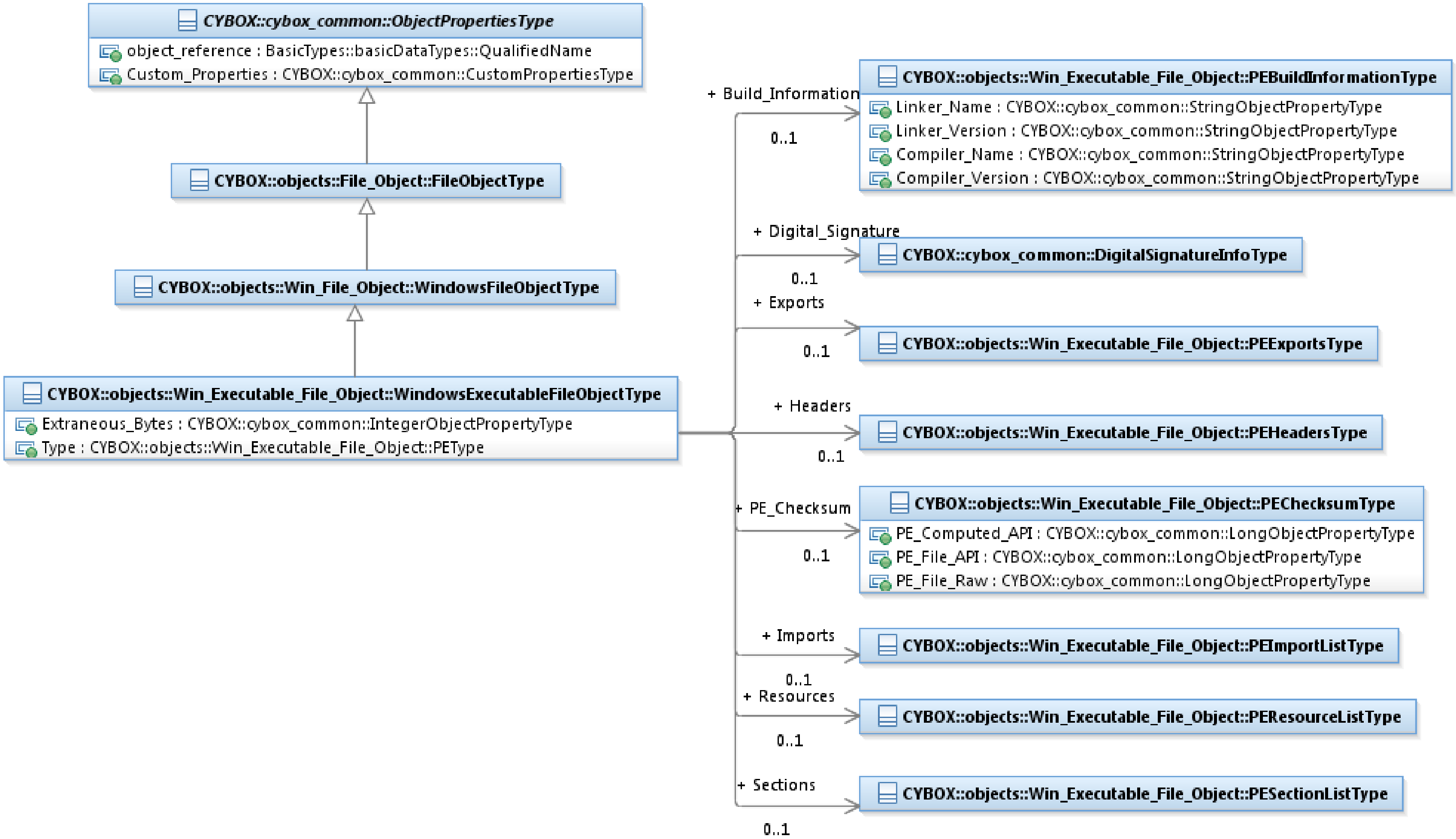

CYBOX::cybox_common::ObjectPropertiesType
object_reference : BasicTypes::basicDataTypes::QualifiedName
Custom_Properties : CYBOX::cybox_common::CustomPropertiesType
CYBOX::objects::File_Object::FileObjectType
CYBOX::objects::Win_File_Object::WindowsFileObjectType
CYBOX::objects::Win_Executable_File_Object::WindowsExecutableFileObjectType
Extraneous_Bytes : CYBOX::cybox_common::IntegerObjectPropertyType
Type : CYBOX::objects::Win_Executable_File_Object::PEType
+ Build_Information
0..1
CYBOX::objects::Win_Executable_File_Object::PEBuildInformationType
Linker_Name : CYBOX::cybox_common::StringObjectPropertyType
Linker_Version : CYBOX::cybox_common::StringObjectPropertyType
Compiler_Name : CYBOX::cybox_common::StringObjectPropertyType
Compiler_Version : CYBOX::cybox_common::StringObjectPropertyType
+ Digital_Signature
0..1
CYBOX::cybox_common::DigitalSignatureInfoType
+ Exports
0..1
CYBOX::objects::Win_Executable_File_Object::PEExportsType
+ Headers
0..1
CYBOX::objects::Win_Executable_File_Object::PEHeadersType
+ PE_Checksum
0..1
CYBOX::objects::Win_Executable_File_Object::PEChecksumType
PE_Computed_API : CYBOX::cybox_common::LongObjectPropertyType
PE_File_API : CYBOX::cybox_common::LongObjectPropertyType
PE_File_Raw : CYBOX::cybox_common::LongObjectPropertyType
+ Imports
0..1
CYBOX::objects::Win_Executable_File_Object::PEImportListType
+ Resources
0..1
CYBOX::objects::Win_Executable_File_Object::PEResourceListType
+ Sections
0..1
CYBOX::objects::Win_Executable_File_Object::PESectionListType

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

The property table of the `WindowsExecutableFileObjectType` class is given in **Table 3-1**.

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

Name	Type	Multiplicity	Description
Build_Information	PEBuildInformationType	0..1	The Build_Information property specifies some information on the tools used to build the PE binary.
Digital_Signature	cyboxCommon: DigitalSignatureInfoType	0..1	The Digital_Signature property specifies the information about the digital signature used to sign the PE binary.
Exports	PEExportsType	0..1	The Exports property characterizes the PE Export table of the PE Binary.
Extraneous_Bytes	cyboxCommon: IntegerObjectPropertyType	0..1	The Extraneous_Bytes property specifies the number of extraneous bytes contained in the PE binary.
Headers	PEHeadersType	0..1	The Headers property contains fields for characterizing aspects the various types of PE headers.
Imports	PEImportListType	0..1	The Imports property characterizes the PE Import Table of the binary.
PE_Checksum	PEChecksumType	0..1	The PE_Checksum property specifies the checksum of the PE file.
Resources	PEResourceListType	0..1	The Resources property characterizes the PE Resources of the binary.
Sections	PESectionListType	0..1	The Sections property characterizes the PE Sections of the binary.
Type	PEType	0..1	The Type property specifies the particular type of the PE binary, e.g. Executable.

3.2 PEChecksumType Class

The PEChecksumType class records the checksum of the PE file, both as found in the file and computed.

The property table of the PEChecksumType class is given in **Table 3-2**.

Table 3-2. Properties of the PEChecksumType class

Name	Type	Multiplicity	Description
PE_Computed_API	cyboxCommon: LongObjectPropertyType	0..1	The PE_Computed_API property specifies a checksum computed by an external algorithm.
PE_File_API	cyboxCommon: LongObjectPropertyType	0..1	The PE_File_API property specifies the checksum computed by IMAGHELP.DLL.
PE_File_Raw	cyboxCommon: LongObjectPropertyType	0..1	The PE_File_Raw property specifies the checksum found in the PE file (in the Optional Header).

3.3 PEExportsType Class

The PEExportsType class specifies the PE File exports data section. The exports data section contains information about symbols exported by the PE File (a DLL) which can be dynamically loaded by other executables. This class and its components abstract the Windows structures. The UML diagram corresponding to the PEExportsType class is shown in **Figure 3-2**.

