THE MITRE CORPORATION

CybOX™ 2.1

Core Specification (v2.1)

July 6, 2015

*The Cyber Observable eXpression (CybOXTM) 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.*

**Acknowledgements**

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

**Trademark Information**

CybOX, the CybOX logo, and STIX are trademarks of The MITRE Corporation. All other trademarks are the property of their respective owners.

**Warnings**

MITRE PROVIDES CYBOX "AS IS" AND MAKES NO WARRANTY, EXPRESS OR IMPLIED, AS TO THE ACCURACY, CAPABILITY, EFFICIENCY, MERCHANTABILITY, OR FUNCTIONING OF CYBOX. IN NO EVENT WILL MITRE BE LIABLE FOR ANY GENERAL, CONSEQUENTIAL, INDIRECT, INCIDENTAL, EXEMPLARY, OR SPECIAL DAMAGES, RELATED TO CYBOX OR ANY DERIVATIVE THEREOF, WHETHER SUCH CLAIM IS BASED ON WARRANTY, CONTRACT, OR TORT, EVEN IF MITRE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.[[1]](#footnote-1)

**Feedback**

The CybOX development team welcomes any feedback regarding this document. Please send any comments, questions, or suggestions to cybox@mitre.org.[[2]](#footnote-2)

Table of Contents

[1 Introduction 1](#_Toc423855163)

[1.1 CybOX Specification Documents 1](#_Toc423855164)

[1.2 Document Conventions 1](#_Toc423855165)

[1.2.1 Keywords 2](#_Toc423855166)

[1.2.2 Fonts 2](#_Toc423855167)

[1.2.3 UML Package References 2](#_Toc423855168)

[1.2.4 UML Diagrams 2](#_Toc423855169)

[1.2.4.1 Class Properties 3](#_Toc423855170)

[1.2.4.2 Diagram Icons and Arrow Types 3](#_Toc423855171)

[1.2.4.3 Color Coding 4](#_Toc423855172)

[1.2.5 Property Table Notation 4](#_Toc423855173)

[1.2.6 Property and Class Descriptions 4](#_Toc423855174)

[2 Background Information 6](#_Toc423855175)

[2.1 Cyber Observables 6](#_Toc423855176)

[3 CybOX Core Data Model 7](#_Toc423855177)

[3.1 Primary Classes 7](#_Toc423855178)

[3.1.1 ActionType Class 7](#_Toc423855179)

[3.1.2 EventType Class 10](#_Toc423855180)

[3.1.3 ObjectType Class 12](#_Toc423855181)

[3.1.4 ObservableType Class 14](#_Toc423855182)

[3.2 Secondary Classes 16](#_Toc423855183)

[3.2.1 ActionArgumentType Class 17](#_Toc423855184)

[3.2.2 ActionPertinentObjectPropertyType Class 17](#_Toc423855185)

[3.2.3 ActionReferenceType Class 18](#_Toc423855186)

[3.2.4 ActionRelationshipType Class 18](#_Toc423855187)

[3.2.5 AssociatedObjectType Class 19](#_Toc423855188)

[3.2.6 FrequencyType Class 20](#_Toc423855189)

[3.2.7 ObfuscationTechniqueType Class 20](#_Toc423855190)

[3.2.8 ObservableCompositionType Class 21](#_Toc423855191)

[3.2.9 PatternFidelityType Class 22](#_Toc423855192)

[3.2.10 RelatedObjectType Class 22](#_Toc423855193)

[3.3 Content Aggregation Classes 23](#_Toc423855194)

[3.3.1 ActionAliasesType Class 23](#_Toc423855195)

[3.3.2 ActionArgumentsType Class 24](#_Toc423855196)

[3.3.3 ActionPertinentObjectPropertiesType Class 24](#_Toc423855197)

[3.3.4 ActionRelationshipsType Class 24](#_Toc423855198)

[3.3.5 ActionsType Class 25](#_Toc423855199)

[3.3.6 AssociatedObjectsType Class 25](#_Toc423855200)

[3.3.7 DomainSpecificObjectPropertiesType Class 26](#_Toc423855201)

[3.3.8 KeywordsType Class 26](#_Toc423855202)

[3.3.9 ObfuscationTechniquesType Class 26](#_Toc423855203)

[3.3.10 ObservablesType Class 27](#_Toc423855204)

[3.3.11 PropertiesType Class 28](#_Toc423855205)

[3.3.12 RelatedObjectsType Class 29](#_Toc423855206)

[3.3.13 ValuesType Class 29](#_Toc423855207)

[3.4 Pool Classes 30](#_Toc423855208)

[3.4.1 PoolsType Class 30](#_Toc423855209)

[3.4.2 EventPoolType Class 31](#_Toc423855210)

[3.4.3 ActionPoolType Class 32](#_Toc423855211)

[3.4.4 ObjectPoolType Class 32](#_Toc423855212)

[3.4.5 PropertyPoolType Class 33](#_Toc423855213)

[3.5 Defined Effect Classes 33](#_Toc423855214)

[3.5.1 DefinedEffectType Class 33](#_Toc423855215)

