CybOX™ Version 2.1.1. Part 03: Core

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/part03-core/cybox-v2.1.1-csprd01-part03-core.docx (Authoritative)
http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part03-core/cybox-v2.1.1-csprd01-part03-core.html
http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part03-core/cybox-v2.1.1-csprd01-part03-core.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/cti/cybox/v2.1.1/part03-core/cybox-v2.1.1-part03-core.docx (Authoritative)
http://docs.oasis-open.org/cti/cybox/v2.1.1/part03-core/cybox-v2.1.1-part03-core.html
http://docs.oasis-open.org/cti/cybox/v2.1.1/part03-core/cybox-v2.1.1-part03-core.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 Core 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-core]

CybOX™ Version 2.1.1. Part 03: Core. Edited by Desiree Beck, Trey Darley, Ivan Kirilov, 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/part03-core/cybox-v2.1.1-csprd01-part03-core.html. Latest version: http://docs.oasis-open.org/cti/cybox/v2.1.1/part03-core/cybox-v2.1.1-part03-core.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 Core 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 **4**.

1.1 CybOX™ Specification Documents

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

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

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

The *CybOX Version 2.1.1 Part 1: Overview* document provides a comprehensive overview of the full set of CybOX data models, which in addition to the Core, Common, and numerous Object data models, includes 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 Core 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.

2 Background Information

In this section, we provide high level information about the Core data model that is necessary to fully understand the specification details given in Section **3**. The CybOX Core data model contains the data models for the core elements of CybOX (Observables, Events, Actions and Objects) as part of a framework hanging all of these elements together in an integrated, flexible and extensible manner.

2.1 Cyber Observables

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

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

2.2 Objects

Objects in CybOX are individual data models for characterizing a particular cyber entity, such as a Windows registry key, or an Email Message, for example. Accordingly, each release of the CybOX language includes a particular set of Objects that are part of the release. The data model for each of these Objects is defined by its own specification that describes the context-specific classes and properties that compose the Object.

2.3 Events and Actions

Events and Actions are data models designed to characterize dynamic cyber observable activity. They support modular expression of any event made up of one or more actions with the ability to relate actions to one another and to relate actions to relevant objects. The Action data model allows expression of the nature of the action, any relevant arguments and relationships to any relevant objects including the nature of the relationship and any specific effects the action has on the object.

3 CybOX Core Data Model

The CybOX Core data model defines a variety of classes. For discussion purposes, we have separated the classes into five categories (Sections **3.1** through **3.4**), and within each category, we primarily define the classes in alphabetical order below, except for the cases when a class is uniquely used in the main class, it will be defined within the same subsection. We list enumerations in Section **3.5**.

3.1 Primary Classes

The following classes are the primary classes in CybOX and enable the capture of Actions, Events, Objects, and Observables (Stateful Measures).

3.1.1 ActionType Class

The `ActionType` class characterizes a cyber observable Action. The UML diagram corresponding to the `ActionType` class is shown in Figure 3-1.

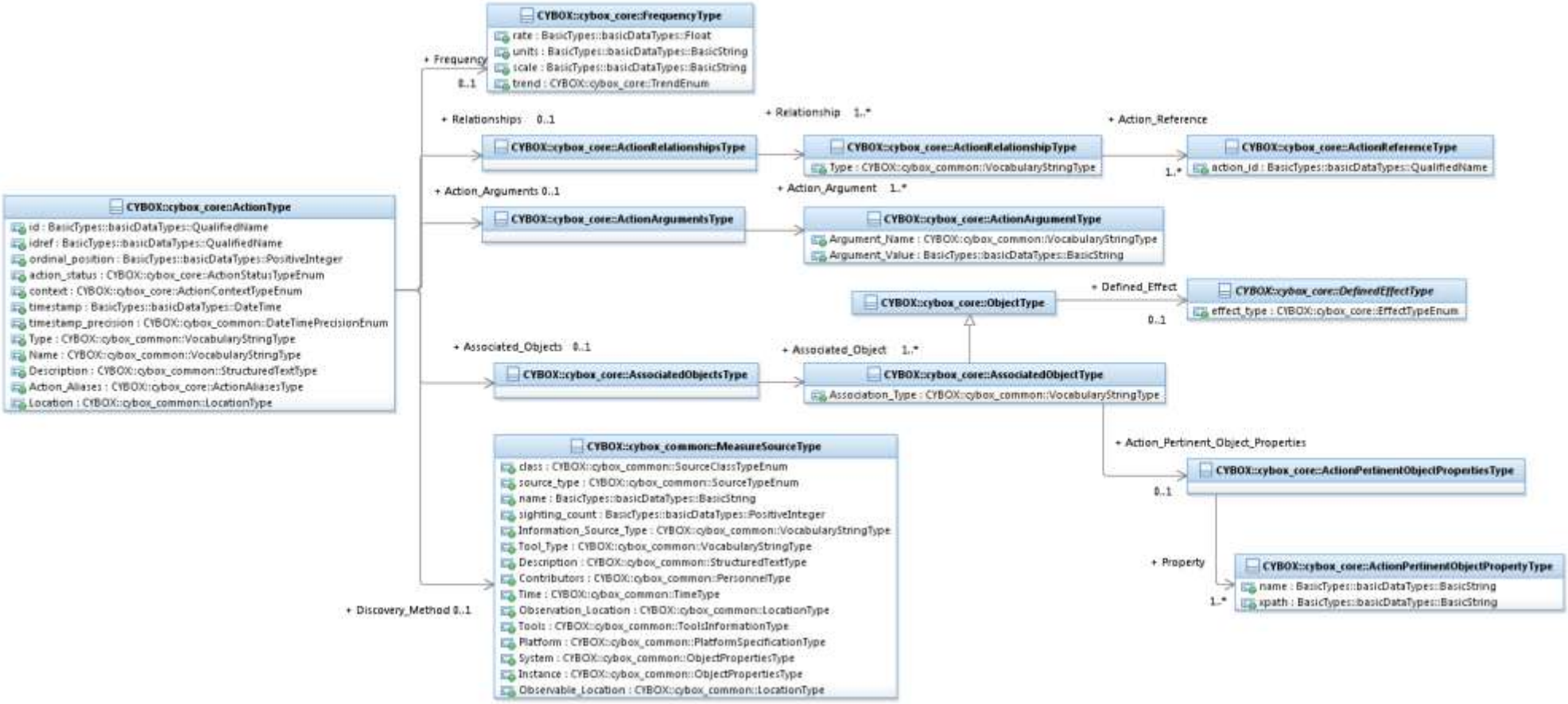

Figure 3-1. UML diagram for the `ActionType` class

*The property table given in **Table 3-1** corresponds to the UML diagram shown in **Figure 3-1**.*

