



# CybOX™ Version 2.1.1. Part 14: Code 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/part14-code/cybox-v2.1.1-csprd01-part14-code.docx> (Authoritative)

<http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part14-code/cybox-v2.1.1-csprd01-part14-code.html>

<http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part14-code/cybox-v2.1.1-csprd01-part14-code.pdf>

#### Previous version:

N/A

#### Latest version:

<http://docs.oasis-open.org/cti/cybox/v2.1.1/part14-code/cybox-v2.1.1-part14-code.docx> (Authoritative)

<http://docs.oasis-open.org/cti/cybox/v2.1.1/part14-code/cybox-v2.1.1-part14-code.html>

<http://docs.oasis-open.org/cti/cybox/v2.1.1/part14-code/cybox-v2.1.1-part14-code.pdf>

#### Technical Committee:

OASIS Cyber Threat Intelligence (CTI) TC

#### Chair:

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

#### Editors:

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

Trey Darley ([trey@kingfisherops.com](mailto:trey@kingfisherops.com)), Individual member

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

Rich Piazza ([rpiazza@mitre.org](mailto: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 Code 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](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-code]**

*CybOX™ Version 2.1.1. Part 14: Code 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/part14-code/cybox-v2.1.1-csprd01-part14-code.html>. Latest version: <http://docs.oasis-open.org/cti/cybox/v2.1.1/part14-code/cybox-v2.1.1-part14-code.html>.

---

## Notices

Copyright © OASIS Open 2016. All Rights Reserved.

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

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

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

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

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

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

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

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

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

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

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

---

# Table of Contents

1	Introduction.....	6
1.1	CybOX™ Specification Documents.....	6
1.2	Document Conventions.....	6
1.2.1	Fonts.....	6
1.2.2	UML Package References.....	7
1.2.3	UML Diagrams.....	7
1.2.4	Property Table Notation.....	8
1.2.5	Property and Class Descriptions.....	8
1.3	Terminology.....	9
1.4	Normative References.....	9
2	Background Information.....	10
2.1	Cyber Observables.....	10
2.2	Objects.....	10
3	Data Model.....	11
3.1	CodeObjectType Class.....	11
3.2	CodeTypeType Data Type.....	13
3.3	CodeSegmentXORType Data Type.....	13
3.4	CodePurposeType Data Type.....	15
3.5	CodeLanguageType Data Type.....	15
3.6	ProcessorTypeType Data Type.....	15
3.7	TargetedPlatformsType Class.....	15
3.8	CodeTypeEnum Enumeration.....	15
3.9	CodePurposeEnum Enumeration.....	16
3.10	CodeLanguageEnum Enumeration.....	17
3.11	ProcessorTypeEnum Enumeration.....	18
4	Conformance.....	20
	Appendix A. Acknowledgments.....	21
	Appendix B. Revision History.....	25

---

# 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 Code 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 Code Object data model. We present the Code 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 Code data model is `CodeObj`. Note that in this specification document, we do not explicitly specify the package prefix for any classes that originate from the Code 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 Code Object data model (see Section 1.2.2).

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

## 1.2.5 Property and Class Descriptions

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

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

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

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

### 1.3 Terminology

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

### 1.4 Normative References

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

---

## 2 Background Information

In this section, we provide high-level information about the Code 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 CodeObjectType Class

The `CodeObjectType` class is intended to characterize a body of computer code. The UML diagram corresponding to the `CodeObjectType` class is shown in [Figure 3-1](#).

