CybOX™ Version 2.1.1. Part 24: 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/part24-file/cybox-v2.1.1-csprd01-part24-file.docx (Authoritative)
http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part24-file/cybox-v2.1.1-csprd01-part24-file.html
http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part24-file/cybox-v2.1.1-csprd01-part24-file.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/cti/cybox/v2.1.1/part24-file/cybox-v2.1.1-part24-file.docx (Authoritative)
http://docs.oasis-open.org/cti/cybox/v2.1.1/part24-file/cybox-v2.1.1-part24-file.html
http://docs.oasis-open.org/cti/cybox/v2.1.1/part24-file/cybox-v2.1.1-part24-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 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-file]

CybOX™ Version 2.1.1. Part 24: 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/part24-file/cybox-v2.1.1-csprd01-part24-file.html. Latest version: http://docs.oasis-open.org/cti/cybox/v2.1.1/part24-file/cybox-v2.1.1-part24-file.html.

Notices

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

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

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

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

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

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

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

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

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

Table of Contents

1 Introduction

[All text is normative unless otherwise labeled.]

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

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

1.1 CybOX™ Specification Documents

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

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

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

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

1.2 Document Conventions

The following conventions are used in this document.

1.2.1 Fonts

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

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

  Examples: Action, Object, Event, Property

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

  Examples: `ActionType`, `cyboxCommon:BaseObjectPropertyType`

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

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

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

1.2.2 UML Package References

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

The package_prefix for the File data model is `FileObj`. Note that in this specification document, we do not explicitly specify the package prefix for any classes that originate from the 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 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 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 FileObjectType Class

The `FileObjectType` class is intended to characterize generic files. The UML diagram corresponding to the `FileObjectType` class is shown in **Figure 3-1**.

Figure 3-1. UML diagram of the FileObjectType class

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

Table 3-1. Properties of the FileObjectType class

Name	Type	Multiplicity	Description
is_packed	basicDataTypes:Boolean	0..1	The is_packed property is used to indicate whether the file is packed or not.
is_masqueraded	basicDataTypes:Boolean	0..1	The is_masqueraded property specifies whether the file is masqueraded as another type of file; e.g., a PDF file that has had its extension changed to TXT to masquerade itself as a text file.
File_Name	cyboxCommon: StringObjectPropertyType	0..1	The File_Name property specifies the base name of the file (including an extension, if present).
File_Path	FilePathType	0..1	The File_Path property specifies the relative or fully-qualified path to the file, not including the path to the device where the file system containing the file resides. Whether the path is relative or fully-qualified can be specified via the 'fully_qualified' property. The File_Path field may include the name of the file; if so, it must not conflict with the File_Name field. If not, the File_Path field should contain the path of the directory containing the file, and should end with a terminating path separator ("\" or "/").
Device_Path	cyboxCommon: StringObjectPropertyType	0..1	The Device_Path property specifies the path to the physical device where the file system containing the file resides.

Full_Path	cyboxCommon: StringObjectPropertyType	0..1	The Full_Path property specifies the complete path to the file, including the device path. It should contain the contents that would otherwise be in the Device_Path and File_Path properties, and can be used in case the producer is unable or does not wish to separate the Device_Path and File_Path properties. If the Full_Path property is specified along with the File_Path and/or Device_Path properties, it must not conflict with either. The Full_Path property may include the name of the file; if so, it must not conflict with the File_Name property. If not, the File_Path property should contain the path of the directory containing the file, and should end with a terminating path separator ("\" or "/").
File_Extension	cyboxCommon: StringObjectPropertyType	0..1	The File_Extension property specifies the extension of the name of the file. The File_Extension property must not conflict with the ending of the File_Name property. The File_Extension property should not begin with a "." character, but may contain a "." character in the case of a compound file extension, such as "tar.gz".
Size_In_Bytes	cyboxCommon: UnsignedLongObjectPropertyType	0..1	The Size_In_Bytes property specifies the size of the file, in bytes.
Magic_Number	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Magic_Number property specifies the particular magic number (typically a hexadecimal constant used to identify a file format) corresponding to the file, if applicable.
File_Format	cyboxCommon: StringObjectPropertyType	0..1	The File_Format property specifies the particular file format of the file, most typically specified by a tool such as the UNIX file command.

