CybOX Version 2.1.1 Part 1: Overview

Working Draft 01

15 December 2015

Technical Committee:

[OASIS Cyber Threat Intelligence (CTI) TC](https://www.oasis-open.org/committees/cti/)

Chair:

Richard Struse ([Richard.Struse@HQ.DHS.GOV](mailto:Richard.Struse@HQ.DHS.GOV)), [DHS Office of Cybersecurity and Communications (CS&C)](http://www.dhs.gov/office-cybersecurity-and-communications)

Editors:

Trey Darley ([trey@soltra.com](mailto:trey@soltra.com)), [Soltra](http://www.soltra.com/)

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

Rich Piazza ([rpiazza@mitre.org](mailto:rpiazza@mitre.org)), [MITRE Corporation](http://www.mitre.org/)

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

Additional artifacts:

This prose specification is one component of a Work Product which consists of:

* *CybOX Version 2.1.1 Part 1: Overview*. (this document)
* *CybOX Version 2.1.1 Part 2: Common*. [URI]
* *CybOX Version 2.1.1 Part 3: Core*. [URI]
* *CybOX Version 2.1.1 Part 4: Default Extensions*. [URI]
* *CybOX Version 2.1.1 Part 5: Default Vocabularies*. [URI]
* *CybOX Version 2.1.1 Part 6: UML Model*. [URI]
* *CybOX Version 2.1.1 Part 7: API Object*. [URI]
* *CybOX Version 2.1.1 Part 8: ARP Cache Object*. [URI]
* *CybOX Version 2.1.1 Part 9: AS Object*. [URI]
* *CybOX Version 2.1.1 Part 10: Account Object*. [URI]
* *CybOX Version 2.1.1 Part 11: Address Object*. [URI]
* *CybOX Version 2.1.1 Part 12: Archive File Object*. [URI]
* *CybOX Version 2.1.1 Part 13: Artifact Object*. [URI]
* *CybOX Version 2.1.1 Part 14: Code Object*. [URI]
* *CybOX Version 2.1.1 Part 15: Custom Object*. [URI]
* *CybOX Version 2.1.1 Part 16: DNS Cache Object*. [URI]
* *CybOX Version 2.1.1 Part 17: DNS Query Object*. [URI]
* *CybOX Version 2.1.1 Part 18: DNS Record Object*. [URI]
* *CybOX Version 2.1.1 Part 19: Device Object*. [URI]
* *CybOX Version 2.1.1 Part 20: Disk Object*. [URI]
* *CybOX Version 2.1.1 Part 21: Disk Partition Object*. [URI]
* *CybOX Version 2.1.1 Part 22: Domain Name Object*. [URI]
* *CybOX Version 2.1.1 Part 23: Email Message Object*. [URI]
* *CybOX Version 2.1.1 Part 24: File Object*. [URI]
* *CybOX Version 2.1.1 Part 25: GUI Dialogbox Object*. [URI]
* *CybOX Version 2.1.1 Part 26: GUI Object*. [URI]
* *CybOX Version 2.1.1 Part 27: GUI Window Object*. [URI]
* *CybOX Version 2.1.1 Part 28: HTTP Session Object*. [URI]
* *CybOX Version 2.1.1 Part 29: Hostname Object*. [URI]
* *CybOX Version 2.1.1 Part 30: Image File Object*. [URI]
* *CybOX Version 2.1.1 Part 31: Library File Object*. [URI]
* *CybOX Version 2.1.1 Part 32: Link Object*. [URI]
* *CybOX Version 2.1.1 Part 33: Linux Package Object*. [URI]
* *CybOX Version 2.1.1 Part 34: Memory Object*. [URI]
* *CybOX Version 2.1.1 Part 35: Mutex Object*. [URI]
* *CybOX Version 2.1.1 Part 36: Network Connection Object*. [URI]
* *CybOX Version 2.1.1 Part 37: Network Flow Object*. [URI]
* *CybOX Version 2.1.1 Part 38: Network Packet Object*. [URI]
* *CybOX Version 2.1.1 Part 39: Network Route Entry Object*. [URI]
* *CybOX Version 2.1.1 Part 40: Network Route Object*. [URI]
* *CybOX Version 2.1.1 Part 41: Network Socket Object*. [URI]
* *CybOX Version 2.1.1 Part 42: Network Subnet Object*. [URI]
* *CybOX Version 2.1.1 Part 43: PDF File Object*. [URI]
* *CybOX Version 2.1.1 Part 44: Pipe Object*. [URI]