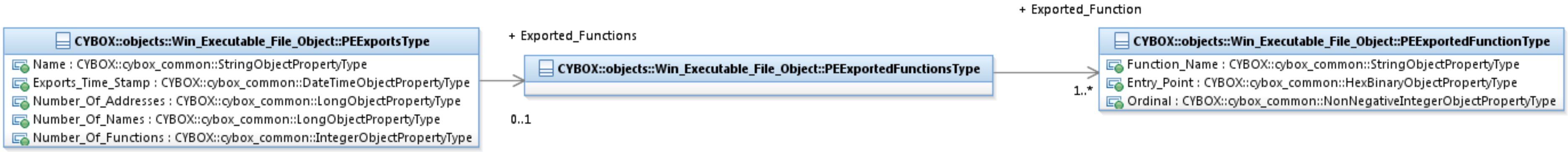

CYBOX::objects::Win_Executable_File_Object::PEExportsType
Name : CYBOX::cybox_common::StringObjectPropertyType
Exports_Time_Stamp : CYBOX::cybox_common::DateTimeObjectPropertyType
Number_Of_Addresses : CYBOX::cybox_common::LongObjectPropertyType
Number_Of_Names : CYBOX::cybox_common::LongObjectPropertyType
Number_Of_Functions : CYBOX::cybox_common::IntegerObjectPropertyType
+ Exported_Functions
0..1
CYBOX::objects::Win_Executable_File_Object::PEExportedFunctionsType
+ Exported_Function
1..*
CYBOX::objects::Win_Executable_File_Object::PEExportedFunctionType
Function_Name : CYBOX::cybox_common::StringObjectPropertyType
Entry_Point : CYBOX::cybox_common::HexBinaryObjectPropertyType
Ordinal : CYBOX::cybox_common::NonNegativeIntegerObjectPropertyType

Figure 3-2. UML diagram for PEExportsType class

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

Table 3-3. Properties of the PEExportsType class

Name	Type	Multiplicity	Description
Name	cyboxCommon: StringObjectPropertyType	0..1	The Name property specifies the actual name of the PE module, as used by the PE loader when it is imported by another executable.
Exported_Functions	PEExportedFunctionsType	0..1	The Exported_Functions property specifies a list of the exported functions in this section.
Exports_Time_Stamp	cyboxCommon: DateTimeObjectPropertyType	0..1	The Exports_Time_Stamp property specifies the date and time the export data was created.
Number_Of_Addresses	cyboxCommon: LongObjectPropertyType	0..1	The Number_Of_Addresses property specifies the number of addresses in the export data section's address table.
Number_Of_Names	cyboxCommon: LongObjectPropertyType	0..1	The Number_Of_Names property specifies the number of names in the export data section's name table.
Number_Of_Functions	cyboxCommon: IntegerObjectPropertyType	0..1	The Number_Of_Functions property specifies the total number of functions that are exported by the PE file.

3.4 PEExportedFunctionsType Class

The PEExportedFunctionsType class specifies a list of PE exported functions.

The property table of the PEExportedFunctionsType class is given in **Table 3-4**.

Table 3-4. Properties of the PEExportedFunctionsType class

Name	Type	Multiplicity	Description
Exported_Function	PEExportedFunctionType	1..*	The Exported_Function property specifies a single property in the list of exported functions.

3.5 PESectionListType Class

The PESectionListType class captures a list of sections that appear in the PE file.

The property table of the PESectionListType class is given in **Table 3-5**.

Table 3-5. Properties of the PESectionListType class

Name	Type	Multiplicity	Description
Section	PESectionType	1..*	Specifies a property of a list of PE file sections.

3.6 EntropyType Class

The EntropyType class captures the result of an entropy computation.

The property table of the EntropyType class is given in **Table 3-6**.

Table 3-6. Properties of the EntropyType class

Name	Type	Multiplicity	Description
Value	cyboxCommon: FloatObjectPropertyType	0..1	The Value property specifies the computed entropy value.

Min	cyboxCommon: FloatObjectPropertyType	0..1	The Min property specifies the smallest possible value for the entropy computation.
Max	cyboxCommon: FloatObjectPropertyType	0..1	The Max property specifies the largest possible value for the entropy computation (e.g., this would be 8 if the entropy computations are based on bits of information).

3.7 PEImportType Class

The PEImportType class is intended as container for the properties relevant to PE binary imports.

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

Table 3-7. Properties of the PEImportType class

Name	**Type**	**Multiplicity**	**Description**
delay_load	basicDataTypes:Boolean	0..1	The delay_load property is a boolean value that is intended to describe whether a PE binary import is delay-load or not.
initially_visible	basicDataTypes:Boolean	0..1	The initially_visible property refers to whether the import is initially visible, with regards to being initially visible or hidden in relation to PE binary packing. A packed binary will typically have few initially visible imports and thus it is necessary to make the distinction between those that are visible initially or only after the binary is unpacked.
File_Name	cyboxCommon: StringObjectPropertyType	0..1	The File_Name property specifies the name of the library (file) that the PE binary imports.
Imported_Functions	PEImportedFunctionsType	0..1	The Imported_Functions property is used to list any functions imported from a particular library.

Virtual_Address	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Virtual_Address property specifies the relative virtual address (RVA) of the PE binary library import.

3.8 PEImportedFunctionsType Class

The PEImportedFunctionsType class captures a list of functions imported by the PE file.

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

Table 3-8. Properties of the PEImportedFunctionsType class

Name	**Type**	**Multiplicity**	**Description**
Imported_Function	PEImportedFunctionType	1..*	Specifies a single property in a list of imported functions.

3.9 PEResourceType Class

The PEResourceType class is intended as container for the properties relevant to PE binary resources.

The property table of the PEResourceType class is given in **Table 3-9**.

Table 3-9. Properties of the PEResourceType class

Name	**Type**	**Multiplicity**	**Description**
Type	PEResourceContentType	0..1	The Type property refers to the type of data referred to by this resource.
Name	cyboxCommon: StringObjectPropertyType	0..1	The Name property specifies the name of the resource used by the PE binary.

Size	cyboxCommon: PositiveIntegerObjectPropertyType	0..1	The Size property specifies the size of the resource, in bytes.
Virtual_Address	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Virtual_Address property specifies the relative virtual address (RVA) of the resource data.
Language	cyboxCommon: StringObjectPropertyType	0..1	The Language property specifies the name of the language (LANG) defined for the resource, if applicable.
Sub_Language	cyboxCommon: StringObjectPropertyType	0..1	The Sub_Language property specifies the name of the sub language (SUBLANG) defined for the resource, if applicable.
Hashes	cyboxCommon:HashListType	0..1	The Hashes property is used to include any hash values computed using the specified PE binary resource as input.
Data	cyboxCommon: StringObjectPropertyType	0..1	The Data property captures the actual data contained in the resource, most commonly as base64-encoded.

3.10 PEVersionInfoResourceType Class

The PEVersionInfoResourceType class characterizes the special VERSIONINFO resource class. For more information please see: http://msdn.microsoft.com/en-us/library/windows/desktop/aa381058(v=vs.85).aspx.

The property table of the PEVersionInfoResourceType class is given in **Table 3-10**.

Table 3-10. Properties of the PEVersionInfoResourceType class

Name	Type	Multiplicity	Description
Comments	cyboxCommon:	0..1	The Comments property captures any additional information that should

	StringObjectPropertyType		be displayed for diagnostic purposes.
CompanyName	cyboxCommon: StringObjectPropertyType	0..1	The `CompanyName` property captures the company that produced the file - for example, "Microsoft Corporation" or "Standard Microsystems Corporation, Inc.".
FileDescription	cyboxCommon: StringObjectPropertyType	0..1	The `FileDescription` property captures the file description to be presented to users. This string may be displayed in a list box when the user is choosing files to install - for example, "Keyboard Driver for AT-Style Keyboards".
FileVersion	cyboxCommon: StringObjectPropertyType	0..1	The `FileVersion` property captures the version number of the file - for example, "3.10" or "5.00.RC2".
InternalName	cyboxCommon: StringObjectPropertyType	0..1	The `InternalName` property captures the internal name of the file, if one exists - for example, a module name if the file is a dynamic-link library. If the file has no internal name, this string should be the original filename, without extension.
LangID	cyboxCommon: StringObjectPropertyType	0..1	The `LangID` property captures the localization language identifier specified in the version-information resource.
LegalCopyright	cyboxCommon: StringObjectPropertyType	0..1	The `LegalCopyright` property captures the copyright notices that apply to the file. This should include the full text of all notices, legal symbols, copyright dates, and so on.
LegalTrademarks	cyboxCommon: StringObjectPropertyType	0..1	The `LegalTrademarks` property captures the trademarks and registered trademarks that apply to the file. This should include the full text of all notices, legal symbols, trademark numbers, and so on.
OriginalFilename	cyboxCommon: StringObjectPropertyType	0..1	The `OriginalFilename` property captures the original name of the file, not including a path. This information enables an application to determine whether a file has been renamed by a user. The format of

			the name depends on the file system for which the file was created.
PrivateBuild	cyboxCommon: StringObjectPropertyType	0..1	The PrivateBuild property captures the information about a private version of the file - for example, "Built by TESTER1 on \TESTBED". This string should be present only if VS_FF_PRIVATEBUILD is specified in the fileflags parameter of the root block.
ProductName	cyboxCommon: StringObjectPropertyType	0..1	The ProductName property captures the name of the product with which the file is distributed. This property SHOULD be provided.
ProductVersion	cyboxCommon: StringObjectPropertyType	0..1	The ProductVersion property captures the version of the product with which the file is distributed - for example, "3.10" or "5.00.RC2".
SpecialBuild	cyboxCommon: StringObjectPropertyType	0..1	The SpecialBuild property captures the text that indicates how this version of the file differs from the standard version - for example, "Private build for TESTER1 solving mouse problems on M250 and M250E computers". This string should be present only if VS_FF_SPECIALBUILD is specified in the fileflags parameter of the root block.

