CybOX™ Version 2.1.1. Part 02: Common

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/part02-common/cybox-v2.1.1-csprd01-part02-common.docx (Authoritative)
http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part02-common/cybox-v2.1.1-csprd01-part02-common.html
http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part02-common/cybox-v2.1.1-csprd01-part02-common.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/cti/cybox/v2.1.1/part02-common/cybox-v2.1.1-part02-common.docx (Authoritative)
http://docs.oasis-open.org/cti/cybox/v2.1.1/part02-common/cybox-v2.1.1-part02-common.html
http://docs.oasis-open.org/cti/cybox/v2.1.1/part02-common/cybox-v2.1.1-part02-common.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 Common data model, which is one of the fundamental 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-common]

CybOX™ Version 2.1.1. Part 02: Common. 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/part02-common/cybox-v2.1.1-csprd01-part02-common.html. Latest version: http://docs.oasis-open.org/cti/cybox/v2.1.1/part02-common/cybox-v2.1.1-part02-common.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 Common Version 2.1.1 data model, which is one of two fundamental data models for CybOX content.

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 Sections **1.4**. In Section **2**, we give background information necessary to fully understand the Core data model. We present the Core data model specification details in Section **3** and conformance information in Section **3.6.16**.

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 externally defined data models, 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. *CybOX Version 2.1.1 Part 1: Overview* contains the full list of CybOX packages, along with the associated prefix notations, descriptions, and examples.

Note that in *this* specification document, we do not explicitly specify the package prefix for any classes that originate from the Common data model.

1.2.3 UML Diagrams

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

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

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

1.2.3.1 Class Properties

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

[RFC3986] **Berners-Lee, T., Fielding, R. and Masinter, L., “Uniform Resource Identifier (URI): Generic Syntax,” STD 66, RFC 3986, January 2005. Available:** https://www.ietf.org/rfc/rfc3986.txt**.**

[RFC2045] Freed, N., Borenstein, N., “Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies”, RCF 2045, November 1996. Available: https://www.ietf.org/rfc/rfc2045.txt

[ISO8601] Date and time format – ISO 8601 (n.d.). International Organization for Standardization (ISO). [Online]. Available: http://www.iso.org/iso/home/standards/iso8601.htm. Accessed: December 15, 2015.

[IEEE 754-1985] IEEE. *IEEE Standard for Binary Floating-Point Arithmetic.* Available: http://standards.ieee.org/reading/ieee/std_public/description/busarch/754-1985_desc.html

[CPE] Common Platform Enumeration (CPE). (2014, Nov. 28). The MITRE Corporation. [Online]. Available: http://cpe.mitre.org.

2 Background Information

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

The CybOX Common data model defines object classes that are shared across the various CybOX data models. There is a wide variety of class types, so to make the specification document content easier to reference and understand, we have organized the data model content into eight categories:

- **Object Property Classes and Data Types** – capture a property of a CybOX object, with support for metadata and patterning.
- **General Shared Classes** – serve a variety of purposes and shared across the CybOX data models.
- **General Classes and Data Types** – support classes and data types defined in the CybOX data models.
- **Vocabulary Data Types** – provide a content creator with choices for defining content.
- **Enumerations** – support the classes defined in the CybOX data models.

Each category is contained in a separate subsection in Section **3**.

3 CybOX Common Data Model

The CybOX Core data model defines a variety of classes and data types. For discussion purposes, we have separated the classes into five categories (Sections **3.1** through **3.5**), and within each category, we primarily define the classes and data types in alphabetical order below, except for the cases when a class or data type is uniquely used in the main class or data type. We list enumerations in Section **3.6**.

3.1 ObjectPropertiesType Class

The `ObjectPropertiesType` class is an abstract class within the CybOX schema enabling the inclusion of contextually varying object properties descriptions. This abstract type is leveraged as the extension base for all predefined CybOX object properties schemas. Through this extension mechanism, any object instance data based on an object properties schema extended from `ObjectPropertiesType` (e.g. `File_Object`, `Address_Object`, etc.) can be directly integrated into any instance document where a property is defined as `ObjectPropertiesType`. For flexibility and extensibility purposes any user of CybOX can specify their own externally defined object properties schemas (outside of or derived from the set of predefined objects) extended from `ObjectPropertiesType` class and utilize them as part of their CybOX content.

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

Name	Type	Multiplicity	Description
object_reference	`basicDataTypes:QualifiedName`	0..1	The `object_reference` property specifies a unique ID reference to an Object defined elsewhere. This property allows for the re-use of the defined Properties of one Object within another, without the need to embed the full Object in the location from which it is being referenced. Thus, this ID reference is intended to resolve to the properties of the Object that it points to.
Custom_Properties	`CustomPropertiesType`	0..1	The `Custom_Properties` property characterizes a set of custom Object Properties that may not be defined in existing properties.

3.2 Object Properties Data Types

Objects in CybOX can have properties of various different data types. This section describes the underlying model for all Object properties, such that they support metadata and pattern matching.

3.2.1 BaseObjectPropertyType Data Type

The `BaseObjectPropertyType` data type represents a common typing foundation for the specification of a single Object Property. The `BaseObjectPropertyType` data type is extended from the `BaseObjectPropertyGroup` data type, which is an abstract data type that contains the auxiliary metadata properties associated with the main property value being represented. In addition, the `BaseObjectPropertyType` data type also inherits from `PatternFieldGroup` data type. This data type incorporates pattern matching capabilities to all specializations of `BaseObjectPropertyType`.

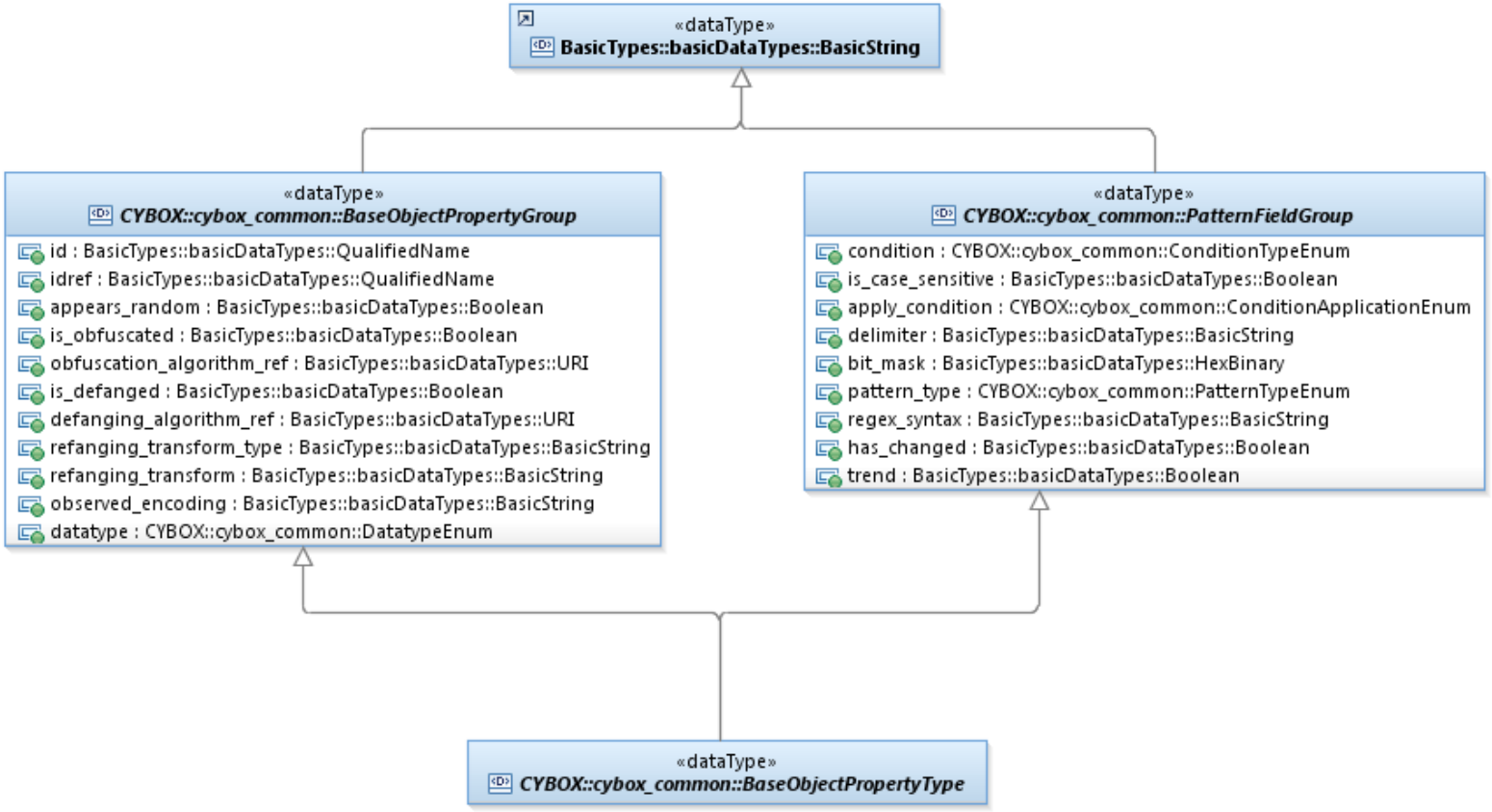

Figure 3-1. UML diagram for `BaseObjectPropertyType` data type

Object Properties that use the `BaseObjectPropertyType` data type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '*##comma##*' (no quotes) but can be overridden through use of the `delimiter` property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

3.2.1.1 BaseObjectPropertyGroup Data Type

The `BaseObjectPropertyGroup` is an abstract data type that aggregates a set of metadata properties associated with an Object instance.

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

Name	Type	Multiplicity	Description
id	`basicDataTypes: QualifiedName`	0..1	The `id` property specifies a globally unique identifier for the Object Property.
idref	`basicDataTypes: QualifiedName`	0..1	The `idref` property specifies an identifier reference to an Object Property instance specified elsewhere. When the `idref` property is used, no other property should be specified.
datatype	`DatatypeEnum`	0..1	The `datatype` property specifies the expected type for the value of the specified property. Data Types that are specializations of this class will usually redefine this property to specify one of the enumeration literals as the default, corresponding to class being modeled.
appears_random	`basicDataTypes: Boolean`	0..1	The `appears_random` property specifies whether the associated object property value appears to somewhat random in nature. An object property with this property set to TRUE need not provide any further information including a value. If more is known about the particular variation of randomness, a regex value could be provided to outline what is known of the structure.
is_obfuscated	`basicDataTypes: Boolean`	0..1	The `is_obfuscated` property specifies whether the associated Object property has been obfuscated.

obfuscation_algorithm_ref	basicDataTypes:URI	0..1	The obfuscation_algorithm_ref property specifies a reference to a description of the algorithm used to obfuscate this Object property.
is_defanged	basicDataTypes:Boolean	0..1	The is_defanged property specifies whether the associated Object property has been defanged (representation changed to prevent malicious effects of handling/processing).
defanging_algorithm_ref	basicDataTypes:URI	0..1	The defanging_algorithm_ref property specifies a reference to a description of the algorithm used to defang (representation changed to prevent malicious effects of handling/processing) this Object property.
refanging_transform_type	basicDataTypes:BasicString	0..1	The refanging_transform_type property specifies the type (e.g. RegEx) of refanging transform specified in the optional accompanying refanging_transform property.
refanging_transform	basicDataTypes:BasicString	0..1	The refanging_transform property captures an automated transform that can be applied to the Object property content in order to refang it to its original format.
observed_encoding	basicDataTypes:BasicString	0..1	The observed_encoding property captures the encoding of the string when it is/was observed. This may be different from the encoding used to represent the string within this property. It is strongly recommended that character set names should be taken from the IANA character set registry (https://www.iana.org/assignments/character-sets/character-sets.xhtml). This property is intended to be applicable only to Object properties which contain string values.

3.2.1.2 PatternFieldGroup Data Type

The PatternFieldGroup is an abstract data type that aggregates a set of properties for the application of patterns.