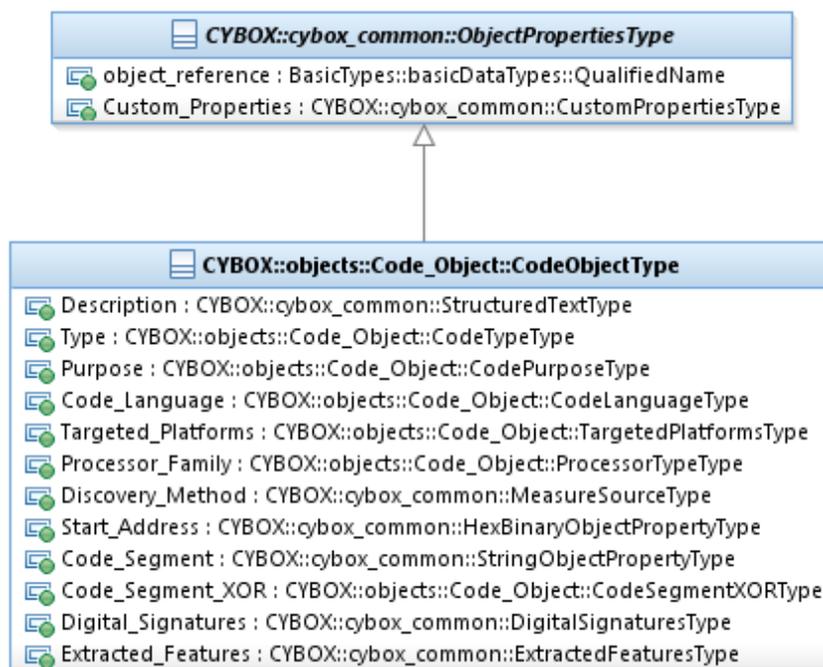


Figure 3-1. UML diagram of the `CodeObjectType` class

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

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

Name	Type	Multiplicity	Description
<b>Description</b>	cyboxCommon: StructuredTextType	0..1	The <code>Description</code> property captures a technical description of the code. Any length is permitted. Optional formatting is supported via the <code>structuring_format</code> property of the <code>StructuredTextType</code> class.
<b>Type</b>	CodeTypeType	0..1	The <code>Type</code> property specifies the type of code being characterized.
<b>Purpose</b>	CodePurposeType	0..1	The <code>Purpose</code> property specifies the purpose or flavor of code being characterized.
<b>Code_Language</b>	CodeLanguageType	0..1	The <code>Code_Language</code> property refers to the code language used in the code characterized in this field.
<b>Targeted_Platforms</b>	TargetedPlatformsType	0..1	The <code>Targeted_Platforms</code> property specifies a list of platforms that this code is targeted for.
<b>Processor_Family</b>	ProcessorTypeType	0..*	The <code>Processor_Family</code> property specifies the class of processor that the code snippet is targeting. This field may be specified multiple times for code snippets that are applicable across multiple processor families.
<b>Discovery_Method</b>	cyboxCommon: MeasureSourceType	0..1	The <code>Discovery_Method</code> property is intended to characterize the method and/or tool used to discover the code.
<b>Start_Address</b>	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Start_Address</code> property can be used to reference the start address of the code, if it was discovered inside a binary.

<b>Code_Segment</b>	cyboxCommon: StringObjectPropertyType	0..1	The <code>Code_Segment</code> property encompasses any arbitrary code segment in unencoded (plaintext or binary) format.
<b>Code_Segment_XOR</b>	CodeSegmentXORType	0..1	The <code>Code_Segment_XOR</code> property encompasses any arbitrary code segment. Its contents should contain the actual code segment XORed with the pattern defined in the <code>xorpattern</code> property. This is so that the code contained in the pattern does not trigger IDS, AV, or other signature-based scanners.
<b>Digital_Signatures</b>	cyboxCommon: DigitalSignaturesType	0..1	The <code>Digital_Signatures</code> property captures one or more digital signatures for the code.
<b>Extracted_Features</b>	cyboxCommon: ExtractedFeaturesType	0..1	The <code>Extracted_Features</code> property specifies a description of features extracted from this code segment.

### 3.2 CodeTypeType Data Type

The `CodeTypeType` data type specifies the type of code. Its core value SHOULD be a literal found in the `CodeTypeEnum` enumeration. It extends the `BaseObjectPropertyType` data type, in order to permit complex (i.e., regular-expression based) specifications.