[3.5.2 StateChangeEffectType Class 34](#_Toc423855216)

[3.5.3 DataReadEffectType Class 34](#_Toc423855217)

[3.5.4 DataWrittenEffectType Class 35](#_Toc423855218)

[3.5.5 DataSentEffectType Class 35](#_Toc423855219)

[3.5.6 DataReceivedEffectType Class 35](#_Toc423855220)

[3.5.7 PropertyReadEffectType Class 36](#_Toc423855221)

[3.5.8 PropertiesEnumeratedEffectType Class 36](#_Toc423855222)

[3.5.9 ValuesEnumeratedEffectType Class 37](#_Toc423855223)

[3.5.10 SendControlCodeEffectType Class 37](#_Toc423855224)

[3.6 Enumerations 38](#_Toc423855225)

[3.6.1 ActionStatusTypeEnum Enumeration 38](#_Toc423855226)

[3.6.2 ActionContextTypeEnum Enumeration 38](#_Toc423855227)

[3.6.3 EaseOfObfuscationEnum Enumeration 39](#_Toc423855228)

[3.6.4 EffectTypeEnum Enumeration 39](#_Toc423855229)

[3.6.5 NoisinessEnum Enumeration 40](#_Toc423855230)

[3.6.6 OperatorTypeEnum Enumeration 41](#_Toc423855231)

[3.6.7 TrendEnum Enumeration 41](#_Toc423855232)

[References 42](#_Toc423855233)

# 

# Introduction

The Cyber Observable eXpression (CybOXTM) 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 1.2 data model, which is one of two fundamental data models for CybOX content.

In Section 1.1 we discuss CybOX specification documents, and in Section 1.2 we give document conventions. In Section 2, we give background information necessary to fully understand the Core data model, and we present the Core data model specification details in Section 3. References are provided in the final section.

## 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 key 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 specification overview document provides a comprehensive overview of the full set of CybOX data models [CybOXO], which in addition to the Core, Common, and numerous Object data models, includes a set of default controlled vocabularies. [CybOXO] also summarizes the relationship of CybOX to other languages, and outlines general CybOX data model conventions.

All specification documents can be found on this CybOX Website [CybOX-SPC].

## Document Conventions

The following conventions are used in this document.

### Keywords

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 *RFC 2119* [RFC2119].

### 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 the CybOX Specification Overview [CybOXO].

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 (withsingle 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*

### UML Package References

Each CybOX data model is captured in a different UML package (e.g., Core package) where the packages together compose the full CybOX UML model. To refer to a particular class of a specific package, we use the format package\_prefix:class, where package\_prefix corresponds to the appropriate UML package. The CybOXTM 2.1 Specification Overview document [CybOXO] 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.

### 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.

#### 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.

#### 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** |
| cid:image003.png@01CFC8F2.37AD2A50 | 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. |

#### Color Coding

The shapes of the UML diagrams are color coded to indicate the data model associated with a class.  The colors used in the Core specification are illustrated via exemplars in Figure 1‑2.

[need diagram]

Figure 1‑2. Data model color coding

### 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 1.2.3).

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.

In addition, properties that are part of a “choice” relationship (e.g., Prop1 OR Prop2 is used but not both) will be denoted by a unique letter subscript (e.g., API\_CallA, CodeB) and single logic expression in the Multiplicity column.  For example, if there is a choice of property API\_CallA and CodeB, the expression “A(1)|B(0..1)” will indicate that the API\_Call property can be chosen with multiplicity 1 or the Code property can be chosen with multiplicity 0 or 1.

### Property and Class Descriptions

Each class and property defined in CybOX is described using the format, “The X property verbY.” 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 chosen 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 Source property characterizes the source of the sighting information. Examples of details captured include identitifying characteristics, time-related attributes, and a list of the tools used to collect the information.  The Description property captures a textual description of the Indicator. |
| 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. |
|  | *Example*:  The Confidence property characterizes the level of confidence in the accuracy of the overall content captured in the Incident.  The ActivityType class characterizes basic information about an activity a defender might use in response to a Campaign. |
| 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 version property specifies the version identifier of the CybOX Core data model used to capture the information associated with… |

# Background Information

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

## 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 and a cyber indicator are not the same. A cyber observable is a statement of fact, capturing what was observed or could be observed in the cyber operational domain. Cyber indiators 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.

# CybOX Core Data Model

The CybOX Core data model defines a variety of classes. For discussion purposes, we have separated the classes into six categories (Sections 3.1 through 3.5), and within each category, we primarily define the classes in alphabetical order below, except for the cases when one class (a superclass) is specialized by other classes, in which case the superclass is defined first (and the other classes are either listed alphabetically or in another order as explained). We list enumerations in Section 3.6.

## Primary Classes

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

### ActionType Class

The ActionType is a complex type representing a single cyber observable action.