Hashes	cyboxCommon: HashListType	0..1	The Hashes property specifies any hashes of the file.
Digital_Signatures	cyboxCommon: DigitalSignaturesType	0..1	The Digital_Signatures property captures one or more digital signatures for the file.
Modified_Time	cyboxCommon: DateTimeObjectPropertyType	0..1	The Modified_Time property specifies the date/time the file was last modified.
Accessed_Time	cyboxCommon: DateTimeObjectPropertyType	0..1	The Accessed_Time property specifies the date/time the file was last accessed.
Created_Time	cyboxCommon: DateTimeObjectPropertyType	0..1	The Created_Time property specifies the date/time the file was created.
File_Attributes_List	FileAttributeType	0..1	The File_Attributes_List property specifies the particular special attributes set for the file. Since this is a platform-specific Object property, it is defined here as an abstract type and then implemented in any platform specific derived file objects.
Permissions	FilePermissionsType	0..1	The Permissions property specifies that particular permissions that a file may have. Since this is a platform-specific Object property, it is defined here as an abstract type and then implemented in any platform specific derived file objects.
User_Owner	cyboxCommon: StringObjectPropertyType	0..1	The User_Owner property specifies the name of the user that owns the file.
Packer_List	PackerListType	0..1	The Packer_List property specifies any packers that the file may be packed with. The term 'packer' here refers

			things like archivers and installers.
Peak_Entropy	cyboxCommon: DoubleObjectPropertyType	0..1	The Peak_Entropy property specifies the calculated peak entropy of the file.
Sym_Links	SymLinksListType	0..1	The Sym_Links property specifies any symbolic links that may exist for the file.
Byte_Runs	cyboxCommon:ByteRunsType	0..1	The Byte_Runs property contains a list of byte runs from the raw file or its storage medium.
Extracted_Features	cyboxCommon: ExtractedFeaturesType	0..1	The Extracted_Features property specifies a description of features extracted from this file.
Encryption_Algorithm	cyboxCommon:CipherType	0..1	The Encryption_Algorithm property specifies the algorithm used to encrypt the file.
Decryption_Key	cyboxCommon: StringObjectPropertyType	0..1	The Decryption_Key property specifies the key used to decrypt the file.
Compression_Method	cyboxCommon: StringObjectPropertyType	0..1	The Compression_Method property specifies the method used to compress the file.
Compression_Version	cyboxCommon: StringObjectPropertyType	0..1	The Compression_Version property specifies the version of the compression method used to compress the file.
Compression_Comment	cyboxCommon: StringObjectPropertyType	0..1	The Compression_Comment property specifies the comment string associated with the compressed file.

3.2 FilePathType Class

The `FilePathType` class specifies the path to the file, not including the device. Whether the path is relative or fully-qualified can be specified via the '`fully_qualified`' property.

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

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

Name	Type	Multiplicity	Description
fully_qualified	`basicDataTypes:Boolean`	0..1	The `fully_qualified` property specifies whether the path is fully qualified.

3.3 FileAttributeType Class

The `FileAttributeType` class specifies attribute(s) of a file. Since this is a platform-specific Object property, it is defined here as an abstract type.

3.4 FilePermissionsType Class

The `FilePermissionsType` class specifies a permission of a file. Since this is a platform-specific Object property, it is defined here as an abstract type and then implemented in any platform specific derived file objects.

3.5 PackerListType Class

The `PackerListType` class specifies a list of file packers.

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

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

Name	Type	Multiplicity	Description
Packer	`PackerType`	1..*	The Packer property specifies a single file packer.

3.6 PackerType Class

The `PackerType` class specifies the fields that characterize a particular file packer, such as name and version. The UML diagram corresponding to the `PackerType` class is shown in Figure 3-2.