* *CybOX Version 2.1.1 Part 45: Port Object*. [URI]
* *CybOX Version 2.1.1 Part 46: Process Object*. [URI]
* *CybOX Version 2.1.1 Part 47: Product Object*. [URI]
* *CybOX Version 2.1.1 Part 48: SMS Message Object*. [URI]
* *CybOX Version 2.1.1 Part 49: Semaphore Object*. [URI]
* *CybOX Version 2.1.1 Part 50: Socket Address Object*. [URI]
* *CybOX Version 2.1.1 Part 51: System Object*. [URI]
* *CybOX Version 2.1.1 Part 52: URI Object*. [URI]
* *CybOX Version 2.1.1 Part 53: URL History Object*. [URI]
* *CybOX Version 2.1.1 Part 54: Unix File Object*. [URI]
* *CybOX Version 2.1.1 Part 55: Unix Network Route Entry Object*. [URI]
* *CybOX Version 2.1.1 Part 56: Unix Pipe Object*. [URI]
* *CybOX Version 2.1.1 Part 57: Unix Process Object*. [URI]
* *CybOX Version 2.1.1 Part 58: Unix User Account Object*. [URI]
* *CybOX Version 2.1.1 Part 59: Unix Volume Object*. [URI]
* *CybOX Version 2.1.1 Part 60: User Account Object*. [URI]
* *CybOX Version 2.1.1 Part 61: User Session Object*. [URI]
* *CybOX Version 2.1.1 Part 62: Volume Object*. [URI]
* *CybOX Version 2.1.1 Part 63: Whois Object*. [URI]
* *CybOX Version 2.1.1 Part 64: Win Computer Account Object*. [URI]
* *CybOX Version 2.1.1 Part 65: Win Critical Section Object*. [URI]
* *CybOX Version 2.1.1 Part 66: Win Driver Object*. [URI]
* *CybOX Version 2.1.1 Part 67: Win Event Log Object*. [URI]
* *CybOX Version 2.1.1 Part 68: Win Event Object*. [URI]
* *CybOX Version 2.1.1 Part 69: Win Executable File Object*. [URI]
* *CybOX Version 2.1.1 Part 70: Win File Object*. [URI]
* *CybOX Version 2.1.1 Part 71: Win Filemapping Object*. [URI]
* *CybOX Version 2.1.1 Part 72: Win Handle Object*. [URI]
* *CybOX Version 2.1.1 Part 73: Win Hook Object*. [URI]
* *CybOX Version 2.1.1 Part 74: Win Kernel Hook Object*. [URI]
* *CybOX Version 2.1.1 Part 75: Win Kernel Object*. [URI]
* *CybOX Version 2.1.1 Part 76: Win Mailslot Object*. [URI]
* *CybOX Version 2.1.1 Part 77: Win Memory Page Region Object*. [URI]
* *CybOX Version 2.1.1 Part 78: Win Mutex Object*. [URI]
* *CybOX Version 2.1.1 Part 79: Win Network Route Entry Object*. [URI]
* *CybOX Version 2.1.1 Part 80: Win Network Share Object*. [URI]
* *CybOX Version 2.1.1 Part 81: Win Pipe Object*. [URI]
* *CybOX Version 2.1.1 Part 82: Win Prefetch Object*. [URI]
* *CybOX Version 2.1.1 Part 83: Win Process Object*. [URI]
* *CybOX Version 2.1.1 Part 84: Win Registry Key Object*. [URI]
* *CybOX Version 2.1.1 Part 85: Win Semaphore Object*. [URI]
* *CybOX Version 2.1.1 Part 86: Win Service Object*. [URI]
* *CybOX Version 2.1.1 Part 87: Win System Object*. [URI]
* *CybOX Version 2.1.1 Part 88: Win System Restore Object*. [URI]
* *CybOX Version 2.1.1 Part 89: Win Task Object*. [URI]
* *CybOX Version 2.1.1 Part 90: Win Thread Object*. [URI]
* *CybOX Version 2.1.1 Part 91: Win User Account Object*. [URI]
* *CybOX Version 2.1.1 Part 92: Win Volume Object*. [URI]
* *CybOX Version 2.1.1 Part 93: Win Waitable Timer Object*. [URI]
* *CybOX Version 2.1.1 Part 94: X509 Certificate Object*. [URI]

Related work:

This specification is related to:

* *STIX Version 1.2.1 (placeholder)*

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 document serves as an overview of those specifications and defines how they are used within the broader CybOX framework.

