CybOX™ Version 2.1.1. Part 84: Win Registry Key Object

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/part84-win-registry-key/cybox-v2.1.1-csprd01-part84-win-registry-key.docx (Authoritative)
http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part84-win-registry-key/cybox-v2.1.1-csprd01-part84-win-registry-key.html
http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part84-win-registry-key/cybox-v2.1.1-csprd01-part84-win-registry-key.pdf

Previous version:
N/A

Latest version:
http://docs.oasis-open.org/cti/cybox/v2.1.1/part84-win-registry-key/cybox-v2.1.1-part84-win-registry-key.docx (Authoritative)
http://docs.oasis-open.org/cti/cybox/v2.1.1/part84-win-registry-key/cybox-v2.1.1-part84-win-registry-key.html
http://docs.oasis-open.org/cti/cybox/v2.1.1/part84-win-registry-key/cybox-v2.1.1-part84-win-registry-key.pdf

Technical Committee:
OASIS Cyber Threat Intelligence (CTI) TC

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

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

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

Related work:
This specification is related to:
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 document was last revised or approved by the OASIS Cyber Threat Intelligence (CTI) TC on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=cti#technical.

TC members should send comments on this specification to the TC’s email list. Others should send comments to the TC’s public comment list, after subscribing to it by following the instructions at the “Send A Comment” button on the TC’s web page at https://www.oasis-open.org/committees/cti/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC’s web page (https://www.oasis-open.org/committees/cti/ipr.php).

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

[CybOX-v2.1.1-win-registry-key]
Notices

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 may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an “AS IS” basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS’ procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name “OASIS” is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark for above guidance.

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
1.1 CybOX™ Specification Documents .......................................................................................... 6
1.2 Document Conventions ............................................................................................................ 6
   1.2.1 Fonts........................................................................................................................................... 6
   1.2.2 UML Package References ...................................................................................................... 7
   1.2.3 UML Diagrams....................................................................................................................... 7
   1.2.4 Property Table Notation .......................................................................................................... 8
   1.2.5 Property and Class Descriptions .......................................................................................... 8
1.3 Terminology ...................................................................................................................................... 9
1.4 Normative References .................................................................................................................. 9
2 Background Information ........................................................................................................................... 10
   2.1 Cyber Observables .................................................................................................................... 10
   2.2 Objects ......................................................................................................................................... 10
3 Data Model ............................................................................................................................................ 11
   3.1 WindowsRegistryKeyType Object Class ................................................................................ 11
   3.2 RegistryValueType Class ............................................................................................................ 13
   3.3 RegistryDatatypeType Data Type ............................................................................................ 13
   3.4 RegistryHiveType Data Type .................................................................................................... 13
   3.5 RegistryValuesType Class ......................................................................................................... 14
   3.6 RegistrySubkeysType Class ....................................................................................................... 14
   3.7 RegistryDatatypesEnum Enumeration ..................................................................................... 14
   3.8 RegistryHiveEnum Enumeration .............................................................................................. 16
4 Conformance ......................................................................................................................................... 18
Appendix A. Acknowledgments .................................................................................................................. 19
Appendix B. Revision History .................................................................................................................... 23
1 Introduction

[All text is normative unless otherwise labeled]

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

This document serves as the specification for the CybOX 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.

1.1 CybOX™ Specification Documents

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

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

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

The CybOX Version 2.1.1 Part 1: Overview document provides a comprehensive overview of the full set of CybOX data models. 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 also summarizes the relationship of CybOX to other languages, and outlines general CybOX data model conventions.

1.2 Document Conventions

The following sections describe the conventions 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.

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.

1.2.3 UML Diagrams

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

In UML diagrams, classes are often presented with their attributes elided, to avoid clutter. The fully described class can usually be found in a related diagram. A class presented with an empty section at the bottom of the icon indicates that there are no attributes other than those that are visualized using associations.

1.2.3.1 Class Properties

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

1.2.3.2 Diagram Icons and Arrow Types

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

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Class Icon]</td>
<td>This diagram icon indicates a class. If the name is in italics, it is an abstract class.</td>
</tr>
<tr>
<td>![Enumeration Icon]</td>
<td>This diagram icon indicates an enumeration.</td>
</tr>
<tr>
<td>![DataType Icon]</td>
<td>This diagram icon indicates a data type.</td>
</tr>
<tr>
<td>![Attribute Icon]</td>
<td>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.</td>
</tr>
<tr>
<td>![Enumeration Literal Icon]</td>
<td>This decorator icon indicates an enumeration literal.</td>
</tr>
<tr>
<td>![Association Arrow]</td>
<td>This arrow type indicates a directed association relationship.</td>
</tr>
<tr>
<td>![Generalization Arrow]</td>
<td>This arrow type indicates a generalization relationship.</td>
</tr>
</tbody>
</table>

### 1.2.4 Property Table Notation

Throughout Section 3, tables are used to describe the properties of each data model class. Each property table consists of a column of names to identify the property, a type column to reflect the datatype of the property, a multiplicity column to reflect the allowed number of occurrences of the property, and a description column that describes the property. Package prefixes are provided for classes outside of the 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.