Table 3-3. Properties of the PatternFieldGroup class

Name	Type	Multiplicity	Description
condition	ConditionTypeEnum	0..1	The condition property specifies the relevant condition to apply to the value.
is_case_sensitive	basicDataTypes:Boolean	0..1	The is_case_sensitive property specifies the case-sensitivity of a pattern which uses an Equals, DoesNotEqual, Contains, DoesNotContain, StartsWith, EndsWith, or FitsPattern condition. The default value for this property is TRUE which indicates that pattern evaluations are to be considered case-sensitive.
apply_condition	ConditionApplicationEnum	0..1	The apply_condition property specifies how a condition should be applied when the Object property body contains a list of values. (Its value is meaningless if the Object property value contains only a single value as all possible values for this property would have the same behavior.) If this property is set to ANY, then a pattern is considered to be matched if the provided condition successfully evaluates for any of the values in the Object property body. If the property is set to ALL, then the pattern only matches if the provided condition successfully evaluates for every value in the Object property body.
delimiter	basicDataTypes: BasicString	0..1	The delimiter property captures the delimiter used when defining lists of values. The default value is "##comma##".
bit_mask	basicDataTypes:HexBinary	0..1	The bit_mask property specifies a bit_mask in conjunction with one of the defined binary conditions (bitwiseAnd, bitwiseOr, and bitwiseXor). This bitmask is then uses as one operand in the indicated bitwise computation.
pattern_type	PatternTypeEnum	0..1	The pattern_type property specifies the type of pattern used if one is specified for the Object property value. This is applicable only if the Condition property is set to 'FitsPattern'.

regex_syntax	`basicDataTypes:BasicString`	0..1	The `regex_syntax` property captures the syntax format used for a regular expression, if one is specified for the property value. This is applicable only if the Condition property is set to 'FitsPattern'. Setting this property with an empty value (e.g., "") or omitting it entirely notifies CybOX consumers and pattern evaluators that the corresponding regular expression utilizes capabilities, character classes, escapes, and other lexical tokens defined by the CybOX Language Specification. Setting this attribute with a non-empty value notifies CybOX consumers and pattern evaluators that the corresponding regular expression utilizes capabilities not defined by the CybOX Language Specification. The regular expression must be evaluated through a compatible regular expression engine in this case.
has_changed	`basicDataTypes:Boolean`	0..1	The `has_changed` property specifies whether a targeted observation pattern of the associated Object property value has changed. This property would be leveraged within a pattern observable triggering on whether the value of a single Object property value has changed.
trend	`basicDataTypes:Boolean`	0..1	The `trend` property specifies whether a targeted observation pattern of the nature of any trend in the associated Object property value. This property would be leveraged within a pattern observable triggering on the matching of a specified trend in the value of a single specified Object property.

3.2.2 AnyURIObjectPropertyType Data Type

The `AnyURIObjectPropertyType` data type represents the specification of a single Object property whose core value is a `BasicString` such that it adheres to the standard defined in **[RFC3986]**. It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain a URI and enables the use of relevant metadata for the property. This class redefines the property `datatype` to have a default value of the *URI* literal from the `DatatypeEnum` enumeration.

3.2.3 Base64BinaryObjectPropertyType Data Type

The `Base64BinaryObjectPropertyType` data type represents the specification of a single Object property whose core value is a `BasicString` such that it adheres to the standard defined in **[RFC2045]**. It extends the base data type `BaseObjectPropertyType`. This class will be assigned to any property of a CybOX object that should contain Base64Binary content and enables the use of relevant metadata for the property. This class redefines the property `datatype` to have a default value of the *base64Binary* literal from the `DatatypeEnum` enumeration.

3.2.4 DateObjectPropertyRestrictionType Data Type

The `DateObjectPropertyRestrictionType` data type is a type is an intermediate type to allow for the addition of the precision property to `DateObjectPropertyType`. It extends the base data type `BaseObjectPropertyType`. This class redefines the property `datatype` to have a default value of the *date* literal from the `DatatypeEnum` enumeration. It should not be used directly.

3.2.4.1 DateObjectPropertyType Data Type

The `DateObjectPropertyType` data type (extended from the `DateObjectPropertyRestrictionType` data type) represents the specification of a single Object property whose core value is a `BasicString` such that it adheres to the standard defined in **[ISO8601]** for expressing a date. This type will be assigned to any property of a CybOX object that should contain Date content and enables the use of relevant metadata for the property.

For properties of this type using CybOX patterning, it is strongly suggested that the `condition` (pattern type) is limited to one of *Equals*, *DoesNotEqual*, *GreaterThan*, *LessThan*, *GreaterThanOrEqual*, *LessThanOrEqual*, *ExclusiveBetween*, or *InclusiveBetween*. The use of other conditions may lead to ambiguity or unexpected results. When evaluating data against a pattern, the evaluator should take into account the precision of the property (as given by the precision property) and any timezone information that is available to perform a data-aware comparison. The usage of simple string comparisons is discouraged due to ambiguities in how precision and timezone information is processed.

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

Name	Type	Multiplicity	Description
precision	`DatePrecisionEnum`	0..1	The `precision` property specifies the granularity with which the date should be considered. If omitted, the default is "day", meaning the full property value. Digits in a timestamp that are beyond the specified precision should be zeroed out. When used in conjunction with CybOX patterning, the pattern should only be evaluated against the target up to the given precision.

3.2.5 DateTimeObjectPropertyRestrictionType Data Type

The `DateTimeObjectPropertyRestrictionType` class is data type is an intermediate type to allow for the addition of the precision property to `DateTimeObjectPropertyType`. It extends the base data type `BaseObjectPropertyType`. This class redefines the property `datatype` to have a default value of the *dateTime* literal from the `DatatypeEnum` enumeration. It should not be used directly.

3.2.5.1 DateTimeObjectPropertyType Data Type

The `DateTimeObjectPropertyType` data type (extended from the `DateTimeObjectPropertyRestrictionType` data type) represents the specification of a single Object property whose core value is a `BasicString` such that it adheres to the standard defined in **[ISO8601]** for expressing a date and time. This type will be assigned to any property of a CybOX object that should contain DateTime content and enables the use of relevant metadata for the property. In order to avoid ambiguity, it is strongly suggested that any property using this class SHOULD include a timezone.

For properties of this type using CybOX patterning, it is strongly suggested that the `condition` (pattern type) is limited to one of *Equals*, *DoesNotEqual*, *GreaterThan*, *LessThan*, *GreaterThanOrEqual*, *LessThanOrEqual*, *ExclusiveBetween*, or *InclusiveBetween*. The use of other conditions may lead to ambiguity or unexpected results. When evaluating data against a pattern, the evaluator should take into account the precision of the property (as given by the precision attribute) and any timezone information that is available to perform a data-aware comparison. The usage of simple string comparisons is discouraged due to ambiguities in how precision and timezone information is processed.

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

Name	Type	Multiplicity	Description
Precision	`DateTimePrecisionEnum`	0..1	The `precision` property specifies the granularity with which the time should be considered, as specified by the DateTimePrecisionEnum enumeration (e.g., *hour*, *minute*). If omitted, the default precision is *second*. Digits in a timestamp that are beyond the specified precision should be zeroed out. When used in conjunction with CybOX patterning, the pattern should only be evaluated against the target up to the given precision.

3.2.6 DoubleObjectPropertyType Data Type

The `DoubleObjectPropertyType` data type represents the specification of a single Object property whose core value is a `BasicString` such that it adheres to the standard defined in [IEEE 754-1985]. It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain Double content and enables the use of relevant metadata for the property. This class redefines the property `datatype` to have a default value of the *double* literal from the `DatatypeEnum` enumeration.

3.2.7 DurationObjectPropertyType Data Type

The `DurationObjectPropertyType` data type represents the specification of a single Object property whose core value is a `BasicString` such that it adheres to the standard defined in **[ISO8601]** for expressing date/time duration. It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain duration content and enables the use of relevant metadata for the property. This class redefines the property `datatype` to have a default value of the *duration* literal from the `DatatypeEnum` enumeration.

3.2.8 FloatObjectPropertyType Data Type

The `FloatObjectPropertyType` data type represents the specification of a single Object property whose core value is value is a `BasicString` such that it adheres to the standard defined in [IEEE 754-1985]. It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain content of type Float and enables the use of relevant metadata for the property. This class redefines the property `datatype` to have a default value of the *float* literal from the `DatatypeEnum` enumeration.

3.2.9 HexBinaryObjectPropertyType Data Type

The `HexBinaryObjectPropertyType` data type represents the specification of a single Object property whose core value is value is a `BasicString` such that it adheres to the regular expression [0-9A-Fa-f]*. It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain content of type HexBinary and enables the use of relevant metadata for the property. This class redefines the property `datatype` to have a default value of the *hexBinary* literal from the `DatatypeEnum` enumeration.

3.2.9.1 SimpleHashValueType Data Type

The `SimpleHashValueType` data type is used for characterizing the output of basic cryptographic hash functions outputting a single hexbinary hash value. It extends the `HexBinaryObjectPropertyType` data type.

3.2.9.2 FuzzyHashValueType Data Type

The `FuzzyHashValueType` data type is used for characterizing the output of cryptographic fuzzy hash functions outputting a single complex string based hash value. It extends the `HexBinaryObjectPropertyType` data type.

3.2.10 IntegerObjectPropertyType Data Type

The `IntegerObjectPropertyType` data type represents the specification of a single Object property whose core value is a `BasicString` such that it corresponds to a sequence of decimal digits, with perhaps a leading minus or plus sign (“-“ or “+”). It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain content of type Integer and enables the use of relevant metadata for the property. This data type redefines the property `datatype` to have a default value of the *int* literal from the `DatatypeEnum` enumeration.

3.2.11 LongObjectPropertyType Data Type

The `LongObjectPropertyType` data type represents the specification of a single Object property whose core value is a `BasicString` such that it corresponds to a sequence of decimal digits, but limited to the values -9223372036854775808 through 9223372036854775807, inclusive. A leading minus or plus sign ("-" or "+") is permitted. It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain content of type Long and enables the use of relevant metadata for the property. This data type redefines the property `datatype` to have a default value of the *long* literal from the `DatatypeEnum` enumeration.

3.2.12 NameObjectPropertyType Data Type

The `NameObjectPropertyType` data type represents the specification of a single Object property whose core value is a `BasicString` that corresponds to legal XML 1.0 names. It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain content of type Name and enables the use of relevant metadata for the property. This data type redefines the property `datatype` to have a default value of the *name* literal from the `DatatypeEnum` enumeration.

3.2.13 NonNegativeIntegerObjectPropertyType Data Type

The `NonNegativeIntegerObjectPropertyType` data type represents the specification of a single Object property whose core value is a `BasicString` such that it corresponds to a sequence of decimal digits, which may only be proceeded by the plus sign ("+"). It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain content of type NonNegativeInteger and enables the use of relevant metadata for the property. This data type redefines the property `datatype` to have a default value of the *nonNegativeInteger* literal from the `DatatypeEnum` enumeration.

3.2.14 PositiveIntegerObjectPropertyType Data Type

The `PositiveIntegerObjectPropertyType` data type represents the specification of a single Object property whose core value is a `BasicString` that corresponds to a positive integer. The value 0 is not permitted. It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain content of type PositiveInteger and enables the use of relevant metadata for the property. This data type redefines the property `datatype` to have a default value of the *positiveInteger* literal from the `DatatypeEnum` enumeration.

3.2.15 StringObjectPropertyType Data Type

The `StringObjectPropertyType` data type represents the specification of a single Object property whose core value is a `BasicString`. It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain content of type String and enables the use of relevant metadata for the property. This data type redefines the property `datatype` to have a default value of the *string* literal from the `DatatypeEnum` enumeration.

3.2.15.1 DataSizeType Data Type

The `DataSizeType` data type specifies the size of the data segment. It extends the data type `StringObjectPropertyType`. In addition to representing the size of the data segment as a `BasicString`, the `units` property can be used to specify the units used to express the size.

Table 3-6. Properties of the `DataSizeType` data type

Name	Type	Multiplicity	Description
units	`DataSizeUnitsEnum`	0..1	The `units` property specifies the Units used in the object size element.

3.2.15.2 PlatformIdentifierType Data Type

The `PlatformIdentiferType` data type is used to specify a name for a platform using a particular naming system and also allowing a reference pointing to more information about that naming scheme. For example, one could provide a CPE (Common Platform Enumeration) [CPE] name using the CPE naming format. In this case, the `system` value could be "CPE" while the `system_ref` value could be "http://scap.nist.gov/specifications/cpe/". It extends the data type `StringObjectPropertyType`.

Table 3-7. Properties of the `PlatformIdentifierType` data type

Name	Type	Multiplicity	Description
system	`basicDataTypes: BasicString`	0..1	The `system` property captures the naming system from which the indicated name was drawn.
system-ref	`basicDataTypes:URI`	0..1	The `system-ref` property specifies a reference to information about the naming system from which the indicated name was drawn.

3.2.16 TimeObjectPropertyRestrictionType Data Type

The `TimeObjectPropertyRestrictionType` data type is a type is an intermediate type to allow for the addition of the precision property to `TimeObjectPropertyType`. It extends the base data type `BaseObjectPropertyType`. This data type redefines the property `datatype` to have a default value of the *time* literal from the `DatatypeEnum` enumeration. It should not be used directly.

