THE MITRE CORPORATION

CybOX™ 2.1

Common Specification (v2.1)

July 14, 2015

*The Cyber Observable eXpression (CybOXTM) is a standardized language for encoding and communicating high-fidelity information about cyber observables, whether dynamic events or stateful measures that are observable in the operational cyber domain. By specifying a common structured schematic mechanism for these cyber observables, the intent is to enable the potential for detailed automatable sharing, mapping, detection and analysis heuristics. This specification document defines the Common data model, which is one of the fundamental data models for CybOX content.*

**Acknowledgements**

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

**Trademark Information**

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

**Warnings**

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

**Feedback**

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

Table of Contents

[1. Introduction 1](#_Toc425409603)

[1.1. CybOX Specification Documents 1](#_Toc425409604)

[1.2. Document Conventions 1](#_Toc425409605)

[1.1.1. Keywords 2](#_Toc425409606)

[1.1.2. Fonts 2](#_Toc425409607)

[1.1.3. UML Package References 2](#_Toc425409608)

[1.1.4. UML Diagrams 2](#_Toc425409609)

[1.1.5. Property Table Notation 4](#_Toc425409610)

[1.1.6. Property and Class Descriptions 4](#_Toc425409611)

[2. Background Information 6](#_Toc425409612)

[3. Data Models 7](#_Toc425409613)

[1.3. MeasureSourceType Class 7](#_Toc425409614)

[1.4. ContributorType Class 10](#_Toc425409615)

[1.5. DateRangeType Class 10](#_Toc425409616)

[1.6. PersonnelType Class 11](#_Toc425409617)

[1.7. TimeType Class 11](#_Toc425409618)

[1.8. ToolSpecificDataType Class 13](#_Toc425409619)

[1.9. ToolsInformationType Class 13](#_Toc425409620)

[1.10. ToolInformationType Class 13](#_Toc425409621)

[1.11. CompensationModelType Class 16](#_Toc425409622)

[1.12. ToolReferencesType Class 16](#_Toc425409623)

[1.13. ToolReferenceType Class 16](#_Toc425409624)

[1.14. ToolConfigurationType Class 17](#_Toc425409625)

[1.15. ConfigurationSettingsType Class 17](#_Toc425409626)

[1.16. ConfigurationSettingType Class 18](#_Toc425409627)

[1.17. DependenciesType Class 18](#_Toc425409628)

[1.18. DependencyType Class 19](#_Toc425409629)

[1.19. UsageContextAssumptionsType Class 19](#_Toc425409630)

[1.20. InternationalizationSettingsType Class 20](#_Toc425409631)

[1.21. InternalStringsType Class 20](#_Toc425409632)

[1.22. BuildInformationType Class 21](#_Toc425409633)

[1.23. BuildUtilityType Class 22](#_Toc425409634)

[1.24. CompilersType Class 22](#_Toc425409635)

[1.25. CompilerType Class 23](#_Toc425409636)

[1.26. CompilerInformalDescriptionType Class 23](#_Toc425409637)

[1.27. BuildConfigurationType Class 24](#_Toc425409638)

[1.28. LibrariesType Class 24](#_Toc425409639)

[1.29. LibraryType Class 25](#_Toc425409640)

[1.30. ExecutionEnvironmentType Class 25](#_Toc425409641)

[1.31. ErrorsType Class 26](#_Toc425409642)

[1.32. ErrorType Class 26](#_Toc425409643)

[1.33. ErrorInstancesType Class 27](#_Toc425409644)

[1.34. ObjectPropertiesType Class 27](#_Toc425409645)

[1.35. CustomPropertiesType Class 28](#_Toc425409646)

[1.36. PropertyType Class 28](#_Toc425409647)

[1.37. BaseObjectPropertyType Class 29](#_Toc425409648)

[1.38. IntegerObjectPropertyType Class 33](#_Toc425409649)

[1.39. StringObjectPropertyType Class 33](#_Toc425409650)

[1.40. NameObjectPropertyType Class 34](#_Toc425409651)

[1.41. DateObjectPropertyRestrictionType Class 34](#_Toc425409652)

[1.42. DateObjectPropertyType Class 34](#_Toc425409653)

[1.43. DateTimeObjectPropertyRestrictionType Class 35](#_Toc425409654)

[1.44. DateTimeObjectPropertyType Class 36](#_Toc425409655)

[1.45. FloatObjectPropertyType Class 37](#_Toc425409656)

[1.46. DoubleObjectPropertyType Class 37](#_Toc425409657)

[1.47. UnsignedLongObjectPropertyType Class 38](#_Toc425409658)

