CybOX™ Version 2.1.1. Part 01: Overview

Committee Specification Draft 01 / Public Review Draft 01

20 June 2016

Specification URIs

This version:

http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part01-overview/cybox-v2.1.1-csprd01-part01-overview.docx (Authoritative)
http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part01-overview/cybox-v2.1.1-csprd01-part01-overview.html
http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part01-overview/cybox-v2.1.1-csprd01-part01-overview.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/cti/cybox/v2.1.1/part01-overview/cybox-v2.1.1-part01-overview.docx (Authoritative)
http://docs.oasis-open.org/cti/cybox/v2.1.1/part01-overview/cybox-v2.1.1-part01-overview.html
http://docs.oasis-open.org/cti/cybox/v2.1.1/part01-overview/cybox-v2.1.1-part01-overview.pdf

Technical Committee:

OASIS Cyber Threat Intelligence (CTI) TC

Chair:

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

Editors:

Trey Darley (trey@kingfisherops.com), Individual member
Ivan Kirillov (ikirillov@mitre.org), MITRE Corporation
Rich Piazza (rpiazza@mitre.org), MITRE Corporation
Desiree Beck (dbeck@mitre.org), MITRE Corporation

Additional artifacts:

This prose specification is one component of a Work Product whose components are listed in http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/cybox-v2.1.1-csprd01-additional-artifacts.html.

Related work:

This specification is related to:

- *STIX™ Version 1.2.1*. Edited by Sean Barnum, Desiree Beck, Aharon Chernin, and Rich Piazza. 05 May 2016. OASIS Committee Specification 01. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part1-overview/stix-v1.2.1-cs01-part1-overview.html.

Abstract:

The Cyber Observable Expression (CybOX) is a standardized language for encoding and communicating high-fidelity information about cyber observables, whether dynamic events or stateful measures that are observable in the operational cyber domain. By specifying a common structured schematic mechanism for these cyber observables, the intent is to enable the potential

for detailed automatable sharing, mapping, detection and analysis heuristics. This document serves as an overview of those specifications and defines how they are used within the broader CybOX framework.

Status:

Citation format:

When referencing this specification the following citation format should be used:

[CybOX-v2.1.1-overview]

CybOX™ Version 2.1.1. Part 01: Overview. Edited by Trey Darley, Ivan Kirillov, Rich Piazza, and Desiree Beck. 20 June 2016. OASIS Committee Specification Draft 01 / Public Review Draft 01. http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part01-overview/cybox-v2.1.1-csprd01-part01-overview.html. Latest version: http://docs.oasis-open.org/cti/cybox/v2.1.1/part01-overview/cybox-v2.1.1-part01-overview.html.

Notices

Table of Contents

1 Introduction

[All text is normative unless otherwise labeled]

The Cyber Observable Expression (CybOX™) provides a common structure for representing cyber observables across and among the operational areas of enterprise cyber security. CybOX improves the consistency, efficiency, and interoperability of deployed tools and processes, and it increases overall situational awareness by enabling the potential for detailed automatable sharing, mapping, detection, and analysis heuristics.

This CybOX specification overview document serves as a unifying document for the full set of CybOX specification documents. In Section **0,** 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 discusses the modularity of CybOX, and summarizes the relationship of CybOX to other languages. In section **3**, we discuss conventions common across all of the data models. Conformance information is also provided in Section **4**.

1.1 CybOX Specification Documents

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

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

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.

1.2 Document Conventions

The following conventions are used in this document.

1.2.1 Fonts

The following font and font style conventions are used in the document:

- Capitalization is used for CybOX high level concepts, which are defined in *CybOX™ Version 2.1.1 Part 1: Overview*.

  Examples: Action, Object, Event, Property

- The `Courier New` font is used for writing UML objects.

  Examples: `ActionType`, `cyboxCommon:BaseObjectPropertyType`

  Note that all high level concepts have a corresponding UML object. For example, the Action high level concept is associated with a UML class named, `ActionType`.

- The '*italic'* font (with single quotes) is used for noting actual, explicit values for CybOX Language properties. The *italic* font (without quotes) is used for noting example values.

Example: *'HashNameVocab-1.0,' high, medium, low*

1.2.2 UML Package References

Each CybOX data model is captured in a different UML package (e.g., Core package) where the packages together compose the full CybOX UML model. To refer to a particular class of a specific package, we use the format `package_prefix:class`, where `package_prefix` corresponds to the appropriate UML package.