**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 and identifier reference to a Action instance specified elsewhere. When the idref property is used, no other property should be specified. |
| **ordinal\_position** | basicDataTypes:  PositiveInteger | 0..1 | The ordinal\_position property is intended to reference the ordinal position of the action with within a series of actions. |
| **action\_status** | ActionStatusTypeEnum | 0..1 | The action\_status property enables description of the status of the action being described. |
| **context** | ActionContextTypeEnum | 0..1 | The context property is optional and enables simple characterization of the broad operational context in which the Action is relevant. |
| **timestamp** | basicDataTypes:  DateTime | 0..1 | The timestamp property represents the local or relative time at which the action occurred or was observed. 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. |
| **timestamp\_precision** | cyboxCommon:  DateTimePrecisionEnum | 0..1 | Represents the precision of the associated timestamp value. If omitted, the default is "second", meaning the timestamp is precise to the full property value. Digits in the timestamp that are required by the xs:dateTime datatype but 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 stixCommon:ControlledVocabularyStringType class. 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 resgistry 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 stixCommon:ControlledVocabularyStringType class. The CybOX default vocabulary class for use in the property is ‘*ActionNameVocab-1.1*’. |
| **Description** | cyboxCommon:  StructuredTextType | 0..1 | The Description property captures a technical description of the Action. Any length is permitted. Optional formatting is supported via the structuring\_format property of the StructuredTextType class. |
| **Action\_Aliases** | ActionAliasesType | 0..1 | The Action\_Aliases property is optional and enables identification of other potentially used names for this Action. |
| **Action\_Arguments** | ActionArgumentsType | 0..1 | The Action\_Arguments property is optional and enables the specification of relevant arguments/parameters for this Action. |
| **Location** | cyboxCommon:  LocationType | 0..1 | The Location property specifies a relevant physical location. This field is implemented through the xsi:type extension mechanism. The default type is CIQAddressInstanceType in the http://cybox.mitre.org/extensions/Identity#CIQAddress-1 namespace. This type is defined in the extensions/location/ciq\_address\_3.0.xsd file or at the URL http://cybox.mitre.org/XMLSchema/extensions/location/ciq\_address/1.0/ciq\_address\_3.0.xsd. Those who wish to express a simple name may also do so by not specifying an xsi:type and using the Name field. |
| **Discovery\_Method** | cyboxCommon:  MeasureSourceType | 0..1 | The Discovery\_Method property is optional and enables descriptive specification of how this 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). |
| **Associated\_Objects** | AssociatedObjectsType | 0..1 | The Associated\_Objects property is optional and enables the description/specification of cyber Objects relevant (either initiating or affected by) this Action. |
| **Relationships** | ActionRelationshipsType | 0..1 | The Relationships property is optional and enables description of other cyber observable actions that are related to this Action. |
| **Frequency** | FrequencyType | 0..1 | The Frequency property conveys a targeted observation pattern of the frequency of the associated event or action. |

### EventType Class

The EventType is a complex type representing a cyber observable event that is dynamic in nature with specific action(s) taken against specific cyber relevant objects (e.g. a file is deleted, a registry key is created or an HTTP Get Request is received).

**Table 3‑2**. 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 and identifier reference to a Event instance specified elsewhere. When the idref property is used, no other property should be specified. |
| **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 stixCommon:ControlledVocabularyStringType class. 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 technical description of the Event. Any length is permitted. Optional formatting is supported via the structuring\_format property of the StructuredTextType class. |
| **Observation\_Method** | cyboxCommon:  MeasureSourceType | 0..1 | The Observation\_Method property is optional and enables descriptive specification of how this 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). |
| **Actions** | ActionsType | 0..1 | The Actions property enables description/specification of one or more cyber observable actions. |
| **Location** | cyboxCommon:  LocationType | 0..1 | The Location property specifies a relevant physical location. This field is implemented through the xsi:type extension mechanism. The default type is CIQAddressInstanceType in the http://cybox.mitre.org/extensions/Identity#CIQAddress-1 namespace. This type is defined in the extensions/location/ciq\_address\_3.0.xsd file or at the URL http://cybox.mitre.org/XMLSchema/extensions/location/ciq\_address/1.0/ciq\_address\_3.0.xsd. Those who wish to express a simple name may also do so by not specifying an xsi:type and using the Name field. |
| **Frequency** | FrequencyType | 0..1 | The Frequency property conveys a targeted observation pattern of the frequency of the associated event or action. |
| **Event** | EventType | 1 | This Event property is included recursively to enable description/specification of composite Events. |

### ObjectType Class

The ObjectType is a complex type representing the characteristics of a specific cyber-relevant object (e.g. a file, a registry key or a process).