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

Name	Type	Multiplicity	Description

id	basicDataTypes: QualifiedName	0..1	The id property specifies a globally unique identifier for the Action.
idref	basicDataTypes: QualifiedName	0..1	The idref property specifies a globally unique identifier for an Action specified elsewhere. When the idref property is used, the id property MUST NOT also be specified and the other properties of the ActionType class SHOULD NOT hold any content.
ordinal_position	basicDataTypes: PositiveInteger	0..1	The ordinal_position property specifies the order (e.g., *1*, *2*, or *3*) of the Action within a potential set of multiple Actions. If only a single Action is present, its ordinality can be assumed to be 1. If multiple Actions are present, the ordinality property SHOULD be specified with unique values for each instance.
action_status	ActionStatusTypeEnum	0..1	The action_status property specifies the status of the Action.
context	ActionContextTypeEnum	0..1	The context property specifies the broad operational context in which the Action is relevant.
timestamp	basicDataTypes: DateTime	0..1	The timestamp property specifies the date and time at which the Action occurred or was observed. To avoid ambiguity, all timestamps SHOULD include a specification of the time zone.
timestamp_precision	cyboxCommon: DateTimePrecisionEnum	0..1	The timestamp_precision property specifies the granularity with which the timestamp property should be considered, as specified by the DateTypePrecisionEnum 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.
Type	cyboxCommon: VocabularyStringType	0..1	The Type property specifies the type of the Action that was performed. Examples of potential types include *compress, replicate*, and *suspend* (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` data type. The CybOX default vocabulary class for use in the property is '*ActionTypeVocab-1.0*'.
Name	`cyboxCommon:VocabularyStringType`	0..1	The `Name` property specifies the name of the Action that was performed. Examples of potential names include *add user, connect to socket*, and *monitor registry key* (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` data type. The CybOX default vocabulary class for use in the property is '*ActionNameVocab-1.1*'.
Description	`cyboxCommon:StructuredTextType`	0..1	The `Description` property captures a textual description of the Action. Any length is permitted. Optional formatting is supported via the `structuring_format` property of the `StructuredTextType` data type.
Action_Aliases	`ActionAliasesType`	0..1	The `Action_Aliases` property specifies a set of one or more alias names for the Action.
Action_Arguments	`ActionArgumentsType`	0..1	The `Action_Arguments` property specifies a set of one or more arguments or parameters relevant to the Action.
Location	`cyboxCommon:LocationType`	0..1	The `Location` property characterizes the actual physical location of the Object. A simple location name may be specified or the underlying class may be extended, in which case, the default and strongly RECOMMENDED subclass is `CIQAddress3.0InstanceType`, as defined in *CybOX Version 2.1.1 Part 4: Default Extensions*.

Discovery_Method	cyboxCommon: MeasureSourceType	0..1	The Discovery_Method property characterizes how the Action was observed (in the case of a cyber observable Action instance) or could potentially be observed (in the case of a cyber observable Action pattern). Examples of details captured include identifying characteristics, time-related attributes, and a list of the tools used to collect the information.
Associated_Objects	AssociatedObjectsType	0..1	The Associated_Objects property specifies a set of one or more cyber Objects relevant to the Action (initiating or affected by).
Relationships	ActionRelationshipsType	0..1	The Relationships property specifies a set of one or more relationships between this Action and other Actions.
Frequency	FrequencyType	0..1	The Frequency property characterizes the frequency of the Action.

3.1.1.1 ActionArgumentType Class

The ActionArgumentType class characterizes an argument or parameter relevant to an Action.

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

Table 3-2. Properties of the ActionArgumentType class

Name	Type	Multiplicity	Description
Argument_Name	cyboxCommon: VocabularyStringType	0..1	The Argument_Name property specifies the name of the argument. Examples of potential names include *application name, base address,* and *size* (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 data type. The CybOX default vocabulary class for use in the property is

			'*ActionArgumentNameVocab-1.0*'.
Argument_Value	basicDataTypes: BasicString	0..1	The Argument_Value property specifies the value of the Action argument or parameter.

3.1.1.2 ActionPertinentObjectPropertyType Class

The ActionPertinentObjectPropertyType class characterizes a property of an Object that is relevant to an Action.

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

Table 3-3. Properties of the ActionPertinentObjectPropertyType class

Name	Type	Multiplicity	Description
name	basicDataTypes:BasicString	0..1	The name property specifies the field name for the Object property.
xpath	basicDataTypes:BasicString	0..1	The xpath[1] property specifies the identifying the pertinent property of the data model that corresponds to the Object's class. Specific syntax is dependent on the particular syntactic implementation (XML, JSON, etc.) of the CybOX language and MUST be explicitly specified in a separate binding specification for that syntactic implementation (e.g. a CybOX XML Binding Specification). For example, a CybOX XML Binding Specification could specify XPath 1.0 as an appropriate choice for the syntax of the xpath property.

3.1.1.3 ActionRelationshipType Class

The ActionRelationshipType class characterizes a relationship between one Action and a related Action.

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

Table 3-4. Properties of the ActionRelationshipType class

Name	Type	Multiplicity	Description
Type	cyboxCommon: VocabularyStringType	0..1	The Type property specifies the type of relationship between two actions. Examples of potential types include *dependent on, preceded by*, and *equivalent to* (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 data type. The CybOX default vocabulary class for use in the property is '*ActionRelationshipTypeVocab-1.0*'.
Action_Reference	ActionReferenceType	1..*	The Action_Reference property captures a reference to the related Action.

The ActionReferenceType class is defined because in some cases it is not appropriate to define an Action *only* in the context of another Action, and in those cases, an otherwise defined Action should be referenced.

3.1.1.4 ActionReferenceType Class

The ActionReferenceType class captures a reference to a related Action.

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

Table 3-5. Properties of the ActionReferenceType class

Name	Type	Multiplicity	Description
action_id	basicDataTypes:QualifiedName	0..1	The action_id property specifies the globally unique identifier of the Action referenced.

3.1.1.5 AssociatedObjectType Class

The `AssociatedObjectType` class characterizes a cyber observable Object associated with a given cyber observable Action (i.e., an Object that is initiated by or affected by the Action). It extends the `ObjectType` superclass (see Section **3.1.3**).

The UML diagram corresponding to the `AssociatedObjectType` class is shown in **Figure 3-2**.

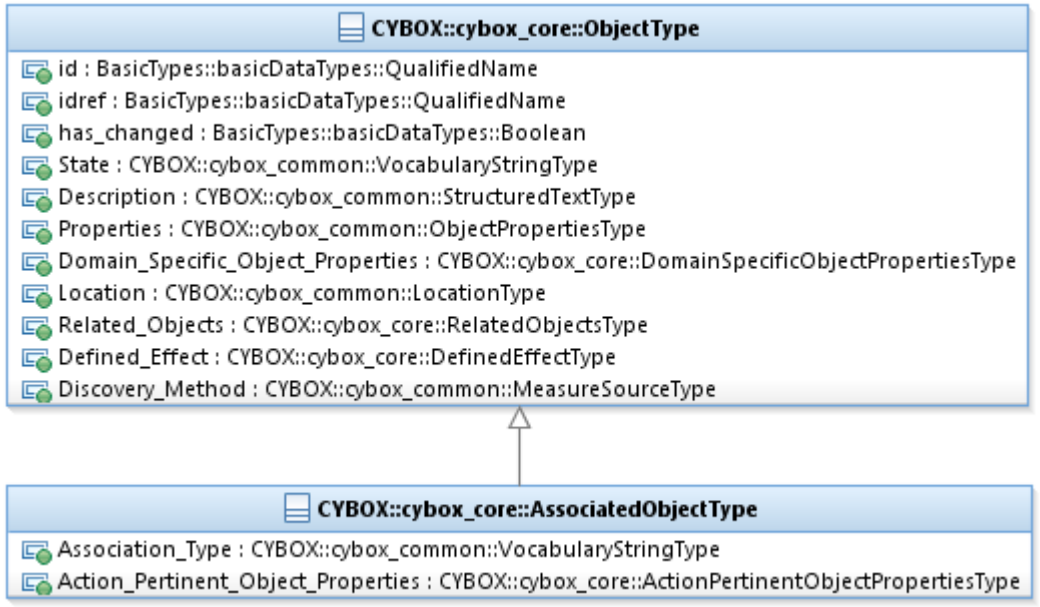

Figure 3-2. UML diagram of the `AssociatedObjectType` class

The property table given in **Table 3-6** corresponds to the UML diagram shown in **Figure 3-2**.

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

Name	Type	Multiplicity	Description
Association_Type	`cyboxCommon: VocabularyStringType`	0..1	The `Association_Type` property specifies types of Action-Object associations. Examples of potential types include *initiating, affected*, and

			utilized (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` data type. The CybOX default vocabulary class for use in the property is '*ActionObjectAssociationTypeVocab-1.0'*.
Action_Pertinent_ Object_Properties	`ActionPertinent ObjectPropertiesType`	0..1	The `Action_Pertinent_Object_Properties` property specifies a set of one or more properties of the Object that are pertinent to the Action.