Status:

This [Working Draft](https://www.oasis-open.org/policies-guidelines/tc-process#dWorkingDraft) (WD) has been produced by one or more TC Members; it has not yet been voted on by the TC or [approved](https://www.oasis-open.org/policies-guidelines/tc-process#committeeDraft) as a Committee Draft (Committee Specification Draft or a Committee Note Draft). The OASIS document [Approval Process](https://www.oasis-open.org/policies-guidelines/tc-process#standApprovProcess) begins officially with a TC vote to approve a WD as a Committee Draft. A TC may approve a Working Draft, revise it, and re-approve it any number of times as a Committee Draft.

URI patterns:

Initial publication URI:  
http://docs.oasis-open.org/cti/stix/v1.2.1/csd01/part9-coa/stix-v1.2.1-csd01-part9-coa.docx

Permanent “Latest version” URI:  
http://docs.oasis-open.org/cti/stix/v1.2.1/stix-v1.2.1-part9-coa.docx

(Managed by OASIS TC Administration; please don’t modify.)

Copyright © OASIS Open 2015. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](https://www.oasis-open.org/policies-guidelines/ipr) 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.

Table of Contents

[1 Introduction 5](#_Toc440019440)

[1.1 CybOX Specification Documents 5](#_Toc440019441)

[1.2 Document Conventions 5](#_Toc440019442)

[1.2.1 Fonts 5](#_Toc440019443)

[1.2.2 UML Package References 6](#_Toc440019444)

[1.2.3 UML Diagrams 6](#_Toc440019445)

[1.2.3.1 Class Properties 6](#_Toc440019446)

[1.2.3.2 Diagram Icons and Arrow Types 6](#_Toc440019447)

[1.2.4 Property Table Notation 7](#_Toc440019448)

[1.2.5 Property and Class Descriptions 7](#_Toc440019449)

[1.3 Terminology 8](#_Toc440019450)

[1.4 Normative References 8](#_Toc440019451)

[2 Background Information 9](#_Toc440019452)

[2.1 Cyber Observables 9](#_Toc440019453)

[3 Language Modularity 10](#_Toc440019454)

[3.1 Core Data Model 10](#_Toc440019455)

[3.2 Objects Models 10](#_Toc440019456)

[3.3 Common Data Model 12](#_Toc440019457)

[3.3.1 Object Property Model 12](#_Toc440019458)

[3.4 Default Extensions Data Model 14](#_Toc440019459)

[3.5 Default Vocabularies 14](#_Toc440019460)

[3.6 Basic Data Types 14](#_Toc440019461)

[3.6.1 Common Basic Data Types 15](#_Toc440019462)

[3.6.2 Specializations of the BasicString Data Type 15](#_Toc440019463)

[4 Data Model Conventions 17](#_Toc440019464)

[4.1 UML Packages 17](#_Toc440019465)

[4.2 Naming Conventions 18](#_Toc440019466)

[4.3 Identifiers 18](#_Toc440019467)

[5 Relationships to Other Externally-defined Data Models 19](#_Toc440019468)

[5.1 Customer Information Quality (CIQ) 19](#_Toc440019469)

[5.2 Common Platform Enumeration (CPE) 19](#_Toc440019470)

[6 Conformance 20](#_Toc440019471)

[Appendix A. Acknowledgments 21](#_Toc440019472)

[Appendix B. Revision History 22](#_Toc440019473)

# 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** and **1.5**. 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**.

## 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](#AdditionalArtifacts) is not necessary.

The [*CybOX Version 2.1.1 Part 1: Overview*](#AdditionalArtifacts) 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*](#AdditionalArtifacts) also summarizes the relationship of CybOX to other externally defined data models, and outlines general CybOX data model conventions.

## Document Conventions

The following conventions are used in this document.

### 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*](#AdditionalArtifacts).

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](#AdditionalArtifacts). 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*](#AdditionalArtifacts)contains the full list of CybOX packages, along with the associated prefix notations, descriptions, and examples.

### 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.

#### 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 ‑. UML diagram icons

|  |  |
| --- | --- |
| **Icon** | **Description** |
| cid:image003.gif@01D05428.2B30AE20 | 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. |

### 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.

### 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 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. |

## 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].

## 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>.

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

**[CIQ]** *Customer Information Quality (CIQ) Specifications Version 3.0*. Edited by Ram Kumar. 8 April 2008. OASIS Public Review Draft 03. Available: <http://docs.oasis-open.org/ciq/v3.0/specs/ciq-specs-v3.html>.