**Table 3‑3**. 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 and identifier reference to a Object instance specified elsewhere. When the idref property is used, no other property should be specified. |
| **has\_changed** | basicDataTypes:  Boolean | 0..1 | The has\_changed property is optional and conveys a targeted observation pattern of whether the associated object specified has changed in some way without requiring further specific detail. This field would be leveraged within a pattern observable triggering on whether the value of an object specification has changed at all. This field 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 stixCommon:ControlledVocabularyStringType class. The CybOX default vocabulary class for use in the property is ‘*ObjectStateVocab-1.0*’. |
| **Description** | cyboxCommon:  StructuredTextType | 0..1 | The Description property captures a technical description of the Object. Any length is permitted. Optional formatting is supported via the structuring\_format property of the StructuredTextType class. |
| **Properties** | cyboxCommon:  ObjectPropertiesType | 0..1 | The Properties property is an abstract placeholder for various predefined Object type schemas (e.g. File, Process or System) that can be instantiated in its place through extension of the ObjectPropertiesType. This mechanism enables the specification of a broad range of Object types with consistent Object Property naming and structure. The set of Properties schemas are maintained independent of the core CybOX schema. |
| **Domain\_Specific\_**  **Object\_Properties** | DomainSpecific  ObjectPropertiesType | 0..1 | The Domain\_Specific\_Object\_Properties property is of an Abstract type placeholder within the CybOX schema enabling the inclusion of domain-specific metadata for an object through the use of a custom type defined as an extension of this base Abstract type. This enables domains utilizing CybOX such as malware analysis or forensics to incorporate non-generalized object metadata from their domains into CybOX objects. |
| **Location** | cyboxCommon:  LocationType | 0..1 | The Location property specifies a relevant physical location. This field is implemented through the xsi:type extension mechanism. The default type is CIQAddressInstanceType in the http://cybox.mitre.org/extensions/Identity#CIQAddress-1 namespace. This type is defined in the extensions/location/ciq\_address\_3.0.xsd file or at the URL http://cybox.mitre.org/XMLSchema/extensions/location/ciq\_address/1.0/ciq\_address\_3.0.xsd. Those who wish to express a simple name may also do so by not specifying an xsi:type and using the Name field. |
| **Related\_Objects** | RelatedObjectsType | 0..1 | The Related\_Objects property is optional and enables the identification and/or specification of Objects with relevant relationships with this Object. |
| **Defined\_Effect** | DefinedEffectType | 0..1 | The Defined\_Effect property is an abstract placeholder for various predefined Object Effect types (e.g. DataReadEffect, ValuesEnumeratedEffect or StateChangeEffect) that can be instantiated in its place through extension of the DefinedEffectType. This mechanism enables the specification of a broad range of types of potential complex action effects on Objects. The set of Defined\_Effect types (extending the DefinedEffectType) are maintained as part of the core CybOX schema. |
| **Discovery\_Method** | cyboxCommon:  MeasureSourceType | 0..1 | The Discovery\_Method property is optional and enables descriptive specification of how this 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). |

### ObservableType Class

The ObservableType is a type representing a description of a single cyber observable.

**Table 3‑4**. 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 and identifier reference to a Observable instance specified elsewhere. When the idref property is used, no other property should be specified. |
| **negate** | basicDataTypes:Boolean | 0..1 | The negate property, when set to true, indicates the absence (rather than the presence) of the given Observable in a CybOX pattern. |
| **sighting\_count** | basicDataTypes:PositiveInteger | 0..1 | The sighting\_count property specifies how many different identical instances of the Observable may have been seen/sighted. |
| **Title** | basicDataTypes:BasicString | 0..1 | The Title property captures the 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 technical description of the Observable. Any length is permitted. Optional formatting is supported via the structuring\_format property of the StructuredTextType class. |
| **Keywords** | KeywordsType | 0..1 | The Keywords property Keywords enables capture of relevant keywords for this cyber observable. |
| **Observable\_Source** | cyboxCommon:  MeasureSourceType | 0..\* | The Observable\_Source property is optional and enables descriptive specification of how this Observable was identified and specified. |
| **Object** | ObjectType | 0..1 | The Object property identifies and specifies the characteristics of a specific cyber-relevant object (e.g. a file, a registry key or a process). |
| **Event** | EventType | 0..1 | The Event property enables specification of a cyber observable event that is dynamic in nature with specific action(s) taken against specific cyber relevant objects (e.g. a file is deleted, a registry key is created or an HTTP Get Request is received). |
| **Observable\_Composition** | ObservableCompositionType | 0..1 | The Observable\_Composition property enables specification of composite observables made up of logical constructions of atomic observables or other composite observables (e.g. Obs5 = (Obs1 OR Obs2) AND (Obs3 OR Obs4)). |
| **Pattern\_Fidelity** | PatternFidelityType | 0..1 | The Pattern\_Fidelity property Pattern\_Fidelity contains elements that enable the characterization of the fidelity of this pattern to its purpose. |

## Secondary Classes

These classes are shared by CybOX classes.

### ActionArgumentType Class

The ActionArgumentType class enables the specification of a single relevant argument (i.e., parameter) for the Action.

**Table 3‑10**. 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 stixCommon:ControlledVocabularyStringType class. 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 for this action argument (i.e., parameter). |

### ActionPertinentObjectPropertyType Class

The ActionPertinentObjectPropertyType identifies one of the Properties of an Object that specifically pertinent to an Action.

**Table 3‑23**. Properties of the ActionPertinentObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **name** | basicDataTypes:BasicString | 0..1 | The name property specifies the field name for the pertinent Object Property. |
| **xpath** | basicDataTypes:BasicString | 0..1 | The xpath property specifies the XPath 1.0 expression identifying the pertinent property within the Properties schema for this object type. |

### ActionReferenceType Class

ActionReferenceType is intended to serve as a method for linking to actions.