3.1.2 EventType Class

The `EventType` class characterizes a cyber observable Event, which is a set of specific Action(s) involving specific cyber relevant Objects. Examples of Events include: a file is deleted, a registry key is created, or an HTTP GET Request is received. The UML diagram corresponding to the `EventType` class is shown in Figure 3-3.

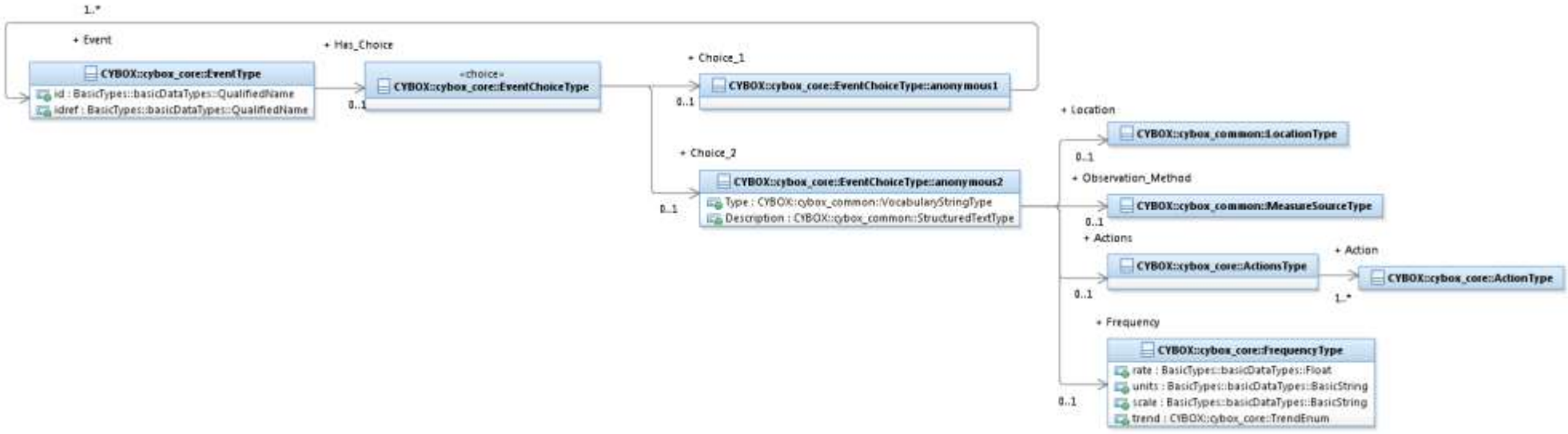

Figure 3-3. UML diagram for the `EventType` class

*The property table given in **Table 3-7** corresponds to the UML diagram shown in **Figure 3-3**.*

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

Name	Type	Multiplicity	Description
id	`basicDataTypes: QualifiedName`	0..1	The `id` property specifies a globally unique identifier for the Event.
idref	`basicDataTypes: QualifiedName`	0..1	The `idref` property specifies a globally unique identifier for an Event specified elsewhere. When the `idref` property is used, the `id` property MUST NOT also be specified and the other properties of the `EventType` class SHOULD NOT hold any content.
Has_Choice	`EventChoiceType`	0..1	The *`Has_Choice`* property is associated with the class `EventChoiceType`. It indicates that there is a choice between the

			`Choice_1` property or the `Choice_2` property. Only one of the properties of `EventChoiceType` class can be populated at any time. See Section Error! Reference source not found. for more d etail.

Name	Type	Multiplicity	Description
Choice_1	`EventChoiceType: Anonymous1`	0..1	The `Choice_1` property is a placeholder for specifying a list of `Events`.
Choice_2	`EventChoiceType: Anonymous2`	0..1	The `Choice_2` property is a placeholder for specifying Event details.

3.1.2.1 Anonymous1 Class

The property table of the Anonymous1 class is given in **Table 3-9**.

Table 3-8. Properties of the `Anonymous1` class

Name	Type	Multiplicity	Description
Event	`Event`	1..*	The `Event` property characterizes a cyber Event.

3.1.2.2 Anonymous2 Class

The property table of the Anonymous2 class is given in **Table 3-9**.

Table 3-9. Properties of the `Anonymous2` class

Name	Type	Multiplicity	Description

Type	cyboxCommon: VocabularyStringType	0..1	The Type property specifies the type of the Event. Examples of potential types include *http traffic, socket operations*, and *autorun* (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 data type. The CybOX default vocabulary class for use in the property is '*EventTypeVocab-1.0.1'*.
Description	cyboxCommon: StructuredTextType	0..1	The Description property captures a textual description of the Event. Any length is permitted. Optional formatting is supported via the structuring_format property of the StructuredTextType data type.
Observation_Method	cyboxCommon: MeasureSourceType	0..1	The Observation_Method property characterizes how the Event was observed (in the case of a cyber observable Event instance) or could potentially be observed (in the case of a cyber observable Event pattern). Examples of details captured include identifying characteristics, time-related attributes, and a list of the tools used to collect the information.
Actions	ActionsType	0..1	The Actions property specifies a set of one or more Actions that compose the Event.
Location	cyboxCommon: LocationType	0..1	The Location property characterizes the actual physical location of the Event. A simple location name may be specified or the underlying class may be extended, in which case, the default and strongly RECOMMENDED subclass is CIQAddress3.0InstanceType, as defined in *CybOX Version 2.1.1 Part 4: Default Extensions*.
Frequency	FrequencyType	0..1	The Frequency property characterizes the frequency of the Event.

3.1.2.3 CompositeEventType Class

The `CompositeEventType` class specifies a set of one or more other Events related to this Event. Notice that there is no defined relationship between the set of events, and no explicit relationship between the original Event and the others beyond simple associativity.

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

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

Name	Type	Multiplicity	Description
Event	`EventType`	1..*	The `Event` property specifies an Event asserted to be associated with the original Event.

3.1.2.4 EventDetailsType Interface

The `EventDetailsType` interface captures the ability to specify one Event or a hierarchy of Events. An `EventType` can be defined either in terms of Event properties, using the `EventPropertiesType` class, or by a set of other Events using the `CompositeEventType` class. The relationships represented in this hierarchy is not explicitly specified.

3.1.3 ObjectType Class

The `ObjectType` class characterizes a cyber-relevant Object (e.g., a file, a registry key or a process). The UML diagram corresponding to the `ObjectType` class is shown in **Figure 3-4**.

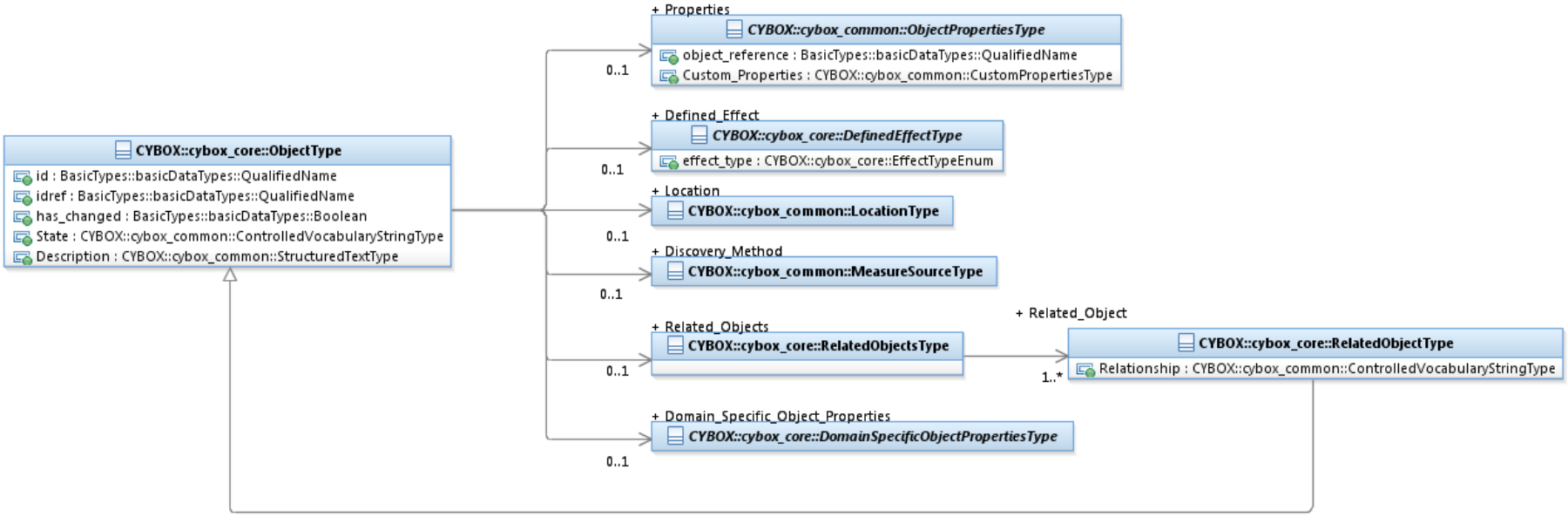

Figure 3-4. UML diagram for the `ObjectType` class

*The property table given in **Table 3-11** corresponds to the UML diagram shown in **Figure 3-4**.*