# 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.

## 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.

# Language Modularity

## Core Data Model

The CybOX Core data model defines the four main classes: Action, Event, Observable and Object which corresponds to the primary structure characterized in CybOX. Please see [*CybOX Version 2.1.1 Part 3: Core*](#AdditionalArtifacts) for complete information on the CybOX Core data model.

## Objects Models

There are eighty-eight different Object data models in CybOX 2.1.1. They cover a varied collection of artifacts that are pertinent to the cyber thread domain. 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 UML package that describes the context-specific classes and properties that comprise the Object. As stated in the introduction, the use of any particular CybOX Object data models is purely optional: users select and use only those Objects and corresponding data models that are needed.

The grouping of Objects below is for expository purposes only. It does not suggest or imply the use of any Object data model.

* **Host-based Artifacts**
  + **File-related**
    - Library
    - Windows Pipe
    - Pipe
    - Unix Pipe
    - Windows Filemapping
    - **File**
      * File
      * Image File
      * PDF File
      * Archive File
      * Windows File
      * Unix File
      * Windows Executable File
  + **Memory-related**
    - Windows Critical Section
    - Semaphore
    - Windows Semaphore
    - Windows Waitable Timer
    - Mutex
    - Windows Mutex
    - Windows Hook
    - Windows Kernel Hook
    - Windows Memory Page Region
    - Windows Handle
    - Memory
    - Windows Mailslot
  + **Mobile**
    - SMS Message
  + **Process-related**
    - Process
    - Windows Thread
    - Unix Process
    - Windows Process
  + **Disk-related**
    - Disk
    - Disk Partition
    - Volume
    - Unix Volume
    - Windows Volume
  + **Network-related**
    - Network Socket
    - Socket Address
    - Network Route Entry
    - Windows Network Route Entry
    - Unix Network Route Entry
    - Windows Network Share
    - Hostname
    - ARP Cache
    - URL History
  + **Device/System**
    - Device
    - System
    - Windows System
  + **OS Metadata**
    - Account
    - User Account
    - **GUI**
      * GUI
      * GUI Window
      * GUI Dialog Box
    - **Unix/Linux**
      * Unix User Account
      * Linux Package
    - **Windows**
      * Windows Prefetch
      * Windows Task
      * Windows Service
      * Windows System Restore
      * Windows Computer Account
      * Windows User Account
      * Windows Driver
      * Windows Service
      * Windows Event
      * Windows Event Log
      * Windows Registry Key
      * Windows Kernel
* **Network-based Artifacts**
  + Email Message
  + **DNS**
    - DNS Record
    - DNS Query
    - DNS Cache
  + Network Route
  + Network Subnet
  + Network Connection
  + Network Flow
  + Network Packet
  + HTTP Session
  + AS
  + Address
  + Email Message
  + URI
  + Port
  + Domain Name
  + Whois
* **Misc.**
  + Code
  + User Session
  + API
  + Custom
  + Link
  + Product
  + X509 Certificate
  + Artifact

The Object data models are split into three major categories: Host-based artifacts, Network-based artifacts, and the catch-all Miscellaneous artifacts. Among the Host-based artifacts, there are many data models that are extensions of others in order to define the Windows or Unix operating system specific properties. Some of the Object data models are very simple (e.g. DNS\_Cache), containing one or two UML classes or data types. Others are quite complex, encompassing many UML classes (e.g., PDF\_File).

Each individual Objects data model has a main toplevel class, which is usually named after the UML package name (e.g., the main class in the PDF\_File\_Object package is PDFFileObjectType).

## Common Data Model

The CybOX Common data model defines object classes that are shared across the various CybOX data models. At a high level, the CybOX Common data model provides object property classes, content aggregation classes, shared classes, and a pattern class for permitting complex (i.e. regular-expression based) specifications. Please see [*CybOX Version 1.2.1 Part 2: Common*](#AdditionalArtifacts) for complete information on the STIX Common data model.

### Object Property Model

The Object Property data model in CybOX is sophisticated, enabling the expression of complicated Observables. Observables can be made up of Objects or Events. In this section, we will concentrate on the Object aspect of an Observable.