3.11 PEExportedFunctionType Class

The PEExportedFunctionType class specifies the class describing exported functions.

The property table of the PEExportedFunctionType class is given in **Table 3-11**.

Table 3-11. Properties of the PEExportedFunctionType class

Name	**Type**	**Multiplicity**	**Description**
Function_Name	cyboxCommon: StringObjectPropertyType	0..1	The Function_Name property specifies the name of the function exported by the PE binary.

Entry_Point	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Entry_Point property specifies the entry point of the function exported by the PE binary.
Ordinal	cyboxCommon: NonNegativeIntegerObjectPropertyType	0..1	The Ordinal property specifies the ordinal value (index) of the function exported by the PE binary.

3.12 PEResourceListType Class

The PEResourceListType class specifies a list of resources found in the PE file. The UML diagram corresponding to the PEResourceListType class is shown in **Figure 3-3.**

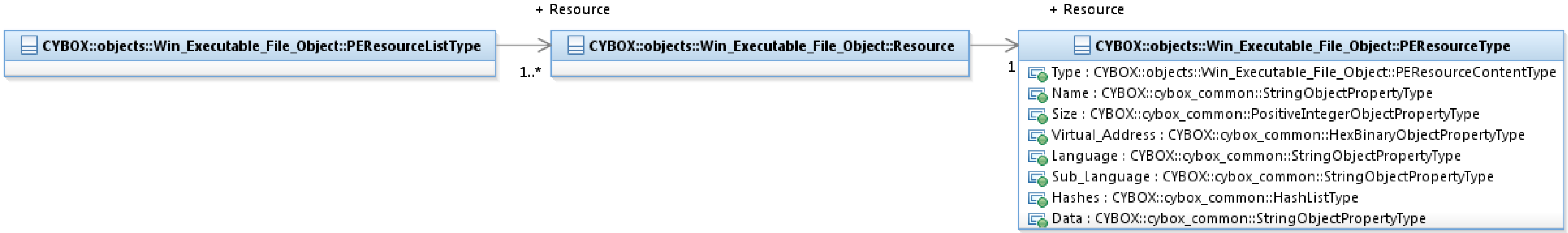

CYBOX::objects::Win_Executable_File_Object::PEResourceListType
+ Resource
1..*
CYBOX::objects::Win_Executable_File_Object::Resource
+ Resource
1
CYBOX::objects::Win_Executable_File_Object::PEResourceType
Type : CYBOX::objects::Win_Executable_File_Object::PEResourceContentType
Name : CYBOX::cybox_common::StringObjectPropertyType
Size : CYBOX::cybox_common::PositiveIntegerObjectPropertyType
Virtual_Address : CYBOX::cybox_common::HexBinaryObjectPropertyType
Language : CYBOX::cybox_common::StringObjectPropertyType
Sub_Language : CYBOX::cybox_common::StringObjectPropertyType
Hashes : CYBOX::cybox_common::HashListType
Data : CYBOX::cybox_common::StringObjectPropertyType

Figure 3-3. UML diagram for the PEResourceListType class

The property table of the PEResourceListType class is given in **Table 3-12**.

Table 3-12. Properties of the PEResourceListType class

Name	**Type**	**Multiplicity**	**Description**
Resource	PEResourceType	1..*	Specifies a property of a list of resources.

3.13 PEImportedFunctionType Class

The PEImportedFunctionType class specifies the class describing imported functions.

The property table of the PEImportedFunctionType class is given in **Table 3-13**.

Table 3-13. Properties of the PEImportedFunctionType class

Name	Type	Multiplicity	Description
Function_Name	cyboxCommon:StringObjectPropertyType	0..1	The Function_Name property specifies the name of the function from the specified library that the PE binary imports.
Hint	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Hint property specifies the index into the export table of the library that the function is found in.
Ordinal	cyboxCommon: NonNegativeIntegerObjectPropertyType	0..1	The Ordinal property specifies the ordinal value (index) of the function in the library that it is found in.
Bound	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Bound property specifies the precomputed address if the imported function is bound.
Virtual_Address	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Virtual_Address property specifies the relative virtual address (RVA) of the PE binary library imported function.

3.14 PEImportListType Class

The `PEImportListType` class specifies a list of functions in an import data section. The UML diagram corresponding to the `PEImportListType` class is shown in **Figure 3-4**.

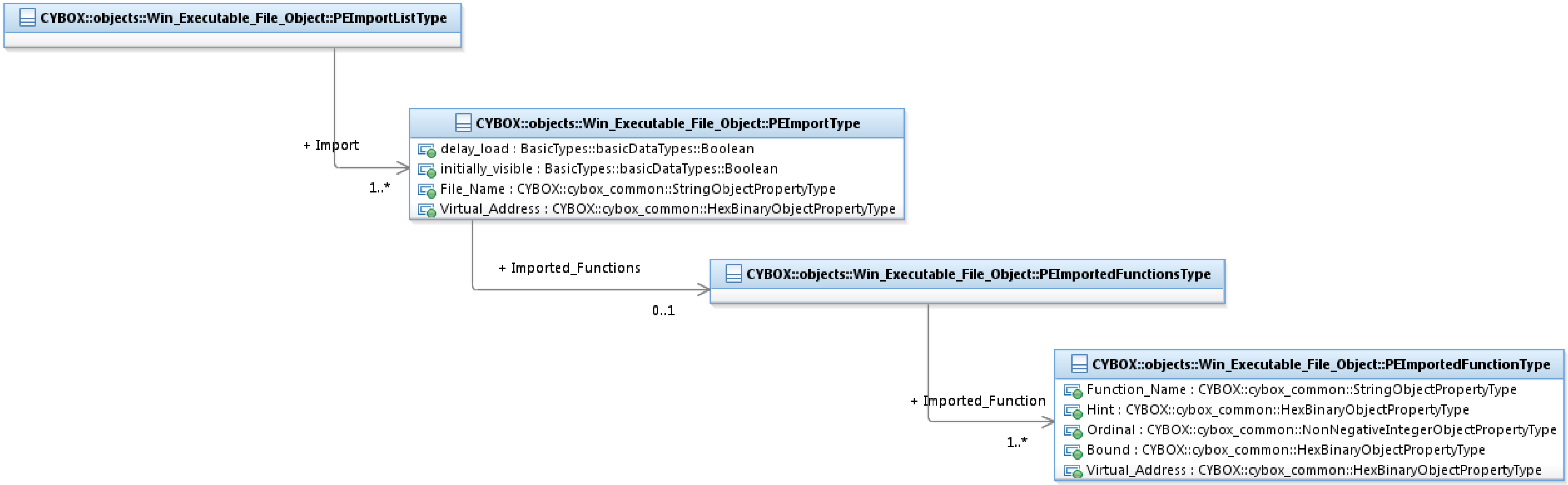

CYBOX::objects::Win_Executable_File_Object::PEImportListType
+ Import
1..*
CYBOX::objects::Win_Executable_File_Object::PEImportType
delay_load : BasicTypes::basicDataTypes::Boolean
initially_visible : BasicTypes::basicDataTypes::Boolean
File_Name : CYBOX::cybox_common::StringObjectPropertyType
Virtual_Address : CYBOX::cybox_common::HexBinaryObjectPropertyType
+ Imported_Functions
0..1
CYBOX::objects::Win_Executable_File_Object::PEImportedFunctionsType
+ Imported_Function
1..*
CYBOX::objects::Win_Executable_File_Object::PEImportedFunctionType
Function_Name : CYBOX::cybox_common::StringObjectPropertyType
Hint : CYBOX::cybox_common::HexBinaryObjectPropertyType
Ordinal : CYBOX::cybox_common::NonNegativeIntegerObjectPropertyType
Bound : CYBOX::cybox_common::HexBinaryObjectPropertyType
Virtual_Address : CYBOX::cybox_common::HexBinaryObjectPropertyType

Figure 3-4. UML diagram for the `PEImportListType` class

The property table of the `PEImportListType` class is given in **Table 3-14**.

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

Name	Type	Multiplicity	Description
Import	`PEImportType`	1..*	The `Import` property specifies a single property in a list of imported functions.