Table 3-11. Properties of the `ObjectType` class

Name	Type	Multiplicity	Description
id	`basicDataTypes:QualifiedName`	0..1	The `id` property specifies a globally unique identifier for the Object.
idref	`basicDataTypes:QualifiedName`	0..1	The `idref` property specifies a globally unique identifier for an Object specified elsewhere. When the `idref` property is used, the `id` property MUST NOT also be specified and the other properties of the `ObjectType` class SHOULD NOT hold any content.

has_changed	basicDataTypes:Boolean	0..1	The has_changed property specifies whether the Object has changed in some way. This property is NOT intended to be used for versioning of CybOX content.
State	cyboxCommon:VocabularyStringType	0..1	The State property specifies the state of the Object. Examples of potential states include *exists, inactive*, and *locked* (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 data type. The CybOX default vocabulary class for use in the property is '*ObjectStateVocab-1.0*'.
Description	cyboxCommon:StructuredTextType	0..1	The Description property captures a textual description of the Object. Any length is permitted. Optional formatting is supported via the structuring_format property of the StructuredTextType data type.
Properties	cyboxCommon:ObjectPropertiesType	0..1	The Properties property captures Object properties. CybOX defines a broad collection of Object data models that can be used to extend the cyboxCommon:ObjectPropertiesType superclass. Examples include the File Object data model, the Address Object data model, and the Network Packet data model.
Domain_Specific_ Object_Properties	DomainSpecific ObjectPropertiesType	0..1	The Domain_Specific_Object_Properties property captures domain specific Object properties, defined outside of the scope of the CybOX data model.
Location	cyboxCommon:LocationType	0..1	The Location property characterizes the actual physical location of the Object. A simple location name may be specified or the underlying abstract class may be extended, in which case, the default and strongly RECOMMENDED subclass is CIQAddress3.0InstanceType, as defined in *CybOX Version 2.1.1 Part 4: Default Extensions*.

Related_Objects	RelatedObjectsType	0..1	The Related_Objects property specifies a set of one or more Objects related to this Object.
Defined_Effect	DefinedEffectType	0..1	The Defined_Effect property characterizes the effect that an Action has on an Object. Examples include DataReadEffectType and StateChangeEffectType (see Section **3.4**).
Discovery_Method	cyboxCommon: MeasureSourceType	0..1	The Discovery_Method property characterizes how the Object was observed (in the case of a cyber observable Object instance) or could potentially be observed (in the case of a cyber observable Object pattern). Examples of details captured include identifying characteristics, time-related attributes, and a list of the tools used to collect the information.

3.1.3.1 DomainSpecificObjectPropertiesType Class

The DomainSpecificObjectPropertiesType class is of an abstract class placeholder enabling the inclusion of domain-specific metadata for an object through the use of a custom class, which MUST be defined as an extension of this base abstract class. This enables domains utilizing CybOX such as malware analysis or forensics to incorporate non-generalized object metadata from their domains into CybOX objects. It has no properties.

3.1.3.2 RelatedObjectType Class

The RelatedObjectType class characterizes a relationship between one Object and a related Object. It extends the ObjectType superclass by specifying the type of the relationship.

The UML diagram corresponding to the RelatedObjectType class is shown in **Figure 3-5**.

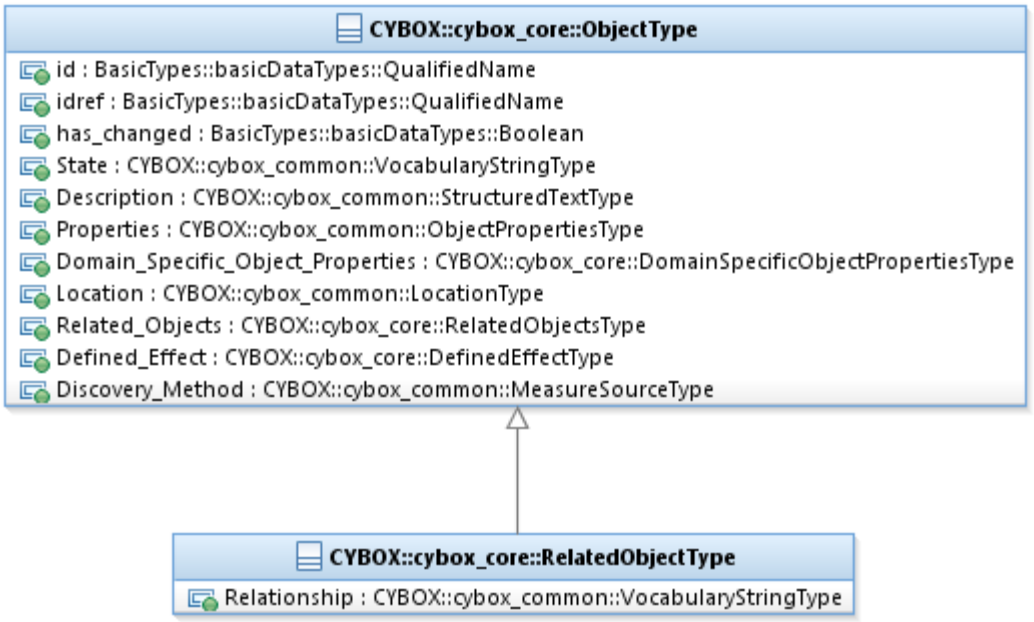

Figure 3-5. UML diagram of the `RelatedObjectType` class

The property table given in **Table 3-12** corresponds to the UML diagram shown in **Figure 3-5**.

Table 3-12. Properties of the `RelatedObjectType` class

Name	Type	Multiplicity	Description
Relationship	`cyboxCommon: VocabularyStringType`	0..1	The `Relationship` property specifies the type of relationship between two Objects. Examples of potential types include *created by, deleted by,* and *read from* (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` data type. The CybOX default vocabulary class for use in the property is '*ObjectRelationshipVocab-1.1*'.

3.1.4 ObservableType Class

The `ObservableType` class characterizes a cyber Observable. As shown in **Figure 3-6** and Error! Reference source not found., a CybOX Observable can either be on a CybOX Object with type corresponding to the CybOX `ObjectType` class (e.g., a File with name X), a CybOX Event with type corresponding to the CybOX `EventType` class (an Event is typically one or more actions taken against one or more Objects; e.g., "delete the File with name X"), or an Observable Composition with type corresponding to the CybOX `ObservableCompositionType` class.