[1.48. UnsignedIntegerObjectPropertyType Class 38](#_Toc425409659)

[1.49. PositiveIntegerObjectPropertyType Class 39](#_Toc425409660)

[1.50. HexBinaryObjectPropertyType Class 39](#_Toc425409661)

[1.51. LongObjectPropertyType Class 40](#_Toc425409662)

[1.52. NonNegativeIntegerObjectPropertyType Class 40](#_Toc425409663)

[1.53. AnyURIObjectPropertyType Class 41](#_Toc425409664)

[1.54. DurationObjectPropertyType Class 41](#_Toc425409665)

[1.55. TimeObjectPropertyRestrictionType Class 41](#_Toc425409666)

[1.56. TimeObjectPropertyType Class 42](#_Toc425409667)

[1.57. Base64BinaryObjectPropertyType Class 43](#_Toc425409668)

[1.58. BaseObjectPropertyGroup Class 43](#_Toc425409669)

[1.59. PatternFieldGroup Class 46](#_Toc425409670)

[1.60. LocationType Class 50](#_Toc425409671)

[1.61. ExtractedFeaturesType Class 50](#_Toc425409672)

[1.62. ExtractedStringsType Class 51](#_Toc425409673)

[1.63. ExtractedStringType Class 51](#_Toc425409674)

[1.64. ImportsType Class 53](#_Toc425409675)

[1.65. FunctionsType Class 53](#_Toc425409676)

[1.66. CodeSnippetsType Class 54](#_Toc425409677)

[1.67. ByteRunsType Class 54](#_Toc425409678)

[1.68. ByteRunType Class 55](#_Toc425409679)

[1.69. HashListType Class 56](#_Toc425409680)

[1.70. HashValueType Class 56](#_Toc425409681)

[1.71. SimpleHashValueType Class 56](#_Toc425409682)

[1.72. FuzzyHashValueType Class 57](#_Toc425409683)

[1.73. FuzzyHashStructureType Class 57](#_Toc425409684)

[1.74. FuzzyHashBlockType Class 58](#_Toc425409685)

[1.75. HashSegmentsType Class 58](#_Toc425409686)

[1.76. HashSegmentType Class 59](#_Toc425409687)

[1.77. HashType Class 59](#_Toc425409688)

[1.78. StructuredTextType Class 60](#_Toc425409689)

[1.79. DataSegmentType Class 61](#_Toc425409690)

[1.80. DataSizeType Class 62](#_Toc425409691)

[1.81. PlatformSpecificationType Class 62](#_Toc425409692)

[1.82. PlatformIdentifierType Class 63](#_Toc425409693)

[1.83. MetadataType Class 64](#_Toc425409694)

[1.84. EnvironmentVariableListType Class 64](#_Toc425409695)

[1.85. EnvironmentVariableType Class 64](#_Toc425409696)

[1.86. DigitalSignaturesType Class 65](#_Toc425409697)

[1.87. DigitalSignatureInfoType Class 65](#_Toc425409698)

[1.88. PatternableFieldType Class 66](#_Toc425409699)

[1.89. ControlledVocabularyStringType Class 68](#_Toc425409700)

[1.90. DateWithPrecisionType Class 68](#_Toc425409701)

[1.91. DateTimeWithPrecisionType Class 69](#_Toc425409702)

[1.92. SIDType Class 69](#_Toc425409703)

[1.93. Layer4ProtocolType Class 70](#_Toc425409704)

[1.94. EndiannessType Class 70](#_Toc425409705)

[1.95. CipherType Class 70](#_Toc425409706)

[1.96. RegionalRegistryType Class 70](#_Toc425409707)

[1.97. SourceClassTypeEnum Class 71](#_Toc425409708)

[1.98. SourceTypeEnum Class 71](#_Toc425409709)

[1.99. CompensationModelEnum Class 72](#_Toc425409710)

[1.100. ToolReferenceTypeEnum Class 73](#_Toc425409711)

[1.101. ConditionTypeEnum Class 73](#_Toc425409712)

[1.102. ConditionApplicationEnum Class 76](#_Toc425409713)

[1.103. DatatypeEnum Class 76](#_Toc425409714)

[1.104. PatternTypeEnum Class 82](#_Toc425409715)

[1.105. DataFormatEnum Class 82](#_Toc425409716)

[1.106. DataSizeUnitsEnum Class 83](#_Toc425409717)

[1.107. DatePrecisionEnum Class 83](#_Toc425409718)

[1.108. TimePrecisionEnum Class 84](#_Toc425409719)

[1.109. SIDTypeEnum Class 84](#_Toc425409720)

[1.110. Layer4ProtocolEnum Class 85](#_Toc425409721)

[1.111. EndiannessTypeEnum Class 86](#_Toc425409722)

[1.112. CipherEnum Class 86](#_Toc425409723)

[1.113. RegionalRegistryTypeEnum Class 87](#_Toc425409724)

# Introduction

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

This document serves as the specification for the CybOX Core Version 1.2 data model, which is one of two fundamental data models for CybOX content.