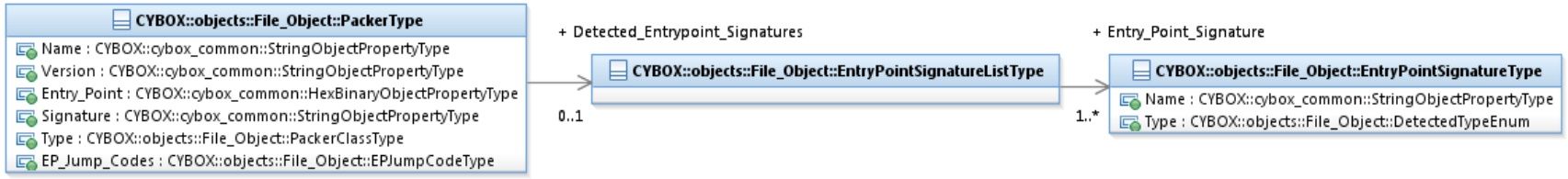

Figure 3-2. UML diagram of the `PackerType` class

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

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

Name	Type	Multiplicity	Description
Name	`cyboxCommon: StringObjectPropertyType`	0..1	The `Name` property specifies the name of the packer.
Version	`cyboxCommon: StringObjectPropertyType`	0..1	The `Version` property specifies the version of the packer.
Entry_Point	`cyboxCommon: HexBinaryObjectPropertyType`	0..1	The `Entry_Point` property specifies the entry point address of the packer, if applicable.
Signature	`cyboxCommon: StringObjectPropertyType`	0..1	The `Signature` property specifies the matching signature detected for the packer, if applicable.

Type	PackerClassType	0..1	The Type property specifies the type of packer being characterized.
Detected_Entrypoint_Signatures	EntryPointSignatureListType	0..1	The Detected_Entrypoint_Signatures property specifies the entrypoint signatures that were detected for the packer.
EP_Jump_Codes	EPJumpCodeType	0..1	The EP_Jump_Codes property characterizes the entry point jump codes of the packer.

3.7 PackerClassType Data Type

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

3.8 EPJumpCodeType Class

The EPJumpCodeType class specifies an entry-point jump code used by a packer.

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

Table 3-5. Properties of the EPJumpCodeType class

Name	Type	Multiplicity	Description
Depth	cyboxCommon: IntegerObjectPropertyType	0..1	The Depth property specified the frequency that a jump instruction is found to be immediately followed by another jump instruction within the PE (Portable Executable) entry point.
Opcodes	cyboxCommon: StringObjectPropertyType	0..1	The Opcodes property specifies the hex value of the bytes located at the jump location for a relative jump identified in the PE (Portable Executable) entry point up to 10 bytes or the end of the RVA (Relative Virtual Address) section.

3.9 EntryPointSignatureListType Class

The `EntryPointSignatureListType` class specifies a list of entry point signatures for a packer.

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

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

Name	Type	Multiplicity	Description
Entry_Point_Signature	`EntryPointSignatureType`	1..*	The `Entry_Point_Signature` property specifies a single property in a list of entry point signatures.

3.10 EntryPointSignatureType Class

The `EntryPointSignatureType` class specifies an entry point signature for a packer.

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

Table 3-7. Properties of the `EntryPointSignatureType` class

Name	Type	Multiplicity	Description
Name	`cyboxCommon:StringObjectPropertyType`	0..1	The `Name` property specifies the signature name.
Type	`DetectedTypeEnum`	0..1	The `Type` property specifies the type of entry point detected (e.g., packer, compiled file).

3.11 SymLinksListType Class

The `SymLinksListType` class specifies a list of symbolic links.

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

Table 3-8. Properties of the SymLinksListType class

Name	Type	Multiplicity	Description
Sym_Link	cyboxCommon: StringObjectPropertyType	1..*	The Sym_Link property specifies a single symbolic link.

3.12 DetectedTypeEnum Enumeration

The literals of the DetectedTypeEnum enumeration are given in **Table 3-9**.