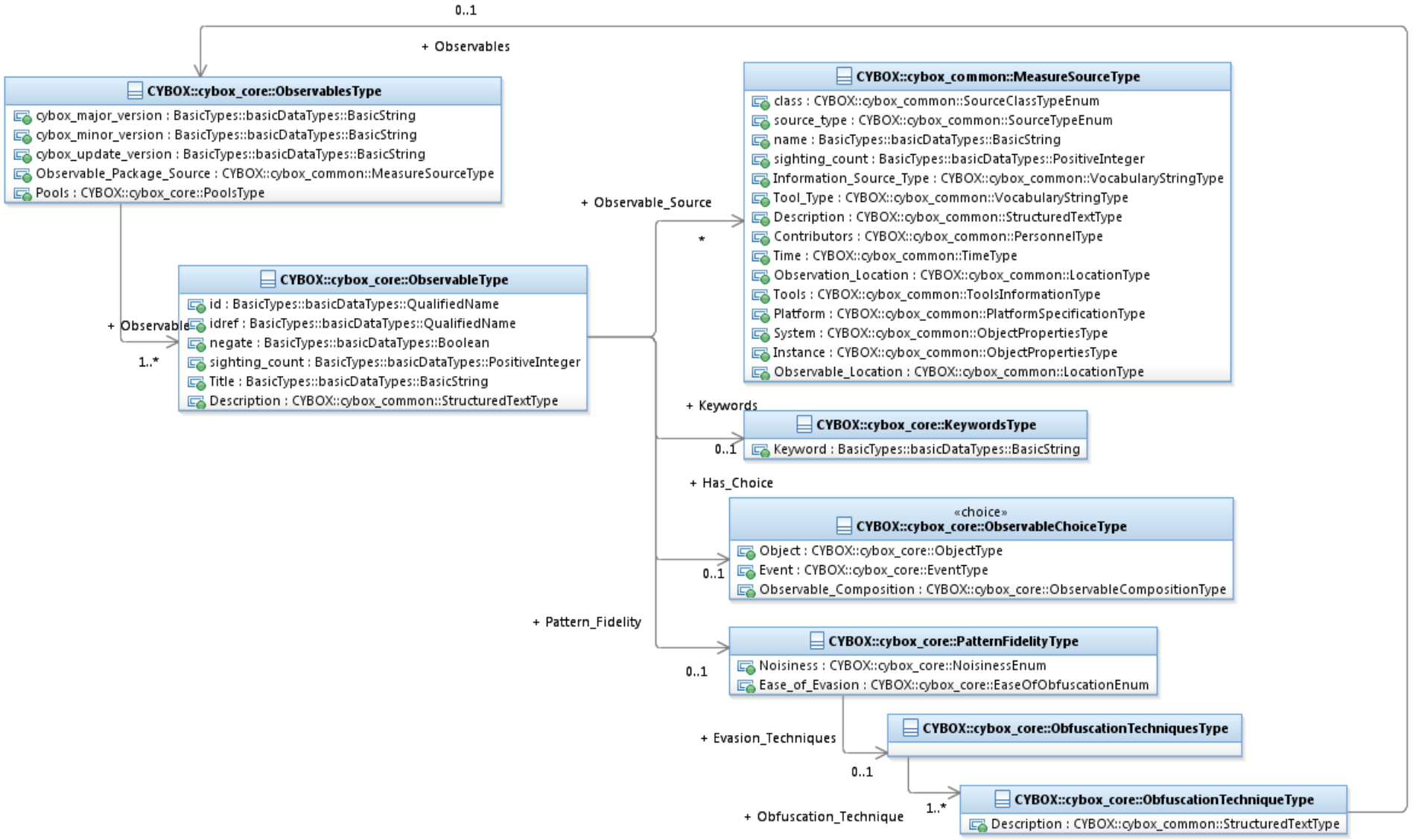

Figure 3-6. UML diagram for the `Observable` class

The property table given in **Table 3-13** corresponds to the UML diagram shown in **Figure 3-6**.

Table 3-13. Properties of the `ObservableType` class

Name	Type	Multiplicity	Description
id	`basicDataTypes:QualifiedName`	0..1	The `id` property specifies a globally unique identifier for the Observable.
idref	`basicDataTypes:QualifiedName`	0..1	The `idref` property specifies a globally unique identifier for an Observable specified elsewhere. When the `idref` property is used, the `id` property MUST NOT also be specified and the other properties of the `ObservableType` class SHOULD NOT hold any content.
negate	`basicDataTypes:Boolean`	0..1	The `negate` property, when set to TRUE, specifies the absence (rather than the presence) of the Observable in a CybOX pattern.
sighting_count	`basicDataTypes:PositiveInteger`	0..1	The `sighting_count` property specifies how many different (but identical) instances of the Observable have been observed.
Title	`basicDataTypes:BasicString`	0..1	The `Title` property captures a title for the Observable and reflects what the content producer thinks the Observable as a whole should be called. The `Title` property is typically used by humans to reference a particular Observable; however, it is not suggested for correlation.
Description	`cyboxCommon:StructuredTextType`	0..1	The `Description` property captures a textual description of the Observable. Any length is permitted. Optional formatting is supported via the `structuring_format` property of the `StructuredTextType` data type.

Keywords	KeywordsType	0..1	The Keywords property captures relevant keywords for the Observable.
Observable_Source	cyboxCommon: MeasureSourceType	0..*	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.
Observable_Expression	ObservableDetailsType	0..1	The Observable_Expression property characterizes an Observable details expression. ObservableDetailsType is a UML interface, which can be realized using either the ObjectType class, the EventType class or the ObservableCompositionType class (see **Section** Error! Reference source not found. for more details).
Pattern_Fidelity	PatternFidelityType	0..1	The Pattern_Fidelity property characterizes the fidelity of the Observable pattern.
Has_Choice	ObservableChoiceType	0..1	The Has_Choice property is associated with the class ObservableChoiceType. It indicates that there is a choice between the IP_Address property or the Hostname property. Only one of the properties of ObservableChoiceType class can be populated at any time. See Section **1.2.3** for more detail.

The ObservableChoiceType class is the type of the Has_Choice property. In the UML model, this class is associated with the <<choice>> UML stereotype, which specifies that only one of the available properties of the ObservableChoiceType class can be populated at any time. The property table of the ObservableChoiceType class is given in **Table 3-14**.

Table 3-14, Properties of the `ObservableChoiceType` class

Name	Type	Multiplicity	Description
Object	`ObjectType`	0..1	The `Object` property characterizes a cyber-relevant Object (e.g., a file, a registry key or a process)
Event	`EventType`	0..1	The `Event` property characterizes a cyber Event.
Observable_Composition	`ObservableCompositionType`	0..1	The `Observable_Composition` property specifies a composite Observable pattern expression through the specification of a single Boolean operator (the operator of the expression) and a list of simple (non-composition) Observable patterns (the operands of the expression).

3.1.4.1 ObservableCompositionType Class

The `ObservableCompositionType` class enables a content creator to define a composite Observable pattern expression through the specification of a single Boolean operator (the operator of the expression) and a list of simple (non-composition) Observable patterns (the operands of the expression).

More complex Observable compositions (of the `ObservableCompositionType` class) can be created using multiple simple Observable patterns and/or other Observable compositions. For example, it may be desired to express the Observable Composition, $OC_X = O_A$ AND (E_A OR E_B). The Observable (O_A) and each of the two Events (E_A and E_B) would be created individually, and then one Observable Composition would be defined as $OC_A = E_A$ OR E_B. This permits OC_X to be rewritten as $OC_X = O_A$ AND OC_A. Note that this example shows just one or several possible constructions that generate OC_X.

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

Name	Type	Multiplicity	Description
operator	`OperatorTypeEnum`	0..1	The `operator` property specifies the logical operator of the Observable.
Observable	`ObservableType`	0..*	The `Observable` property characterizes a cyber Observable.

3.1.4.2 PatternFidelityType Class

The `PatternFidelityType` class characterizes the fidelity of an Observable pattern.

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

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

Name	Type	Multiplicity	Description
Noisiness	`NoisinessEnum`	0..1	The `Noisiness` property specifies the noisiness of the Observable (i.e., the false positives of the Observable).
Ease_of_Evasion	`EaseOfObfuscationEnum`	0..1	The `Ease_of_Evasion` property specifies the difficulty a threat actor may have obfuscating the Observable.
Evasion_Techniques	`ObfuscationTechniquesType`	0..1	The `Evasion_Techniques` property specifies a set of one or more potential techniques an attacker could leverage to obfuscate an Observable.