### 1.2.5 Property and Class Descriptions

Each class and property defined in CybOX is described using the format, “The X property verb Y.” For example, in the specification for the CybOX Core data model, we write, “The id property specifies a globally unique identifier for the Action.” In fact, the verb “specifies” could have been replaced by any number of alternatives: “defines,” “describes,” “contains,” “references,” etc.

However, we thought that using a wide variety of verb phrases might confuse a reader of a specification document because the meaning of each verb could be interpreted slightly differently. On the other hand, we didn’t want to use a single, generic verb, such as “describes,” because although the different verb choices may or may not be meaningful from an implementation standpoint, a distinction could be useful to those interested in the modeling aspect of CybOX.
Consequently, we have preferred to use the three verbs, defined as follows, in class and property descriptions:

<table>
<thead>
<tr>
<th>Verb</th>
<th>CybOX Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>captures</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>characterizes</th>
<th>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.</th>
</tr>
</thead>
</table>

**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.

<table>
<thead>
<tr>
<th>specifies</th>
<th>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.</th>
</tr>
</thead>
</table>

**Example:**
The cybox_major_version property specifies the major version of the CybOX language used for the set of Observables.

### 1.3 Terminology

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

### 1.4 Normative References

2 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.

2.1 Cyber Observables

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

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

2.2 Objects

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.
3 Data Model

3.1 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.

![UML diagram of the WindowsRegistryKeyObjectType class](image)

Figure 3-1. UML diagram of the WindowsRegistryKeyObjectType class

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3-1. Properties of the WindowsRegistryKeyObjectType class
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>cyboxCommon:StringObjectPropertyType</td>
<td>0..1 The <strong>Key</strong> property specifies the full key to the Windows registry object, not including the hive.</td>
</tr>
<tr>
<td>Hive</td>
<td>RegistryHiveType</td>
<td>0..1 The <strong>Hive</strong> property specifies the Windows registry hive to which the registry object belongs to.</td>
</tr>
<tr>
<td>Number_Values</td>
<td>cyboxCommon:UnsignedIntegerObjectPropertyType</td>
<td>0..1 The <strong>Number_Values</strong> property specifies the number of values found in the registry key.</td>
</tr>
<tr>
<td>Values</td>
<td>RegistryValuesType</td>
<td>0..1 The <strong>Values</strong> property specifies the values (with their name/data pairs) held within the registry key.</td>
</tr>
<tr>
<td>Modified_Time</td>
<td>cyboxCommon:DateTimeObjectPropertyType</td>
<td>0..1 The <strong>Modified_Time</strong> property specifies the last date/time that the registry object was modified.</td>
</tr>
<tr>
<td>Creator_Username</td>
<td>cyboxCommon:StringObjectPropertyType</td>
<td>0..1 The <strong>Creator_Username</strong> property specifies the name of the user who created the registry object.</td>
</tr>
<tr>
<td>Handle_List</td>
<td>WinHandleObj:WindowsHandleListType</td>
<td>0..1 The <strong>Handle_List</strong> property specifies a list of open Handles for this registry object.</td>
</tr>
<tr>
<td>Number_Subkeys</td>
<td>cyboxCommon:UnsignedIntegerObjectPropertyType</td>
<td>0..1 The <strong>Number_Subkeys</strong> property specifies the number of subkeys contained under the registry key.</td>
</tr>
<tr>
<td>Subkeys</td>
<td>RegistrySubkeysType</td>
<td>0..1 The <strong>Subkeys</strong> property specifies the set of subkeys contained under the registry key.</td>
</tr>
<tr>
<td>Byte_Runs</td>
<td>cyboxCommon:ByteRunsType</td>
<td>0..1 The <strong>Byte_Runs</strong> property contains a list of byte runs from the raw registry.</td>
</tr>
</tbody>
</table>
3.2 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>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>cyboxCommon: StringObjectPropertyType</td>
<td>0..1</td>
<td>The Name property specifies the name of the registry value. For specifying</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the default value in a registry key, an empty string should be used.</td>
</tr>
<tr>
<td>Data</td>
<td>cyboxCommon: StringObjectPropertyType</td>
<td>0..1</td>
<td>The Data property specifies the data contained in the registry value.</td>
</tr>
<tr>
<td>Datatype</td>
<td>RegistryDatatypeType</td>
<td>0..1</td>
<td>The Datatype property specifies the registry (REG_*) datatype used in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>registry value.</td>
</tr>
<tr>
<td>Byte_Runs</td>
<td>cyboxCommon: ByteRunsType</td>
<td>0..1</td>
<td>The Byte_Runs property contains a list of byte runs from the raw registry</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>key entry.</td>
</tr>
</tbody>
</table>