**Table 3‑21**. Properties of the ActionReferenceType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **action\_id** | basicDataTypes:QualifiedName | 0..1 | The action\_id property refers to the id of the action being referenced. |

### ActionRelationshipType Class

The ActionRelationshipType characterizes a relationship between a specified cyber observable action and another cyber observable action.

**Table 3‑20**. 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 stixCommon:ControlledVocabularyStringType class. The CybOX default vocabulary class for use in the property is ‘*ActionRelationshipTypeVocab-1.0*’. |
| **Action\_Reference** | ActionReferenceType | 1..\* | The Action\_Reference property captures references to other Actions. |

### AssociatedObjectType Class

The AssociatedObjectType is a complex type representing the characterization of a cyber observable Object associated with a given cyber observable Action. It extends the ObjectType class. The extended props are below.

**Table 3‑22**. 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 stixCommon:ControlledVocabularyStringType class. 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 is optional and identifies which of the Properties of this Object are specifically pertinent to this Action. |

### FrequencyType Class

The FrequencyType is a type representing the specification of a frequency for a given action or event.

**Table 3‑24**. Properties of the FrequencyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **rate** | xs:float | 0..1 | This property specifies the rate for this defined frequency. |
| **units** | basicDataTypes:BasicString | 0..1 | This property specifies the units for this defined frequency. |
| **scale** | basicDataTypes:BasicString | 0..1 | This property specifies the time scale for this defined frequency. |
| **trend** | TrendEnum | 0..1 | This property is optional and conveys a targeted observation pattern of the nature of any trend in the frequency of the associated event or action. This field would be leveraged within an event or action pattern observable triggering on the matching of a specified trend in the frequency of an event or action. |

### ObfuscationTechniqueType Class

The ObfuscationTechniqueType enables the description of a single potential technique an attacker could leverage to obfuscate the observability of this Observable.

**Table 3‑25**. Properties of the ObfuscationTechniqueType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Description** | cyboxCommon:StructuredTextType | 1 | The Description property captures a technical description of the ObfuscationTechnique. Any length is permitted. Optional formatting is supported via the structuring\_format property of the StructuredTextType class. |
| **Observables** | ObservablesType | 0..1 | The Observables property is optional and enables description of potential cyber observables that could indicate the use of this particular obfuscation technique. |

### ObservableCompositionType Class

The ObservablesCompositionType enables the specification of higher-order composite observables composed of logical combinations of other observables.

**Table 3‑5**. Properties of the ObservableCompositionType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **operator** | OperatorTypeEnum | 0..1 | The operator property enables the specification of complex compositional cyber observables by providing logical operators for defining interrelationships between constituent cyber observables defined utilizing the recursive Observable element. |
| **Observable** | ObservableType | 0..\* | The Observable property represents a description of a single cyber observable. |

### PatternFidelityType Class

**Table 3‑6**. Properties of the PatternFidelityType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Noisiness** | NoisinessEnum | 0..1 | The Noisiness property is optional and enables simple characterization of how noisy this Observable typically could be. In other words, how likely is it to generate false positives. |
| **Ease\_of\_Evasion** | EaseOfObfuscationEnum | 0..1 | The Ease\_of\_Obfuscation property is optional and enables simple characterization of how easy it would be for an attacker to obfuscate the observability of this Observable. |
| **Evasion\_Techniques** | ObfuscationTechniquesType | 0..1 | The Obfuscation\_Techniques property is optional and enables the description of potential techniques an attacker could leverage to obfuscate the observability of this Observable. |

### RelatedObjectType Class

The RelatedObjectType enables the identification and/or specification of an Object with a relevant relationship with this Object. This extends the ObjectType class – extended props are below (one).

**Table 3‑26**. 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 stixCommon:ControlledVocabularyStringType class. The CybOX default vocabulary class for use in the property is ‘*ObjectRelationshipVocab-1.1*’. |

## Content Aggregation Classes

Each component type has an associated aggregation class that has one main property – a set of instances of that component type.

### ActionAliasesType Class

The ActionAliasesType enables identification of other potentially used names for this Action.

**Table 3‑8**. Properties of the ActionAliasesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Action\_Alias** | basicDataTypes:BasicString | 1..\* | The Action\_Alias property is optional and enables identification of a single other potentially used name for this Action. |

### ActionArgumentsType Class

The ActionArgumentsType class enables the specification of relevant arguments/parameters for the Action.

**Table 3‑9**. Properties of the ActionArgumentsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Action\_Argument** | ActionArgumentType | 1..\* | The Action\_Argument property is optional and enables the specification of a single relevant argument/parameter for this Action. |

### ActionPertinentObjectPropertiesType Class

The ActionPertinentObjectPropertiesType identifies which of the Properties of this Object are specifically pertinent to this Action.

**Table 3‑14**. Properties of the ActionPertinentObjectPropertiesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Property** | ActionPertinentObjectPropertyType | 1..\* | The Property property identifies a single Object Property that is specifically pertinent to this Action. |

### ActionRelationshipsType Class

The ActionRelationshipsType captures 1-n relationships between an Action and another Action.