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

## CybOX Specification Documents

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

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

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

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

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

## Document Conventions

The following conventions are used in this document.

### Keywords

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in *RFC 2119* [RFC2119].

### Fonts

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

* Capitalization is used for CybOX high level concepts, which are defined in the CybOX Specification Overview [CybOXO].

Examples: Action, Object, Event, Property

* The Courier New font is used for writing UML objects.

Examples: ActionType, cyboxCommon:BaseObjectPropertyType

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

* The ‘*italic’* font (withsingle quotes) is used for noting actual, explicit values for CybOX Language properties. The *italic* font (without quotes) is used for noting example values.

Example:  *‘HashNameVocab-1.0,’ high, medium, low*

### UML Package References

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

Note that in *this* specification document, we do not explicitly specify the package prefix for any classes that originate from the Core data model.

### UML Diagrams

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

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

#### Class Properties

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

#### Diagram Icons and Arrow Types

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

Table 1‑1. UML diagram icons

|  |  |
| --- | --- |
| **Icon** | **Description** |
| cid:image003.png@01CFC8F2.37AD2A50 | This diagram icon indicates a class. If the name is in italics, it is an abstract class. |
|  | This diagram icon indicates an enumeration. |
|  | This diagram icon indicates a data type. |
|  | This decorator icon indicates an attribute of a class. The green circle means its visibility is public. If the circle is red or yellow, it means its visibility is private or protected. |
|  | This decorator icon indicates an enumeration literal. |
|  | This arrow type indicates a directed association relationship. |
|  | This arrow type indicates a generalization relationship. |

#### Color Coding

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

[need diagram]

Figure 1‑1. Data model color coding

### Property Table Notation

Throughout Section 3, tables are used to describe the properties of each data model class. Each property table consists of a column of names to identify the property, a type column to reflect the datatype of the property, a multiplicity column to reflect the allowed number of occurrences of the property, and a description column that describes the property. Package prefixes are provided for classes outside of the Core data model (see Section 1.2.3).

Note that if a class is a specialization of a superclass, only the properties that constitute the specialization are shown in the property table (i.e., properties of the superclass will not be shown). However, details of the superclass may be shown in the UML diagram.

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

### Property and Class Descriptions

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

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

Consequently, we have chosen to use the three verbs, defined as follows, in class and property descriptions:

|  |  |
| --- | --- |
| **Verb** | **CybOX Definition** |
| captures | Used to record and preserve information without implying anything about the structure of a class or property. Often used for properties that encompass general content. This is the least precise of the three verbs. |
|  | *Examples*:  The Source property characterizes the source of the sighting information. Examples of details captured include identitifying characteristics, time-related attributes, and a list of the tools used to collect the information.  The Description property captures a textual description of the Indicator. |
| characterizes | Describes the distinctive nature or features of a class or property. Often used to describe classes and properties that themselves comprise one or more other properties. |
|  | *Example*:  The Confidence property characterizes the level of confidence in the accuracy of the overall content captured in the Incident.  The ActivityType class characterizes basic information about an activity a defender might use in response to a Campaign. |
| specifies | Used to clearly and precisely identify particular instances or values associated with a property. Often used for properties that are defined by a controlled vocabulary or enumeration; typically used for properties that take on only a single value. |
|  | *Example*:  The version property specifies the version identifier of the CybOX Core data model used to capture the information associated with… |

# Background Information

# Data Models

## MeasureSourceType Class

The MeasureSourceType class is a type representing a description of a single cyber observation source.