Objects have two main properties, Properties and Related\_Objects. This is shown in **Figure 3‑1**.

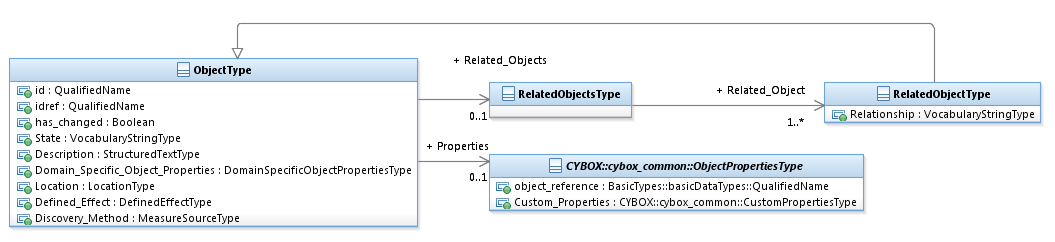


Figure ‑. UML diagram for ObjectType class

The Object data models described in **Section 3.2**, are not extensions of ObjectType, but instead are associated with and ObjectType class instance via the Properties property. Therefore, they are extensions of *ObjectPropertiesType*, an abstract class. Notice that only one toplevel instance from one of Objects data models is specified in the Properties property. Other Objects instances can be referred to via the Related\_Objects property.

Once the Objects data model is selected for an ObjectType instance, its properties can be selected. There are four different kinds of properties that can be populated:

* BasicTypes
* ObjectPropertyType extension classes
* Other classes defined in the Objects domain model
* Other toplevel classes from other Objects domain models.

The following figures illustrates an example.

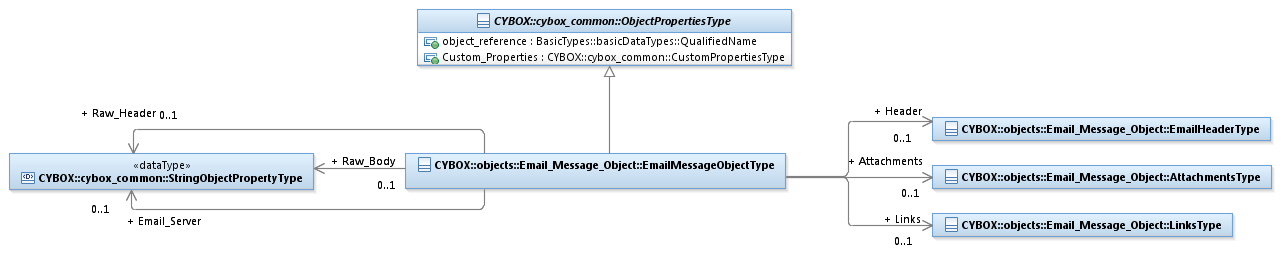


Figure ‑. Toplevel class of Email\_Message\_Object data model

The EmailMessageObjectType class extends the ObjectPropertiesType, and therefore can be used in the ObjectType class. The properties of EmailMessageObjectType are examples of using other classes from the same UML package and the use of StringObjectPropertyType, an extension of ObjectPropertyType.

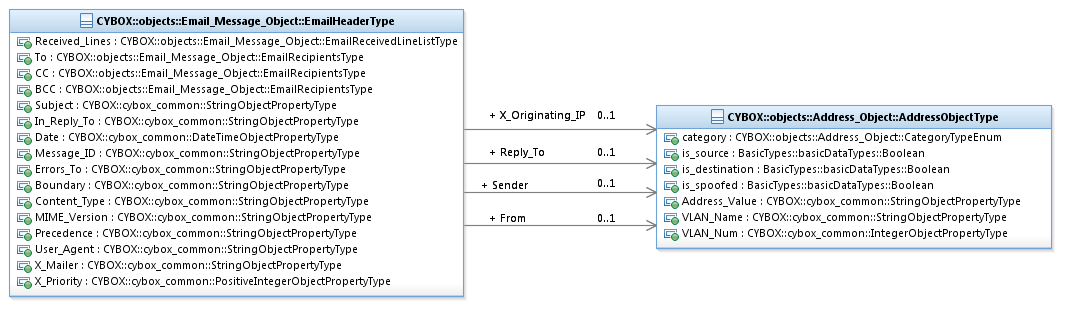


Figure ‑. Details of other classes of the Email\_Message\_Object data model

Expanding the detail of the EmailHeaderType class, in addition to properties of data types extended from the ObjectPropertyType class it also makes use of a class from another Objects domain model (AddressObjectType). Notice that AddressObjectType contains properties that use data types from the BasicTypes package (see **Section 3.6**).