3.2.16.1 TimeObjectPropertyType Data Type

The `TimeObjectPropertyType` data type (extended from the `TimeObjectPropertyRestrictionType` data type) represents the specification of a single Object property whose core value is a `BasicString` such that it adheres to the standard defined in **[ISO8601]**. This type will be assigned to any property of a CybOX object that should contain content of type Time and enables the use of relevant metadata for the property. In order to avoid ambiguity, it is strongly suggested that any property using this data type SHOULD include a timezone.

For properties of this type using CybOX patterning, it is strongly suggested that the `condition` (pattern type) is limited to one of *Equals*, *DoesNotEqual*, *GreaterThan*, *LessThan*, *GreaterThanOrEqual*, *LessThanOrEqual*, *ExclusiveBetween*, or *InclusiveBetween*. The use of other conditions may lead to ambiguity or unexpected results. When evaluating data against a pattern, the evaluator should take into account the precision of the property (as given by the precision attribute) and any timezone information that is available to perform a data-aware comparison. The usage of simple string comparisons is discouraged due to ambiguities in how precision and timezone information is processed.

Table 3-8. Properties of the `TimeObjectPropertyType` data type

Name	Type	Multiplicity	Description
precision	`TimePrecisionEnum`	0..1	The `precision` property specifies the granularity with which a timestamp should be considered as specified by the `TimePrecisionEnum` enumeration (e.g., *hour*, *minute*). If omitted, the default precision is *second*. Digits in a timestamp that are beyond a specified precision SHOULD be zeroed out. When used in conjunction with CybOX patterning, the pattern should only be evaluated against the target up to the given precision.

3.2.17 UnsignedIntegerObjectPropertyType Data Type

The `UnsignedIntegerObjectPropertyType` data type represents the specification of a single Object property whose core value is a `BasicString` such that it corresponds to a sequence of decimal digits, but limited to the values 0 through 4294967295, inclusive. It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain content of type an unsigned integer and enables the use of relevant metadata for the property. This data type redefines the property `datatype` to have a default value of the *unsignedInt* literal from the `DatatypeEnum` enumeration.

3.2.18 UnsignedLongObjectPropertyType Data Type

The `UnsignedLongObjectPropertyType` data type represents the specification of a single Object property whose core value is a `BasicString` such that it corresponds to a sequence of decimal digits, but limited to the values 0 through 18446744073709551615, inclusive. It extends the base data type `BaseObjectPropertyType`. This type will be assigned to any property of a CybOX object that should contain content of type unsigned long integer

and enables the use of relevant metadata for the property. This data type redefines the property `datatype` to have a default value of the *unsignedLong* literal from the `DatatypeEnum` enumeration.

3.2.19 ObjectPropertyType Data Types Related to Enumerations

The data types described in this section represent the specification of a single Object property whose core value is a `BasicString`, which SHOULD be one of the literals found in the corresponding enumeration; however, any free form text string is permitted.

3.2.19.1 CipherType Data Type

The `CipherType` specifies encryption algorithms. Its core value SHOULD be a literal from the `CipherEnum` enumeration. It extends the `BaseObjectPropertyType` data type, for permitting complex (i.e. regular-expression based) specifications.

3.2.19.2 CompensationModelType Data Type

The `CompensationModelType` data type characterizes the compensation model for a tool. Its core value SHOULD be a literal from the `CompensationModelEnum` enumeration. It extends the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.2.19.3 EndiannessType Data Type

The `EndiannessType` specifies names for byte ordering methods. Its core value SHOULD be a literal from the `EndiannessTypeEnum` enumeration. It extends the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.2.19.4 Layer4ProtocolType Data Type

The `Layer4ProtocolType` data type specifies Layer 4 protocol types. Its core value SHOULD be a literal from the `Layer4ProtocolEnum` enumeration. It extends the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.2.19.5 RegionalRegistryType Data Type

The `RegionalRegistryType` data type specifies a Regional Internet Registry (RIR) for a given WHOIS entry. Its core value SHOULD be a literal from the `RegionalRegistryTypeEnum` enumeration. It extends the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.2.19.6 SIDType Data Type

The `SIDType` data type specifies the Windows Security ID (SID) types. Its core values SHOULD be one of the literals from the `SIDTypeEnum` enumeration. It extends the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.3 General Shared Classes

3.3.1 MeasureSourceType Class

The `MeasureSourceType` class is a type representing a description of a single cyber observation source.

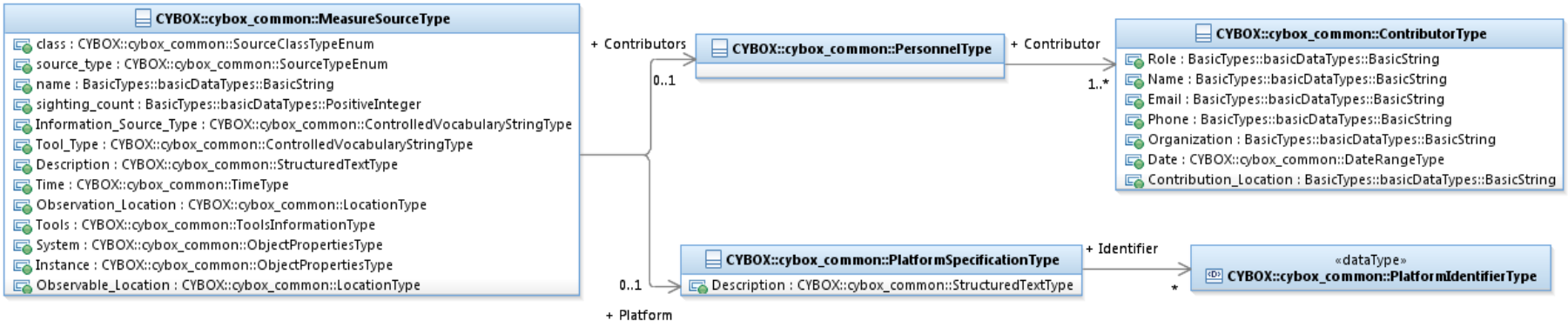

Figure 3-2. UML diagram for the `MeasureSourceType` class

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

Name	Type	Multiplicity	Description
class	`SourceClassTypeEnum`	0..1	The `class` property specifies the identification of the high-level source of this cyber observation source.
source_type	`SourceTypeEnum`	0..1	The `source_type` property specifies the identification of the broad type of this cyber observation source.
name	`basicDataTypes: BasicString`	0..1	The `name` property specifies the assignment of a relevant name to this Discovery Method.
sighting_count	`basicDataTypes: PositiveInteger`	0..1	The `sighting_count` property specifies how many different identical instances of a given Observable may have

			been seen/sighted by the observation source.
Information_Source_Type	VocabularyStringType	0..1	The Information_Source_Type property specifies the type of information source. Examples of potential types *are application logs, help desk* and *TPM* (these specific values are only provided to help explain the property: they are neither recommended values nor necessarily part of any existing vocabulary). The content creator may choose any arbitrary value or may constrain the set of possible values by referencing an externally-defined vocabulary or leveraging a formally defined vocabulary extending from the cyboxCommon:ControlledVocabularyStringType class. The CybOX default vocabulary class for use in the property is *'InformationSourceTypeVocab-1.0*'.
Tool_Type	VocabularyStringType	0..1	The Tool_Type property specifies the type of the tool. Examples of potential types are *NIDS*, *asset scanner*, and *malware analysis* (these specific values are only provided to help explain the property: they are neither recommended values nor necessarily part of any existing vocabulary). The content creator may choose any arbitrary value or may constrain the set of possible values by referencing an externally-defined vocabulary or leveraging a formally defined vocabulary extending from the cyboxCommon:ControlledVocabularyStringType class. The CybOX default vocabulary class for use in the property is *'ToolTypeVocab-1.1'*.
Description	StructuredTextType	0..1	The Description property captures a technical description of the measure source. Any length is permitted. Optional formatting is supported via the structuring_format property of the StructuredTextType data type.
Contributors	PersonnelType	0..1	The Contributors property characterizes the description of the individual contributors involved in this

			cyber observation source.
Time	TimeType	0..1	The Time property specifies the various time-related properties for this cyber observation source instance.
Observation_Location	LocationType	0..1	The Observation_Location property specifies a relevant physical location for the associated Observation. The underlying abstract class MUST be extended. The default and strongly RECOMMENDED subclass is CIQAddressInstanceType, as defined in *CybOX Version 2.1.1 Part 4: Default Extensions*.
Tools	ToolsInformationType	0..1	The Tools property characterizes the tools utilized for this cyber observation source.
Platform	PlatformSpecificationType	0..1	The Platform property characterizes a formal, standardized specification of the platform for this cyber observation source.
System	ObjectPropertiesType	0..1	The System property characterizes the system on which the mechanism of cyber observation executed. System SHOULD be an object of type SystemObj:SystemObjectType.
Instance	ObjectPropertiesType	0..1	The Instance property characterizes the process instance in which the mechanism of cyber observation executed. Instance SHOULD be of type ProcessObj:ProcessObjectType.
Observable_Location	LocationType	0..1	The Observable_Location property specifies a relevant physical location for the associated Observable. The underlying abstract class MUST be extended. The default and strongly RECOMMENDED subclass is

			`CIQAddressInstanceType`, as defined in the *CybOX Version 2.1.1 Part 4: Default Extensions.*

3.3.2 Build-Related Classes

3.3.2.1 BuildInformationType Class

The `BuildInformationType` class contains information describing how this tool was built.

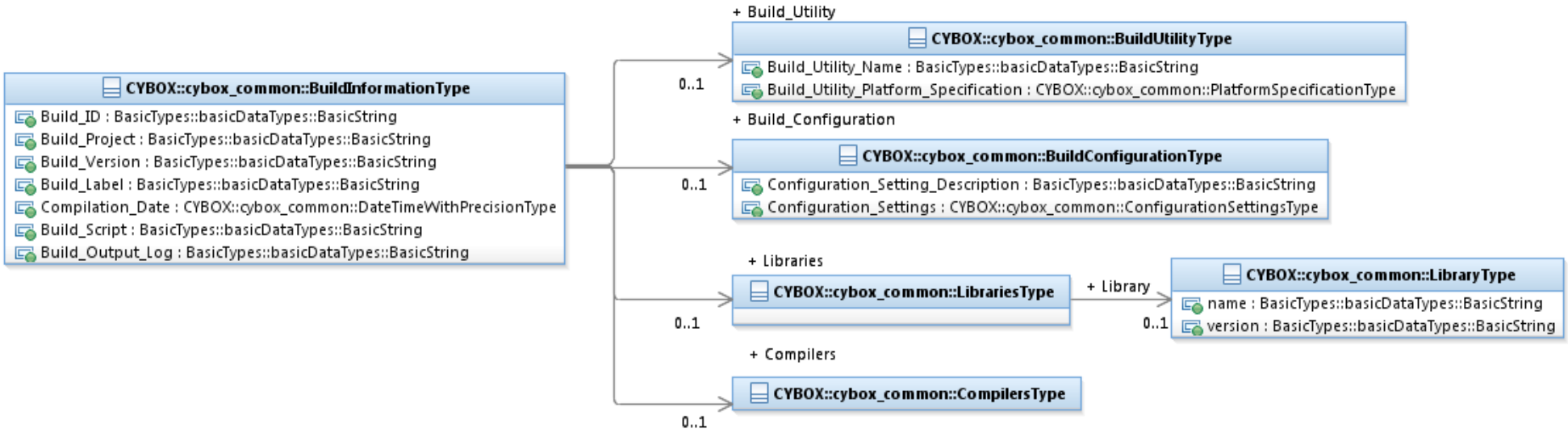

Figure 3-3. UML diagram for the `BuildInformationType` class

Table 3-10. Properties of the `BuildInformationType` class

Name	Type	Multiplicity	Description

Build_ID	basicDataTypes: BasicString	0..1	The Build_ID property captures an externally defined unique identifier of this build of this application instance.
Build_Project	basicDataTypes: BasicString	0..1	The Build_Project property captures the project name of this build of this application instance.
Build_Utility	BuildUtilityType	0..1	The Build_Utility property characterizes the utility used to build this application.
Build_Version	basicDataTypes: BasicString	0..1	The Build_Version property captures the appropriate version descriptor of this build of this application instance.
Build_Label	basicDataTypes: BasicString	0..1	The Build_Label property captures any relevant label for this build of this application instance.
Compilers	CompilersType	0..1	The Compilers property characterizes compilers utilized during this build of this application.
Compilation_Date	DateTimeWithPrecisionType	0..1	The Completion_Date property specifies the compilation date for the build of the tool. In order to avoid ambiguity, it is strongly suggest that all timestamps in this field include a specification of the timezone if it is known.
Build_Configuration	BuildConfigurationType	0..1	The Build_Configuration property characterizes how the build utility was configured for this build of this application.
Build_Script	basicDataTypes: BasicString	0..1	The Build_Script property captures the actual build script for this build of this application instance.
Libraries	LibrariesType	0..1	The Libraries property characterizes the libraries incorporated into the build of the tool.