3.3 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.

3.4 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.
3.5 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>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>RegistryValueType</td>
<td>1..*</td>
<td>The <code>Value</code> property specifies the value (with name/data pair) held within the registry key.</td>
</tr>
</tbody>
</table>

3.6 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>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subkey</td>
<td>WindowsRegistryKeyObjectType</td>
<td>1..*</td>
<td>The <code>Subkey</code> property specifies a single subkey contained under the registry key.</td>
</tr>
</tbody>
</table>

3.7 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](https://msdn.microsoft.com/en-us/library/windows/desktop/ms724884%28v=vs.85%29.aspx).

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration Literal</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>REG_NONE</td>
<td>No defined value type.</td>
<td></td>
</tr>
<tr>
<td>REG_SZ</td>
<td>A null-terminated string. This will be either a Unicode or an ANSI string, depending on whether you use the Unicode or ANSI functions.</td>
<td></td>
</tr>
<tr>
<td>REG_EXPAND_SZ</td>
<td>A null-terminated string that contains unexpanded references to environment variables (for example, &quot;%PATH%&quot;). It will be a Unicode or ANSI string depending on whether you use the Unicode or ANSI functions.</td>
<td></td>
</tr>
<tr>
<td>REG_BINARY</td>
<td>Binary data in any form.</td>
<td></td>
</tr>
<tr>
<td>REG_DWORD</td>
<td>A 32-bit number.</td>
<td></td>
</tr>
<tr>
<td>REG_DWORD_BIG_ENDIAN</td>
<td>A 32-bit number in big-endian format. Some UNIX systems support big-endian architectures.</td>
<td></td>
</tr>
<tr>
<td>REG_LINK</td>
<td>A null-terminated Unicode string that contains the target path of a symbolic link.</td>
<td></td>
</tr>
<tr>
<td>REG_MULTI_SZ</td>
<td>A sequence of null-terminated strings, terminated by an empty string (\0).</td>
<td></td>
</tr>
<tr>
<td>REGRESOURCE_LIST</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>
### 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

### REG_QWORD
A 64-bit number.

### REG_INVALID_TYPE
Specifies an invalid key.

### 3.8 RegistryHiveEnum Enumeration
The literals of the `RegistryHiveEnum` enumeration are given in **Table 3-6**.


**Table 3-6. Literals of the `RegistryHiveEnum` enumeration**

<table>
<thead>
<tr>
<th>Enumeration Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKEY_CLASSES_ROOT</td>
<td>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.</td>
</tr>
<tr>
<td>HKEY_CURRENT_CONFIG</td>
<td>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</td>
</tr>
<tr>
<td>Registry Key</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HKEY_CURRENT_USER</td>
<td>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.</td>
</tr>
<tr>
<td>HKEY_LOCAL_MACHINE</td>
<td>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.</td>
</tr>
<tr>
<td>HKEY_USERS</td>
<td>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.</td>
</tr>
<tr>
<td>HKEY_CURRENT_USER_LOCAL_SETTINGS</td>
<td>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.</td>
</tr>
<tr>
<td>HKEY_PERFORMANCE_DATA</td>
<td>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.</td>
</tr>
<tr>
<td>HKEY_PERFORMANCE_NLSTEXT</td>
<td>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.</td>
</tr>
<tr>
<td>HKEY_PERFORMANCE_TEXT</td>
<td>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.</td>
</tr>
</tbody>
</table>
4 Conformance

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

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

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

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

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 Dragoljivic
   Joep Gommers
   Sergey Polzunov
The authors would also like to thank the larger CybOX Community for its input and help in reviewing this document.
## Appendix B. Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Editor</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>wd01</td>
<td>15 December 2015</td>
<td>Desiree Beck, Trey Darley, Ivan Kirillov, Rich Piazza</td>
<td>Initial transfer to OASIS template</td>
</tr>
</tbody>
</table>