### 3.3 CodeSegmentXORType Data Type

Used to encapsulate a segment of code that has been XORed with a pattern in order to avoid tripping anti-virus detection. The UML diagram corresponding to the `CodeSegmentXORType` class is shown in [Figure 3-2](#).

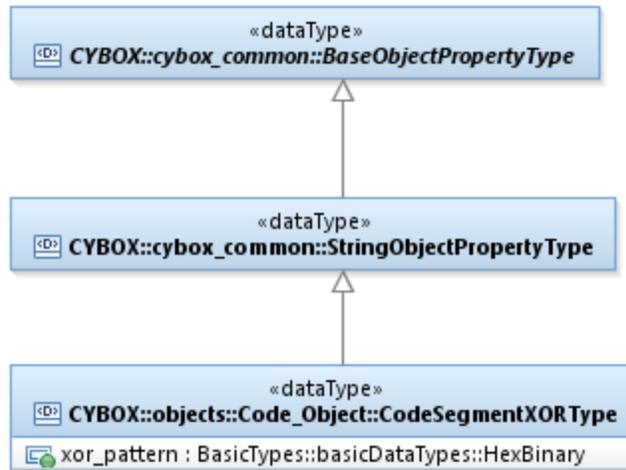


Figure 3-2. UML diagram of the CodeSegmentXORType data type

The property table of the CodeSegmentXORType class is given in [Table 3-2](#).

Table 3-2. Properties of the CodeSegmentXORType class

Name	Type	Multiplicity	Description
<b>xor_pattern</b>	basicDataTypes:HexBinary	0..1	The xor_pattern property contains a 16-hexadecimal-character hex string, which represents the pattern that the Code_Segment_XOR field should be XORed with in order to recover the actual code. The default value is 55AA55AA55AA55BB, as specified by IETF RFC 5901.

### 3.4 CodePurposeType Data Type

The `CodePurposeType` data type specifies the intended purposes of the code. Its core value SHOULD be a literal found in the `CodePurposeEnum` enumeration. It extends the `BaseObjectPropertyType` data type, in order to permit complex (i.e., regular-expression based) specifications.

### 3.5 CodeLanguageType Data Type

The `CodeLanguageType` data type specifies the language of code. Its core value SHOULD be a literal from the `CodeLanguageEnum` enumeration. It extends the `BaseObjectPropertyType` data type, in order to permit complex (i.e., regular-expression based) specifications.

### 3.6 ProcessorTypeType Data Type

The `ProcessorTypeType` data type specifies the relevant processor families. Its core value SHOULD be a literal from the `ProcessorTypeEnum` enumeration. It extends the `BaseObjectPropertyType` data type, in order to permit complex (i.e., regular-expression based) specifications.

### 3.7 TargetedPlatformsType Class

The `TargetedPlatformsType` class contains a list of the targeted platforms.

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

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

Name	Type	Multiplicity	Description
<b>Targeted_Platform</b>	<code>cyboxCommon:PlatformSpecificationType</code>	1..*	The <code>Targeted_Platform</code> property specifies a particular platform that this code is targeted for.

### 3.8 CodeTypeEnum Enumeration

The literals of the `CodeTypeEnum` enumeration are given in [Table 3-4](#).

Table 3-4. Literals of the `CodeTypeEnum` enumeration

Enumeration Literal	Description
<b>Source_Code</b>	The code represented is in the form of source code.
<b>Byte_Code</b>	The code represented is in the form of byte code.
<b>Binary_Code</b>	The code represented is in the form of binary code.

### 3.9 CodePurposeEnum Enumeration

The literals of the `CodePurposeEnum` enumeration are given in [Table 3-5](#).

Table 3-5. Literals of the `CodePurposeEnum` enumeration

Enumeration Literal	Description
<b>Application_Code</b>	The code represented is intended as application code.
<b>Library_Code</b>	The code represented is intended as library code.
<b>Shellcode</b>	The code represented is intended as shellcode.
<b>Exploit_Code</b>	The code represented is intended as exploit code.
<b>Unknown</b>	The code represented is intended for unknown purposes.
<b>Other</b>	The code represented is intended for a purpose other than those listed in this enumeration.

