CybOXTM Version 2.1.1 Part 84: Win Registry Key Object

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:

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

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:ikirillov@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 01: Overview*. [URI]
* *CybOX™ Version 2.1.1 Part 02: Common*. [URI]
* *CybOX™ Version 2.1.1 Part 03: Core*. [URI]
* *CybOX™ Version 2.1.1 Part 04: Default Extensions*. [URI]
* *CybOX™ Version 2.1.1 Part 05: Default Vocabularies*. [URI]
* *CybOX™ Version 2.1.1 Part 06: UML Model*. [URI]
* *CybOX™ Version 2.1.1 Part 07: API Object*. [URI]
* *CybOX™ Version 2.1.1 Part 08: ARP Cache Object*. [URI]
* *CybOX™ Version 2.1.1 Part 09: AS Object*. [URI]
* *CybOXTM Version 2.1.1 Part 10: Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 11: Address Object*. [URI]
* *CybOXTM Version 2.1.1 Part 12: Archive File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 13: Artifact Object*. [URI]
* *CybOXTM Version 2.1.1 Part 14: Code Object*. [URI]
* *CybOXTM Version 2.1.1 Part 15: Custom Object*. [URI]
* *CybOXTM Version 2.1.1 Part 16: DNS Cache Object*. [URI]
* *CybOXTM Version 2.1.1 Part 17: DNS Query Object*. [URI]
* *CybOXTM Version 2.1.1 Part 18: DNS Record Object*. [URI]
* *CybOXTM Version 2.1.1 Part 19: Device Object*. [URI]
* *CybOXTM Version 2.1.1 Part 20: Disk Object*. [URI]
* *CybOXTM Version 2.1.1 Part 21: Disk Partition Object*. [URI]
* *CybOXTM Version 2.1.1 Part 22: Domain Name Object*. [URI]
* *CybOXTM Version 2.1.1 Part 23: Email Message Object*. [URI]
* *CybOXTM Version 2.1.1 Part 24: File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 25: GUI Dialogbox Object*. [URI]
* *CybOXTM Version 2.1.1 Part 26: GUI Object*. [URI]
* *CybOXTM Version 2.1.1 Part 27: GUI Window Object*. [URI]
* *CybOXTM Version 2.1.1 Part 28: HTTP Session Object*. [URI]
* *CybOXTM Version 2.1.1 Part 29: Hostname Object*. [URI]
* *CybOXTM Version 2.1.1 Part 30: Image File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 31: Library File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 32: Link Object*. [URI]
* *CybOXTM Version 2.1.1 Part 33: Linux Package Object*. [URI]
* *CybOXTM Version 2.1.1 Part 34: Memory Object*. [URI]
* *CybOXTM Version 2.1.1 Part 35: Mutex Object*. [URI]
* *CybOXTM Version 2.1.1 Part 36: Network Connection Object*. [URI]
* *CybOXTM Version 2.1.1 Part 37: Network Flow Object*. [URI]
* *CybOXTM Version 2.1.1 Part 38: Network Packet Object*. [URI]
* *CybOXTM Version 2.1.1 Part 39: Network Route Entry Object*. [URI]
* *CybOXTM Version 2.1.1 Part 40: Network Route Object*. [URI]
* *CybOXTM Version 2.1.1 Part 41: Network Socket Object*. [URI]
* *CybOXTM Version 2.1.1 Part 42: Network Subnet Object*. [URI]
* *CybOXTM Version 2.1.1 Part 43: PDF File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 44: Pipe Object*. [URI]
* *CybOXTM Version 2.1.1 Part 45: Port Object*. [URI]
* *CybOXTM Version 2.1.1 Part 46: Process Object*. [URI]
* *CybOXTM Version 2.1.1 Part 47: Product Object*. [URI]