3.1.4.3 ObfuscationTechniqueType Class

The `ObfuscationTechniqueType` class characterizes a technique an attacker could potentially leverage to obfuscate the observability of the Observable.

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

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

Name	Type	Multiplicity	Description
Description	`cyboxCommon: StructuredTextType`	1	The `Description` property captures a textual description of the obfuscation technique. Any length is permitted. Optional formatting is supported via the `structuring_format` property of the `StructuredTextType` data type.
Observables	`ObservablesType`	0..1	The `Observables` property captures potential cyber Observables that could indicate the use of the obfuscation technique.

3.1.5 FrequencyType Class

The `FrequencyType` class characterizes the frequency of a given Action or Event.

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

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

Name	Type	Multiplicity	Description
rate	`basicDataTypes:Decimal`	0..1	The `rate` property specifies the rate of the frequency.
units	`basicDataTypes:BasicString`	0..1	The `units` property specifies the units of the frequency.
scale	`basicDataTypes:BasicString`	0..1	The `scale` property specifies the time scale of the frequency.
trend	`TrendEnum`	0..1	The `trend` property specifies the trend of the frequency. This property could be used as a trigger within an Event or Action pattern observable.

3.2 Content Aggregation Classes

A content aggregation class captures a collection of one or more CybOX objects.

3.2.1 ActionAliasesType Class

The `ActionAliasesType` class specifies a set of one or more names of an Action.

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

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

Name	Type	Multiplicity	Description
Action_Alias	`basicDataTypes:BasicString`	1..*	The `Action_Alias` property specifies an alias name for the Action.

3.2.2 ActionArgumentsType Class

The `ActionArgumentsType` class specifies a set of one or more arguments or parameters relevant to an Action.

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

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

Name	Type	Multiplicity	Description
Action_Argument	`ActionArgumentType`	1..*	The `Action_Argument` property characterizes an argument or parameter relevant to the Action.

3.2.3 ActionPertinentObjectPropertiesType Class

The `ActionPertinentObjectPropertiesType` class specifies a set of one or more Object properties pertinent to an Action.

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

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

Name	Type	Multiplicity	Description
Property	`ActionPertinentObjectPropertyType`	1..*	The `Property` property characterizes an Object property pertinent to the Action.

3.2.4 ActionRelationshipsType Class

The `ActionRelationshipsType` specifies a set of one or more relationships between one Action and other Actions.

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

Table 3-22. Properties of the ActionRelationshipsType class

Name	Type	Multiplicity	Description
Relationship	ActionRelationshipType	1..*	The Relationship property characterizes a relationship between the Action and another, related Action.

3.2.5 ActionsType Class

The ActionsType class specifies a set of one or more cyber observable Actions.

The properties of the ActionsType class are given in **Table 3-23**.

Table 3-23. Properties of the ActionsType class

Name	Type	Multiplicity	Description
Action	ActionType	1..*	The Action property characterizes a cyber observable Action.

3.2.6 AssociatedObjectsType Class

The AssociatedObjectsType class specifies a set of one or more cyber Objects relevant to an Action.

The properties of the AssociatedObjectsType class are given in **Table 3-24**.

Table 3-24. Properties of the AssociatedObjectsType class

Name	Type	Multiplicity	Description
Associated_Object	AssociatedObjectType	1..*	The Associated_Object property characterizes a cyber Object associated with the Action. An associated Object may be an Object that initiated the Action, an Object that is affected by or utilized by the Action, or an Object that is returned as a result of the Action.

3.2.7 KeywordsType Class

The `KeywordsType` class specifies a set of one or more keywords.

The properties of the `KeywordsType` class are given in **Table 3-25**.

Table 3-25. Properties of the `KeywordsType` class

Name	Type	Multiplicity	Description
Keyword	`basicDataTypes:BasicString`	1..*	The `Keyword` property captures a keyword.

3.2.8 ObfuscationTechniquesType Class

The `ObfuscationTechniquesType` class specifies a set of one or more potential techniques an attacker could leverage to obfuscate an Observable.

The properties of the `ObfuscationTechniquesType` class are given in **Table 3-26**.

Table 3-26. Properties of the `ObfuscationTechniquesType` class

Name	Type	Multiplicity	Description
Obfuscation_Technique	`ObfuscationTechniqueType`	1..*	The `Obfuscation_Technique` property characterizes a technique an attacker could potentially leverage to obfuscate the Observable.

3.2.9 ObservablesType Class

The `ObservablesType` class characterizes a set of one or more cyber Observables.

The properties of the `ObservablesType` class are given in **Table 3-27**.

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

Name	Type	Multiplicity	Description

cybox_major_version	basicDataTypes: BasicString	0..1	The cybox_major_version property specifies the major version of the CybOX language used for the set of Observables.
cybox_minor_version	basicDataTypes: BasicString	0..1	The cybox_minor_version property specifies the minor version of the CybOX language used for the set of Observables.
cybox_update_version	basicDataTypes: BasicString	0..1	The cybox_update_version property specifies the update version of the CybOX language used for this set of Observables. This property MUST be used when using an update version of CybOX.
Observable_Package_Source	cyboxCommon: MeasureSourceType	0..1	The Observable_Package_Source property characterizes the source of the Observables information. Examples of details captured include identifying characteristics, time-related attributes, and a list of the tools used to collect the information.
Observable	ObservableType	1..*	The Observable property characterizes a cyber Observable.
Pools	PoolsType	0..1	The Pools property captures the Events, Actions, Objects and Properties (in a space-efficient, pooled manner) that are referenced by the cyber Observable.

3.2.10 PropertiesType Class

The PropertiesType class specifies a set of one or more properties enumerated as a result of the effect of the Action on the Object.

The properties of the PropertiesType class are given in **Table 3-28**.

Table 3-28. Properties of the PropertiesType class

Name	Type	Multiplicity	Description
Property	basicDataTypes: BasicString	1..*	The Property property specifies a property resulting from an Action on an

			Object.

3.2.11 RelatedObjectsType Class

The `RelatedObjectsType` class specifies a set of one or more relationships between one Object and other Objects.

The properties of the `RelatedObjectsType` class are given in **Table 3-29**.

Table 3-29. Properties of the `RelatedObjectsType` class

Name	Type	Multiplicity	Description
Related_Object	`RelatedObjectType`	1..*	The `Related_Object` property specifies an Object related to the Object of focus and characterizes the relationship between the Objects.

3.2.12 ValuesType Class

The `ValuesType` class specifies a set of one or more values that are enumerated as a result of an Action on an Object.

The properties of the `ValuesType` class are given in **Table 3-30**.

Table 3-30. Properties of the `ValuesType` class

Name	Type	Multiplicity	Description
Value	`basicDataTypes:BasicString`	1..*	The `Value` property specifies a single value that is enumerated as a result of the Action on the Object.

3.3 Pool Classes

Pool classes enable observable elements – Events, Actions, Objects, and Properties – to be described in a space-efficient manner. Rather than defining identical observable elements multiple times within a set of defined Observables, observable elements are defined in type-specific pools (i.e., sets) and are then referenced by Observable structures.

3.3.1 PoolsType Class

The `PoolsType` class captures one or more pools, each of which contains one type of observable element.

The UML diagram corresponding to the `PoolsType` class is shown in **Figure 3-7**.