3.15 PESectionType Class

The PESectionType class is intended as container for the properties relevant to PE binary sections. A PE Section consists of a header and data. The PESectionType contains properties that describe the Section Header and metadata computed about the section (e.g., hashes, entropy). The UML diagram corresponding to the PESectionType class is shown in **Figure 3-5.**

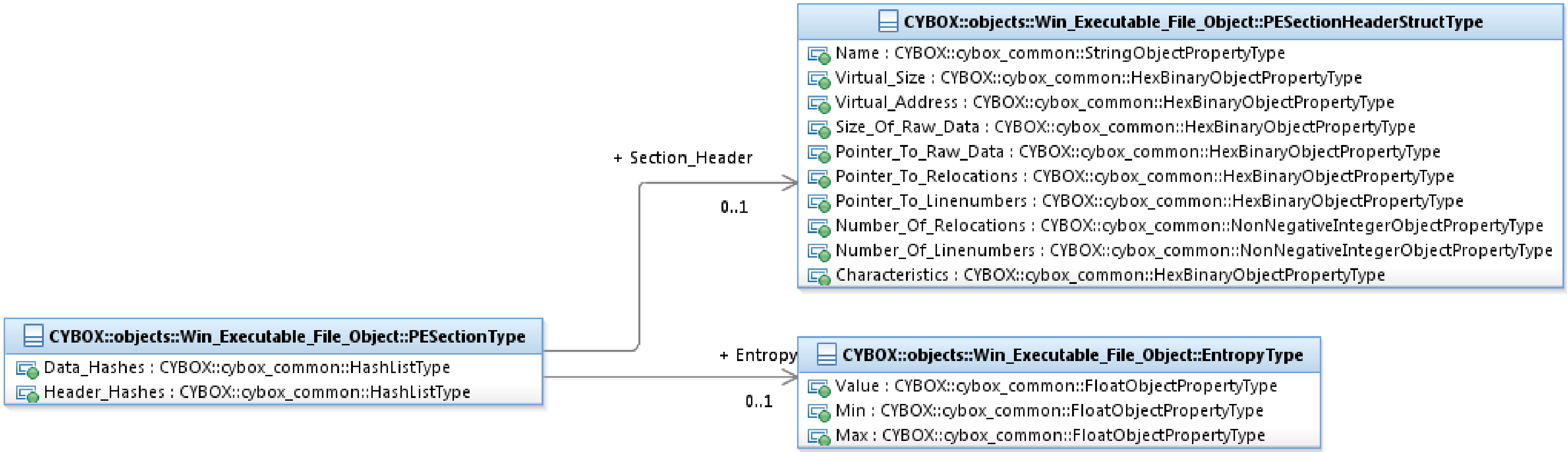

CYBOX::objects::Win_Executable_File_Object::PESectionHeaderStructType
Name : CYBOX::cybox_common::StringObjectPropertyType
Virtual_Size : CYBOX::cybox_common::HexBinaryObjectPropertyType
Virtual_Address : CYBOX::cybox_common::HexBinaryObjectPropertyType
Size_Of_Raw_Data : CYBOX::cybox_common::HexBinaryObjectPropertyType
Pointer_To_Raw_Data : CYBOX::cybox_common::HexBinaryObjectPropertyType
Pointer_To_Relocations : CYBOX::cybox_common::HexBinaryObjectPropertyType
Pointer_To_Linenumbers : CYBOX::cybox_common::HexBinaryObjectPropertyType
Number_Of_Relocations : CYBOX::cybox_common::NonNegativeIntegerObjectPropertyType
Number_Of_Linenumbers : CYBOX::cybox_common::NonNegativeIntegerObjectPropertyType
Characteristics : CYBOX::cybox_common::HexBinaryObjectPropertyType
+ Section_Header
0..1
CYBOX::objects::Win_Executable_File_Object::PESectionType
Data_Hashes : CYBOX::cybox_common::HashListType
Header_Hashes : CYBOX::cybox_common::HashListType
+ Entropy
0..1
CYBOX::objects::Win_Executable_File_Object::EntropyType
Value : CYBOX::cybox_common::FloatObjectPropertyType
Min : CYBOX::cybox_common::FloatObjectPropertyType
Max : CYBOX::cybox_common::FloatObjectPropertyType

Figure 3-5. UML diagram for the PESectionType class

The property table of the PESectionType class is given in **Table 3-15**.

Table 3-15. Properties of the PESectionType class

Name	Type	Multiplicity	Description
Section_Header	PESectionHeaderStructType	0..1	The Section_Header property contains characteristics of the section's section header structure.
Data_Hashes	cyboxCommon:HashListType	0..1	The Data_Hashes property is used to include any hash values computed using the data contained in the specified PE binary section as input.

Entropy	EntropyType	0..1	The Entropy property specifies the calculated entropy of the PE binary section.
Header_Hashes	cyboxCommon:HashListType	0..1	The Header_Hashes property is used to include any hash values computed using the header of the specified PE binary section as input.

3.16 PEDataDirectoryStructType Class

The PEDataDirectoryStructType class is intended as container for the properties relevant to a PE binary's data directory structure.

The property table of the PEDataDirectoryStructType class is given in **Table 3-16**.

Table 3-16. Properties of the PEDataDirectoryStructType class

Name	Type	Multiplicity	Description
Virtual_Address	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Virtual_Address property specifies the relative virtual address (RVA) of the data structure.
Size	cyboxCommon: NonNegativeIntegerObjectPropertyType	0..1	The Size property specifies the size of the data structure, in bytes.

3.17 PESectionHeaderStructType Class

The PESectionHeaderStructType class is intended as container for the properties relevant to a PE binary's section header structure.

The property table of the PESectionHeaderStructType class is given in **Table 3-17**.

Table 3-17. Properties of the PESectionHeaderStructType class

Name	Type	Multiplicity	Description
Name	cyboxCommon: StringObjectPropertyType	0..1	The Name property specifies the name of the PE binary section.
Virtual_Size	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Virtual_Size property is the total size of the PE binary section when loaded into memory. It is valid only for executables and should be 0 for object files.
Virtual_Address	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Virtual_Address property specifies the relative virtual address (RVA) of the PE binary section.
Size_Of_Raw_Data	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Size_Of_Raw_Data property specifies the size of the data contained in the PE binary section.
Pointer_To_Raw_Data	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Pointer_To_Raw_Data property specifies the file offset of the beginning of the PE binary section.
Pointer_To_Relocations	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Pointer_To_Relocations property specifies the offset of the PE binary section relocations, if applicable.
Pointer_To_Linenumbers	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Pointer_To_Linenumbers property specifies the beginning of line-number entries for the section.
Number_Of_Relocations	cyboxCommon: NonNegativeIntegerObjectPropertyType	0..1	The Number_Of_Relocations property specifies the number of relocations defined for the specified PE binary section.
Number_Of_Linenumbers	cyboxCommon: NonNegativeIntegerObjectPropertyType	0..1	The Number_Of_Linenumbers property specifies the number of line number entries for the section. Should be 0.

Characteristics	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Chara21cteristics property specifies any flags defined for the specified PE binary section.

3.18 DOSHeaderType Class

The DOSHeaderType class is a container for the characteristics of the _IMAGE_DOS_HEADER structure, which can be found in Winnt.h and pe.h. See http://www.csn.ul.ie/~caolan/pub/winresdump/winresdump/doc/pefile.html for more information about the winnt.h file, and http://www.tavi.co.uk/phobos/exeformat.html for even more clarification.

The property table of the DOSHeaderType class is given in **Table 3-18**.

Table 3-18. Properties of the DOSHeaderType class

Name	Type	Multiplicity	Description
e_magic	cyboxCommon: HexBinaryObjectPropertyType	0..1	The e_magic property specifies the magic number, specifically the Windows OS signature value, which can either take on MZ for DOS (which is, for all intents and purposes, most Windows executables), NE for OS2, LE for OS2 LE, or PE00 for NT.
e_cblp	cyboxCommon: HexBinaryObjectPropertyType	0..1	The e_cblp property specifies the number of bytes actually used in the last page, with the special case of a full page being represented by a value of zero (since the last page is never empty). For example, assuming a page size of 512 bytes, this value would be 0x0000 for a 1024 byte file, and 0x0001 for a 1025 byte file (since it only contains one valid byte).
e_cp	cyboxCommon: HexBinaryObjectPropertyType	0..1	The e_cp property specifies the number of pages required to hold the file. For example, if the file contains 1024 bytes, and we assume the file has pages of a size of 512 bytes, this word would contain 0x0002; if the file contains 1025 bytes, this word would contain 0x0003.
e_crlc	cyboxCommon:	0..1	The e_crlc property specifies the number of relocation items, i.e. the number of entries that exist in the relocation pointer table. If there are no