Build_Output_Log	basicDataTypes: BasicString	0..1	The Build_Output_Log property captures the output log of the build process.

3.3.2.2 BuildUtilityType Class

The BuildUtilityType class contains information identifying the utility used to build this application.

Table 3-11. Properties of the BuildUtilityType class

Name	Type	Multiplicity	Description
Build_Utility_Name	basicDataTypes: BasicString	1	The Build_Utility_Name property captures the informally defined name of the utility used to build this application instance.
Build_Utility_Platform_Specification	PlatformSpecificationType	1	The Build_Utility_Platform_Specification property characterizes the build utility used to build this application.

3.3.2.3 BuildConfigurationType Class

The BuildConfigurationType class describes how the build utility was configured for this build of this application.

Table 3-12. Properties of the BuildConfigurationType class

Name	Type	Multiplicity	Description
Configuration_Setting_Description	basicDataTypes: BasicString	0..1	The Configuration_Setting_Description property captures the configuration settings for this build of this application instance.

Configuration_Settings	ConfigurationSettingsType	1	The Configuration_Settings property characterizes the configuration settings for this build of this application instance.

3.3.2.4 ExecutionEnvironmentType Class

The ExecutionEnvironmentType class contains information describing the execution environment of the tool.

Table 3-13. Properties of the ExecutionEnvironmentType class

Name	Type	Multiplicity	Description
System	ObjectPropertiesType	0..1	The System property characterizes the system on which the tool was executed. This property should be of class SystemObj:SystemObjectType.
User_Account_Info	ObjectPropertiesType	0..1	The User_Account_Info property characterizes the user account that executed the tool. This property should be of class UserAccountObj:UserAccountObjectType.
Command_Line	basicDataTypes: BasicString	0..1	The Command_Line property captures the command line string used to run the tool.
Start_Time	DateTimeWithPrecisionType	0..1	The Start_Time property specifies when the tool was run. In order to avoid ambiguity, it is strongly suggest that all timestamps in this field include a specification of the timezone if it is known.

3.3.3 ByteRunsType Class

The ByteRunsType class is used for representing a list of byte runs from within a raw object.

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

Name	Type	Multiplicity	Description
Byte_Run	`ByteRunType`	1..*	The `Byte_Run` property characterizes a single byte run from the raw object.

3.3.3.1 ByteRunType Class

The `ByteRunType` class is used for representing a single byte run from within a raw object.

Table 3-15. Properties of the `ByteRunType` class

Name	Type	Multiplicity	Description
Offset	`IntegerObjectPropertyType`	0..1	The `Offset` property characterizes the offset of the beginning of the byte run as measured from the beginning of the object.
Byte_Order	`EndiannessType`	0..1	The `Byte_Order` property characterizes the endianness of the unpacked (e.g., unencoded, unencrypted, etc.) data contained within the `Byte_Run_Data` property.
File_System_Offset	`IntegerObjectPropertyType`	0..1	The `File_System_Offset` property characterizes the offset of the beginning of the byte run as measured from the beginning of the relevant file system. It is relevant only for byte runs of files in forensic analysis.
Image_Offset	`IntegerObjectPropertyType`	0..1	The `Image_Offset` property characterizes the offset of the beginning of the byte run as measured from the beginning of the relevant forensic image. It is provided for forensic analysis purposes.
Length	`IntegerObjectPropertyType`	0..1	The `Length` property characterizes the number of bytes in the

			byte run.
Hashes	`HashListType`	0..1	The `Hashes` property specifies computed hash values for this the data in this byte run.
Byte_Run_Data	`HexBinaryObjectPropertyType`	0..1	The `Byte_Run_Data` property captures a raw dump of the byte run data.

3.3.4 CodeSnippetsType Class

The `CodeSnippetsType` class is intended to represent a set of code snippets extracted from within a CybOX object.

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

Name	Type	Multiplicity	Description
Code_Snippet	`ObjectPropertiesType`	1..*	The `Code Snippet` property characterizes a single code snippet extracted from a raw cyber object. This property should be of class `CodeObj:CodeObjectType`.

3.3.5 Compiler-Related Classes

3.3.5.1 CompilersType Class

The `CompilersType` class describes the compilers utilized during this build of this application.

Table 3-17. Properties of the `CompilersType` class

Name	Type	Multiplicity	Description
Compiler	`CompilerType`	1..*	The `Compiler` property characterizes a single compiler utilized during this build of this

			application.

3.3.5.2 CompilerType Class

The `CompilerType` class describes a single compiler utilized during this build of this application.

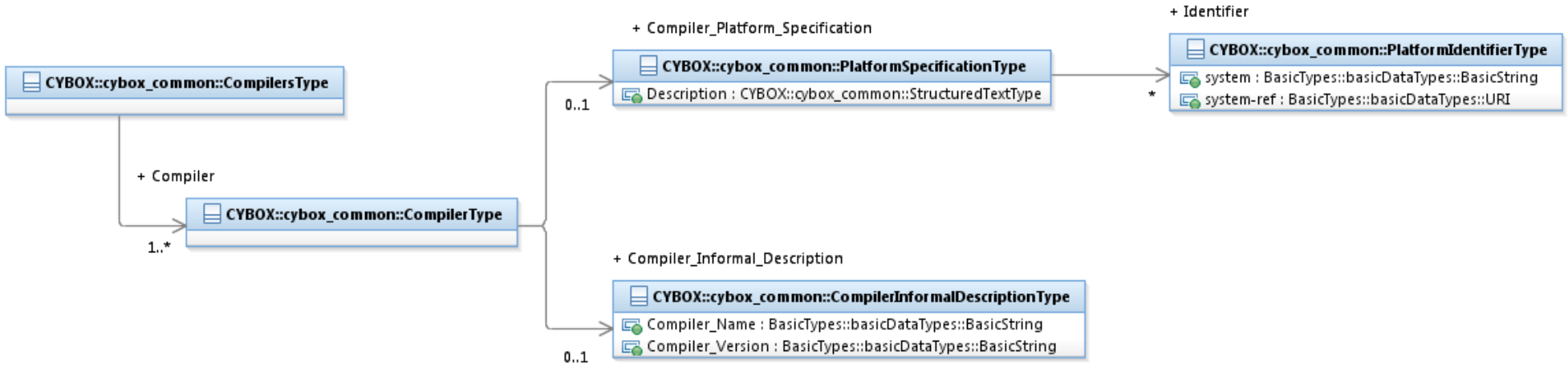

Figure 3-4. UML diagram for the `CompilerType` class

Table 3-18. Properties of the `CompilerType` class

Name	Type	Multiplicity	Description
Compiler_Informal_Description	`CompilerInformalDescriptionType`	0..1	The `Compiler_Informal_Description` property characterizes the informal description this compiler instance.
Compiler_Platform_Specification	`PlatformSpecificationType`	0..1	The `Compiler_Platform_Specification`

			property characterizes this compiler instance.

3.3.5.3 CompilerInformalDescriptionType Class

The `CompilerInformalDescriptionType` class contains the informal description of this compiler instance.

Table 3-19. Properties of the `CompilerInformalDescriptionType` class

Name	Type	Multiplicity	Description
Compiler_Name	`basicDataTypes: BasicString`	1	The `Compiler_Name` property captures the name of the compiler.
Compiler_Version	`basicDataTypes: BasicString`	0..1	The `Compiler_Version` property captures the version of the compiler.

3.3.6 ConfigurationSettingsType Class

The `ConfigurationSettingsType` class is a modularized data type used to provide a consistent approach to describing configuration settings for a tool, application or other cyber object.

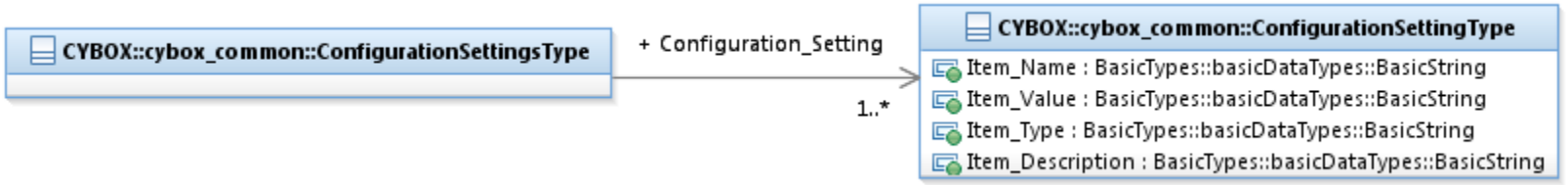

Figure 3-5. UML diagram for the `ConfigurationSettingsType` class

Table 3-20. Properties of the `ConfigurationSettingsType` class

Name	Type	Multiplicity	Description
Configuration_Setting	`ConfigurationSettingType`	1..*	The `Configuration_Setting` property specifies a single

			configuration setting instance.

3.3.6.1 ConfigurationSettingType Class

The `ConfigurationSettingType` class is a modularized data type used to provide a consistent approach to describing a particular configuration setting for a tool, application or other cyber object.

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

Name	Type	Multiplicity	Description
Item_Name	`basicDataTypes:BasicString`	1	The `Item_Name` property captures the name of the configuration item referenced by this configuration setting instance.
Item_Value	`basicDataTypes:BasicString`	1	The `Item_Value` property captures the value of this configuration setting instance.
Item_Type	`basicDataTypes:BasicString`	0..1	The `Item_Type` property captures the type of the configuration item referenced in this configuration setting instance.
Item_Description	`basicDataTypes:BasicString`	0..1	The `Item_Description` property captures a description of the configuration item referenced in this configuration setting instance.

3.3.7 CustomPropertiesType Class

The `CustomPropertiesType` class enables the specification of a set of custom Object Properties that may not be defined by existing Property data types.

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

Name	Type	Multiplicity	Description

Property	PropertyType	1..*	The Property property characterizes a single custom Object Property.

3.3.7.1 PropertyType Class

The PropertyType class is a type representing the specification of a single Object Property.

Table 3-23. Properties of the PropertyType class

Name	Type	Multiplicity	Description
name	basicDataTypes: BasicString	0..1	The name property captures the name for this custom property.
description	basicDataTypes: BasicString	0..1	The description property captures a description of what this custom property represents.

3.3.8 DataSegmentType Class

The DataSegmentType is intended to provide a relatively abstract way of characterizing data segments that may be written/read/transmitted or otherwise utilized in actions or behaviors.

Table 3-24. Properties of the DataSegmentType class

Name	Type	Multiplicity	Description
id	basicDataTypes: QualifiedName	0..1	The id property specifies a globally unique identifier for the Data Segment.
Data_Format	DataFormatEnum	0..1	The Data_Format property characterizes the type of data contained in the Data_Segment property.

Data_Size	DataSizeType	0..1	The Data_Size property characterizes the size of the data contained in this element.
Byte_Order	EndiannessType	0..1	The Byte_Order property characterizes the endianness of the unpacked (e.g., decoded, unencrypted, etc.) data stored within the Data_Segment property.
Data_Segment	StringObjectPropertyType	0..1	The Data_Segment property characterizes the actual segment of data being characterized.
Offset	IntegerObjectPropertyType	0..1	The Offset property characterizes where to start searching for the specified data segment in an object, in bytes.
Search_Distance	IntegerObjectPropertyType	0..1	The Search_Distance property characterizes how far into an object should be ignored, in bytes, before starting to search for the specified data segment relative to the end of the previous data segment.
Search_Within	IntegerObjectPropertyType	0..1	The Search_Within property characterizes that at most N bytes are between data segments in related objects.

3.3.9 DependenciesType Class

The DependenciesType class contains information describing a set of dependencies for this tool.