1.2.3 UML Diagrams

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

In UML diagrams, classes are often presented with their attributes elided, to avoid clutter. A class presented with an empty section at the bottom of the icon indicates that there are no attributes other than those that are visualized using associations.

1.2.3.1 Class Properties

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

1.2.3.2 Diagram Icons and Arrow Types

Diagram icons are used in a UML diagram to indicate whether a shape is a class, enumeration, or a data type, and decorative icons are used to indicate whether an element is an attribute of a class or an enumeration literal. In addition, two different arrow styles indicate either a directed association relationship (regular arrowhead) or a generalization relationship (triangle-shaped arrowhead). The icons and arrow styles we use are shown and described in **Table 1-1**.

Table 1-1. UML diagram icons

Icon	**Description**
▤	This diagram icon indicates a class. If the name is in italics, it is an abstract class.
‹E›	This diagram icon indicates an enumeration.
‹D›	This diagram icon indicates a data type.
▣●	This decorator icon indicates an attribute of a class. The green circle means its visibility is public. If the circle is red or yellow, it means its visibility is private or protected.

▭	This decorator icon indicates an enumeration literal.
→	This arrow type indicates a directed association relationship.
—▷	This arrow type indicates a generalization relationship.

1.3 Terminology

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

1.4 Normative References

[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.
[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.
[W3DT]	**“**XML Schema Part 2: Datatypes Second Edition,” W3C Recommendation, 28 October 2004. Available: http://www.w3.org/TR/xmlschema-2.
[CAPEC]	Common Attack Pattern Enumeration and Classification (CAPEC). (2014, Nov. 7). The MITRE Corporation. [Online]. Available: http://capec.mitre.org.
[CEE]	Common Event Expression (CEE). (2014, Nov. 28). The MITRE Corporation. [Online]. Available: http://cee.mitre.org.
[CVE]	Common Vulnerabilities and Exposures (CVE). (2015, Jul. 28). The MITRE Corporation. [Online]. Available: http://cve.mitre.org.
[CWE]	Common Weakness Enumeration (CWE). (2014, Jul. 31). The MITRE Corporation. [Online]. Available: http://cwe.mitre.org.
[ISO8601]	Date and time format – ISO 8601 (n.d.). International Organization for Standardization (ISO). [Online]. Available: http://www.iso.org/iso/home/standards/iso8601.htm. Accessed Aug. 23, 2015.
[RFC3986]	Berners-Lee, T., Fielding, R. and Masinter, L., “Uniform Resource Identifier (URI): Generic Syntax,” STD 66, RFC 3986, January 2005. Available: https://www.ietf.org/rfc/rfc3986.txt.
[RFC5646]	Phillips, A. and Davis, M., “Tags for Identifying Languages,” BCP 47, RFC 5646, September 2009. Available: http://www.ietf.org/rfc/rfc5646.txt.
[W3Name]	**“**Namespaces in XML 1.0 (Third Edition),” W3C Recommendation, 8 December 2009. Available: http://www.w3.org/TR/REC-xml-names.

1.5 Non-Normative References

[UML-2.4.1]	Documents associated with Unified Modeling Language (UML), V2.4.1. (Aug. 2011). The Object Management Group (OMG). [Online]. Available: http://www.omg.org/spec/UML/2.4.1/.

2 Language Modularity

2.1 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* for complete information on the CybOX Core data model.

2.2 Object Models

Each release of the CybOX language includes a particular set of Objects that are part of the release. 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 threat domain. 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 need to use of any particular 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 properties specific to the Windows or Unix operating systems. 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 Object 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 data model is `PDFFileObjectType`).

2.3 Events and Actions Data Models

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

2.4 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 2.1.1 Part 2: Common* for complete information on the CybOX Common data model.

2.4.1 Object Property Model

The Object Property data model in CybOX is sophisticated, enabling the description of complex 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 2-1**.