Figure ‑. UML diagram of the BaseObjectPropertyType data type

Use of UML data types that extend from ObjectPropertyType enables a producer of content to include much more detail about the value of an Object’s property. ObjectPropertyType data type extends from both BaseObjectPropertyGroup, which is an abstract data type that contains the auxiliary metadata properties associated with the main property value being represented. In addition, the BaseObjectPropertyType data type also inherits from PatternFieldGroup data type. This data type incorporates pattern matching capabilities to all specializations of BaseObjectPropertyType.

## Default Extensions Data Model

A primary design principle of CybOX is to avoid duplicating data models that already exist for capturing cyber threat information. Therefore, CybOX leverages a number of other structured languages and identifiers through the use of default extensions.

More precisely, the CybOX Default Extensions data model provides loose-coupling mechanisms and default extensions for leveraging constituent data models Common Platform Enumeration [**[CPE]**](#cpe) and OASIS Customer Information Quality model [[CIQ]](#CIQ).

High level summary information is given in Section **4**. Please see [*CybOX Version 2.1.1 Part 4: Default Extensions*](#AdditionalArtifacts) for complete information on the STIX Default Extensions data model.

## Default Vocabularies

For some properties captured in CybOX, a content creator may choose to constrain the set of possible values by referencing an externally-defined vocabulary or by leveraging a default vocabulary class defined within CybOX. Alternatively, the content creator may use an arbitrary value without specifying any vocabulary. Please see  [*CybOX Version 2.1.1 Part 5: Vocabularies*](#AdditionalArtifacts) for more information about the default vocabularies defined in CybOX.

## Basic Data Types

The Basic Data Types data model defines UML data types used CybOX. As stated in the [**[UML 2.4.1]**](#UML241) specification, UML data types are similar to UML classes, but also different:

“A data type is a special kind of classifier, similar to a class. It differs from a class in that instances of a data type are identified only by their value. All copies of an instance of a data type and any instances of that data type with the same value are considered to be equal instances. Instances of a data type that have attributes (i.e., is a structured data type) are considered to be equal if the structure is the same and the values of the corresponding attributes are equal. If a data type has attributes, then instances of that data type will contain attribute values matching the attributes.”

Although four of the requisite primitive data types (Boolean, Integer, String, UnlimitedNatural) are defined in UML, the need for a broader set in CybOX drove the decision to define a complete set of basic data types in a separate, stand-alone UML package (the Basic Data Types data model). We explicitly define the data types in the Basic Data Types data model in Sections **2.7.1** and **2.7.2**.

### Common Basic Data Types

Common data types, such as string and integer, are defined in the Basic DataTypes data model and adhere to the following definitions shown in **Table 2‑1**. These definitions are based on the specification of the corresponding data types found in [**[W3DT]**](#W3DT).

Table ‑. Common basic data types

|  |  |
| --- | --- |
| **Data Type** | **Definition** |
| BasicString | The BasicString data type is a sequence of characters. Currently, characters are defined using the UTF-8 character encoding. The number of characters allowed is finite, but unbounded. |
| Boolean | The Boolean data type is defined with two possible literals: ‘*true*’ and ‘*false*’. |
| Decimal | The Decimal data type is a sequence of decimal digits, with perhaps an intervening decimal point, “.”. The number of digits on either side of the decimal point is finite, but unbounded. Often used to express currency amounts. |
| Integer | The Integer data type is a sequence of decimal digits, with perhaps a leading minus sign “-“. The number of decimal digits allowed is finite, but unbounded. |
| NonNegativeInteger | The NonNegativeInteger data type is a restriction on the Integer data type such that the leading minus sign is not allowed. |
| PositiveInteger | The PositiveInteger data type is a restriction on the NonNegativeInteger data type that disallows zero (0). |

### Specializations of the BasicString Data Type

The data types in **Table 3‑2** correspond to strings that have semantics associated with them. Because of this, they usually are restricted to a certain pattern, defined via a regular expression, and/or more formally defined in a standardization document.