Table 3-9. Literals of the DetectedTypeEnum enumeration

Enumeration Literal	Description
None	Specifies a type other than those listed.
Compiler	Specifies an executable that acts as a compiler.
Packer	Specifies an executable that acts as a packer.
Installer	Specifies an executable that acts as an installer.

3.13 PackerClassEnum Enumeration

The literals of the PackerClassEnum enumeration are given in **Table 3-10**.

Table 3-10. Literals of the PackerClassEnum enumeration

Enumeration Literal	Description
Archiver	Indicates that the packer is an archiver.
Installer	Indicates that the packer is an installer.
Self-Extracting Archiver	Indicates that the packer is a self-extracting archiver.
Crypter	Indicates that the packer is a crypter.
Packer	Indicates a packer.
Protector	Indicates that the packer is a protector.
Bundler	Indicates that the packer is a bundler.
Other	Indicates a different type of packer from the ones listed.

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

Kenichi Terashita

Airbus Group SAS

Joerg Eschweiler

Marcos Orallo

Anomali

Ryan Clough

Wei Huang

Hugh Njemanze

Katie Pelusi

Aaron Shelmire

Jason Trost

Bank of America

Alexander Foley

Center for Internet Security (CIS)

Sarah Kelley

Check Point Software Technologies

Ron Davidson

Cisco Systems

Syam Appala

Ted Bedwell

David McGrew

Pavan Reddy

Omar Santos

Jyoti Verma

Cyber Threat Intelligence Network, Inc. (CTIN)

Doug DePeppe

Jane Ginn

Ben Othman

DHS Office of Cybersecurity and Communications (CS&C)

Richard Struse

Marlon Taylor

EclecticIQ

Marko Dragoljevic

Joep Gommers

Sergey Polzunov

Rutger Prins

Fujitsu Limited
Neil Edwards
Frederick Hirsch
Ryusuke Masuoka
Daisuke Murabayashi

Google Inc.
Mark Risher

Hitachi, Ltd.
Kazuo Noguchi
Akihito Sawada
Masato Terada

iboss, Inc.
Paul Martini

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

Intel Corporation
Tim Casey
Kent Landfield

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

LookingGlass
Allan Thomson
Lee Vorthman

Mitre Corporation
Greg Back
Jonathan Baker
Sean Barnum
Desiree Beck
Nicole Gong
Jasen Jacobsen
Ivan Kirillov
Richard Piazza
Jon Salwen
Charles Schmidt
Andrei Sîrghi
Raymon van der Velde

eSentire, Inc.
Jacob Gajek

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

Fox-IT
Sarah Brown

Georgetown University
Eric Burger

Hewlett Packard Enterprise (HPE)
Tomas Sander

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

IID
Chris Richardson

Integrated Networking Technologies, Inc.
Patrick Maroney

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

Kaiser Permanente
Russell Culpepper
Beth Pumo

Lumeta Corporation
Brandon Hoffman

MTG Management Consultants, LLC.
James Cabral

Emmanuelle Vargas-Gonzalez
John Wunder

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

NEC Corporation
Takahiro Kakumaru

North American Energy Standards Board
David Darnell

Object Management Group
Cory Casanave

Palo Alto Networks
Vishaal Hariprasad

Queralt, Inc.
John Tolbert

Resilient Systems, Inc.
Ted Julian

Securonix
Igor Baikalov

Siemens AG
Bernd Grobauer

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

National Security Agency
Mike Boyle
Jessica Fitzgerald-McKay

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

OASIS
James Bryce Clark
Robin Cover
Chet Ensign

Open Identity Exchange
Don Thibeau

PhishMe Inc.
Josh Larkins

Raytheon Company-SAS
Daniel Wyschogrod

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

Semper Fortis Solutions
Joseph Brand

Splunk Inc.
Cedric LeRoux
Brian Luger
Kathy Wang

TELUS
Greg Reaume
Alan Steer

Threat Intelligence Pty Ltd
Tyron Miller
Andrew van der Stock

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

TruSTAR Technology
Chris Roblee

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

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