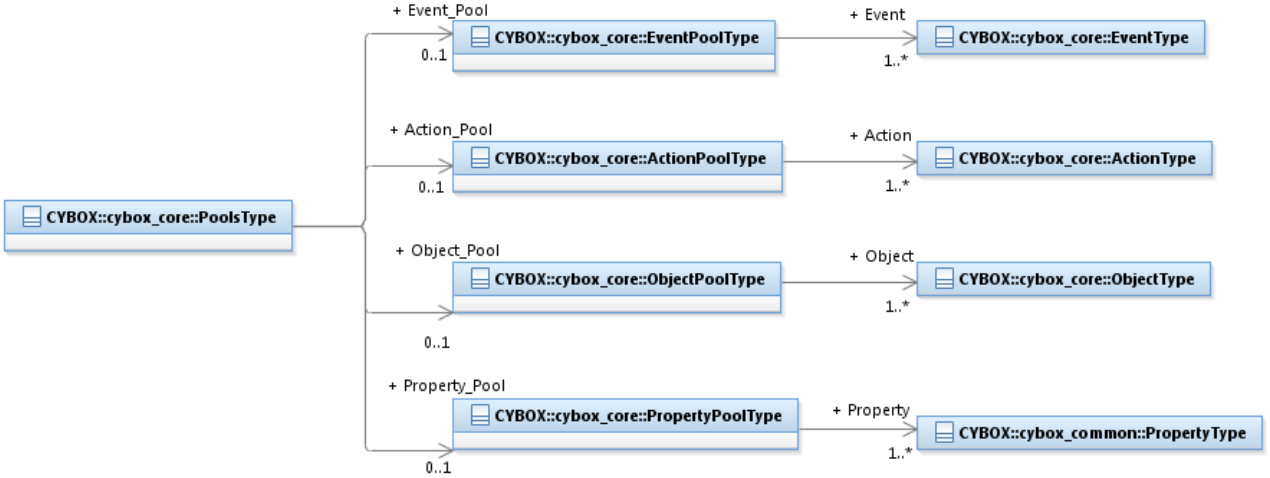

Figure 3-7. UML diagram of the `PoolsType` class

The property table given in **Table 3-31** corresponds to the UML diagram shown in **Figure 3-7**.

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

Name	Type	Multiplicity	Description
Event_Pool	`EventPoolType`	0..1	The `Event_Pool` property specifies a pool of one or more Events.
Action_Pool	`ActionPoolType`	0..1	The `Action_Pool` property specifies a pool of one or more Actions.
Object_Pool	`ObjectPoolType`	0..1	The `Object_Pool` property specifies a pool of one or more Objects.
Property_Pool	`PropertyPoolType`	0..1	The `Property_Pool` property specifies a pool of one or more Properties.

3.3.2 EventPoolType Class

The `EventPoolType` class specifies a pool of one or more Events.

The properties of the `EventPoolType` class are given in **Table 3-32**.

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

Name	Type	Multiplicity	Description
Event	`EventType`	1..*	The `Event` property characterizes a cyber observable Event.

3.3.3 ActionPoolType Class

The `ActionPoolType` class specifies a pool of one or more Actions.

The properties of the `ActionPoolType` class are given in **Table 3-33.**

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

Name	Type	Multiplicity	Description
Action	`ActionType`	1..*	The `Action` property characterizes a cyber observable Action.

3.3.4 ObjectPoolType Class

The `ObjectPoolType` class specifies a pool of one or more Objects.

The properties of the `ObjectPoolType` class are given in **Table 3-34**.

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

Name	Type	Multiplicity	Description
Object	`ObjectType`	1..*	The `Object` property characterizes a cyber-relevant Object.

3.3.5 PropertyPoolType Class

The `PropertyPoolType` class specifies a pool of one or more Properties.

The properties of the `PropertyPoolType` class are given in **Table 3-35**.

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

Name	Type	Multiplicity	Description
Property	`cyboxCommon:PropertyType`	1..*	The `Property` property characterizes an Object property.

3.4 Defined Effect Classes

The CybOX Core data model defines a number of classes to characterize a broad range of potential effects of an Action on an Object. Characterization is achieved through specialization of the `DefinedEffectType` abstract class, which is defined in Section **3.4.1**. The defined effect-type classes that specialize the `DefinedEffectType` class are presented in Sections **3.4.2** through **3.4.10**, which corresponds to the order that they are listed in the `EffectTypeEnum` enumeration (Section **3.5.4**).

3.4.1 DefinedEffectType Class

The `DefinedEffectType` class specifies the type of the effect that an Action has on an Object. It is an abstract class, and it MUST be extended via a subclass to specify a complete effect. Use of the `DefinedEffectType` class enables a broad range of complex Action effects on Objects to be specified.

A UML diagram corresponding to the `DefinedEffectType` class is shown in **Figure 3-8**.

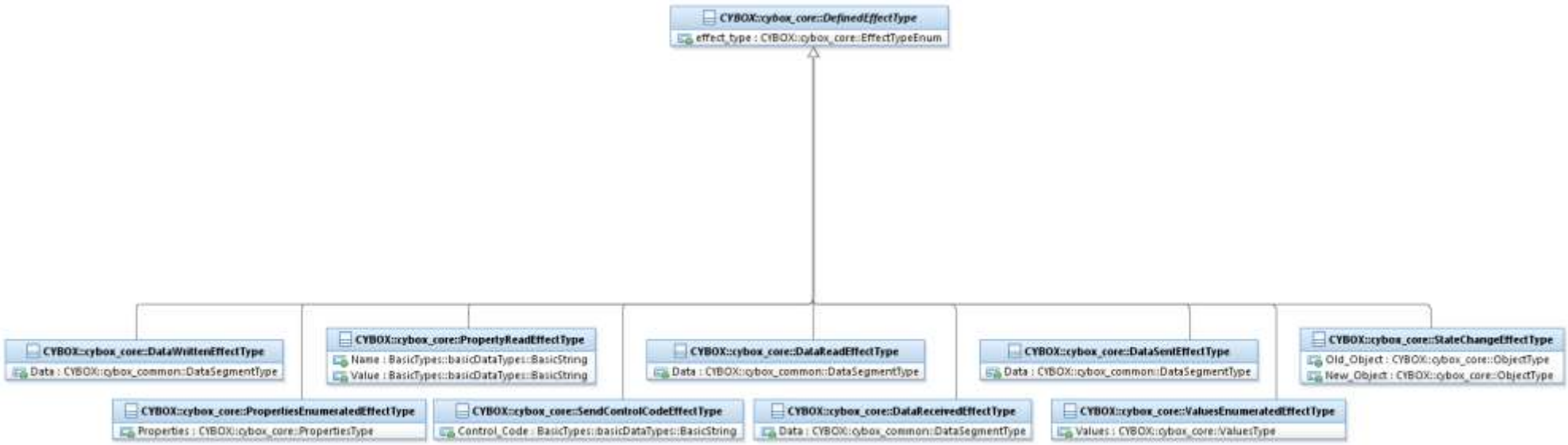

Figure 3-8. UML diagram of the `DefinedEffectType` class

The property table given in **Table 3-36** corresponds to the UML diagram shown in **Figure 3-8**.

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

Name	Type	Multiplicity	Description
effect_type	`EffectTypeEnum`	0..1	The `effect_type` property specifies the effect of the Action on the Object.

3.4.2 StateChangeEffectType Class

The `StateChangeEffectType` class extends the `DefinedEffectType` superclass by characterizing the effects of Actions upon Objects where the state of the Object is changed.

The properties of the `StateChangeEffectType` specialization are given in **Table 3-37**.

Table 3-37. Properties of the StateChangeEffectType class

Name	Type	Multiplicity	Description
Old_Object	ObjectType	0..1	The Old_Object property characterizes the Object before the state change effect occurred.
New_Object	ObjectType	1	The New_Object property characterizes the Object after the state change effect occurred.

3.4.3 DataReadEffectType Class

The DataReadEffectType class extends the DefinedEffectType superclass by characterizing the effects of Actions upon Objects where some data is read, such as from a file or a pipe.

The properties of the DataReadEffectType specialization are given in **Table 3-38**.

Table 3-38. Properties of the DataReadEffectType class

Name	Type	Multiplicity	Description
Data	cyboxCommon:DataSegmentType	1	The Data property characterizes the data that was read from the Object by the Action.

3.4.4 DataWrittenEffectType Class

The DataWrittenEffectType class extends the DefinedEffectType superclass by characterizing the effects of Actions upon Objects where some data is written, such as to a file or a pipe.