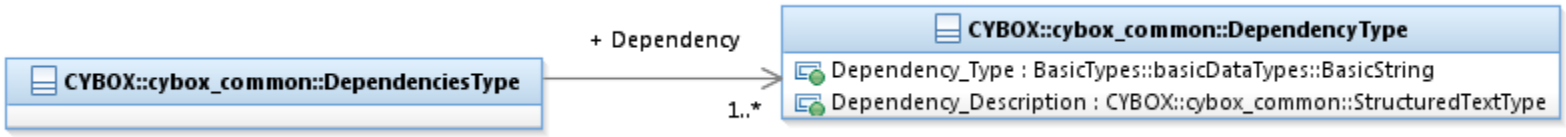

Figure 3-6. UML diagram for the DependencyType class

Table 3-25. Properties of the DependenciesType class

Name	Type	Multiplicity	Description
Dependency	DependencyType	1..*	The Dependency property characterizes a single dependency for this tool.

3.3.9.1 DependencyType Class

The DependencyType class contains information describing a single dependency for this tool.

Table 3-26. Properties of the DependencyType class

Name	Type	Multiplicity	Description
Dependency_Type	basicDataTypes: BasicString	0..1	The Dependency_Type property captures the type of this dependency instance.
Dependency_Description	StructuredTextType	1	The Dependency_Description property captures a description of this dependency instance. Any length is permitted. Optional formatting is supported via the structuring_format property of the StructuredTextType data type.

3.3.10 DigitalSignaturesType Class

The DigitalSignaturesType class is used for representing a list of digital signatures.

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

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

Name	Type	Multiplicity	Description
Digital_Signature	`DigitalSignatureInfoType`	0..*	The `Digital_Signature` property characterizes a single digital signature for this Object.

3.3.10.1 DigitalSignatureInfoType Class

The `DigitalSignatureInfoType` class is used as a way to represent some of the basic information about a digital signature.

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

Name	Type	Multiplicity	Description
signature_exists	`basicDataTypes:Boolean`	0..1	The `signature_exists` property specifies whether the digital signature exists.
signature_verified	`basicDataTypes:Boolean`	0..1	The `signature_verified` property specifies if the digital signature is verified.
Certificate_Issuer	`StringObjectPropertyType`	0..1	The `Certificate_Issuer` property characterizes the certificate issuer of the digital signature.

Certificate_Subject	StringObjectPropertyType	0..1	The Certificate_Subject property characterizes the certificate subject of the digital signature.
Signature_Description	StringObjectPropertyType	0..1	The Signature_Description property characterizes a description of the digital signature.

3.3.11 EnvironmentVariableListType Class

The EnvironmentVariableListType class is used for representing a list of environment variables.

Table 3-29. Properties of the EnvironmentVariableListType class

Name	Type	Multiplicity	Description
Environment_Variable	EnvironmentVariableType	1..*	The Environment_Variable property is used for capturing environment variables using a name/value pair.

3.3.11.1 EnvironmentVariableType Class

The EnvironmentVariableType class is used for representing environment variables using a name/value pair.

Table 3-30. Properties of the EnvironmentVariableType class

Name	Type	Multiplicity	Description
Name	StringObjectPropertyType	1	The Name property characterizes the name of the environment variable.
Value	StringObjectPropertyType	0..1	The Value property characterizes the value of the environment variable.

3.3.12 Error-Related Classes

3.3.12.1 ErrorsType Class

The `ErrorsType` class captures any errors generated during the run of the tool.

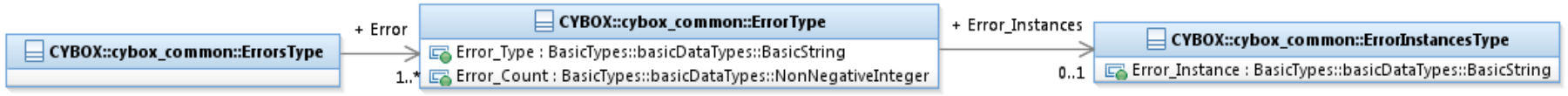

Figure 3-8. UML diagram for the `ErrorType` class

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

Name	Type	Multiplicity	Description
Error	`ErrorType`	1..*	The `Error` property captures a single type of error generated during the run of the tool.

3.3.12.2 ErrorType Class

The `ErrorType` class captures a single error generated during the run of the tool.

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

Name	Type	Multiplicity	Description
Error_Type	`basicDataTypes: BasicString`	1	The `Error_Type` property captures the type for this tool run error.
Error_Count	`basicDataTypes: PositiveInteger`	0..1	The `Error_Count` property specifies the count of instances for this error in the tool run.
Error_Instances	`ErrorInstancesType`	0..1	The `Error_Instances` property captures the actual error output for each

			instance of this type of error.

3.3.12.3 ErrorInstancesType Class

The `ErrorInstancesType` class captures the actual error output for each instance of this type of error.

Table 3-33. Properties of the `ErrorInstancesType` class

Name	Type	Multiplicity	Description
Error_Instance	`basicDataTypes: BasicString`	1..*	The `Error_Instance` property captures the actual error output for a single instance of this type of error.

3.3.13 ExtractedFeaturesType Class

The `ExtractedFeaturesType` class is a type representing a description of features extracted from an object such as a file.

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

Name	Type	Multiplicity	Description
Strings	`ExtractedStringsType`	0..1	The `Strings` property characterizes a set of static strings extracted from a raw cyber object.
Imports	`ImportsType`	0..1	The `Imports` property characterizes a set of references to external resources imported by a raw cyber object.
Functions	`FunctionsType`	0..1	The `Functions` property characterizes a set of references to functions called by a raw cyber object.
Code_Snippets	`CodeSnippetsType`	0..1	The `Code_Snippets` property characterizes a set of code snippets extracted from a raw cyber object.

3.3.14 ExtractedStringsType Class

The `ExtractedStringsType` class is intended as a container for strings extracted from CybOX objects.

Table 3-35. Properties of the `ExtractedStringsType` class

Name	Type	Multiplicity	Description
String	`ExtractedStringType`	1..*	The `String` property characterizes a single static string extracted from a raw cyber object.

3.3.14.1 ExtractedStringType Class

The `ExtractedStringType` class is intended as a container for a single string extracted from a CybOX object.

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

Name	Type	Multiplicity	Description
Encoding	`VocabularyStringType`	0..1	The `Encoding` property specifies the character encoding used for the `String_Value` property. Examples of potential values include *ASCII*, *UTF-8*, *Windows-1250* (these specific values are only provided to help explain the property: they are neither recommended values nor necessarily part of any existing vocabulary). The content creator may choose any arbitrary value or may constrain the set of possible values by referencing an externally-defined vocabulary or leveraging a formally defined vocabulary extending from the `cyboxCommon:ControlledVocabularyStringType` class. The CybOX default vocabulary class for use in the property is '*CharacterEncodingEnum-1.0*'.

String_Value	StringObjectPropertyType	0..1	The String_Value property characterizes the actual value of the string extracted from the CybOX object, if it is capable of being represented in the encoding scheme used in the document (most commonly UTF-8).
Byte_String_Value	HexBinaryObjectPropertyType	0..1	The Byte_String_Value property characterizes the raw, byte-string representation of the string extracted from the CybOX object, in hexadecimal format.
Hashes	HashListType	0..1	The Hashes property specifies any hash values computed using the string extracted from the CybOX object as input.
Address	HexBinaryObjectPropertyType	0..1	The Address property characterizes the location or offset of the specified string in the CybOX objects.
Length	PositiveIntegerObjectPropertyType	0..1	The Length property characterizes the length, in characters, of the string extracted from the CybOX object.
Language	StringObjectPropertyType	0..1	The Language property characterizes the language the string is written in, e.g. English. For consistency, we strongly recommend using a ISO 639-2 language code, if available. Please see http://www.loc.gov/standards/iso639-2/php/code_list.php for a list of ISO 639-2 codes.
English_Translation	StringObjectPropertyType	0..1	The English_Translation property characterizes the English translation of the string, if it is not written in English.

3.3.15 FunctionsType Class

The FunctionsType class is intended to represent an extracted list of functions leveraged within a CybOX object.

Table 3-37. Properties of the `FunctionsType` class

Name	Type	Multiplicity	Description
Function	`StringObjectPropertyType`	1..*	The `Function` property characterizes a single reference to a function called by a raw cyber object.

3.3.16 Hash-Related Classes

3.3.16.1 HashListType Class

The `HashListType` class is used for representing a list of hash values.

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

Name	Type	Multiplicity	Description
Hash	`HashType`	1..*	The `Hash` property specifies a single calculated hash value.

3.3.16.2 HashType Class

The `HashType` class is intended to characterize hash values.

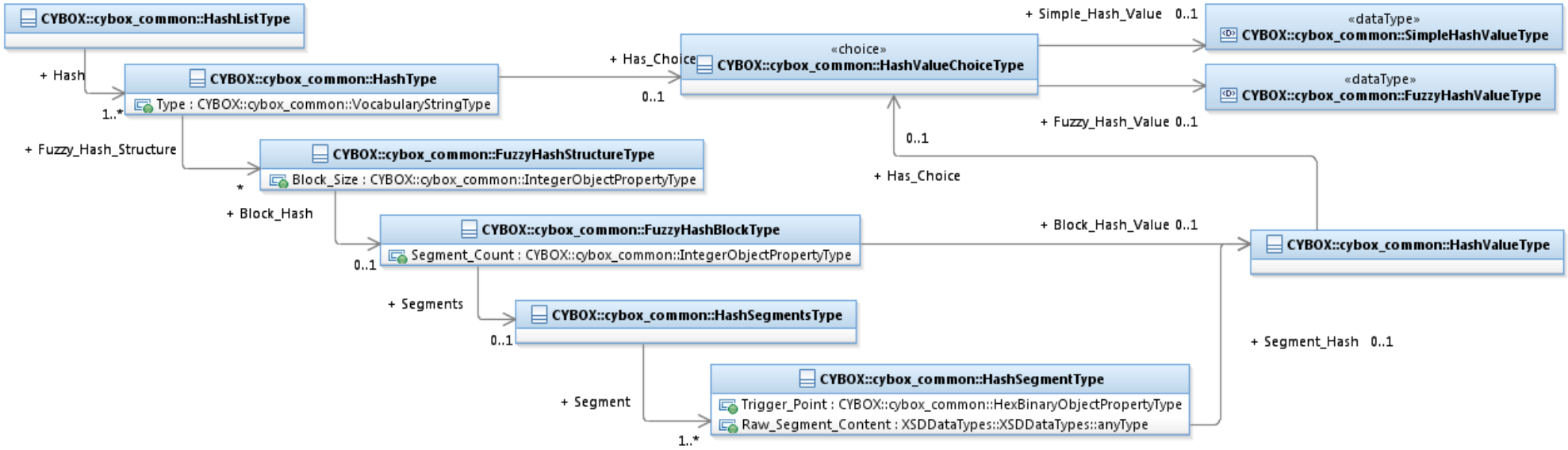

Figure 3-9. UML diagram for the `HashType` class

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

Name	Type	Multiplicity	Description
Type	`VocabularyStringType`	0..1	The `Type` property specifies the type of hash algorithm used to create the hash value. Examples of potential types of hashes are *MD5, SHA1* and *SHA256* (these specific values are only provided to help explain the property: they are neither recommended values nor necessarily part of any existing vocabulary). The content creator may choose any arbitrary value or may constrain the set of possible values by referencing an externally-defined vocabulary or leveraging a formally defined vocabulary extending from the `cyboxCommon:ControlledVocabularyStringType` class. The CybOX default vocabulary class for use in the property is '*HashNameEnum-1.0*'.

Fuzzy_Hash_Structure	FuzzyHashStructureType	0..*	The Fuzzy_Hash_Structure property enables the characterization of the key internal components of a fuzzy hash calculation with a given block size.
Has_Choice	HashValueChoiceType	0..1	The Has_Choice property is associated with the class HashValueChoiceType. It indicates that there is a choice between the Simple_Hash_Value property or the Fuzzy_Hash_Value property. Only one of the properties of HashValueChoiceType class can be populated at any time. See Section 1.2.3 for more detail.

3.3.16.3 HashValueType Class

The HashValueType class is used for specifying the resulting value from a hash calculation.

Table 3-40. Properties of the HashType class

Name	Type	Multiplicity	Description
Has_Choice	HashValueChoiceType	0..1	The Has_Choice property is associated with the class HashValueChoiceType. It indicates that there is a choice between the Simple_Hash_Value property or the Fuzzy_Hash_Value property. Only one of the properties of HashValueChoiceType class can be populated at any time. See Section 1.2.3 for more detail.

3.3.16.4 HashValueChoiceType Class

The HashValueChoiceType class is used for specifying the choice between different formats of the resulting value from a hash calculation. In the UML model, this class is associated with the <<choice>> UML stereotype, which specifies that only one of the available properties of the HashValueChoiceType class can be populated at any time.

See **Section 3.2.9** for details on `SimpleHashValueType` and `FuzzyHashValueType` data types.