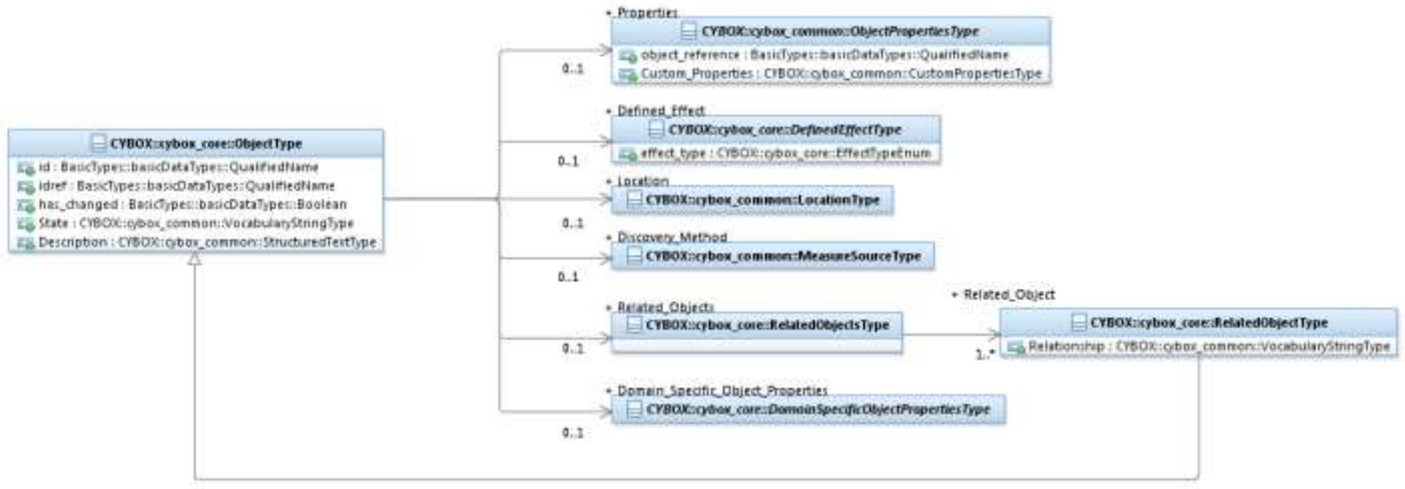
CYBOX::cybox_core::ObjectType
id : BasicTypes::basicDataTypes::QualifiedName
idref : BasicTypes::basicDataTypes::QualifiedName
has_changed : BasicTypes::basicDataTypes::Boolean
State : CYBOX::cybox_common::VocabularyStringType
Description : CYBOX::cybox_common::StructuredTextType
+ Properties
0..1
CYBOX::cybox_common::ObjectPropertiesType
object_reference : BasicTypes::basicDataTypes::QualifiedName
Custom_Properties : CYBOX::cybox_common::CustomPropertiesType
+ Defined_Effect
0..1
CYBOX::cybox_core::DefinedEffectType
effect_type : CYBOX::cybox_core::EffectTypeEnum
+ Location
0..1
CYBOX::cybox_common::LocationType
+ Discovery_Method
0..1
CYBOX::cybox_common::MeasureSourceType
+ Related_Objects
0..1
CYBOX::cybox_core::RelatedObjectsType
+ Related_Object
1..*
CYBOX::cybox_core::RelatedObjectType
Relationship : CYBOX::cybox_common::VocabularyStringType
+ Domain_Specific_Object_Properties
0..1
CYBOX::cybox_core::DomainSpecificObjectPropertiesType

Figure 2-1. UML diagram for ObjectType class

The Object data models, described in **Section 2.2**, are not extensions of `ObjectType`, but instead are used in an `ObjectType` class instance via the `Properties` property. Accordingly, they are extensions of `ObjectPropertiesType`, an abstract class. Notice that only a single toplevel instance from one of the Object data models is specified in the `Properties` property. Instances of other Objects can be referred to via the `Related_Objects` property.

Once the Object data model to be used in an ObjectType instance is selected, its properties can be assigned values. Such Object properties are based on one of four different kinds of property types:

- BasicTypes
- `ObjectPropertyType` extension classes
- Other classes defined in the same Object domain model
- Other toplevel classes from other Object domain models.

The following figures provide an example.