	HexBinaryObjectPropertyType		relocation entries, this value is zero.
e_cparhdr	cyboxCommon: HexBinaryObjectPropertyType	0..1	The e_cparhdr property specifies the size of the executable header in terms of paragraphs (16 byte chunks). It indicates the offset of the program's compiled/assembled and linked image (the load module) within the executable file. The size of the load module can be deduced by subtracting this value (converted to bytes) from the overall file size derived from combining the e_cp (number of file pages) and e_cblp (number of bytes in last page) values. The header always spans an even number of paragraphs.
e_minalloc	cyboxCommon: HexBinaryObjectPropertyType	0..1	The e_minalloc property specifies the minimum number of extra paragraphs needed to be allocated to begin execution. This is *in addition* to the memory required to hold the load module. This value normally represents the total size of any uninitialized data and/or stack segments that are linked at the end of a program. This space is not directly included in the load module since there are no particular initializing values and it would simply waste disk space.
e_maxalloc	cyboxCommon: HexBinaryObjectPropertyType	0..1	The e_maxalloc property specifies the maximum number of extra paragraphs needed to be allocated by the program before it begins execution. This indicates *additional* memory over and above that required by the load module and the value specified by MINALLOC. If the request cannot be satisfied, the program is allocated as much memory as is available.
e_ss	cyboxCommon: HexBinaryObjectPropertyType	0..1	The e_ss property specifies the initial SS value, which is the paragraph address of the stack segment relative to the start of the load module. At load time, this value is relocated by adding the address of the start segment of the program to it, and the resulting value is placed in the SS register before the program is started. In DOS, the start segment of the program is the first segment boundary in memory after the PSP.
e_sp	cyboxCommon: HexBinaryObjectPropertyType	0..1	The e_sp property specifies the initial SP value, which is the absolute value that must be loaded into the SP register before the program is given control. Since the actual stack segment is determined by the loader, and

			this is merely a value within that segment, it does not need to be relocated.
e_csum	`cyboxCommon:HexBinaryObjectPropertyType`	0..1	The `e_csum` property specifies the checksum of the contents of the executable file. It is used to ensure the integrity of the data within the file. For full details on how this checksum is calculated, see http://www.tavi.co.uk/phobos/exeformat.html#checksum.
e_ip	`cyboxCommon:HexBinaryObjectPropertyType`	0..1	The `e_ip` property specifies the initial IP value, which is the absolute value that should be loaded into the IP register in order to transfer control to the program. Since the actual code segment is determined by the loader, and this is merely a value within that segment, it does not need to be relocated.
e_cs	`cyboxCommon:HexBinaryObjectPropertyType`	0..1	The `e_cs` property specifies the pre-relocated initial CS value, relative to the start of the load module that should be placed in the CS register in order to transfer control to the program. At load time, this value is relocated by adding the address of the start segment of the program to it, and the resulting value is placed in the CS register when control is transferred.
e_lfarlc	`cyboxCommon:HexBinaryObjectPropertyType`	0..1	The `e_lfarlc` property specifies the file address of the relocation table, or more specifically, the offset from the start of the file to the relocation pointer table. This value must be used to locate the relocation pointer table (rather than assuming a fixed location) because variable-length information pertaining to program overlays can occur before this table, causing its position to vary. A value of 0x40 in this property generally indicates a different kind of executable file, not a DOS 'MZ' type.
e_ovro	`cyboxCommon:HexBinaryObjectPropertyType`	0..1	The `e_ovro` property specifies the overlay number, which is normally set to 0x0000, because few programs actually have overlays. It changes only in files containing programs that use overlays. See http://www.tavi.co.uk/phobos/exeformat.html#overlaynote for more information about overlays.

reserved1	cyboxCommon: HexBinaryObjectPropertyType	0..4	The reserved1 property specifies reserved words for the program (known in winnt.h as e_res[4]), usually set to zero by the linker. In this case, just use a single reserved1 set to zero; if not zero create four reserved1 with the correct value.
e_oemid	cyboxCommon: HexBinaryObjectPropertyType	0..1	The e_oemid property specifies the identifier for the OEM for e_oeminfo.
e_oeminfo	cyboxCommon: HexBinaryObjectPropertyType	0..1	The e_oeminfo property specifies the OEM information for a specific value of e_oeminfo.
reserved2	cyboxCommon: HexBinaryObjectPropertyType	0..1	The reserved2 property specifies reserved words for the program (known in winnt.h as e_res[10]), usually set to zero by the linker. In this case, just use a single reserved1 set to zero; if not zero create ten reserved1 with the correct value.
e_lfanew	cyboxCommon: HexBinaryObjectPropertyType	0..1	The e_lfanew property specifies the file address of the of the new exe header. In particular, it is a 4-byte offset into the file where the PE file header is located. It is necessary to use this offset to locate the PE header in the file.
Hashes	cyboxCommon:HashListType	0..1	The Hashes property is used to include any hash values computed using the specified PE binary MS-DOS header as input.

3.19 PEHeadersType Class

The PEHeadersType class specifies the headers found in PE and COFF files. The UML diagram corresponding to the PEHeadersType class is shown in **Figure 3-6.**