### 3.10 CodeLanguageEnum Enumeration

The literals of the CodeLanguageEnum enumeration are given in [Table 3-6](#).

Table 3-6. Literals of the CodeLanguageEnum enumeration

Enumeration Literal	Description
<b>C</b>	Indicates the code is written in the C programming language.
<b>C++</b>	Indicates the code is written in the C++ programming language.
<b>C#</b>	Indicates the code is written in the C# programming language.
<b>Java</b>	Indicates the code is written in the Java programming language.
<b>JSP</b>	Indicates the code is written in the JSP (Java Server Pages) language.
<b>Javascript</b>	Indicates the code is written in the Javascript programming language.
<b>ASP.NET</b>	Indicates the code is written in the ASP.NET programming language.
<b>SQL</b>	Indicates the code is written in SQL (Standard Query Language).
<b>Python</b>	Indicates the code is written in the Python programming language.
<b>Perl</b>	Indicates the code is written in the Perl programming language.
<b>PHP</b>	Indicates the code is written in the PHP programming language.
<b>SOAP</b>	Indicates the code is written as a SOAP message.

<b>Ruby</b>	Indicates the code is written in the Ruby programming language.
<b>Shell</b>	Indicates the code is written as a Shell script.
<b>PseudoCode</b>	Indicates the code is written as pseudo code.
<b>.NET</b>	Indicates the code utilizes the .NET framework.
<b>Assembly</b>	Indicates the code is written in an assembly language.
<b>XML</b>	Indicates the code is written in XML (eXtensible Markup Language).
<b>HTML</b>	Indicates the code is written in HTML (HyperText Markup Language).
<b>Other</b>	Indicates the code is written in a language not found in this enumeration.

### 3.11 ProcessorTypeEnum Enumeration

The literals of the `ProcessorTypeEnum` enumeration are given in [Table 3-7](#).

Table 3-7. Literals of the `ProcessorTypeEnum` enumeration

<b>Enumeration Literal</b>	<b>Description</b>
<b>x86-32</b>	Indicates an x86 32bit processor.
<b>x86-64</b>	Indicates an x86 64bit processor.
<b>IA-64</b>	Indicates an IA (Intel Itanium) 64bit processor.

<b>PowerPC</b>	Indicates a PowerPC processor.
<b>ARM</b>	Indicates an ARM processor.
<b>Alpha</b>	Indicates an Alpha processor.
<b>SPARC</b>	Indicates a SPARC processor.
<b>z/Architecture</b>	Indicates a z/Architecture (IBM) processor.
<b>eSi-RISC</b>	Indicates an eSi-RISC processor.
<b>MIPS</b>	Indicates a MIPS processor.
<b>Motorola 68k</b>	Indicates a Motorola 68k processor.
<b>Other</b>	Indicates a processor outside of this enumeration.

---

## 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

### **Eclectiq**

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 Sirghi  
Raymon van der Velde

**eSentire, Inc.**

Jacob Gajek

**FireEye, Inc.**

Phillip Boles  
Pavan Gorakav  
Anuj Kumar  
Shyamal Pandya  
Paul Patrick  
Scott Shreve

**Fox-IT**

Sarah Brown

**Georgetown University**

Eric Burger

**Hewlett Packard Enterprise (HPE)**

Tomas Sander

**IBM**

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

**IID**

Chris Richardson

**Integrated Networking Technologies, Inc.**

Patrick Maroney

**Johns Hopkins University Applied Physics Laboratory**

Karin Marr  
Julie Modlin  
Mark Moss  
Pamela Smith

**Kaiser Permanente**

Russell Culpepper  
Beth Pumo

**Lumeta Corporation**

Brandon Hoffman

**MTG Management Consultants, LLC.**

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