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

Name	Type	Multiplicity	Description
Simple_Hash_Value	`SimpleHashValueType`	0..1	The `Simple_Hash_Value` property characterizes a single result value of a basic cryptographic hash function outputting a single hexbinary hash value. The `Simple_Hash_Value` and `Fuzzy_Hash_Value` properties MUST NOT both have a value.
Fuzzy_Hash_Value	`FuzzyHashValueType`	0..1	The `Fuzzy_Hash_Value` property characterizes a single result value of a cryptographic fuzzy hash function outputting a single complex string based hash value. (e.g., SSDEEP's Block1hash:Block2hash format). The `Simple_Hash_Value` and `Fuzzy_Hash_Value` properties MUST NOT both have a value.

3.3.16.5 FuzzyHashStructureType Class

The `FuzzyHashStructureType` class is used for characterizing the internal components of a cryptographic fuzzy hash algorithmic calculation.

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

Name	Type	Multiplicity	Description
Block_Size	`IntegerObjectPropertyType`	0..1	The `Block_Size` property characterizes the calculated block size for this fuzzy hash calculation.
Block_Hash	`FuzzyHashBlockType`	0..1	The `Block_Hash` property characterizes specification of the elemental components utilized for a fuzzy hash calculation on the hashed object

			utilizing the `Block_Size` property to calculate trigger points.

3.3.16.6 FuzzyHashBlockType Class

The `FuzzyHashBlockType` class is used for characterizing the internal components of a single block in a cryptographic fuzzy hash algorithmic calculation.

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

Name	Type	Multiplicity	Description
Block_Hash_Value	`HashValueType`	0..1	The `Block_Hash_Value` property characterizes a fuzzy hash calculation result value for this block.
Segment_Count	`IntegerObjectPropertyType`	0..1	The `Segment_Count` property characterizes the number of segments identified and utilized within this fuzzy hash calculation.
Segments	`HashSegmentsType`	0..1	The `Segments` property characterizes the set of segments identified and utilized within this fuzzy hash calculation.

3.3.16.7 HashSegmentsType Class

The `HashSegmentsType` class is used for characterizing the internal components of a set of trigger point-delimited segments in a cryptographic fuzzy hash algorithmic calculation.

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

Name	Type	Multiplicity	Description
Segment	`HashSegmentType`	1..*	The `Segment` property characterizes a single segment identified and utilized within this fuzzy hash calculation.

3.3.16.8 HashSegmentType Class

The `HashSegmentType` class is used for characterizing the internal components of a single trigger point-delimited segment in a cryptographic fuzzy hash algorithmic calculation.

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

Name	Type	Multiplicity	Description
Trigger_Point	`HexBinaryObjectPropertyType`	0..1	The `Trigger_Point` property characterizes the offset within the hashed object of the trigger point for this segment.
Segment_Hash	`HashValueType`	0..1	The `Segment_Hash` property characterizes a calculated hash value for this segment.
Raw_Segment_Content	`HexBinaryObjectPropertyType`	0..1	The `Raw_Segment_Content` property captures the raw content of this segment of the hashed object.

3.3.17 ImportsType Class

The `ImportsType` class is intended to represent an extracted list of imports specified within a CybOX object.

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

Name	Type	Multiplicity	Description
Import	`StringObjectPropertyType`	1..*	The `Import` property characterizes a single reference to an external resource imported by a raw cyber object.

3.3.18 InternationalizationSettingsType Class

The `InternationalizationSettingsType` class contains information describing relevant internationalization setting for this tool.

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

Name	Type	Multiplicity	Description
Internal_Strings	`InternalStringsType`	1..*	The `Internal_Strings` property captures a single internal string instance for this internationalization setting instance.

3.3.18.1 InternalStringsType Class

The `InternalStringsType` class contains a single internal string instance for this internationalization setting instance.

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

Name	Type	Multiplicity	Description
Key	`basicDataTypes: BasicString`	1	The `Key` property captures the actual key of this internal string instance.
Content	`basicDataTypes: BasicString`	1	The `Content` property captures the actual content of this internal string instance.

3.3.19 LibrariesType Class

The `LibrariesType` class identifies the libraries incorporated into the build of the tool.

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

Name	Type	Multiplicity	Description

Library	LibraryType	0..1	The Library property characterizes a library incorporated into the build of the tool.

3.3.19.1 LibraryType Class

The LibraryType class identifies a single library incorporated into the build of the tool.

Table 3-50. Properties of the LibraryType class

Name	Type	Multiplicity	Description
name	basicDataTypes: BasicString	0..1	The name property captures the name of the library.
version	basicDataTypes: BasicString	0..1	The version property captures the version of the library.

3.3.20 MetadataType Class

The MetadataType class is intended as mechanism to capture any non-context-specific metadata.

Table 3-51. Properties of the MetadataType class

Name	Type	Multiplicity	Description
type	basicDataTypes: BasicString	0..1	The type property captures the type of the name of a single metadata property.
Value	basicDataTypes: BasicString	0..1	The Value property captures the value of the name of a single metadata property.
SubDatum	MetadataType	0..*	The SubDatum property uses recursion of the MetadataType to characterize subdatum structures for this metadata property.

3.3.21 PersonnelType Class

The `PersonnelType` class is an abstracted data type to standardize the description of sets of personnel.

Table 3-52. Properties of the `PersonnelType` class

Name	Type	Multiplicity	Description
Contributor	`ContributorType`	1..*	The `Contributor` property characterizes the identity, resources and timing of involvement for a single contributor.

3.3.21.1 ContributorType Class

The `ContributorType` class represents a description of an individual who contributed as a source of cyber observation data.

Table 3-523. Properties of the `ContributorType` class

Name	Type	Multiplicity	Description
Role	`basicDataTypes: BasicString`	0..1	The `Role` property captures the role played by this contributor.
Name	`basicDataTypes: BasicString`	0..1	The `Name` property captures the name of this contributor.
Email	`basicDataTypes: BasicString`	0..1	The `Email` property captures the email of this contributor.
Phone	`basicDataTypes: BasicString`	0..1	The `Phone` property captures a telephone number of this contributor.
Organization	`basicDataTypes: BasicString`	0..1	The `Organization` property captures the organization name of this contributor.
Date	`DateRangeType`	0..1	The `Date` property characterizes a description (bounding) of the timing of this contributor's involvement.

Contribution_Location	basicDataTypes: BasicString	0..1	The Contribution_Location property captures the location at which the contributory activity occurred.

3.3.22 PlatformSpecificationType Class

The PlatformSpecificationType class is a modularized data type intended for providing a consistent approach to uniquely specifying the identity of a specific platform. In addition to capturing basic information, this type is intended to be extended to enable the structured description of a platform instance using the XML Schema extension feature. The CybOX default extension uses the Common Platform Enumeration (CPE) Applicability Language to do so.

Table 3-54. Properties of the PlatformSpecificationType class

Name	Type	Multiplicity	Description
Description	StructuredTextType	0..1	The Description property captures a technical description of the Platform Specification. Any length is permitted. Optional formatting is supported via the structuring_format property of the StructuredTextType class.
Identifier	PlatformIdentifierType	0..*	The Identifier property characterizes a pre-defined name for the given platform using some naming scheme. For example, one could provide a CPE (Common Platform Enumeration) name using the CPE naming format.

3.3.23 Tools-Related Classes

3.3.23.1 ToolsInformationType Class

The ToolsInformationType class represents a description of a set of automated tools.

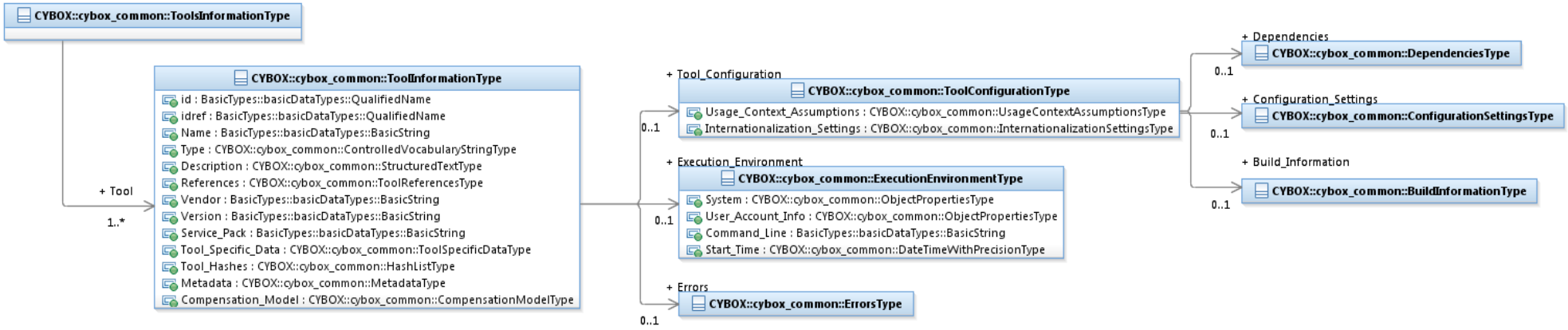

Figure 3-10. UML diagram for `ToolsInformationType` class

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

Name	Type	Multiplicity	Description
Tool	`ToolInformationType`	1..*	The `Tool` property characterizes a single tool utilized for this cyber observation source.

3.3.23.2 ToolInformationType Class

The `ToolInformationType` class is intended to characterize the properties of a hardware or software tool, including those related to instances of its use.

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

Name	Type	Multiplicity	Description
id	`basicDataTypes: QualifiedName`	0..1	The `id` property specifies a globally unique identifier for the Tool Information.

idref	basicDataTypes: QualifiedName	0..1	The idref property specifies an identifier reference to a ToolInformation instance specified elsewhere. When the idref property is used, no other property should be specified.
Name	basicDataTypes: BasicString	0..1	The Name property captures the name of the tool leveraged.
Type	VocabularyStringType	0..*	The Type property specifies the type of the tool. Examples of potential types are *NIDS*, *asset scanner*, and *malware analysis* (these specific values are only provided to help explain the property: they are neither recommended values nor necessarily part of any existing vocabulary). The content creator may choose any arbitrary value or may constrain the set of possible values by referencing an externally-defined vocabulary or leveraging a formally defined vocabulary extending from the cyboxCommon:ControlledVocabularyStringType class. The CybOX default vocabulary class for use in the property is *'ToolTypeVocab-1.1'*.
Description	StructuredTextType	0..1	The Description property captures a technical description of the Tool Information. Any length is permitted. Optional formatting is supported via the structuring_format property of the StructuredTextType class.
References	ToolReferencesType	0..1	The References property captures references to instances or additional information for this tool.
Vendor	basicDataTypes: BasicString	0..1	The Vendor property captures information identifying the vendor organization for this tool.
Version	basicDataTypes: BasicString	0..1	The Version property captures an appropriate version descriptor of this tool.

Service_Pack	basicDataTypes:BasicString	0..1	The Service_Pack property captures an appropriate service pack descriptor for this tool.
Tool_Specific_Data	ToolSpecificDataType	0..1	The Tool_Specific_Data property characterizes tool-specific data to be included.
Tool_Hashes	HashListType	0..1	The Tool_Hashes property captures a hash value computed on the tool file content in order to verify its integrity.
Tool_Configuration	ToolConfigurationType	0..1	The Tool_Configuation property characterizes the configuration and usage of the tool.
Execution_Environment	ExecutionEnvironmentType	0..1	The Execution_Environment property characterizes the execution environment of the tool.
Errors	ErrorsType	0..1	The Errors property captures any errors generated during the run of the tool.
Metadata	MetadataType	0..*	The Metadata property captures other relevant metadata including tool-specific properties.
Compensation_Model	CompensationModelType	0..1	The Compensation_Model property captures the name of the compensation model used for the tool.

3.3.23.3 ToolSpecificDataType Class

The ToolSpecificDataType class is an abstract class placeholder within the CybOX enabling the inclusion of metadata for a specific type of tool through the use of a custom type defined as an extension of this class.

3.3.23.4 ToolConfigurationType Class

The ToolConfigurationType class characterizes the configuration for a tool used as a cyber observation source.

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

Name	Type	Multiplicity	Description
Configuration_Settings	`ConfigurationSettingsType`	0..1	The `Configuration_Settings` property characterizes the configuration settings of this tool instance.
Dependencies	`DependenciesType`	0..1	The `Dependencies` property characterizes the relevant dependencies for this tool.
Usage_Context_Assumptions	`UsageContextAssumptionsType`	0..1	The `Usage_Context_Assumptions` property characterizes the various relevant usage context assumptions for this tool.
Internationalization_Settings	`Internationalization SettingsType`	0..1	The `Internationalization_Settings` property characterizes the relevant internationalization setting for this tool.
Build_Information	`BuildInformationType`	0..1	The `Build_Information` property characterizes how this tool was built.