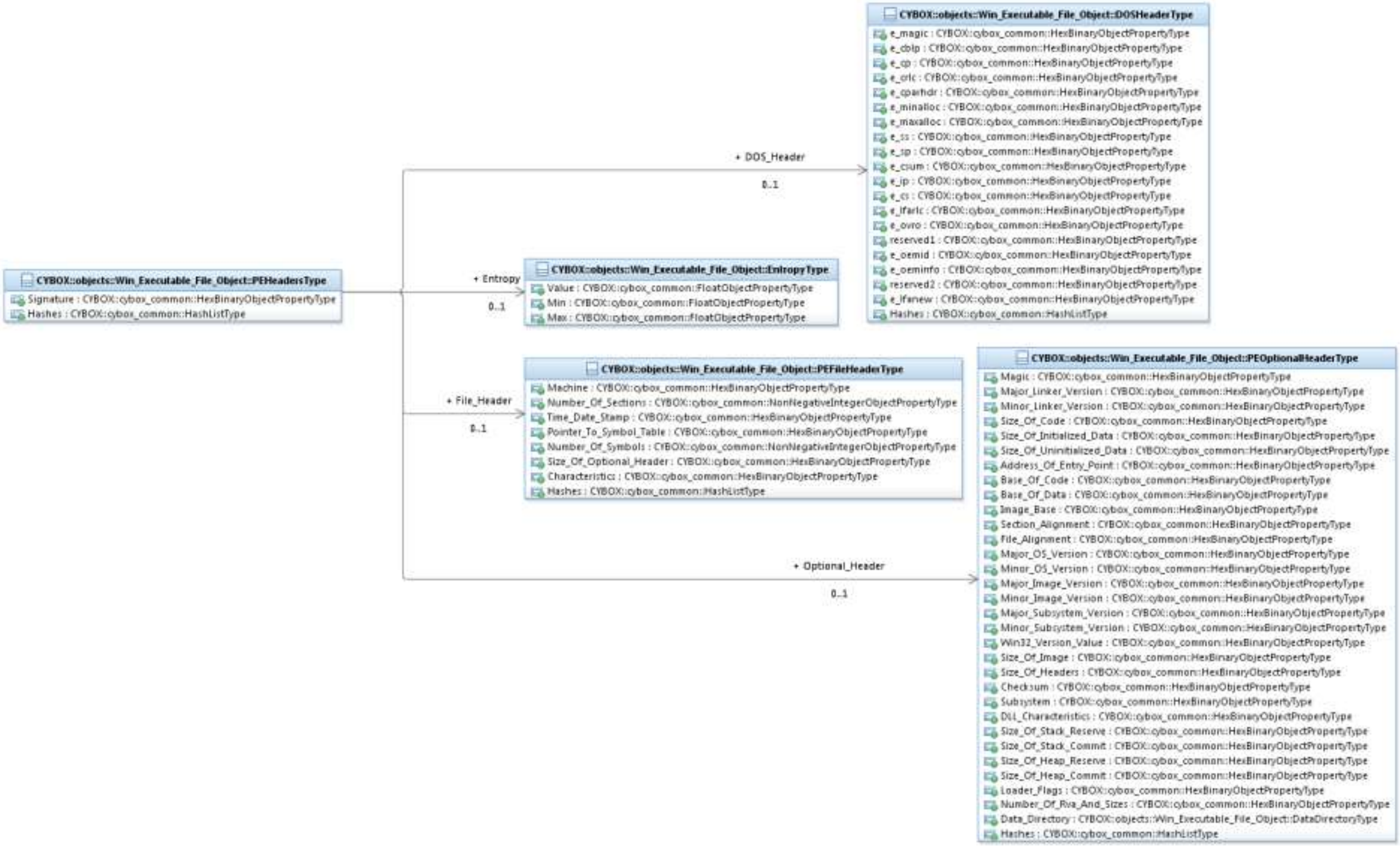

CYBOX::objects::Win_Executable_File_Object::PEHeadersType
Signature : CYBOX::cybox_common::HexBinaryObjectPropertyType
Hashes : CYBOX::cybox_common::HashListType
+ DOS_Header
0..1
CYBOX::objects::Win_Executable_File_Object::DOSHeaderType
e_magic : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_cblp : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_cp : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_crlc : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_cparhdr : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_minalloc : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_maxalloc : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_ss : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_sp : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_csum : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_ip : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_cs : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_lfarlc : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_ovro : CYBOX::cybox_common::HexBinaryObjectPropertyType
reserved1 : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_oemid : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_oeminfo : CYBOX::cybox_common::HexBinaryObjectPropertyType
reserved2 : CYBOX::cybox_common::HexBinaryObjectPropertyType
e_lfanew : CYBOX::cybox_common::HexBinaryObjectPropertyType
Hashes : CYBOX::cybox_common::HashListType
+ Entropy
0..1
CYBOX::objects::Win_Executable_File_Object::EntropyType
Value : CYBOX::cybox_common::FloatObjectPropertyType
Min : CYBOX::cybox_common::FloatObjectPropertyType
Max : CYBOX::cybox_common::FloatObjectPropertyType
+ File_Header
0..1
CYBOX::objects::Win_Executable_File_Object::PEFileHeaderType
Machine : CYBOX::cybox_common::HexBinaryObjectPropertyType
Number_Of_Sections : CYBOX::cybox_common::NonNegativeIntegerObjectPropertyType
Time_Date_Stamp : CYBOX::cybox_common::HexBinaryObjectPropertyType
Pointer_To_Symbol_Table : CYBOX::cybox_common::HexBinaryObjectPropertyType
Number_Of_Symbols : CYBOX::cybox_common::NonNegativeIntegerObjectPropertyType
Size_Of_Optional_Header : CYBOX::cybox_common::HexBinaryObjectPropertyType
Characteristics : CYBOX::cybox_common::HexBinaryObjectPropertyType
Hashes : CYBOX::cybox_common::HashListType
+ Optional_Header
0..1
CYBOX::objects::Win_Executable_File_Object::PEOptionalHeaderType
Magic : CYBOX::cybox_common::HexBinaryObjectPropertyType
Major_Linker_Version : CYBOX::cybox_common::HexBinaryObjectPropertyType
Minor_Linker_Version : CYBOX::cybox_common::HexBinaryObjectPropertyType
Size_Of_Code : CYBOX::cybox_common::HexBinaryObjectPropertyType
Size_Of_Initialized_Data : CYBOX::cybox_common::HexBinaryObjectPropertyType
Size_Of_Uninitialized_Data : CYBOX::cybox_common::HexBinaryObjectPropertyType
Address_Of_Entry_Point : CYBOX::cybox_common::HexBinaryObjectPropertyType
Base_Of_Code : CYBOX::cybox_common::HexBinaryObjectPropertyType
Base_Of_Data : CYBOX::cybox_common::HexBinaryObjectPropertyType
Image_Base : CYBOX::cybox_common::HexBinaryObjectPropertyType
Section_Alignment : CYBOX::cybox_common::HexBinaryObjectPropertyType
File_Alignment : CYBOX::cybox_common::HexBinaryObjectPropertyType
Major_OS_Version : CYBOX::cybox_common::HexBinaryObjectPropertyType
Minor_OS_Version : CYBOX::cybox_common::HexBinaryObjectPropertyType
Major_Image_Version : CYBOX::cybox_common::HexBinaryObjectPropertyType
Minor_Image_Version : CYBOX::cybox_common::HexBinaryObjectPropertyType
Major_Subsystem_Version : CYBOX::cybox_common::HexBinaryObjectPropertyType
Minor_Subsystem_Version : CYBOX::cybox_common::HexBinaryObjectPropertyType
Win32_Version_Value : CYBOX::cybox_common::HexBinaryObjectPropertyType
Size_Of_Image : CYBOX::cybox_common::HexBinaryObjectPropertyType
Size_Of_Headers : CYBOX::cybox_common::HexBinaryObjectPropertyType
Checksum : CYBOX::cybox_common::HexBinaryObjectPropertyType
Subsystem : CYBOX::cybox_common::HexBinaryObjectPropertyType
DLL_Characteristics : CYBOX::cybox_common::HexBinaryObjectPropertyType
Size_Of_Stack_Reserve : CYBOX::cybox_common::HexBinaryObjectPropertyType
Size_Of_Stack_Commit : CYBOX::cybox_common::HexBinaryObjectPropertyType
Size_Of_Heap_Reserve : CYBOX::cybox_common::HexBinaryObjectPropertyType
Size_Of_Heap_Commit : CYBOX::cybox_common::HexBinaryObjectPropertyType
Loader_Flags : CYBOX::cybox_common::HexBinaryObjectPropertyType
Number_Of_Rva_And_Sizes : CYBOX::cybox_common::HexBinaryObjectPropertyType
Data_Directory : CYBOX::objects::Win_Executable_File_Object::DataDirectoryType
Hashes : CYBOX::cybox_common::HashListType

Figure 3-6. UML diagram for the `PEHeaderType` class

The property table of the PEHeadersType class is given in **Table 3-19**.

Table 3-19. Properties of the PEHeadersType class

Name	Type	Multiplicity	Description
DOS_Header	DOSHeaderType	0..1	The DOS_Header property refers to the MS-DOS PE header and its associated characteristics.
Signature	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Signature property specifies the 4-bytes signature that identifies the file as a PE file.
File_Header	PEFileHeaderType	0..1	The File_Header property refers to the PE file header (sometimes referred to as the COFF header) and its associated characteristics.
Optional_Header	PEOptionalHeaderType	0..1	The Optional_Header property refers to the PE optional header and its associated characteristics. The Optional Header is required for executable (PE) files, but optional for object (COFF) files.
Entropy	EntropyType	0..1	The Entropy property specifies the calculated entropy of the PE file header.
Hashes	cyboxCommon:HashListType	0..1	The Hashes property is used to include any hash values computed using the specified PE binary file header as input.

3.20 PEFileHeaderType Class

The PEFileHeaderType class refers to the PE file header (sometimes referred to as the COFF header) and its associated characteristics.

The property table of the PEFileHeaderType class is given in **Table 3-20**.

Table 3-20. Properties of the PEFileHeaderType class

Name	Type	Multiplicity	Description
Machine	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Machine property specifies the type of target machine.
Number_Of_Sections	cyboxCommon: NonNegativeIntegerObjectPropertyType	0..1	The Number_Of_Sections property specifies the number of sections in the file.
Time_Date_Stamp	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Time_Date_Stamp property specifies the time when the file was created (the low 32 bits of the number of seconds since epoch).
Pointer_To_Symbol_Table	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Pointer_To_Symbol_Table property specifies the file offset of the COFF symbol table (should be 0).
Number_Of_Symbols	cyboxCommon: NonNegativeIntegerObjectPropertyType	0..1	The Number_Of_Symbols property specifies the number of entries in the symbol table (should be 0).
Size_Of_Optional_Header	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Size_Of_Optional_Header property specifies the size of the optional header. It should be 0 for object files and non-zero for executables.
Characteristics	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Characteristics property specifies the flags that indicate the file's characteristics.
Hashes	cyboxCommon:HashListType	0..1	The Hashes property specifies any hashes computed for the Optional Header.

3.21 SubsystemType Data Type

The SubsystemType data type specifies the subsystem type. Its core value SHOULD be a literal from the SubsystemTypeEnum enumeration. It extends the BaseObjectPropertyType data type, in order to permit complex (i.e. regular-expression based) specifications.

3.22 PEType Data Type

The PEType data type specifies the PE file type. Its core value SHOULD be a literal from the PETypeEnum enumeration. It extends the BaseObjectPropertyType data type, in order to permit complex (i.e. regular-expression based) specifications.

3.23 PEOptionalHeaderType Class

The PEOptionalHeaderType class describes the PE Optional Header structure. Additional computed metadata, e.g., hashes of the header, are also included.

The property table of the PEOptionalHeaderType class is given in **Table 3-21**.

Table 3-21. Properties of the PEOptionalHeaderType class

Name	Type	Multiplicity	Description
Magic	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Magic property specifies the unsigned integer that indicates the type of executable file.
Major_Linker_Version	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Major_Linker_Version property specifies the linker major version number.
Minor_Linker_Version	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Minor_Linker_Version property specifies the linker minor version number.
Size_Of_Code	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Size_Of_Code property specifies the size of the code (text) section. If there are multiple sections, size is the sum of the sizes of each.

Size_Of_Initialized_Data	`cyboxCommon: HexBinaryObjectPropertyType`	0..1	The `Size_Of_Initialized_Data` property specifies the size of the initialized data section. If there are multiple sections, size is the sum of the sizes of each.
Size_Of_Uninitialized_Data	`cyboxCommon: HexBinaryObjectPropertyType`	0..1	The `Size_Of_Uninitialized_Data` property specifies the size of the uninitialized (bss) data section. If there are multiple sections, size is the sum of the sizes of each.
Address_Of_Entry_Point	`cyboxCommon: HexBinaryObjectPropertyType`	0..1	The `Address_Of_Entry_Point` property specifies the address of the entry point relative to the image base when the executable is loaded into memory. When there is no entry point (e.g., optional for DLLs), the value should be 0.
Base_Of_Code	`cyboxCommon: HexBinaryObjectPropertyType`	0..1	The `Base_Of_Code` property specifies the address that is relative to the image base of the beginning-of-code section when it is loaded into memory.
Base_Of_Data	`cyboxCommon: HexBinaryObjectPropertyType`	0..1	The `Base_Of_Data` property specifies the address that is relative to the image base of the beginning-of-data section when it is loaded into memory.
Image_Base	`cyboxCommon: HexBinaryObjectPropertyType`	0..1	The `Image_Base` property specifies the preferred address of the first byte of image when loaded into memory; must be a multiple of 64 K.
Section_Alignment	`cyboxCommon: HexBinaryObjectPropertyType`	0..1	The `Section_Alignment` property specifies the alignment (in bytes) of sections when they are loaded into memory.
File_Alignment	`cyboxCommon: HexBinaryObjectPropertyType`	0..1	The `File_Alignment` property specifies the factor (in bytes) that is used to align the raw data of sections in the image file.