Table ‑. Specializations of the BasicString Data Type

|  |  |
| --- | --- |
| **Data Type** | **Definition** |
| CAPEC\_ID | The CAPEC\_ID data type is a restriction on the BasicString data type, such that it adheres to the regular expression “CAPEC-\d+”. The CAPEC\_ID values should correspond to those defined at [**[CAPEC]**](#capec). |
| CCE\_ID | The CCE\_ID data type is a restriction on the BasicString data type such that it adheres to the regular expression “CCE-\d+\d”. The CCE\_ID values should correspond to those defined at [**[CEE]**](#cee). |
| CVE\_ID | The CVE\_ID data type is a restriction on the BasicString data type such that it adheres to the regular expression “CVE-\d\d\d\d+\d+”. The CVE\_ID values should correspond to those defined at [**[CVE]**](#cve). |
| CWE\_ID | The CWE\_ID data type is a restriction on the BasicString data type such that it adheres to the regular expression “CWE-\d+”. The CWE\_ID values should correspond to those defined at [**[CWE]**](#cwe). |
| DateTime | The DateTime data type is a restriction on the BasicString data type such that it adheres to the standard defined in [**[ISO8601]**](#iso8601). |
| HexBinary | The HexBinary data type is a restriction on the BasicString data type such that it adheres to the regular expression [0-9A-Fa-f]\*. The number of characters allowed is finite but unbounded. The number of digits must be even in length. |
| LanguageCode | The LanguageCode data type is a restriction on the BasicString data type, such that it adheres to the standard defined in [**[RFC5646]**](#rfc5646). |
| QualifiedName | The QualifiedName data type is a restriction on the BasicString data type such that it adheres to the requirements specified in [**[W3Name]**](#W3Name). |
| NoEmbeddedQuoteString | The NoEmbeddedQuoteString data type is a restriction on the BasicString data type such that it does not include any double quote characters. This data type captures properties that were attributes in the XML model. |
| URI | The URI data type is a restriction on the BasicString data type such that it adheres to the standard defined at [**[RFC3986]**](#rfc3986). |

# Data Model Conventions

The following general information and conventions are used to define the individual data models in [*STIX Version 1.2.1 Part 15: UML Model*](#AdditionalArtifacts). It should be noted that the STIX data models actually evolved as XML schemas, and as a consequence, our UML model follows some conventions so as to be compatible with the preexisting XML implementation. However, we have abstracted away from the XML implementation as much as possible.

## UML Packages

Each STIX data model is captured in a different UML package (e.g., Core package, Campaign package, etc.). 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. **Table 4‑1** lists the basic packages used throughout the CybOX data model specification documents, along with the prefix notation and an example. Descriptions of the packages are provided in Section **2**.

Table ‑. Package prefixes used by the CybOX Language

|  |  |
| --- | --- |
| Package | CybOX Core |
| **Prefix** | **cyboxCore** |
| Description | The CybOX Core data model defines a STIX Package that encompasses all other objects of STIX. |
| Example | cyboxCore:Observable |
|  | |
| Package | CybOX Common |
| **Prefix** | **cyboxCommon** |
| Description | The CybOX Common data model defines classes that are shared across the various CybOX data models. |
| Example | cyboxCommon:ConfidenceType |
|  |  |
| Package | CybOX Default Vocabularies |
| **Prefix** | **cyboxVocabs** |
| Description | The CybOX default vocabularies define the classes for default controlled vocabularies used within CybOX. |
| Example | cyboxVocabs:ActionTypeVocab |
|  | |
| Package | CybOX Basic Data Types |
| **Prefix** | **basicDataTypes** |
| Description | The STIX Basic Data Types data model defines the types used within STIX. |
| Example | basicDataTypes:URI |
|  |  |

## Naming Conventions

The UML classes, enumerations, and properties defined in STIX follow the particular naming conventions outlined in **Table 4‑2**.

Table ‑. Naming formats of different object types

|  |  |  |
| --- | --- | --- |
| **Object Type** | **Format** | **Example** |
| Class | CamelCase ending with “Type” | IndicatorBaseType |
| Property (simple) | Lowercase with underscores between words | capec\_id |
| Property (complex) | Capitalized with underscores between words | Associated\_Actor |
| Enumeration | CamelCase ending with “Enum” or “Type | DateTimePrecisionEnum; IndicatorVersionType |
| Enumeration value | *varies* | Flash drive; Public Disclosure; Externally-Located |
| Data type | CamelCase or if the words are acronyms, all capitalized with underscores between words. | PositiveInteger; CVE\_ID |

## Identifiers

Optional identifiers (IDs) can be assigned to several STIX constructs so that the constructs can be unambiguously referenced. Technically, the decision to specify an ID on a given construct is optional based on the specifics of the usage context. As a general rule, specifying IDs on particular instances of constructs enables clear referencing, relating, and pivoting.