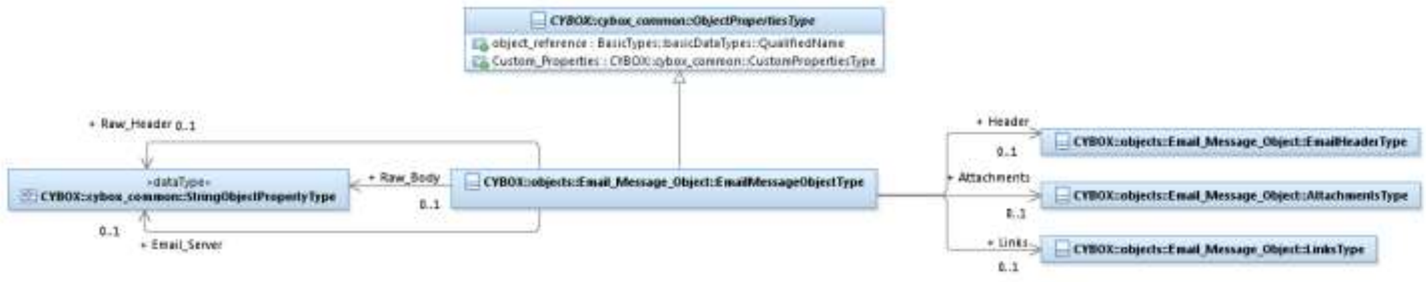
CYBOX::cybox_common::ObjectPropertiesType
object_reference : BasicTypes::basicDataTypes::QualifiedName
Custom_Properties : CYBOX::cybox_common::CustomPropertiesType
+ Raw_Header 0..1
«dataType»
CYBOX::cybox_common::StringObjectPropertyType
0..1
+ Email_Server
+ Raw_Body
0..1
CYBOX::objects::Email_Message_Object::EmailMessageObjectType
+ Header
0..1
CYBOX::objects::Email_Message_Object::EmailHeaderType
+ Attachments
0..1
CYBOX::objects::Email_Message_Object::AttachmentsType
+ Links
0..1
CYBOX::objects::Email_Message_Object::LinksType

Figure 2-2. Toplevel class of the `Email_Message_Object` data model

The `EmailMessageObjectType` class extends the `ObjectPropertiesType`, and therefore can be used in the `ObjectType` class (**Figure 2-2**). The properties of `EmailMessageObjectType` use other classes from the same UML package as well as `StringObjectPropertyType`, an extension of `ObjectPropertyType`.

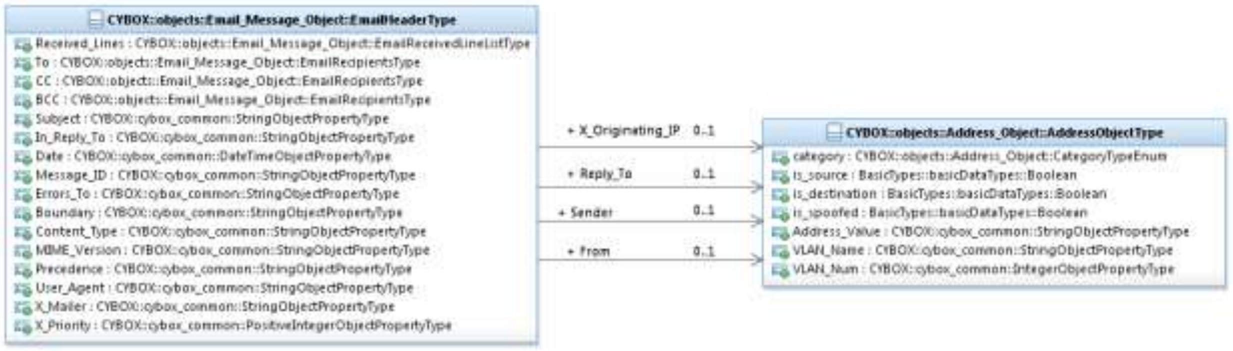
CYBOX::objects::Email_Message_Object::EmailHeaderType
Received_Lines : CYBOX::objects::Email_Message_Object::EmailReceivedLineListType
To : CYBOX::objects::Email_Message_Object::EmailRecipientsType
CC : CYBOX::objects::Email_Message_Object::EmailRecipientsType
BCC : CYBOX::objects::Email_Message_Object::EmailRecipientsType
Subject : CYBOX::cybox_common::StringObjectPropertyType
In_Reply_To : CYBOX::cybox_common::StringObjectPropertyType
Date : CYBOX::cybox_common::DateTimeObjectPropertyType
Message_ID : CYBOX::cybox_common::StringObjectPropertyType
Errors_To : CYBOX::cybox_common::StringObjectPropertyType
Boundary : CYBOX::cybox_common::StringObjectPropertyType
Content_Type : CYBOX::cybox_common::StringObjectPropertyType
MIME_Version : CYBOX::cybox_common::StringObjectPropertyType
Precedence : CYBOX::cybox_common::StringObjectPropertyType
User_Agent : CYBOX::cybox_common::StringObjectPropertyType
X_Mailer : CYBOX::cybox_common::StringObjectPropertyType
X_Priority : CYBOX::cybox_common::PositiveIntegerObjectPropertyType
+ X_Originating_IP 0..1
+ Reply_To 0..1
+ Sender 0..1
+ From 0..1
CYBOX::objects::Address_Object::AddressObjectType
category : CYBOX::objects::Address_Object::CategoryTypeEnum
is_source : BasicTypes::basicDataTypes::Boolean
is_destination : BasicTypes::basicDataTypes::Boolean
is_spoofed : BasicTypes::basicDataTypes::Boolean
Address_Value : CYBOX::cybox_common::StringObjectPropertyType
VLAN_Name : CYBOX::cybox_common::StringObjectPropertyType
VLAN_Num : CYBOX::cybox_common::IntegerObjectPropertyType