Major_OS_Version	`cyboxCommon:`
	`HexBinaryObjectPropertyType`
		0..1	The `Major_OS_Version` property specifies the major version number of the required operating system.

Minor_OS_Version	`cyboxCommon:`
	`HexBinaryObjectPropertyType`
		0..1	The `Minor_OS_Version` property specifies the minor version number of the required operating system.

Major_Image_Version	`cyboxCommon:`
	`HexBinaryObjectPropertyType`
		0..1	The `Major_Image_Version` property specifies the major version number of the image.

Minor_Image_Version	`cyboxCommon:`
	`HexBinaryObjectPropertyType`
		0..1	The `Minor_Image_Version` property specifies the minor version number of the image.

Major_Subsystem_Version	`cyboxCommon:`
	`HexBinaryObjectPropertyType`
		0..1	The `Major_Subsystem_Version` property specifies the major version number of the subsystem.

Minor_Subsystem_Version	`cyboxCommon:`
	`HexBinaryObjectPropertyType`
		0..1	The `Minor_Subsystem_Version` property specifies the minor version number of the subsystem.

Win32_Version_Value	`cyboxCommon:`
	`HexBinaryObjectPropertyType`
		0..1	The `Win32_Version_Value` property is reserved; must be 0.

Size_Of_Image	`cyboxCommon:`
	`HexBinaryObjectPropertyType`
		0..1	The `Size_Of_Image` property specifies the size (in bytes) of the image, including all headers, as the image is loaded in memory.

Size_Of_Headers	`cyboxCommon:`
	`HexBinaryObjectPropertyType`
		0..1	The `Size_Of_Headers` property specifies the combined size of the MS DOS header, PE header, and section headers rounded up to a multiple of FileAlignment.

Checksum	`cyboxCommon:`	0..1	The `Checksum` property specifies the checksum of the PE file.

	HexBinaryObjectPropertyType		
Subsystem	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Subsystem property specifies the subsystem (e.g., GUI, device driver) that is required to run this image.
DLL_Characteristics	cyboxCommon: HexBinaryObjectPropertyType	0..1	The DLL_Characteristics property specifies flags that characterize the PE file.
Size_Of_Stack_Reserve	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Size_Of_Stack_Reserve property specifies the size of the stack to reserve.
Size_Of_Stack_Commit	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Size_Of_Stack_Commit property specifies the size of the stack to commit.
Size_Of_Heap_Reserve	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Size_Of_Heap_Reserve property specifies the size of the local heap space to reserve.
Size_Of_Heap_Commit	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Size_Of_Heap_Commit property specifies the size of the local heap space to commit.
Loader_Flags	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Loader_Flags property is reserved; must be 0.
Number_Of_Rva_And_Sizes	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Number_Of_Rva_And_Sizes property specifies the number of data-directory entries in the remainder of the optional header.
Data_Directory	DataDirectoryType	0..1	The Data_Directory property specifies the data directories in the remainder in the optional header.

Hashes	cyboxCommon: HashListType	0..1	The Hashes property is used to include any hash values computed using the specified PE binary optional header as input.

3.24 DataDirectoryType Class

The DataDirectoryType class specifies the data directories that can appear in the PE file's optional header. The data directories, except the Certificate Table, are loaded into memory so they can be used at runtime.

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

Table 3-22. Properties of the DataDirectoryType class

Name	Type	Multiplicity	Description
Export_Table	PEDataDirectoryStructType	0..1	The Export_Table property specifies the export table data directory.
Import_Table	PEDataDirectoryStructType	0..1	The Import_Table property specifies the import table data directory.
Resource_Table	PEDataDirectoryStructType	0..1	The Resource_Table property specifies the resource table data directory.
Exception_Table	PEDataDirectoryStructType	0..1	The Exception_Table property specifies the exception table data directory.
Certificate_Table	PEDataDirectoryStructType	0..1	The Certificate_Table property specifies the certificate table data directory. The table of certificates is in a file which the data directory points to.
Base_Relocation_Table	PEDataDirectoryStructType	0..1	The Base_Relocation_Table property specifies the base

			relocation table data directory.
Debug	`PEDataDirectoryStructType`	0..1	The `Debug` property specifies the debug data directory.
Architecture	`PEDataDirectoryStructType`	0..1	The `Architecture` property is reserved, must be 0.
Global_Ptr	`PEDataDirectoryStructType`	0..1	The `Global_Ptr` property specifies the RVA of the value to be stored in the global pointer register.
TLS_Table	`PEDataDirectoryStructType`	0..1	The `TLS_Table` property specifies the thread local storage (TLS) table data directory.
Load_Config_Table	`PEDataDirectoryStructType`	0..1	The `Load_Config_Table` property specifies the load configuration table data directory.
Bound_Import	`PEDataDirectoryStructType`	0..1	The `Bound_Import` property specifies the bound import table data directory.
Import_Address_Table	`PEDataDirectoryStructType`	0..1	The `Import_Address_Table` property specifies the import address table (IAT) data directory.
Delay_Import_Descriptor	`PEDataDirectoryStructType`	0..1	The `Delay_Import_Descriptor` property specifies the delay import descriptor data directory.
CLR_Runtime_Header	`PEDataDirectoryStructType`	0..1	The `CLR_Runtime_Header` property specifies the Common Language Runtime (CLR) header data directory.
Reserved	`PEDataDirectoryStructType`	0..1	The `Reserved` property is reserved; must be 0.

3.25 PEBuildInformationType Class

The PEBuildInformationType class captures information about the tools used to build the PE binary, including the compiler and linker.

The property table of the PEBuildInformationType class is given in **Table 3-23**.

Table 3-23. Properties of the PEBuildInformationType class

Name	Type	Multiplicity	Description
Linker_Name	cyboxCommon: StringObjectPropertyType	0..1	The Linker_Name property specifies the name of the linker used to link the PE binary.
Linker_Version	cyboxCommon: StringObjectPropertyType	0..1	The Linker_Version property specifies the version of the linker used to link the PE binary.
Compiler_Name	cyboxCommon: StringObjectPropertyType	0..1	The Compiler_Name property specifies the name of the compiler used to compile the binary.
Compiler_Version	cyboxCommon: StringObjectPropertyType	0..1	The Compiler_Version property specifies the version of the compiler used to compile the binary.

3.26 PEResourceContentType Data Type

The PEResourceContentType data type specifies the PE resource type. Its core value SHOULD be a literal from the PEResourceTypeEnum enumeration. It extends the BaseObjectPropertyType data type, in order to permit complex (i.e. regular-expression based) specifications.

3.27 SubsystemTypeEnum Enumeration

The literals of the SubsystemTypeEnum enumeration are given in **Table 3-24**.

Table 3-24. Literals of the SubsystemTypeEnum enumeration

Enumeration Literal	Description
Unknown	Specifies an unknown subsystem.
Native	Specifies that no subsystem is required to run the image (i.e. only device drivers and native system processes are needed).
Windows_GUI	Specifies the Windows Graphical user interface (GUI) subsystem.
Windows_CUI	Specifies the Windows character-mode user interface (CUI) subsystem.
OS2_CUI	Specifies the OS/2 CUI subsystem.
POSIX_CUI	Specifies the POSIX CUI subsystem.
Native_Win9x_Driver	Specifies the Native Windows 9x drivers. This is denoted by the value IMAGE_SUBSYSTEM_NATIVE_WINDOWS or 0x8.
Windows_CE_GUI	Specifies the Windows CE system with a GUI.
EFI_Application	Specifies the Extensible Firmware Interface (EFI) application.
EFI_Boot_Service_Driver	Specifies the Extensible Firmware Interface (EFI) driver with boot services.
EFI_Runtime_Driver	Specifies the Extensible Firmware Interface (EFI) driver with run-time services.
EFI_ROM	Specifies the Extensible Firmware Interface (EFI) image.

XBOX	Specifies the XBOX system.
Windows_Boot_Application	Specifies the Windows Boot application.

3.28 PETypeEnum Enumeration

The literals of the `PETypeEnum` enumeration are given in **Table 3-25**.