**Table 3‑11**. Properties of the ActionRelationshipsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Relationship** | ActionRelationshipType | 1..\* | The Relationship property is required and enables description of a single other cyber observable Action that is related to this Action. |

### ActionsType Class

The ActionsType class is a complex type representing a set of cyber observable actions.

The properties of the ActionsType class are given in Table 3‑7.

**Table 3‑7**. Properties of the ActionsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Action** | ActionType | 1..\* | The Action property enables description/specification of a single cyber observable action. |

### AssociatedObjectsType Class

The AssociatedObjectsType enables the description/specification of cyber Objects relevant to an Action.

**Table 3‑12**. Properties of the AssociatedObjectsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Associated\_Object** | AssociatedObjectType | 1..\* | The Associated\_Object property enables the description of cyber Objects associated with this Action. This could include Objects that initiated the action, are the target Objects affected by the Action, are utilized by the Action or are the returned result of the Action. |

### DomainSpecificObjectPropertiesType Class

The DomainSpecificObjectPropertiesType is an abstract type placeholder within the CybOX schema enabling the inclusion of domain-specific metadata for an object through the use of a custom type defined as an extension of this base Abstract type. This enables domains utilizing CybOX such as malware analysis or forensics to incorporate non-generalized object metadata from their domains into CybOX objects. There is no property table.

### KeywordsType Class

**Table 3‑18**. Properties of the KeywordsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Keyword** | basicDataTypes:BasicString | 1..\* | The Keyword property Each keyword element contains one keyword. |

### ObfuscationTechniquesType Class

The ObfuscationTechniquesType enables the description of a set of potential techniques an attacker could leverage to obfuscate the observability of this Observable.

**Table 3‑17**. Properties of the ObfuscationTechniquesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Obfuscation\_Technique** | ObfuscationTechniqueType | 1..\* | The Obfuscation\_Technique property is optional and enables the description of a single potential technique an attacker could leverage to obfuscate the observability of this Observable. |

### ObservablesType Class

The ObservablesType class is a type representing a collection of cyber observables.

The properties of the ActionsType class are given Table 3‑13.

**Table 3‑13**. 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 utilized for this set of Observables. |
| **cybox\_minor\_version** | basicDataTypes:BasicString | 0..1 | The cybox\_minor\_version property specifies the minor version of the CybOX language utilized for this set of Observables. |
| **cybox\_update\_version** | basicDataTypes:BasicString | 0..1 | The cybox\_update\_version property specifies the update version of the CybOX language utilized for this set of Observables. This field MUST be used when using an update version of CybOX. |
| **Observable\_Package\_Source** | cyboxCommon:MeasureSourceType | 0..1 | The Observable\_Package\_Source property is optional and enables descriptive specification of how this package of Observables was identified and specified. |
| **Observable** | ObservableType | 1..\* | The Observable property |
| **Pools** | PoolsType | 0..1 | The Pools property enables the description of Events, Actions, Objects and Properties in a space-efficient pooled manner with the actual Observable structures defined in the CybOX schema containing references to the pooled elements. This reduces redundancy caused when identical observable elements occur multiple times within a set of defined Observables. |

### PropertiesType Class

The PropertiesType specifies the properties that were enumerated as a result of the action on the object.

**Table 3‑15**. Properties of the PropertiesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Property** | basicDataTypes:BasicString | 1..\* | The Property property The Property element specifies a single property that was enumerated as a result of the action on the object. |

### RelatedObjectsType Class

The RelatedObjectsType enables the identification and/or specification of Objects with relevant relationships with this Object.

**Table 3‑19**. Properties of the RelatedObjectsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Related\_Object** | RelatedObjectType | 1..\* | The Related\_Object property is optional and enables the identification and/or specification of a single Objects with relevant relationships with this Object. |

### ValuesType Class

The ValuesType specifies the values that were enumerated as a result of the action on the object.

**Table 3‑16**. Properties of the ValuesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Value** | basicDataTypes:BasicString | 1..\* | The Value property specifies a single value that was enumerated as a result of the action on the object. |

## Pool Classes

### PoolsType Class

The PoolsType class enables the description of Events, Actions, Objects and Properties in a space-efficient pooled manner with the actual Observable structures defined in the CybOX schema containing references to the pooled elements. This reduces redundancy caused when identical observable elements occur multiple times within a set of defined Observables.