* *CybOXTM Version 2.1.1 Part 48: SMS Message Object*. [URI]
* *CybOXTM Version 2.1.1 Part 49: Semaphore Object*. [URI]
* *CybOXTM Version 2.1.1 Part 50: Socket Address Object*. [URI]
* *CybOXTM Version 2.1.1 Part 51: System Object*. [URI]
* *CybOXTM Version 2.1.1 Part 52: URI Object*. [URI]
* *CybOXTM Version 2.1.1 Part 53: URL History Object*. [URI]
* *CybOXTM Version 2.1.1 Part 54: Unix File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 55: Unix Network Route Entry Object*. [URI]
* *CybOXTM Version 2.1.1 Part 56: Unix Pipe Object*. [URI]
* *CybOXTM Version 2.1.1 Part 57: Unix Process Object*. [URI]
* *CybOXTM Version 2.1.1 Part 58: Unix User Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 59: Unix Volume Object*. [URI]
* *CybOXTM Version 2.1.1 Part 60: User Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 61: User Session Object*. [URI]
* *CybOXTM Version 2.1.1 Part 62: Volume Object*. [URI]
* *CybOXTM Version 2.1.1 Part 63: Whois Object*. [URI]
* *CybOXTM Version 2.1.1 Part 64: Win Computer Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 65: Win Critical Section Object*. [URI]
* *CybOXTM Version 2.1.1 Part 66: Win Driver Object*. [URI]
* *CybOXTM Version 2.1.1 Part 67: Win Event Log Object*. [URI]
* *CybOXTM Version 2.1.1 Part 68: Win Event Object*. [URI]
* *CybOXTM Version 2.1.1 Part 69: Win Executable File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 70: Win File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 71: Win Filemapping Object*. [URI]
* *CybOXTM Version 2.1.1 Part 72: Win Handle Object*. [URI]
* *CybOXTM Version 2.1.1 Part 73: Win Hook Object*. [URI]
* *CybOXTM Version 2.1.1 Part 74: Win Kernel Hook Object*. [URI]
* *CybOXTM Version 2.1.1 Part 75: Win Kernel Object*. [URI]
* *CybOXTM Version 2.1.1 Part 76: Win Mailslot Object*. [URI]
* *CybOXTM Version 2.1.1 Part 77: Win Memory Page Region Object*. [URI]
* *CybOXTM Version 2.1.1 Part 78: Win Mutex Object*. [URI]
* *CybOXTM Version 2.1.1 Part 79: Win Network Route Entry Object*. [URI]
* *CybOXTM Version 2.1.1 Part 80: Win Network Share Object*. [URI]
* *CybOXTM Version 2.1.1 Part 81: Win Pipe Object*. [URI]
* *CybOXTM Version 2.1.1 Part 82: Win Prefetch Object*. [URI]
* *CybOXTM Version 2.1.1 Part 83: Win Process Object*. [URI]
* *CybOXTM Version 2.1.1 Part 84: Win Registry Key Object*. (this document)
* *CybOXTM Version 2.1.1 Part 85: Win Semaphore Object*. [URI]
* *CybOXTM Version 2.1.1 Part 86: Win Service Object*. [URI]
* *CybOXTM Version 2.1.1 Part 87: Win System Object*. [URI]
* *CybOXTM Version 2.1.1 Part 88: Win System Restore Object*. [URI]
* *CybOXTM Version 2.1.1 Part 89: Win Task Object*. [URI]
* *CybOXTM Version 2.1.1 Part 90: Win Thread Object*. [URI]
* *CybOXTM Version 2.1.1 Part 91: Win User Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 92: Win Volume Object*. [URI]
* *CybOXTM Version 2.1.1 Part 93: Win Waitable Timer Object*. [URI]
* *CybOXTM Version 2.1.1 Part 94: X509 Certificate Object*. [URI]