3.3.23.5 ToolReferencesType Class

The `ToolReferencesType` class is used to indicate one or more references to tool instances and information.

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

Name	Type	Multiplicity	Description
Reference	`ToolReferenceType`	1..*	The `Reference` property specifies one reference to information or instances of a given tool.

3.3.23.6 ToolReferenceType Class

Contains one reference to information or instances of a given tool.

Table 3-59. Properties of the ToolReferenceType class

Name	Type	Multiplicity	Description
reference_type	`ToolReferenceTypeEnum`	0..1	The `reference_type` property specifies the nature of the referenced material (documentation, source, executable, etc.).

3.3.24 UsageContextAssumptionsType Class

The `UsageContextAssumptionsType` class contains descriptions of the various relevant usage context assumptions for this tool.

Table 3-60. Properties of the UsageContextAssumptionsType class

Name	Type	Multiplicity	Description
Usage_Context_Assumption	`StructuredTextType`	1..*	The `Usage_Context_Assumption` property captures a single usage context assumption for this tool.

3.4 Vocabulary Data Types

There are three vocabulary-related UML data types defined in the Common data model, and together they provide a content creator with four choices for defining content, listed below in order of formality. Please see *CybOX Version 2.1.1 Part 5: Vocabularies* for further information on CybOX vocabularies.

- Leverage a default vocabulary using the `ControlledVocabularyStringType` data type. CybOX v2.2.1 defines a collection of default vocabularies and associated enumerations that are based on input from the CybOX community (see *CybOX Version 2.1.1 Part 5: Vocabularies*); however, not all vocabulary properties have an assigned default vocabulary.
- Formally define a custom vocabulary using the `ControlledVocabularyStringType` data type. To achieve value enforcement, a custom vocabulary must be formally added to the CybOX Vocabulary data model. Because this is an extension of the CybOX Vocabulary data model, producers and consumers MUST be aware of the addition to the data model for successful sharing of CybOX documents.

- Reference an externally-defined, custom vocabulary using the `UnenforcedVocabularyStringType` data type to constrain the set of values. Externally-defined vocabularies are publically defined, but have not been included as formally specified vocabularies within the CybOX Vocabulary data model using the `ControlledVocabularyStringType` data type. In this case, it is sufficient to specify the name of the vocabulary and a URL that defines that vocabulary.
- Choose an arbitrary and unconstrained value using the `VocabularyStringType` data type.

While not required by the general CybOX language, default vocabularies should be used whenever possible to ensure the greatest level of compatibility between CybOX users. If an appropriate default vocabulary is not available a formally defined custom vocabulary can be specified and leveraged. In addition to compatibility advantages, using formally defined vocabularies (whether default vocabularies or otherwise defined) enables enforced use of valid enumeration values; please see *CybOX Version 2.1.1 Part 5: Vocabularies* for the associated policy.

If a formally defined vocabulary is not sufficient for a content producer's purposes, the CybOX Vocabulary data model allows the two alternatives listed above: externally defined custom vocabularies and arbitrary string values, which dispense with enumerated vocabularies altogether. If a custom vocabulary is not formally added to the Vocabulary data model then no enforcement policy of appropriate values is specified.

The UML diagram shown in **Figure 3-11** illustrates the relationships between the three vocabulary data types defined in the CybOX Common data model. As illustrated, all controlled vocabularies formally defined within the CybOX Vocabulary data model are defined using an enumeration derived from the `ControlledVocabularyStringType` data type.

As shown, the `HashNameVocab-1.0` enumeration (used as a defined controlled vocabulary exemplar) is defined as a specialization of the `ControlledVocabularyStringType` data type, and therefore it is also a specialization of the `VocabularyStringType` data type.

Further details of each vocabulary class are provided in Subsections **3.4.1** through **3.4.3**.

Figure 3-11. UML diagram of the CybOX Vocabulary data model

3.4.1 VocabularyStringType Data Type

The `VocabularyStringType` data type is the basic data type of all vocabularies. Therefore, all properties in the collection of CybOX data models that makes use of the Vocabulary data model must be defined to use the `VocabularyStringType` data type. Because this data type is a specialization of the `basicDataTypes:BasicString` data type, it can be used to support the arbitrary string option for vocabularies.

3.4.2 UnenforcedVocabularyStringType Data Type

The `UnenforcedVocabularyStringType` data type specifies custom vocabulary values via an enumeration defined outside of the CybOX Vocabulary data model. It extends the `VocabularyStringType` data type. Note that the CybOX vocabulary data model does not define any enforcement policy for this data type.

The property table of the `UnenforcedVocabularyStringType` data type is given in **Table 3-60**.

Table 3-61. Properties of the `UnenforcedVocabularyStringType` data type

Name	Type	Multiplicity	Description
vocab_name	`basicDataTypes:NoEmbeddedQuoteString`	0..1	The `vocab_name` property specifies the name of the externally defined vocabulary.
vocab_reference	`basicDataTypes:URI`	0..1	The `vocab_reference` property specifies the location of the externally defined vocabulary using a Uniform Resource Identifier (URI).

3.4.3 ControlledVocabularyStringType Data Type

The `ControlledVocabularyStringType` data type specifies a formally defined vocabulary. It is an abstract data type so it MUST be extended via an enumeration from the CybOX Vocabulary data model (descriptions of all default vocabularies defined within the CybOX Vocabulary data model are found in *CybOX Version 2.1.1 Part 5: Vocabularies*[1]). Any custom vocabulary must be defined via an enumeration added to the CybOX Vocabulary data model, if appropriate enumeration values are to be enforced.

The `ControlledVocabularyStringType` class has no properties of its own, so there is no associated property table.

3.5 General Classes and Data Types

3.5.1 DateRangeType Class

The `DateRangeType` class specifies a range of dates.

Table 3-62. Properties of the `DateRangeType` class

Name	Type	Multiplicity	Description
Start_Date	`DateWithPrecisionType`	0..1	The `Start_Date` property specifies the start date for this contributor's involvement. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date, the `Start` property MAY also capture a precision property to specify the granularity with which the time should be considered, as specified by the `DateTypePrecisionEnum` enumeration (e.g., '*day*').
End_Date	`DateWithPrecisionType`	0..1	The `End_Date` property specifies the end date for this contributor's involvement. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date, the `End` property MAY also capture a precision property to specify the granularity with which the time should be considered, as specified by the `DateTypePrecisionEnum` enumeration (e.g., *'day'*).

3.5.2 DateTimeWithPrecisionType Data Type

The `DateTimeWithPrecisionType` data type specializes the `basicDataTypes:DateTime` data type by capturing precision information. In order to avoid ambiguity, all uses SHOULD include a specification of the time zone.

If the precision is given, consumers must ignore the portions of this property that is more precise than the given precision. Producers should zero-out (fill with zeros) digits that are beyond the specified precision.

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

Name	Type	Multiplicity	Description

precision	DateTimePrecisionEnum	0..1	The precision property specifies the granularity with which a timestamp should be considered as specified by the DateTimePrecisionEnum enumeration (e.g., '*hour*,' '*minute*'). If omitted, the default precision is '*second*.' Digits in a timestamp that are beyond the specified precision SHOULD be zeroed out.

3.5.3 DateWithPrecisionType Data Type

The DateWithPrecisionType data type specializes the basicDataTypes:Date data type by capturing precision information.

If the precision is given, consumers must ignore the portions of this property that is more precise than the given precision. Producers should zero-out (fill with zeros) digits in the date that are beyond the specified precision.

Table 3-64. Properties of the DateWithPrecisionType class

Name	Type	Multiplicity	Description
precision	DatePrecisionEnum	0..1	The precision property specifies the granularity with which a date should be considered as specified by the DatePrecisionEnum enumeration (e.g., '*year*,' '*month, and 'day*"). If omitted, the default precision is '*day*.' Digits in a timestamp that are beyond the specified precision SHOULD be zeroed out

3.5.4 LocationType Class

The LocationType class is used to express geographic location information. This class is usually extended to incorporate specific location information. The default extension type is CIQAddress3.0InstanceType (see *CybOX Version 2.1.1 Part 4: Default Extensions*). Those who wish to express a simple name may also do so by simply using the Name property of this type.

Table 3-65. Properties of the LocationType class

Name	Type	Multiplicity	Description
id	basicDataTypes: QualifiedName	0..1	The id property specifies a globally unique identifier for the Location.

idref	basicDataTypes: QualifiedName	0..1	The idref property specifies an identifier reference to a Location instance specified elsewhere. When the idref property is used, no other property should be specified.
Name	basicDataTypes: BasicString	0..1	The Name property captures a location through a simple name.

3.5.5 StructuredTextType Data Type

The StructuredTextType class is a type representing a generalized structure for capturing structured or unstructured textual information such as descriptions of things.

Table 3-66. Properties of the StructuredTextType class

Name	Type	Multiplicity	Description
structuring_format	basicDataTypes: BasicString	0..1	The structuring_format property specifies a particular structuring format (e.g., HTML5) used within an instance of StructuredTextType. If this property is absent, then markup MUST NOT be used.

3.5.6 TimeType Class

The TimeType class specifies various time properties for this construct.

Table 3-67. Properties of the TimeType class

Name	Type	Multiplicity	Description
Start_Time	DateTimeWithPrecisionType	0..1	The Start_Time property specifies the starting time for this property. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date and time, the Start_Time property MAY also capture a precision property to specify the granularity with which the time should be considered, as specified by the DateTypePrecisionEnum enumeration (e.g., '*hour*,' '*minute*').

End_Time	DateTimeWithPrecisionType	0..1	The End_Time property specifies the ending time for this property. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date and time, the End_Time property MAY also capture a precision property to specify the granularity with which the time should be considered, as specified by the DateTypePrecisionEnum enumeration (e.g., '*hour*,' '*minute*').
Produced_Time	DateTimeWithPrecisionType	0..1	The Produced_Time property specifies the time that this property was produced. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date and time, the Produced_Time property MAY also capture a precision property to specify the granularity with which the time should be considered, as specified by the DateTypePrecisionEnum enumeration (e.g., '*hour*,' '*minute*').
Received_Time	DateTimeWithPrecisionType	0..1	The Received_Time property specifies the time that this property was received. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date and time, the Received_Time property MAY also capture a precision property to specify the granularity with which the time should be considered, as specified by the DateTypePrecisionEnum enumeration (e.g., '*hour*,' '*minute*').

3.6 Enumerations

3.6.1 CipherEnum Enumeration

Table 3-68. Literals of the `CipherEnum` enumeration

Enumeration Literal	Description
3DES	Specifies the Triple Data Encryption Standard (DES) algorithm.
AES	Specifies the Advanced Encryption Standard (AES) algorithm.
Blowfish	Specifies the Blowfish algorithm.
CAST-128	Specifies the CAST-128 algorithm.
CAST-256	Specifies the CAST-256 algorithm.
DES	Specifies the Data Encryption Standard (DES) algorithm.
IDEA	Specifies the International Data Encryption Algorithm (IDEA).
Rijndael	Specifies the Rijndael algorithm.
RC5	Specifies the RC5 algorithm.
Skipjack	Specifies the Skipjack algorithm.

3.6.2 CompensationModelEnum Enumeration

Table 3-69. Literals of the `CompensationModelEnum` enumeration

Enumeration Literal	Description
Freeware	Specifies that the tool is available for use at no monetary cost as the compensation model.
Shareware	Specifies that the tool is proprietary and offers a limited use license as the compensation model.
Commercial	Specifies that the tool was produced for sale or

	serves commercial purposes as the compensation model.
Adware	Specifies that the tool uses automatically rendered advertisements as the compensation model.

3.6.3 ConditionApplicationEnum Enumeration

Table 3-70. Literals of the `ConditionApplicationEnum` enumeration

Enumeration Literal	Description
ANY	Indicates that a pattern holds if the given condition can be successfully applied to any of the field values.
ALL	Indicates that a pattern holds only if the given condition can be successfully applied to all of the field values.
NONE	Indicates that a pattern holds only if the given condition can be successfully applied to none of the field values.

3.6.4 ConditionTypeEnum Enumeration

Table 3-71. Literals of the `ConditionTypeEnum` enumeration

Enumeration Literal	Description
Equals	Specifies the equality or = condition.
DoesNotEqual	Specifies the "does not equal" or != condition.
Contains	Specifies the "contains" condition.
DoesNotContain	Specifies the "does not contain" condition.
StartsWith	Specifies the "starts with" condition.
EndsWith	Specifies the "ends with" condition.
GreaterThan	Specifies the "greater than" condition.
GreaterThanOrEqual	Specifies the "greater than or equal to" condition.