Table 3-25. Literals of the `PETypeEnum` enumeration

Enumeration Literal	Description
Executable	Specifies an executable image (not an OBJ or LIB).
Dll	Specifies a dynamic link library, not a program.
Invalid	Specifies an invalid executable file (i.e. not one of the listed types).

3.29 PEResourceTypeEnum Enumeration

The literals of the `PEResourceTypeEnum` enumeration are given in **Table 3-26**.

Table 3-26. Literals of the `PEResourceTypeEnum` enumeration

Enumeration Literal	Description
Cursor	Specifies a resource that is a cursor or animated cursor defined by naming it and specifying the name of the file that contains it. (To use a particular cursor, the application requests it by name.).

Bitmap	Specifies a resource that is a bitmap defined by naming it and specifying the name of the file that contains it. (To use a particular cursor, the application requests it by name.).
Icon	Specifies a resource that is an icon or animated icon by naming it and specifying the name of the file that contains it. (To use a particular icon, the application requests it by name.).
Menu	Specifies a resource that captures the appearance and function of a menu. Does not define help or regular identifiers, nor uses the MFT_* type and MFS_* state flags.
MenuEX	Specifies a resource that captures the appearance and function of a menu, which can also utilize help or regular identifiers, as well as the MFT_* type and MFS_* state flags.
Popup	Specifies a resource that captures a menu item that can contain menu items and submenus.
Dialog	Specifies a resource that captures a template that an application can use to create dialog boxes. This type is considered obsolete in Windows and newer applications use the DIALOGEX resource.
DialogEX	Specifies a resource that captures a template that newer applications can use to create dialog boxes.
String	Specifies a resource that is a string.
StringTable	Specifies a resource that captures string tables. String resources are Unicode or ASCII strings that can be loaded from the executable file.
Fontdir	Specifies a resource that is a font directory.

Font	Specifies a resource that captures the name of a file that contains a font.
Accelerators	Specifies a resource that captures menu accelerator keys.
RCData	Specifies a resource that captures data resources. Data resources let you include binary data in the executable file.
MessageTable	Specifies a resource that captures a message table by naming it and specifying the name of the file that contains it. The file is a binary resource file generated by the message compiler.
GroupCursor	Specifies a resource that is a group cursor.
GroupIcon	Specifies a resource that is a group icon.
VersionInfo	Specifies a resource that captures version-information. Contains information such as the version number, intended operating system, and so on.
DLGInclude	Specifies a resource that is a dialog include.
PlugPlay	This resource is obsolete and included for completeness.
TextInclude	This is a special resource that is interpreted by Visual C++. For more information see http://go.microsoft.com/FWLink/?LinkId=83951.
TypeLib	This is a special resource that is used with /TLBID and /TLBOUT linker options. For more information see http://go.microsoft.com/FWLink/?LinkId=83960 (for /TLBID) and http://go.microsoft.com/FWLink/?LinkId=83947 (for /TLBOUT).
Vxd	This resource is obsolete and included for completeness.

AniCursor	Specifies a resource that is an animated cursor.
AniIcon	Specifies a resource that is an animated icon.
HTML	Specifies a resource that captures an HTML file.
Manifest	Specifies a resource that captures a manifest file.
MessageTableEntry	Specifies a resource that captures a message table entry.

4 Conformance

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

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

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

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

Appendix A. Acknowledgments

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

Aetna

David Crawford

AIT Austrian Institute of Technology

Roman Fiedler

Florian Skopik

Australia and New Zealand Banking Group (ANZ Bank)

Dean Thompson

Blue Coat Systems, Inc.

Owen Johnson

Bret Jordan

Century Link

Cory Kennedy

CIRCL

Alexandre Dulaunoy

Andras Iklody

Raphaël Vinot

Citrix Systems

Joey Peloquin

Dell

Will Urbanski

Jeff Williams

DTCC

Dan Brown

Gordon Hundley

Chris Koutras

EMC

Robert Griffin

Jeff Odom

Ravi Sharda

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

David Eilken

Chris Ricard

Fortinet Inc.

Gavin Chow

Airbus Group SAS

Joerg Eschweiler

Marcos Orallo

Anomali

Ryan Clough

Wei Huang

Hugh Njemanze

Katie Pelusi

Aaron Shelmire

Jason Trost

Bank of America

Alexander Foley

Center for Internet Security (CIS)

Sarah Kelley

Check Point Software Technologies

Ron Davidson

Cisco Systems

Syam Appala

Ted Bedwell

David McGrew

Pavan Reddy

Omar Santos

Jyoti Verma

Cyber Threat Intelligence Network, Inc. (CTIN)

Doug DePeppe

Jane Ginn

Ben Othman

DHS Office of Cybersecurity and Communications (CS&C)

Richard Struse

Marlon Taylor

EclecticIQ

Marko Dragoljevic

Joep Gommers

Sergey Polzunov

Kenichi Terashita

Fujitsu Limited

Neil Edwards

Frederick Hirsch

Ryusuke Masuoka

Daisuke Murabayashi

Google Inc.

Mark Risher

Hitachi, Ltd.

Kazuo Noguchi

Akihito Sawada

Masato Terada

iboss, Inc.

Paul Martini

Individual

Jerome Athias

Peter Brown

Elysa Jones

Sanjiv Kalkar

Bar Lockwood

Terry MacDonald

Alex Pinto

Intel Corporation

Tim Casey

Kent Landfield

JPMorgan Chase Bank, N.A.

Terrence Driscoll

David Laurance

LookingGlass

Allan Thomson

Lee Vorthman

Mitre Corporation

Greg Back

Jonathan Baker

Sean Barnum

Desiree Beck

Nicole Gong

Jasen Jacobsen

Ivan Kirillov

Richard Piazza

Jon Salwen

Rutger Prins

Andrei Sîrghi

Raymon van der Velde

eSentire, Inc.

Jacob Gajek

FireEye, Inc.

Phillip Boles

Pavan Gorakav

Anuj Kumar

Shyamal Pandya

Paul Patrick

Scott Shreve

Fox-IT

Sarah Brown

Georgetown University

Eric Burger

Hewlett Packard Enterprise (HPE)

Tomas Sander

IBM

Peter Allor

Eldan Ben-Haim

Sandra Hernandez

Jason Keirstead

John Morris

Laura Rusu

Ron Williams

IID

Chris Richardson

Integrated Networking Technologies, Inc.

Patrick Maroney

Johns Hopkins University Applied Physics Laboratory

Karin Marr

Julie Modlin

Mark Moss

Pamela Smith

Kaiser Permanente

Russell Culpepper

Beth Pumo

Lumeta Corporation

Brandon Hoffman

MTG Management Consultants, LLC.

Charles Schmidt
Emmanuelle Vargas-Gonzalez
John Wunder

National Council of ISACs (NCI)
Scott Algeier
Denise Anderson
Josh Poster

NEC Corporation
Takahiro Kakumaru

North American Energy Standards Board
David Darnell

Object Management Group
Cory Casanave

Palo Alto Networks
Vishaal Hariprasad

Queralt, Inc.
John Tolbert

Resilient Systems, Inc.
Ted Julian

Securonix
Igor Baikalov

Siemens AG
Bernd Grobauer

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

James Cabral

National Security Agency
Mike Boyle
Jessica Fitzgerald-McKay

New Context Services, Inc.
John-Mark Gurney
Christian Hunt
James Moler
Daniel Riedel
Andrew Storms

OASIS
James Bryce Clark
Robin Cover
Chet Ensign

Open Identity Exchange
Don Thibeau

PhishMe Inc.
Josh Larkins

Raytheon Company-SAS
Daniel Wyschogrod

Retail Cyber Intelligence Sharing Center (R-CISC)
Brian Engle

Semper Fortis Solutions
Joseph Brand

Splunk Inc.
Cedric LeRoux
Brian Luger
Kathy Wang

TELUS
Greg Reaume
Alan Steer

Threat Intelligence Pty Ltd
Tyron Miller
Andrew van der Stock

ThreatConnect, Inc.
Wade Baker
Cole Iliff
Andrew Pendergast
Ben Schmoker
Jason Spies

TruSTAR Technology

Clayton Long
Michael Pepin
Natalie Suarez
David Waters
Benjamin Yates

Symantec Corp.
Curtis Kostrosky

The Boeing Company
Crystal Hayes

ThreatQuotient, Inc.
Ryan Trost

U.S. Bank
Mark Angel
Brad Butts
Brian Fay
Mona Magathan
Yevgen Sautin

US Department of Defense (DoD)
James Bohling
Eoghan Casey
Gary Katz
Jeffrey Mates

VeriSign
Robert Coderre
Kyle Maxwell
Eric Osterweil

Chris Roblee

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

US Department of Homeland Security
Evette Maynard-Noel
Justin Stekervetz

ViaSat, Inc.
Lee Chieffalo
Wilson Figueroa
Andrew May

Yaana Technologies, LLC
Anthony Rutkowski

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

Appendix B. Revision History

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