Figure 2-3. Details of other classes of the Email_Message_Object data model

Expanding the detail of the `EmailHeaderType` class (**Figure 2-3**), one can see that in addition to data type-based properties extended from the `ObjectPropertyType` class (e.g., `IntegerObjectPropertyType`) it also makes use of a class from the domain model of another Object (`AddressObjectType`). Notice that `AddressObjectType` contains properties that use data types from the `BasicTypes` package (see **Section 2.7**).

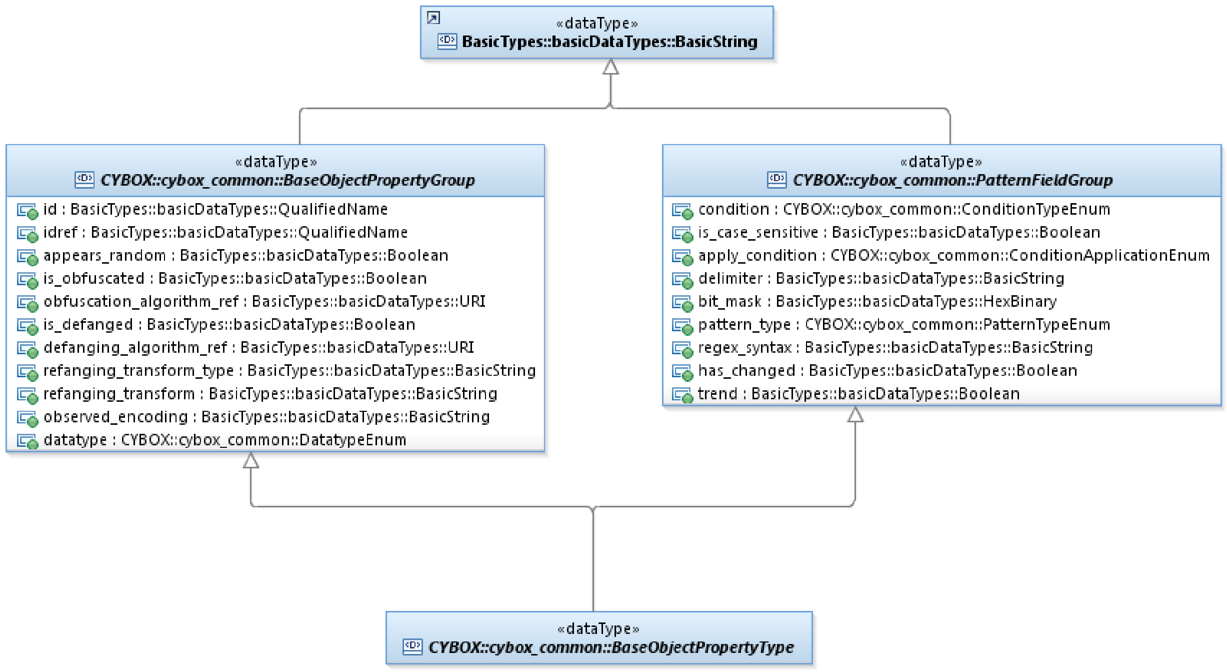
«dataType»
BasicTypes::basicDataTypes::BasicString
«dataType»
CYBOX::cybox_common::BaseObjectPropertyGroup
id : BasicTypes::basicDataTypes::QualifiedName
idref : BasicTypes::basicDataTypes::QualifiedName
appears_random : BasicTypes::basicDataTypes::Boolean
is_obfuscated : BasicTypes::basicDataTypes::Boolean
obfuscation_algorithm_ref : BasicTypes::basicDataTypes::URI
is_defanged : BasicTypes::basicDataTypes::Boolean
defanging_algorithm_ref : BasicTypes::basicDataTypes::URI
refanging_transform_type : BasicTypes::basicDataTypes::BasicString
refanging_transform : BasicTypes::basicDataTypes::BasicString
observed_encoding : BasicTypes::basicDataTypes::BasicString
datatype : CYBOX::cybox_common::DatatypeEnum
«dataType»
CYBOX::cybox_common::PatternFieldGroup
condition : CYBOX::cybox_common::ConditionTypeEnum
is_case_sensitive : BasicTypes::basicDataTypes::Boolean
apply_condition : CYBOX::cybox_common::ConditionApplicationEnum
delimiter : BasicTypes::basicDataTypes::BasicString
bit_mask : BasicTypes::basicDataTypes::HexBinary
pattern_type : CYBOX::cybox_common::PatternTypeEnum
regex_syntax : BasicTypes::basicDataTypes::BasicString
has_changed : BasicTypes::basicDataTypes::Boolean
trend : BasicTypes::basicDataTypes::Boolean
«dataType»
CYBOX::cybox_common::BaseObjectPropertyType