**Table 3‑27**. Properties of the PoolsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Event\_Pool** | EventPoolType | 0..1 | The Event\_Pool property enables the description of CybOX Events in a space-efficient pooled manner with the actual Observable structures defined in the CybOX schema containing references to the pooled Event elements. This reduces redundancy caused when identical Events occur multiple times within a set of defined Observables. |
| **Action\_Pool** | ActionPoolType | 0..1 | The Action\_Pool property enables the description of CybOX Actions in a space-efficient pooled manner with the actual Observable structures defined in the CybOX schema containing references to the pooled Action elements. This reduces redundancy caused when identical Actions occur multiple times within a set of defined Observables. |
| **Object\_Pool** | ObjectPoolType | 0..1 | The Object\_Pool property enables the description of CybOX Objects in a space-efficient pooled manner with the actual Observable structures defined in the CybOX schema containing references to the pooled Object elements. This reduces redundancy caused when identical Objects occur multiple times within a set of defined Observables. |
| **Property\_Pool** | PropertyPoolType | 0..1 | The Property\_Pool property enables the description of CybOX Properties in a space-efficient pooled manner with the actual Observable structures defined in the CybOX schema containing references to the pooled Properties elements. This reduces redundancy caused when identical Properties occur multiple times within a set of defined Observables. |

### EventPoolType Class

The EventPoolType enables the description of CybOX Events in a space-efficient pooled manner with the actual Observable structures defined in the CybOX schema containing references to the pooled Event elements. This reduces redundancy caused when identical Events occur multiple times within a set of defined Observables.

**Table 3‑29**. Properties of the EventPoolType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Event** | EventType | 1..\* | The Event property enables specification of a cyber observable event that is dynamic in nature with specific action(s) taken against specific cyber relevant objects (e.g. a file is deleted, a registry key is created or an HTTP Get Request is received). |

### ActionPoolType Class

The ActionPoolType enables the description of CybOX Actions in a space-efficient pooled manner with the actual Observable structures defined in the CybOX schema containing references to the pooled Action elements. This reduces redundancy caused when identical Actions occur multiple times within a set of defined Observables.

**Table 3‑28**. Properties of the ActionPoolType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Action** | ActionType | 1..\* | The Action property enables description/specification of a single cyber observable action. |

### ObjectPoolType Class

The ObjectPoolType enables the description of CybOX Objects in a space-efficient pooled manner with the actual Observable structures defined in the CybOX schema containing references to the pooled Object elements. This reduces redundancy caused when identical Objects occur multiple times within a set of defined Observables.

**Table 3‑30**. Properties of the ObjectPoolType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Object** | ObjectType | 1..\* | The Object property identifies and specifies the characteristics of a specific cyber-relevant object (e.g. a file, a registry key or a process). |

### PropertyPoolType Class

The PropertyPoolType enables the description of CybOX Properties in a space-efficient pooled manner with the actual Observable structures defined in the CybOX schema containing references to the pooled Properties elements. This reduces redundancy caused when identical Properties occur multiple times within a set of defined Observables.

**Table 3‑31**. Properties of the PropertyPoolType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Property** | cyboxCommon:PropertyType | 1..\* | The Property property enables the specification of a single Object Property. |

## Defined Effect Classes

The order of the other defined effect classes corresponds to the EffectTypeEnum enumeration.

### DefinedEffectType Class

The DefinedEffectType class is an abstract placeholder for various predefined Object Effect types (e.g., DataReadEffect, ValuesEnumeratedEffect, or StateChangeEffect) that can be instantiated in its place through extension of the DefinedEffectType class. This mechanism enables the specification of a broad range of types of potential complex action effects on Objects. The set of Defined\_Effect subclasses (extending the DefinedEffectType class) are maintained as part of the core CybOX schema.

**Table 3‑37**. Properties of the DefinedEffectType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **effect\_type** | EffectTypeEnum | 0..1 | The effect\_type property specifies the nature of the Defined Effect instantiated in the place of the Defined\_Effect element. |

### StateChangeEffectType Class

The StateChangeEffectType is intended as a generic way of characterizing the effects of actions upon objects where the some state of the object is changed. EXTENDS the DefinedEffectType class (an abstract class).

**Table 3‑32**. Properties of the StateChangeEffectType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Old\_Object** | ObjectType | 0..1 | The Old\_Object property specifies the object and its properties as they were before the state change effect occurred. |
| **New\_Object** | ObjectType | 1 | The New\_Object property specifies the object and its properties as they are after the state change effect occurred. |

### DataReadEffectType Class

The DataReadEffectType type is intended to characterize the effects of actions upon objects where some data is read, such as from a file or a pipe. EXTENDS the DefinedEffectType class (an abstract class).

**Table 3‑33**. Properties of the DataReadEffectType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Data** | cyboxCommon:DataSegmentType | 1 | The Data property specifies the data that was read from the object by the action. |

### DataWrittenEffectType Class

The DataWrittenEffectType type is intended to characterize the effects of actions upon objects where some data is written, such as to a file or a pipe. EXTENDS the DefinedEffectType class (an abstract class).

**Table 3‑34**. Properties of the DataWrittenEffectType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Data** | cyboxCommon:DataSegmentType | 1 | The Data property specifies the data that was written to the object by the action. |

### DataSentEffectType Class

The DataSentEffectType type is intended to characterize the effects of actions upon objects where some data is sent, such as a byte sequence on a socket. EXTENDS the DefinedEffectType class (an abstract class).

**Table 3‑35**. Properties of the DataSentEffectType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Data** | cyboxCommon:DataSegmentType | 1 | The Data property specifies the data that was sent on the object, or from the object, by the action. |

### DataReceivedEffectType Class

The DataReceivedEffectType type is intended to characterize the effects of actions upon objects where some data is received, such as a byte sequence on a socket. EXTENDS the DefinedEffectType class (an abstract class).