Assigning IDs supports several very common STIX use cases such as:

* Enabling individual portions of content to be externally referenced unambiguously (e.g., a report talking about a specific Campaign or Threat Actor)
* Enabling the sharing/resharing of portions of STIX content (e.g., PartyB resharing two of a set of 100 Indicators received from PartyA)
* Enabling versioning of content
* Enabling the specification of potentially complex webs of interconnection and correlation between portions of STIX content (e.g., connecting particular TTPs and Indicators to specific Campaigns over time)
* Enabling analysis pivoting on content with multiple contexts (e.g., the same IP Address seen in multiple Incidents and with connections to multiple TTPs and Indicators)

In STIX v1.2.1, each STIX ID is a fully qualified name, which consists of a producer namespace and a unique identifier. The producer namespace is a short-hand prefix, which is separated from the unique identifier by a colon (“:”). For example:

[producer namespace]:[unique identifier]

This format provides high assurance that IDs will be both meaningful and unique. Meaning comes from producer namespace, which denotes who is producing it, and uniqueness comes from the unique identifier.

# Relationships to Other Externally-defined Data Models

STIX Version 1.2.1 leverages several other externally-defined data models that are relevant to the cyber threat domain. However, the STIX specification documents do not define any classes that are part of a non-STIX data model (e.g., [CybOX](#RelatedWork) classes are not defined in STIX specification documents). An alphabetical listing of these other data models is given below.

Please see [*CybOX Version 2.2.1 Part 4: Default Extensions*](#AdditionalArtifacts) for further information on all of the externally-defined data models STIX leverages by default (with the exception of CybOX, for which a different reference is given in Section **4.4**).

## Customer Information Quality (CIQ)

The OASIS Customer Information Quality (CIQ) Version 3.0 is a set of XML specifications for representing characteristic information about individuals and organizations [**[CIQ]**](#ciq). By extending the STIX Common AddressAbstractType and IdentityType classes, CybOX Version 2.2.1 leverages CIQ Version 3.0 to capture geographic address information and identity information associated with Threat Actors, victims, and sources of information.

## Common Platform Enumeration (CPE)

CPE is a structured naming scheme for information technology systems, software, and packages. Based upon the generic syntax for Uniform Resource Identifiers (URI), CPE includes a formal name format, a method for checking names against a system, and a description format for binding text and tests to a name. An XSD schema for version 2.3 can be found at [CPE].

# 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.

1. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Dean Thompson, Australia and New Zealand Banking Group (ANZ Bank)

Bret Jordan, Blue Coat Systems, Inc.

Adnan Baykal, Center for Internet Security (CIS)

Liron Schiff, Comilion (mobile) Ltd.

Jane Ginn, Cyber Threat Intelligence Network, Inc. (CTIN)

Richard Struse, DHS Office of Cybersecurity and Communications (CS&C)

Ryusuke Masuoka, Fujitsu Limited

Eric Burger, Georgetown University

Jason Keirstead, IBM

Paul Martini, iboss, Inc.

Jerome Athias, Individual

Sanjiv Kalkar, Individual

Terry MacDonald, Individual

Alex Pinto, Individual

Patrick Maroney, Integrated Networking Technologies, Inc.

Wouter Bolsterlee, Intelworks BV

Joep Gommers, Intelworks BV

Sergey Polzunov, Intelworks BV

Rutger Prins, Intelworks BV

Andrei Sîrghi, Intelworks BV

Jonathan Baker, MITRE Corporation

Sean Barnum, MITRE Corporation

Mark Davidson, MITRE Corporation

Ivan Kirillov, MITRE Corporation

John Wunder, MITRE Corporation

Mike Boyle, National Security Agency

Jessica Fitzgerald-McKay, National Security Agency

Takahiro Kakumaru, NEC Corporation

John-Mark Gurney, New Context Services, Inc.

Christian Hunt, New Context Services, Inc.

Andrew Storms, New Context Services, Inc.

Igor Baikalov, Securonix

Bernd Grobauer, Siemens AG

John Anderson, Soltra

Trey Darley, Soltra

Paul Dion, Soltra

Brandon Hanes, Soltra

Ali Khan, Soltra

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

1. Revision History

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