Related work:

This specification is related to:

* *STIXTM 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 specification document defines the Win Registry Key Object data model, which is one of the Object data models for CybOX content.

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/part1-overview/stix-v1.2.1-csd01-part1-overview.docx

Permanent “Latest version” URI:  
http://docs.oasis-open.org/cti/stix/v1.2.1/stix-v1.2.1-part1-overview.docx

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

Copyright © OASIS Open 2016. 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.

Portions copyright © United States Government 2012-2016.  All Rights Reserved.  
  
STIX™, TAXII™, AND CybOX™ (STANDARD OR STANDARDS) AND THEIR COMPONENT PARTS ARE PROVIDED “AS IS” WITHOUT ANY WARRANTY OF ANY KIND, EITHER EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY THAT THESE STANDARDS OR ANY OF THEIR COMPONENT PARTS WILL CONFORM TO SPECIFICATIONS, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR FREEDOM FROM INFRINGEMENT, ANY WARRANTY THAT THE STANDARDS OR THEIR COMPONENT PARTS WILL BE ERROR FREE, OR ANY WARRANTY THAT THE DOCUMENTATION, IF PROVIDED, WILL CONFORM TO THE STANDARDS OR THEIR COMPONENT PARTS. IN NO EVENT SHALL THE UNITED STATES GOVERNMENT OR ITS CONTRACTORS OR SUBCONTRACTORS BE LIABLE FOR ANY DAMAGES, INCLUDING, BUT NOT LIMITED TO, DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES, ARISING OUT OF, RESULTING FROM, OR IN ANY WAY CONNECTED WITH THESE STANDARDS OR THEIR COMPONENT PARTS OR ANY PROVIDED DOCUMENTATION, WHETHER OR NOT BASED UPON WARRANTY, CONTRACT, TORT, OR OTHERWISE, WHETHER OR NOT INJURY WAS SUSTAINED BY PERSONS OR PROPERTY OR OTHERWISE, AND WHETHER OR NOT LOSS WAS SUSTAINED FROM, OR AROSE OUT OF THE RESULTS OF, OR USE OF, THE STANDARDS, THEIR COMPONENT PARTS, AND ANY PROVIDED DOCUMENTATION. THE UNITED STATES GOVERNMENT DISCLAIMS ALL WARRANTIES AND LIABILITIES REGARDING THE STANDARDS OR THEIR COMPONENT PARTS ATTRIBUTABLE TO ANY THIRD PARTY, IF PRESENT IN THE STANDARDS OR THEIR COMPONENT PARTS AND DISTRIBUTES IT OR THEM “AS IS.”

Table of Contents

[1 Introduction 6](#_Toc450042463)

[1.1 CybOXTM Specification Documents 6](#_Toc450042464)

[1.2 Document Conventions 6](#_Toc450042465)

[1.2.1 Fonts 6](#_Toc450042466)

[1.2.2 UML Package References 7](#_Toc450042467)

[1.2.3 UML Diagrams 7](#_Toc450042468)

[1.2.3.1 Class Properties 7](#_Toc450042469)

[1.2.3.2 Diagram Icons and Arrow Types 7](#_Toc450042470)

[1.2.4 Property Table Notation 8](#_Toc450042471)

[1.2.5 Property and Class Descriptions 8](#_Toc450042472)

[1.3 Terminology 9](#_Toc450042473)

[1.4 Normative References 9](#_Toc450042474)

[2 Background Information 10](#_Toc450042475)

[2.1 Cyber Observables 10](#_Toc450042476)

[2.2 Objects 10](#_Toc450042477)

[3 Data Model 11](#_Toc450042478)

[3.1 WindowsRegistryKeyObjectType Class 11](#_Toc450042479)

[3.2 RegistryValueType Class 12](#_Toc450042480)

[3.3 RegistryDatatypeType Data Type 13](#_Toc450042481)

[3.4 RegistryHiveType Data Type 13](#_Toc450042482)

[3.5 RegistryValuesType Class 13](#_Toc450042483)

[3.6 RegistrySubkeysType Class 14](#_Toc450042484)

[3.7 RegistryDataTypesEnum Enumeration 14](#_Toc450042485)

[3.8 RegistryHiveEnum Enumeration 15](#_Toc450042486)

[4 Conformance 18](#_Toc450042487)

[Appendix A. Acknowledgments 19](#_Toc450042488)

[Appendix B. Revision History 20](#_Toc450042489)

# Introduction

[All text is normative unless otherwise labeled]

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 Win Registry Key Object Version 2.1.1 data model, which is one of eighty-eight CybOX Object data models.

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 Section **1.4**. In Section **2**, we give background information necessary to fully understand the Win Registry Key Object data model. We present the Win Registry Key Object data model specification details in Section **3** and conformance information in Section **4**.

## CybOXTM Specification Documents

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

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

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

The [*CybOX Version 2.1.1 Part 1: Overview*](#AdditionalArtifacts) document provides a comprehensive overview of the full set of CybOX data models. In addition to the Core, Common, and numerous Object data models, the full set of CybOX data models includes various extension data models and a vocabularies data model, which contains a set of default controlled vocabularies. [*CybOX Version 2.1.1 Part 1: Overview*](#AdditionalArtifacts) also summarizes the relationship of CybOX to other languages, and outlines general CybOX data model conventions.