Figure 2-4. UML diagram of the `BaseObjectPropertyType` data type

Use of UML data types that extend from `BaseObjectPropertyType` enables a producer of content to include much more detail about the value of an Object's property (**Figure 2-4**). `BaseObjectPropertyType` data type extends from `BaseObjectPropertyGroup`, which is an abstract data type that contains the auxiliary metadata properties associated with the main property value being represented. For example, when using `StringObjectPropertyType` to describe the `File_Name` property of a File Object, one might want to include that the observed encoding of the string is "windows-1251".

In addition, the `BaseObjectPropertyType` data type also inherits from `PatternFieldGroup` data type. This data type incorporates pattern matching capabilities to all specializations of `BaseObjectPropertyType`. An example of using properties from `PatternFieldGroup` would be to use a `StringObjectPropertyType` to specify that the File Object's `File_Extension` property satisfied the condition: not equal to "exe".

Accordingly, it follows that properties from `BaseObjectPropertyGroup` are used only when specifying an actual observation and properties from `PatternFieldGroup` are used only when specifying an observable pattern.

2.5 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 such as the Common Platform Enumeration **[CPE]** and OASIS Customer Information Quality model **[CIQ]**.

Please see *CybOX™ Version 2.1.1 Part 4: Default Extensions* for complete information on the CybOX Default Extensions data model.

2.6 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* for more information about the default vocabularies defined in CybOX.

2.7 Basic Data Types

The Basic Data Types data model defines basic UML data types used in CybOX. As stated in the **[UML 2.4.1]** 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**.

Note, that the use of UML data types from the Basic Data Types data model (e.g., `BasicString`, `Boolean`) is not the same as using UML data types defined as specializations of `BaseObjectPropertyType` (e.g., `StringObjectPropertyType`, `HexBinaryObjectPropertyType`, etc.). The latter data types permits the use of properties from `BaseObjectPropertyGroup` and `PatternFieldGroup`, which allow for a much richer characterization of cyber observables. Also, it's worth noting that not all data types defined in the Basic Data Types data model are used in CybOX.

2.7.1 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]**.

Table 2-1. 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).

2.7.2 Specializations of the BasicString Data Type

The data types in **Table 2-2** 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 2-2. 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]**.
`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]**.
`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]**.
`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]**.
`DateTime`	The `DateTime` data type is a restriction on the `BasicString` data type such that it adheres to the standard defined in **[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]**.
QualifiedName	The QualifiedName data type is a restriction on the BasicString data type such that it adheres to the requirements specified in **[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]**.

3 Data Model Conventions

The following general information and conventions are used to define the individual data models in *CybOX™ Version 2.1.1 Part 6: UML Model*. It should be noted that the CybOX 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.