**Table 3‑36**. Properties of the DataReceivedEffectType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Data** | cyboxCommon:DataSegmentType | 1 | The Data property specifies the data that was received on the object, or from the object, by the action. |

### PropertyReadEffectType Class

The PropertyReadEffectType type is intended to characterize the effects of actions upon objects where some specific property is read from an object, such as the current running state of a process. EXTENDS the DefinedEffectType class (an abstract class).

**Table 3‑38**. 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. |

### PropertiesEnumeratedEffectType Class

The PropertiesEnumeratedEffectType type is intended to characterize the effects of actions upon objects where some properties of the object are enumerated, such as the startup parameters for a process. EXTENDS the DefinedEffectType class (an abstract class).

**Table 3‑39**. Properties of the PropertiesEnumeratedEffectType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Properties** | PropertiesType | 1 | The Properties property specifies the properties that were enumerated as a result of the action on the object. |

### ValuesEnumeratedEffectType Class

The ValuesEnumeratedEffectType type is intended to characterize the effects of actions upon objects where some values of the object are enumerated, such as the values of a registry key. EXTENDS the DefinedEffectType class (an abstract class).

**Table 3‑40**. Properties of the ValuesEnumeratedEffectType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Values** | ValuesType | 1 | The Values property specifies the values that were enumerated as a result of the action on the object. |

### SendControlCodeEffectType Class

The SendControlCodeEffectType is intended to characterize the effects of actions upon objects where some control code, or other control-oriented communication signal, is sent to the object. For example, an action may send a control code to change the running state of a process. EXTENDS the DefinedEffectType class (an abstract class).

**Table 3‑41**. Properties of the SendControlCodeEffectType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Control\_Code** | basicDataTypes:BasicString | 1 | The Control\_Code property specifies the actual control code that was sent to the object. |

## Enumerations

### ActionStatusTypeEnum Enumeration

**Table 3‑42**. Literals of the ActionStatusTypeEnum enumeration

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

### ActionContextTypeEnum Enumeration

**Table 3‑43**. Literals of the ActionContextTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **Host** | Specifies that the cyber observable action occurred on a host. |
| **Network** | Specifies that the cyber observable action occurred on a network. |

### EaseOfObfuscationEnum Enumeration

**Table 3‑48**. Literals of the EaseOfObfuscationEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **High** | Specifies that this observable is very easy to obfuscate and hide. |
| **Medium** | Specifies that this observable is somewhat easy to obfuscate and hide. |
| **Low** | Specifies that this observable is not very easy to obfuscate and hide. |

### EffectTypeEnum Enumeration

**Table 3‑46**. Literals of the EffectTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **State\_Changed** | Specifies that the associated Action had an effect on the Object of changing its state. |
| **Data\_Read** | Specifies that the associated Action had an effect on the Object of reading data from it. |
| **Data\_Written** | Specifies that the associated Action had an effect on the Object of writing data to it. |
| **Data\_Sent** | Specifies that the associated Action had an effect on the Object of sending data to it. |
| **Data\_Received** | Specifies that the associated Action had an effect on the Object of receiving data from it. |
| **Properties\_Read** | Specifies that the associated Action had an effect on the Object of reading properties from it. |
| **Properties\_Enumerated** | Specifies that the associated Action had an effect on the Object of enumerating properties from it. |
| **Values\_Enumerated** | Specifies that the associated Action had an effect on the Object of enumerating values from it. |
| **ControlCode\_Sent** | Specifies that the associated Action had an effect on the Object of having a control code sent to it. |

### NoisinessEnum Enumeration

**Table 3‑47**. Literals of the NoisinessEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **High** | Specifies that this observable has a high level of noisiness meaning a potentially high level of false positives. |
| **Medium** | Specifies that this observable has a medium level of noisiness meaning a potentially medium level of false positives. |
| **Low** | Specifies that this observable has a low level of noisiness meaning a potentially low level of false positives. |

### OperatorTypeEnum Enumeration

**Table 3‑44**. Literals of the OperatorTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **AND** | Specifies the AND logical composition operation. |
| **OR** | Specifies the OR logical composition operation. |

### TrendEnum Enumeration

**Table 3‑45**. Literals of the TrendEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **Increasing** | Specifies an increasing trend. |
| **Decreasing** | Specifies a decreasing trend. |

# References

References made in this document are listed below.

[REL] CybOXTM Core Data Model as implement in XSD

<http://cybox.mitre.org/XMLSchema/core/2.1/cybox_core.xsd>

[RFC2119] RFC 2119 – Key words for use in RFCs to Indicate Requirement Levels

<http://www.ietf.org/rfc/rfc2119.txt>

[CybOX] CybOXTM Web Site

<https://cybox.mitre.org>

[CybOX-SPC] CybOXTM Project Github Site

<http://github.com/CybOXProject/specifications>

[TOU] Terms of Use

<http://stix.mitre.org/about/termsofuse.html>

1. For detailed information see [TOU]. [↑](#footnote-ref-1)
2. For more information about the CybOX Language, please visit [CybOX]. [↑](#footnote-ref-2)