## Document Conventions

The following sections describe the conventions 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*](file:///C:\Users\drothenberg\Documents\GitHub\cyboxproject\specifications\documents\cybox-v2.1.1-wd01-part43-pdf-file-object.docx#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. 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 package\_prefix for the Windows Registry Key data model is WinRegistryKeyObj. Note that in this specification document, we do not explicitly specify the package prefix for any classes that originate from the Win Registry Key Object 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 ‑. UML diagram icons

|  |  |
| --- | --- |
| **Icon** | **Description** |
|  | This diagram icon indicates a class. If the name is in italics, it is an abstract class. |
|  | This diagram icon indicates an enumeration. |
|  | This diagram icon indicates a data type. |
|  | This decorator icon indicates an attribute of a class. The green circle means its visibility is public. If the circle is red or yellow, it means its visibility is private or protected. |
|  | This decorator icon indicates an enumeration literal. |
|  | This arrow type indicates a directed association relationship. |
|  | This arrow type indicates a generalization relationship. |

### 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 Win Registry Key Object 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 preferred to use the three verbs, defined as follows, in class and property descriptions:

|  |  |
| --- | --- |
| **Verb** | **CybOX Definition** |
| captures | Used to record and preserve information without implying anything about the structure of a class or property. Often used for properties that encompass general content. This is the least precise of the three verbs. |
|  | *Examples*:  The Observable\_Source property characterizes the source of the Observable information. Examples of details captured include identifying characteristics, time-related attributes, and a list of the tools used to collect the information.  The Description property captures a textual description of the Action. |
| characterizes | Describes the distinctive nature or features of a class or property. Often used to describe classes and properties that themselves comprise one or more other properties. |
|  | *Examples*:  The Action property characterizes a cyber observable Action.  The Obfuscation\_Technique property characterizes a technique an attacker could potentially leverage to obfuscate the Observable. |
| specifies | Used to clearly and precisely identify particular instances or values associated with a property. Often used for properties that are defined by a controlled vocabulary or enumeration; typically used for properties that take on only a single value. |
|  | *Example*:  The cybox\_major\_version property specifies the major version of the CybOX language used for the set of Observables. |

## 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>.

# Background Information

In this section, we provide high level information about the Win Registry Key Object 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.

## Objects

Cyber observable objects (Files, IP Addresses, etc) in CybOX are characterized with a combination of two levels of data models.

The first level is the Object data model which specifies a base set of properties universal to all types of Objects and enables them to integrate with the overall cyber observable framework specified in the CybOX Core data model.

The second level are the object property models which specify the properties of a particular type of Object via individual data models each focused on a particular cyber entity, such as a Windows registry key, or an Email Message. Accordingly, each release of the CybOX language includes a particular set of Objects that are part of the release. The data model for each of these Objects is defined by its own specification that describes the context-specific classes and properties that compose the Object.

Any specific instance of an Object is represented utilizing the particular object properties data model within the general Object data model.

# Data Model

## WindowsRegistryKeyObjectType Class

The WindowsRegistryKeyObjectType class is intended to characterize Windows registry objects, including Keys and Key/Value pairs.

The UML diagram corresponding to the WindowsRegistryKeyObjectType class is shown in **Figure 3‑1**.