3.1 UML Packages

Each CybOX data model is captured in a different UML package (e.g., Core package, File_Object 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 3-1** 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 3-1. Package prefixes used by the CybOX Language

Package	CybOX Core
Prefix	**cybox**
Description	The CybOX Core data model defines the main classes of the CybOX data model, such as ActionType, EventType, ObservableType and ObjectType.
Example	`cyboxCore:ObservableType`
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 Basic Data Types data model defines the types used within CybOX.
Example	`basicDataTypes:URI`

3.2 Naming Conventions

The UML classes, enumerations, and properties defined in CybOX follow the particular naming conventions outlined in **Table 3-2**.

Table 3-2. 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”	DateTimePrecisionEnum
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

3.3 Identifiers

Optional identifiers (IDs) can be assigned to several CybOX 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.

In CybOX v2.1.1, each CybOX 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 (“:”):

```
[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.

4 Relationships to Other Externally-defined Data Models

CybOX™ Version 2.1.1 leverages several other externally-defined data models that are relevant to the cyber threat domain. A listing of these other data models is given below.

Please see *CybOX™ Version 2.2.1 Part 4: Default Extensions* for further information on all of the externally-defined data models CybOX leverages by default (with the exception of CybOX, for which a different reference is given in Section Error! Reference source not found.).

4.1 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]**. By extending the CybOX 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.

4.2 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 CPE version 2.3 can be found at **[CPE].**

5 Conformance

Implementations have discretion over which parts (components, properties, extensions, controlled vocabularies, etc.) of CybOX they implement (e.g., Observable/Object).

[1] Conformant implementations must conform to all normative structural specifications of the UML model or additional normative statements within this document that apply to the portions of CybOX they implement (e.g., Implementers of the entire Observable class must conform to all normative structural specifications of the UML model regarding the Observable class or additional normative statements contained in the document that describes the Observable class).

[2] Conformant implementations are free to ignore normative structural specifications of the UML model or additional normative statements within this document that do not apply to the portions of CybOX they implement (e.g., Non-implementers of any particular properties of the Observable class are free to ignore all normative structural specifications of the UML model regarding those properties of the Observable class or additional normative statements contained in the document that describes the Observable class).

The conformance section of this document is intentionally broad and attempts to reiterate what already exists in this document.

Appendix A. Acknowledgments

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

Aetna
- David Crawford

AIT Austrian Institute of Technology
- Roman Fiedler
- Florian Skopik

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

Blue Coat Systems, Inc.
- Owen Johnson
- Bret Jordan

Century Link
- Cory Kennedy

CIRCL
- Alexandre Dulaunoy
- Andras Iklody
- Raphaël Vinot

Citrix Systems
- Joey Peloquin

Dell
- Will Urbanski
- Jeff Williams

DTCC
- Dan Brown
- Gordon Hundley
- Chris Koutras

EMC
- Robert Griffin
- Jeff Odom
- Ravi Sharda

Financial Services Information Sharing and Analysis Center (FS-ISAC)
- David Eilken
- Chris Ricard

Fortinet Inc.
- Gavin Chow
- Kenichi Terashita

Fujitsu Limited
- Neil Edwards

Airbus Group SAS
- Joerg Eschweiler