The properties of the DataWrittenEffectType specialization are given in **Table 3-39**.

Table 3-39. Properties of the DataWrittenEffectType class

Name	Type	Multiplicity	Description
Data	cyboxCommon:DataSegmentType	1	The Data property characterizes the data that was written to the Object by the Action.

3.4.5 DataSentEffectType Class

The `DataSentEffectType` class extends the `DefinedEffectType` superclass by characterizing the effects of Actions upon Objects where some data is sent, such as a byte sequence on a socket.

The properties of the `DataSentEffectType` specialization are given in **Table 3-40**.

Table 3-40. Properties of the `DataSentEffectType` class

Name	Type	Multiplicity	Description
Data	`cyboxCommon:DataSegmentType`	1	The `Data` property characterizes the data that was sent on the Object, or from the Object, by the Action.

3.4.6 DataReceivedEffectType Class

The `DataReceivedEffectType` class extends the `DefinedEffectType` superclass by characterizing the effects of Actions upon Objects where some data is received, such as a byte sequence on a socket.

The properties of the `DataReceivedEffectType` specialization are given in **Table 3-41**.

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

Name	Type	Multiplicity	Description
Data	`cyboxCommon:DataSegmentType`	1	The `Data` property characterizes the data that was received on the Object, or from the Object, by the Action.

3.4.7 PropertyReadEffectType Class

The `PropertyReadEffectType` class extends the `DefinedEffectType` superclass by characterizing the effects of Actions upon Objects where some specific property is read from an Object, such as the current running state of a process.

The properties of the `PropertyReadEffectType` specialization are given in **Table 3-42**.

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

Name	Type	Multiplicity	Description

Name	basicDataTypes:BasicString	0..1	The Name property specifies the name of the property being read.
Value	basicDataTypes:BasicString	0..1	The Value property specifies the value of the property being read.

3.4.8 PropertiesEnumeratedEffectType Class

The PropertiesEnumeratedEffectType class extends the DefinedEffectType superclass by characterizing the effects of Actions upon Objects where some properties of the Object are enumerated, such as the startup parameters for a process.

The properties of the PropertiesEnumeratedEffectType specialization are given in **Table 3-43**.

Table 3-43. Properties of the PropertiesEnumeratedEffectType class

Name	Type	Multiplicity	Description
Properties	PropertiesType	1	The Properties property specifies a set of one or more values that are enumerated as a result of an Action on an Object.

3.4.9 ValuesEnumeratedEffectType Class

The ValuesEnumeratedEffectType class extends the DefinedEffectType superclass by characterizing the effects of Actions upon Objects where some values of the Object are enumerated, such as the values of a registry key.

The properties of the ValuesEnumeratedEffectType specialization are given in **Table 3-44**.

Table 3-44. Properties of the ValuesEnumeratedEffectType class

Name	Type	Multiplicity	Description
Values	ValuesType	1	The Values property specifies a set of one or more values that are enumerated as a result of an Action on an Object.

3.4.10 SendControlCodeEffectType Class

The SendControlCodeEffectType class extends the DefinedEffectType superclass by characterizing the effects of Actions upon Objects where some control code, or other control-oriented communication signal, is sent to the Object.

The properties of the SendControlCodeEffectType specialization are given in **Table 3-45**.

Table 3-45. Properties of the SendControlCodeEffectType class

Name	Type	Multiplicity	Description
Control_Code	basicDataTypes:BasicString	1	The Control_Code property specifies the actual control code sent to the Object.

3.5 Enumerations

3.5.1 ActionStatusTypeEnum Enumeration

The ActionStatusTypeEnum enumeration is an inventory of Action statuses. The enumeration literals are given in **Table 3-46**.

Table 3-46. Literals of the ActionStatusTypeEnum enumeration

Enumeration Literal	Description
Success	The cyber observable Action was successful.
Fail	The cyber observable Action failed.
Error	The cyber observable Action resulted in an error.
Complete/Finish	The cyber observable Action completed or finished. This action status does not specify the result of the Action (e.g., Success, Error).
Pending	The cyber observable Action is pending.
Ongoing	The cyber observable Action is ongoing.
Unknown	The cyber observable Action has unknown status.

3.5.2 ActionContextTypeEnum Enumeration

The `ActionContextTypeEnum` enumeration is an inventory of Action contexts. The enumeration literals are given in **Table 3-47**.

Table 3-47. Literals of the `ActionContextTypeEnum` enumeration

Enumeration Literal	Description
Host	The cyber observable Action occurred on a host.
Network	The cyber observable Action occurred on a network.

3.5.3 EaseOfObfuscationEnum Enumeration

The `EaseOfObfuscationEnum` enumeration is an inventory of values representing the difficulty a threat actor may have obfuscating an Observable. The enumeration literals are given in **Table 3-48**.

Table 3-48. Literals of the `EaseOfObfuscationEnum` enumeration

Enumeration Literal	Description
High	The Observable is very easy to obfuscate and hide.
Medium	The Observable is somewhat easy to obfuscate and hide.
Low	The Observable is not very easy to obfuscate and hide.

3.5.4 EffectTypeEnum Enumeration

The `EffectTypeEnum` enumeration is an inventory of values representing the effect of an Action on an Object. The enumeration literals are given in **Table 3-49**.

Table 3-49. Literals of the `EffectTypeEnum` enumeration

Enumeration Literal	Description
State_Changed	The Action changed the state of the Object.

Data_Read	The Action read data from the Object.
Data_Written	The Action wrote data to the Object.
Data_Sent	The Action sent data to the Object.
Data_Received	The Action received data from the Object.
Properties_Read	The Action read properties of the Object.
Properties_Enumerated	The Action enumerated properties of the Object.
Values_Enumerated	The Action enumerated values of the Object.
ControlCode_Sent	The Action sent control code to the Object.

3.5.5 NoisinessEnum Enumeration

The `NoisinessEnum` enumeration is an inventory of values for representing the noisiness of the Observable (i.e., false positives of the Observable). The enumeration literals are given in **Table 3-50**.

Table 3-50. Literals of the `NoisinessEnum` enumeration

Enumeration Literal	Description
High	The Observable has a high level of noisiness (i.e., a potentially high level of false positives).
Medium	The Observable has a medium level of noisiness (i.e., a potentially medium level of false positives).
Low	The Observable has a low level of noisiness (i.e., a potentially low level of false positives).

3.5.6 OperatorTypeEnum Enumeration

The `OperatorTypeEnum` enumeration is an inventory of valid operator types. The enumeration literals are given in **Table 3-51**.

Table 3-51. Literals of the `OperatorTypeEnum` enumeration

Enumeration Literal	Description
AND	Logical AND operator
OR	Logical OR operator

3.5.7 TrendEnum Enumeration

The `TrendEnum` enumeration is an inventory of valid trend types. The enumeration literals are given in **Table 3-52**.

Table 3-52. Literals of the `TrendEnum` enumeration

Enumeration Literal	Description
Increasing	The trend is increasing.
Decreasing	The trend is decreasing.

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
Fujitsu Limited
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
Andrei Sîrghi

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
John Wunder

National Council of ISACs (NCI)
Scott Algeier
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

National Security Agency
Mike Boyle
Jessica Fitzgerald-McKay

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
David Waters
Benjamin Yates

Symantec Corp.
Curtis Kostrosky

The Boeing Company

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

United Kingdom Cabinet Office
Iain Brown
Adam Cooper
Mike McLellan
Chris O'Brien

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
James Penman
Howard Staple
Chris Taylor
Laurie Thomson
Alastair Treharne
Julian White
Bethany Yates
US Department of Homeland Security
Evette Maynard-Noel
Justin Stekervetz
ViaSat, Inc.
Lee Chieffalo
Wilson Figueroa
Andrew May
Yaana Technologies, LLC
Anthony Rutkowski

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

Appendix B. Revision History

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

[1] The property name `xpath` is a remnant from the STIX 1.2 XML schema.