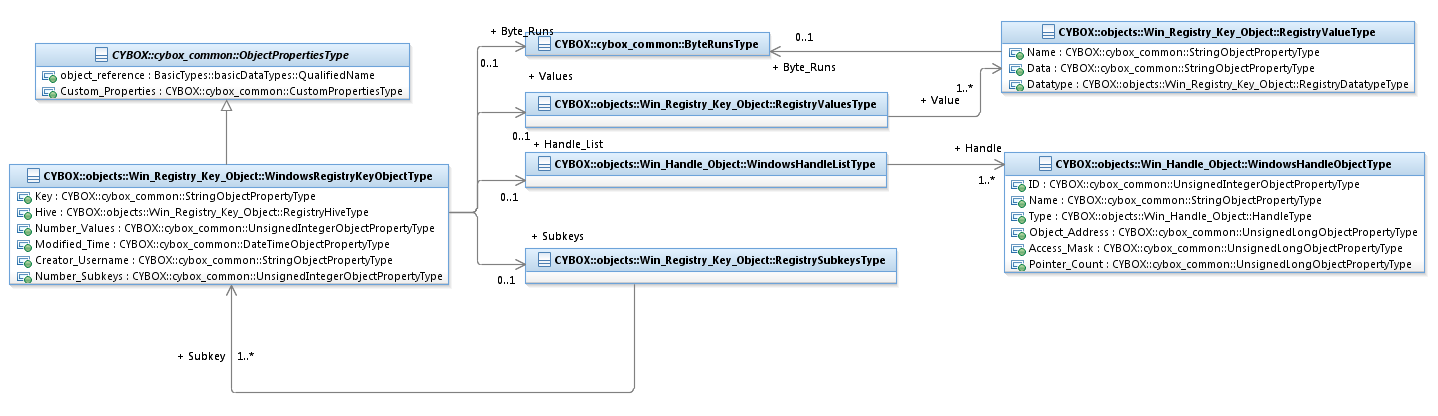


Figure ‑. UML diagram of the WindowsRegistryKeyObjectType class

The property table of the WindowsRegistryKeyObjectType class is given in **Table 3‑1**.

Table ‑. Properties of the WindowsRegistryKeyObjectType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Key** | cyboxCommon:  StringObjectPropertyType | 0..1 | The Key property specifies the full key to the Windows registry object, not including the hive. |
| **Hive** | RegistryHiveType | 0..1 | The Hive property specifies the Windows registry hive to which the registry object belongs to. |
| **Number\_Values** | cyboxCommon:  UnsignedIntegerObjectPropertyType | 0..1 | The Number\_Values property specifies the number of values found in the registry key. |
| **Values** | RegistryValuesType | 0..1 | The Values property specifies the values (with their name/data pairs) held within the registry key. |
| **Modified\_Time** | cyboxCommon:  DateTimeObjectPropertyType | 0..1 | The Modified\_Time property specifies the last date/time that the registry object was modified. |
| **Creator\_Username** | cyboxCommon:  StringObjectPropertyType | 0..1 | The Creator\_Username property specifies the name of the user who created the registry object. |
| **Handle\_List** | WinHandleObj:  WindowsHandleListType | 0..1 | The Handle\_List property specifies a list of open Handles for this registry object. |
| **Number\_Subkeys** | cyboxCommon:  UnsignedIntegerObjectPropertyType | 0..1 | The Number\_Subkeys property specifies the number of subkeys contained under the registry key. |
| **Subkeys** | RegistrySubkeysType | 0..1 | The Subkeys property specifies the set of subkeys contained under the registry key. |
| **Byte\_Runs** | cyboxCommon:ByteRunsType | 0..1 | The Byte\_Runs property contains a list of byte runs from the raw registry. |

## RegistryValueType Class

The RegistryValueType class is intended to characterize Windows Registry Value name/data pairs.

The property table of the RegistryValueType class is given in **Table 3‑2**.

Table ‑. Properties of the RegistryValueType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Name** | cyboxCommon:  StringObjectPropertyType | 0..1 | The Name property specifies the name of the registry value. For specifying the default value in a registry key, an empty string should be used. |
| **Data** | cyboxCommon:  StringObjectPropertyType | 0..1 | The Data property specifies the data contained in the registry value. |
| **Datatype** | RegistryDatatypeType | 0..1 | The Datatype property specifies the registry (REG\_\*) datatype used in the registry value. |
| **Byte\_Runs** | cyboxCommon:  ByteRunsType | 0..1 | The Byte\_Runs property contains a list of byte runs from the raw registry key entry. |

## RegistryDatatypeType Data Type

The RegistryDatatypeType data type specifies the Windows registry datatype. Its core value SHOULD be a literal from the RegistryDataTypesEnum enumeration. It extends the BaseObjectPropertyType data type, in order to permit complex (i.e., regular-expression based) specifications.

## RegistryHiveType Data Type

The RegistryHiveType data type specifies the Windows registry hive type. Its core value SHOULD be a literal from the RegistryHiveEnum enumeration. It extends the BaseObjectPropertyType data type, in order to permit complex (i.e., regular-expression based) specifications.

## RegistryValuesType Class

The RegistryValuesType class specifies the values (with their name/data pairs) held within the registry key.

The property table of the RegistryValuesType class is given in **Table 3‑3**.

Table ‑. Properties of the RegistryValuesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Value** | RegistryValueType | 1..\* | The Value property specifies the value (with name/data pair) held within the registry key. |

## RegistrySubkeysType Class

The RegistrySubkeysType class specifies the set of subkeys contained under the registry key.

The property table of the RegistrySubkeysType class is given in **Table 3‑4**.