- Marcos Orallo

Anomali
- Ryan Clough
- Wei Huang
- Hugh Njemanze
- Katie Pelusi
- Aaron Shelmire
- Jason Trost

Bank of America
- Alexander Foley

Center for Internet Security (CIS)
- Sarah Kelley

Check Point Software Technologies
- Ron Davidson

Cisco Systems
- Syam Appala
- Ted Bedwell
- David McGrew
- Pavan Reddy
- Omar Santos
- Jyoti Verma

Cyber Threat Intelligence Network, Inc. (CTIN)
- Doug DePeppe
- Jane Ginn
- Ben Othman

DHS Office of Cybersecurity and Communications (CS&C)
- Richard Struse
- Marlon Taylor

EclecticIQ
- Marko Dragoljevic
- Joep Gommers
- Sergey Polzunov
- Rutger Prins
- Andrei Sîrghi
- Raymon van der Velde

Frederick Hirsch
Ryusuke Masuoka
Daisuke Murabayashi

Google Inc.
Mark Risher

Hitachi, Ltd.
Kazuo Noguchi
Akihito Sawada
Masato Terada

iboss, Inc.
Paul Martini

Individual
Jerome Athias
Peter Brown
Elysa Jones
Sanjiv Kalkar
Bar Lockwood
Terry MacDonald
Alex Pinto

Intel Corporation
Tim Casey
Kent Landfield

JPMorgan Chase Bank, N.A.
Terrence Driscoll
David Laurance

LookingGlass
Allan Thomson
Lee Vorthman

Mitre Corporation
Greg Back
Jonathan Baker
Sean Barnum
Desiree Beck
Nicole Gong
Jasen Jacobsen
Ivan Kirillov
Richard Piazza
Jon Salwen
Charles Schmidt
Emmanuelle Vargas-Gonzalez
John Wunder

National Council of ISACs (NCI)
Scott Algeier
Denise Anderson

eSentire, Inc.
Jacob Gajek

FireEye, Inc.
Phillip Boles
Pavan Gorakav
Anuj Kumar
Shyamal Pandya
Paul Patrick
Scott Shreve

Fox-IT
Sarah Brown

Georgetown University
Eric Burger

Hewlett Packard Enterprise (HPE)
Tomas Sander

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

IID
Chris Richardson

Integrated Networking Technologies, Inc.
Patrick Maroney

Johns Hopkins University Applied Physics Laboratory
Karin Marr
Julie Modlin
Mark Moss
Pamela Smith

Kaiser Permanente
Russell Culpepper
Beth Pumo

Lumeta Corporation
Brandon Hoffman

MTG Management Consultants, LLC.
James Cabral

National Security Agency
Mike Boyle
Jessica Fitzgerald-McKay

New Context Services, Inc.

Josh Poster

NEC Corporation

Takahiro Kakumaru

North American Energy Standards Board

David Darnell

Object Management Group

Cory Casanave

Palo Alto Networks

Vishaal Hariprasad

Queralt, Inc.

John Tolbert

Resilient Systems, Inc.

Ted Julian

Securonix

Igor Baikalov

Siemens AG

Bernd Grobauer

Soltra

John Anderson

Aishwarya Asok Kumar

Peter Ayasse

Jeff Beekman

Michael Butt

Cynthia Camacho

Aharon Chernin

Mark Clancy

Brady Cotton

Trey Darley

Mark Davidson

Paul Dion

Daniel Dye

Robert Hutto

Raymond Keckler

Ali Khan

Chris Kiehl

Clayton Long

Michael Pepin

Natalie Suarez

David Waters

Benjamin Yates

Symantec Corp.

Curtis Kostrosky

The Boeing Company

Crystal Hayes

John-Mark Gurney

Christian Hunt

James Moler

Daniel Riedel

Andrew Storms

OASIS

James Bryce Clark

Robin Cover

Chet Ensign

Open Identity Exchange

Don Thibeau

PhishMe Inc.

Josh Larkins

Raytheon Company-SAS

Daniel Wyschogrod

Retail Cyber Intelligence Sharing Center (R-CISC)

Brian Engle

Semper Fortis Solutions

Joseph Brand

Splunk Inc.

Cedric LeRoux

Brian Luger

Kathy Wang

TELUS

Greg Reaume

Alan Steer

Threat Intelligence Pty Ltd

Tyron Miller

Andrew van der Stock

ThreatConnect, Inc.

Wade Baker

Cole Iliff

Andrew Pendergast

Ben Schmoker

Jason Spies

TruSTAR Technology

Chris Roblee

United Kingdom Cabinet Office

Iain Brown

Adam Cooper

Mike McLellan

Chris O’Brien

James Penman

ThreatQuotient, Inc.
Ryan Trost
U.S. Bank
Mark Angel
Brad Butts
Brian Fay
Mona Magathan
Yevgen Sautin
US Department of Defense (DoD)
James Bohling
Eoghan Casey
Gary Katz
Jeffrey Mates
VeriSign
Robert Coderre
Kyle Maxwell
Eric Osterweil
Howard Staple
Chris Taylor
Laurie Thomson
Alastair Treharne
Julian White
Bethany Yates
US Department of Homeland Security
Evette Maynard-Noel
Justin Stekervetz
ViaSat, Inc.
Lee Chieffalo
Wilson Figueroa
Andrew May
Yaana Technologies, LLC
Anthony Rutkowski

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

Appendix B. Revision History

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