LessThan	Specifies the "less than" condition.
LessThanOrEqual	Specifies the "less than or equal" condition.
InclusiveBetween	The pattern is met if the given value lies between the values indicated in the field value body, inclusive of the bounding values themselves. The field value body MUST contain at least 2 values to be valid. If the field value body contains more than 2 values, then only the greatest and least values are considered. (I.e., If the body contains "2,4,6", then an InclusiveBetween condition would be satisfied if the observed value fell between 2 and 6, inclusive. Since this is an inclusive range, an observed value of 2 or 6 would fit the pattern in this example.) As such, always treat the InclusiveBetween condition as applying to a single range for the purpose of evaluating the apply_condition attribute.
ExclusiveBetween	The pattern is met if the given value lies between the values indicated in the field value body, exclusive of the bounding values themselves. The field value body MUST contain at least 2 values to be valid. If the field value body contains more than 2 values, then only the greatest and least values are considered. (I.e., If the body contains "2,4,6", then an InclusiveBetween condition would be satisfied if the observed value fell between 2 and 6, exclusive. Since this is an exclusive range, an observed value of 2 or 6 would not fit the pattern in this example.) As such, always treat the ExclusiveBetween condition as applying to a single range for the purpose of evaluating the apply_condition attribute.
FitsPattern	Specifies the condition that a value fits a given pattern.
BitwiseAnd	Specifies the condition of bitwise AND. Specifically, when applying this pattern, a given value is bitwise-ANDed with the bit_mask attribute value (which must be present). If the result is identical to the value provided in the body of this field value, the pattern is considered fulfilled.
BitwiseOr	Specifies the condition of bitwise OR. Specifically, when applying this pattern, a given value is bitwise-ORed with the bit_mask attribute value (which must be present). If the result is identical to the value provided in the body of this field value, the pattern is considered fulfilled.

BitwiseXor	Specifies the condition of bitwise XOR. Specifically, when applying this pattern, a given value is bitwise-XORed with the bit_mask attribute value (which must be present). If the result is identical to the value provided in the body of this field value, the pattern is considered fulfilled.

3.6.5 DataFormatEnum Enumeration

Table 3-72. Literals of the `DataFormatEnum` enumeration

Enumeration Literal	Description
Binary	Specifies binary data.
Hexadecimal	Specifies hexadecimal data.
Text	Specifies text.
Other	Specifies any other type of data from the ones listed.

3.6.6 DataSizeUnitsEnum Enumeration

Table 3-73. Literals of the `DataSizeUnitsEnum` enumeration

Enumeration Literal	Description
Bytes	Specifies an object size in Bytes.
Kilobytes	Specifies an object size in Kilobytes.
Megabytes	Specifies an object size in Megabytes.

3.6.7 DatatypeEnum Enumeration

Table 3-74. Literals of the `DatatypeEnum` enumeration

Enumeration Literal	Description
string	Specifies the string datatype as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#string for

	more information.
int	Specifies the int datatype as it applies to the W3C standard for int. See http://www.w3.org/TR/xmlschema-2/#int for more information.
float	Specifies the float datatype as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#float for more information.
date	Specifies a date, which is usually in the form yyyy-mm-dd as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#date for more information.
positiveInteger	Specifies a positive integer in the infinite set {1,2,...} as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#positiveInteger for more information.
unsignedInt	Specifies an unsigned integer, which is a nonnegative integer in the set {0,1,2,...,4294967295} as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#unsignedInt for more information.
dateTime	Specifies a date in full format including both date and time as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#dateTime for more information.
time	Specifies a time as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#time for more information.
boolean	Specifies a boolean value in the set {true,false,1,0} as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#boolean for more information.
name	Specifies a name (which represents XML Names) as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#Name and http://www.w3.org/TR/2000/WD-xml-2e-20000814#dt-name for more information.
long	Specifies a long integer, which is an integer whose maximum value is 9223372036854775807

	and minimum value is -9223372036854775808 as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#long for more information.
unsignedLong	Specifies an unsigned long integer, which is an integer whose maximum value is 18446744073709551615 and minimum value is 0 as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#unsignedLong for more information.
duration	Specifies a length of time in the extended format PnYn MnDTnH nMnS, where nY represents the number of years, nM the number of months, nD the number of days, 'T' is the date/time separator, nH the number of hours, nM the number of minutes and nS the number of seconds, as it applies to the W3 standard. See http://www.w3.org/TR/xmlschema-2/#duration for more information.
double	Specifies a decimal of datatype double as it is patterned after the IEEE double-precision 64-bit floating point type (IEEE 754-1985) and as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#double for more information.
nonNegativeInteger	Specifies a non-negative integer in the infinite set {0,1,2,...} as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#nonNegativeInteger for more information.
hexBinary	Specifies arbitrary hex-encoded binary data as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#hexBinary for more information.
anyURI	Specifies a Uniform Resource Identifier Reference (URI) as it applies to the W3C standard and to RFC 2396, as amended by RFC 2732. See http://www.w3.org/TR/xmlschema-2/#anyURI for more information.
base64Binary	Specifies base64-encoded arbitrary binary data as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#base64Binary for more information.
IPv4 Address	Specifies an IPV4 address in dotted decimal form.

	CIDR notation is also accepted.
IPv6 Address	Specifies an IPV6 address, which is represented by eight groups of 16-bit hexadecimal values separated by colons (:) in the form a:b:c:d:e:f:g:h. CIDR notation is also accepted.
Host Name	Specifies a host name. For compatibility reasons, this could be any string. Even so, it is best to use the proper notation for the given host type. For example, web hostnames should be written as fully qualified hostnames in practice.
MAC Address	Specifies a MAC address, which is represented by six groups of 2 hexadecimal digits, separated by hyphens (-) or colons (:) in transmission order.
Domain Name	Specifies a domain name, which is represented by a series of labels concatenated with dots conforming to the rules in RFC 1035, RFC 1123, and RFC 2181.
URI	Specifies a Uniform Resource Identifier, which identifies a name or resource and can act as a URL or URN.
TimeZone	Specifies a timezone in UTC notation (UTC+number).
Octal	Specifies arbitrary octal (base-8) encoded data.
Binary	Specifies arbitrary binary encoded data.
BinHex	Specifies arbitrary data encoded in the Mac OS-originated BinHex format.
Subnet Mask	Specifies a subnet mask in IPv4 or IPv6 notation.
UUID/GUID	Specifies a globally/universally unique ID represented as a 32-character hexadecimal string. See ISO/IEC 11578:1996 Information technology -- Open Systems Interconnection -- Remote Procedure Call - http://www.iso.ch/cate/d2229.html.
Collection	Specifies data represented as a container of multiple data of a shared elemental type.

CVE ID	Specifies a CVE ID, expressed as CVE- appended by a four-digit integer, a - and another four-digit integer, as in CVE-2012-1234.
CWE ID	Specifies a CWE ID, expressed as CWE- appended by an integer.
CAPEC ID	Specifies a CAPEC ID, expressed as CAPEC- appended by an integer.
CCE ID	Specifies a CCE ID, expressed as CCE- appended by an integer.
CPE Name	Specifies a CPE Name. See http://cpe.mitre.org/specification/archive/version2.0/cpe-specification_2.0.pdf for more information.

3.6.8 DatePrecisionEnum Enumeration

Table 3-75. Literals of the `DatePrecisionEnum` enumeration

Enumeration Literal	Description
year	Date is precise to the given year.
month	Date is precise to the given month.
day	Date is precise to the given day.

3.6.9 EndiannessTypeEnum Enumeration

Table 3-76. Literals of the `EndiannessTypeEnum` enumeration

Enumeration Literal	Description
Big-endian	The Big-endian value specifies a big-endian byte ordering.
Little-endian	The Little-endian value specifies a little-endian byte ordering.
Middle-endian	The Middle-endian value specifies a middle-endian byte ordering.

3.6.10 Layer4ProtocolEnum Enumeration

Table 3-77. Literals of the `Layer4ProtocolEnum` enumeration

Enumeration Literal	Description
TCP	Specifies the Transmission Control Protocol.
UDP	Specifies the User Datagram Protocol.
AH	Specifies the Authentication Header protocol.
ESP	Specifies the Encapsulating Security Payload protocol.
GRE	Specifies the Generic Routing Encapsulation protocol.
IL	Specifies the Internet Link protocol.
SCTP	Specifies the Stream Control Transmission Protocol.
Sinec H1	Specifies the Siemens Sinec H1 protocol.
SPX	Specifies the Sequenced Packet Exchange protocol.
DCCP	Specifies the Datagram Congestion Control Protocol.

3.6.11 PatternTypeEnum Enumeration

Table 3-78. Literals of the `PatternTypeEnum` enumeration

Enumeration Literal	Description
Regex	Specifies the regular expression pattern type.
Binary	Specifies the binary (bit operations) pattern type.
XPath	Specifies the XPath 1.0 expression pattern type.

3.6.12 RegionalRegistryTypeEnum Enumeration

Table 3-79. Literals of the `RegionalRegistryTypeEnum` enumeration

Enumeration Literal	Description
AfriNIC	AfriNIC stands for African Network Information Centre, and is the RIR for Africa.
ARIN	ARIN stands for American Registry for Internet Numbers, and is the RIR for the United States, Canada, several parts of the Caribbean Region, and Antarctica.
APNIC	APNIC stands for Asia-Pacific Network Information Centre, and is the RIR for Asia, Australia, New Zealand, and neighboring countries.
LACNIC	LACNIC stands for Latin American and Caribbean Network Information Centre, and is the RIR for Latin America and parts of the Caribbean region.
RIPE NCC	RIPE NCC stands for Réseaux IP Européens Network Coordination Centre, and is the RIR for Europe, Russia, the Middle East, and Central Asia.

3.6.13 SIDTypeEnum Enumeration

Table 3-80. Literals of the `SIDTypeEnum` enumeration

Enumeration Literal	Description
SidTypeUser	Indicates a SID of type User.
SidTypeGroup	Indicates a SID of type Group.
SidTypeDomain	Indicates a SID of type Domain.
SidTypeAlias	Indicates a SID of type Alias.
SidTypeWellKnownGroup	Indicates a SID for a well-known group.
SidTypeDeletedAccount	Indicates a SID for a deleted account.

SidTypeInvalid	Indicates an invalid SID.
SidTypeUnknown	Indicates a SID of unknown type.
SidTypeComputer	Indicates a SID for a computer.
SidTypeLabel	Indicates a mandatory integrity label SID.

3.6.14 SourceClassTypeEnum Enumeration

Table 3-81. Literals of the `SourceClassTypeEnum` enumeration

Enumeration Literal	Description
Network	Describes a Network-based cyber observation.
System	Describes a System-based cyber observation.
Software	Describes a Software-based cyber observation.

3.6.15 SourceTypeEnum Enumeration

Table 3-82. Literals of the `SourceTypeEnum` enumeration

Enumeration Literal	Description
Tool	Describes a cyber observation made using various tools, such as scanners, firewalls, gateways, protection systems, and detection systems. See ToolTypeEnum for a more complete list of tools that CybOX supports.
Analysis	Describes a cyber observation made from analysis methods, such as Static and Dynamic methods. See `AnalysisMethodTypeEnum` for a more complete list of methods that CybOX supports.
Information Source	Describes a cyber observation made using other information sources, such as logs, Device Driver APIs, and TPM output data. See `InformationSourceTypeEnum` for a more complete list of information sources that CybOX supports.

3.6.16 TimePrecisionEnum Enumeration

Table 3-83. Literals of the `TimePrecisionEnum` enumeration

Enumeration Literal	Description
hour	Time is precise to the given hour.
minute	Time is precise to the given minute.
second	Time is precise to the given second (including fractional seconds).

3.6.17 ToolReferenceTypeEnum Enumeration

Table 3-84. Literals of the `ToolReferenceTypeEnum` enumeration

Enumeration Literal	Description
Documentation	The reference is to documentation about the identified tool.
Source	The reference is to source code for the identified tool.
Download	The reference is to where an executable version of the tool can be downloaded.
Execute	The reference is to the tool implemented as an online service.
Other	The reference is to material about the tool not covered by other values in 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. Acknowledgements

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
Emmanuelle Vargas-Gonzalez
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

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

Michael Pepin

Natalie Suarez

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

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.

---

[1] Note that all defined vocabulary enumerations have version numbers in their names to facilitate additions to the enumerations that are backward compatible.