Table ‑. Properties of the RegistrySubkeysType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Subkey** | WindowsRegistryKeyObjectType | 1..\* | The Subkey property specifies a single subkey contained under the registry key. |

## RegistryDataTypesEnum Enumeration

The literals of the RegistryDataTypesEnum enumeration are given in **Table 3‑5**. For more information, see <https://msdn.microsoft.com/en-us/library/windows/desktop/ms724884%28v=vs.85%29.aspx>.

Table ‑. Literals of the RegistryDataTypesEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **REG\_NONE** | No defined value type. |
| **REG\_SZ** | A null-terminated string. This will be either a Unicode or an ANSI string, depending on whether you use the Unicode or ANSI functions. |
| **REG\_EXPAND\_SZ** | A null-terminated string that contains unexpanded references to environment variables (for example, "%PATH%"). It will be a Unicode or ANSI string depending on whether you use the Unicode or ANSI functions. |
| **REG\_BINARY** | Binary data in any form. |
| **REG\_DWORD** | A 32-bit number. |
| **REG\_DWORD\_BIG\_ENDIAN** | A 32-bit number in big-endian format. Some UNIX systems support big-endian architectures. |
| **REG\_LINK** | A null-terminated Unicode string that contains the target path of a symbolic link. |
| **REG\_MULTI\_SZ** | A sequence of null-terminated strings, terminated by an empty string (\0). |
| **REG\_RESOURCE\_LIST** | A series of nested arrays designed to store a resource list used by a hardware device driver or one of the physical devices it controls. This data is detected and written into the ResourceMap tree by the system and is displayed in Registry Editor in hexadecimal format as a Binary Value. |
| **REG\_FULL\_RESOURCE\_DESCRIPTOR** | A series of nested arrays designed to store a resource list used by a physical hardware device. This data is detected and written into the HardwareDescription tree by the system and is displayed in Registry Editor in hexadecimal format as a Binary Value. |
| **REG\_RESOURCE\_REQUIREMENTS\_LIST** | Device driver list of hardware resource requirements in Resource Map tree. See <http://www.mdgx.com/reg.htm>. |
| **REG\_QWORD** | A 64-bit number. |
| **REG\_INVALID\_TYPE** | Specifies an invalid key. |

## RegistryHiveEnum Enumeration

The literals of the RegistryHiveEnum enumeration are given in **Table 3‑6**.

Also, see <https://msdn.microsoft.com/en-us/library/windows/desktop/ms724836%28v=vs.85%29.aspx>.

Table ‑. Literals of the RegistryHiveEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **HKEY\_CLASSES\_ROOT** | Registry entries subordinate to this key define types (or classes) of documents and the properties associated with those types. Shell and COM applications use the information stored under this key. |
| **HKEY\_CURRENT\_CONFIG** | Contains information about the current hardware profile of the local computer system. The information under HKEY\_CURRENT\_CONFIG describes only the differences between the current hardware configuration and the standard configuration. |
| **HKEY\_CURRENT\_USER** | Registry entries subordinate to this key define the preferences of the current user. These preferences include the settings of environment variables, data about program groups, colors, printers, network connections, and application preferences. This key makes it easier to establish the current user's settings; the key maps to the current user's branch in HKEY\_USERS. |
| **HKEY\_LOCAL\_MACHINE** | Registry entries subordinate to this key define the physical state of the computer, including data about the bus type, system memory, and installed hardware and software. |
| **HKEY\_USERS** | Registry entries subordinate to this key define the default user configuration for new users on the local computer and the user configuration for the current user. |
| **HKEY\_CURRENT\_USER\_**  **LOCAL\_SETTINGS** | Registry entries subordinate to this key define preferences of the current user that are local to the machine. These entries are not included in the per-user registry portion of a roaming user profile. |
| **HKEY\_PERFORMANCE\_DATA** | Registry entries subordinate to this key allow you to access performance data. The data is not actually stored in the registry; the registry functions cause the system to collect the data from its source. |
| **HKEY\_PERFORMANCE\_NLSTEXT** | Registry entries subordinate to this key reference the text strings that describe counters in the local language of the area in which the computer system is running. These entries are not available to Regedit.exe and Regedt32.exe. |
| **HKEY\_PERFORMANCE\_TEXT** | Registry entries subordinate to this key reference the text strings that describe counters in US English. These entries are not available to Regedit.exe and Regedt32.exe. |

# 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 |