Table 3‑1. Properties of the MeasureSourceType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **class** | SourceClassTypeEnum | 0..1 | The class property specifies the identification of the high-level class of this cyber observation source. |
| **source\_type** | SourceTypeEnum | 0..1 | The source\_type property specifies the identification of the broad type of this cyber observation source. |
| **name** | basicDataTypes:  BasicString | 0..1 | The name property specifies the assignment of a relevant name to this Discovery Method. |
| **sighting\_count** | basicDataTypes:  PositiveInteger | 0..1 | The sighting\_count property specifies how many different identical instances of a given Observable may have been seen/sighted by the observation source. |
| **Information\_Source\_Type** | VocabularyStringType | 0..1 | The Information\_Source\_Type property specifies the type of information source. Examples of potential types *are application logs, help desk* and *TPM* (these specific values are only provided to help explain the property: they are neither recommended values nor necessarily part of any existing vocabulary). The content creator may choose any arbitrary value or may constrain the set of possible values by referencing an externally-defined vocabulary or leveraging a formally defined vocabulary extending from the stixCommon:ControlledVocabularyStringType class. The STIX default vocabulary class for use in the property is *‘InformationSourceTypeVocab-1.0*’. |
| **Tool\_Type** | VocabularyStringType | 0..1 | The Tool\_Type property specifies the type of the tool. Examples of potential types are *NIDS*, *asset scanner*, and *malware analysis* (these specific values are only provided to help explain the property: they are neither recommended values nor necessarily part of any existing vocabulary). The content creator may choose any arbitrary value or may constrain the set of possible values by referencing an externally-defined vocabulary or leveraging a formally defined vocabulary extending from the stixCommon:ControlledVocabularyStringType class. The STIX default vocabulary class for use in the property is *‘ToolTypeVocab-1.1’*. |
| **Description** | StructuredTextType | 0..1 | The Description property captures a technical description of the measure source. Any length is permitted. Optional formatting is supported via the structuring\_format property of the StructuredTextType class. |
| **Contributors** | PersonnelType | 0..1 | The Contributors property characterizes the description of the individual contributors involved in this cyber observation source. |
| **Time** | TimeType | 0..1 | The Time property specifies the various time-related properties for this cyber observation source instance. |
| **Observation\_Location** | LocationType | 0..1 | The Observation\_Location property specifies a relevant physical location for the associated Observation. The underlying abstract class MUST be extended. The default and strongly RECOMMENDED subclass is CIQAddressInstanceType, as defined in the “CybOX Default Extensions Specification” document [CYBOXEXT]. |
| **Tools** | ToolsInformationType | 0..1 | The Tools property characterizes the tools utilized for this cyber observation source. |
| **Platform** | PlatformSpecificationType | 0..1 | The Platform property characterizes a formal, standardized specification of the platform for this cyber observation source. |
| **System** | ObjectPropertiesType | 0..1 | The System property characterizes the system on which the mechanism of cyber observation executed. System SHOULD be an object of type SystemObj:SystemObjectType. |
| **Instance** | ObjectPropertiesType | 0..1 | The Instance property characterizes the process instance in which the mechanism of cyber observation executed. Instance SHOULD be of type ProcessObj:ProcessObjectType. |
| **Observable\_Location** | LocationType | 0..1 | The Observable\_Location property specifies a relevant physical location for the associated Observable. The underlying abstract class MUST be extended. The default and strongly RECOMMENDED subclass is CIQAddressInstanceType, as defined in the “CybOX Default Extensions Specification” document [CYBOXEXT]. |

## ContributorType Class

The ContributorType class represents a description of an individual who contributed as a source of cyber observation data.

Table 3‑2. Properties of the ContributorType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Role** | basicDataTypes:  BasicString | 0..1 | The Role property captures the role played by this contributor. |
| **Name** | basicDataTypes:  BasicString | 0..1 | The Name property captures the name of this contributor. |
| **Email** | basicDataTypes:  BasicString | 0..1 | The Email property captures the email of this contributor. |
| **Phone** | basicDataTypes:  BasicString | 0..1 | The Phone property captures a telephone number of this contributor. |
| **Organization** | basicDataTypes:  BasicString | 0..1 | The Organization property captures the organization name of this contributor. |
| **Date** | DateRangeType | 0..1 | The Date property characterizes a description (bounding) of the timing of this contributor's involvement. |
| **Contribution\_Location** | basicDataTypes:  BasicString | 0..1 | The Contribution\_Location property captures the location at which the contributory activity occurred. |

## DateRangeType Class

The DateRangeType class specifies a range of dates.

Properties of the DateRangeType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Start\_Date** | DateWithPrecisionType | 0..1 | The Start\_Date property specifies the start date for this contributor's involvement. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date and time, the Start property MAY also capture a precision property to specify the granularity with which the time should be considered, as specified by the DateTypePrecisionEnum enumeration (e.g., '*hour*,' '*minute*'). |
| **End\_Date** | DateWithPrecisionType | 0..1 | The End\_date property specifies the end date for this contributor's involvement. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date and time, the Start property MAY also capture a precision property to specify the granularity with which the time should be considered, as specified by the DateTypePrecisionEnum enumeration (e.g., '*hour*,' '*minute*'). |

## PersonnelType Class

The PersonnelType class is an abstracted data type to standardize the description of sets of personnel.

Table 3‑3. Properties of the PersonnelType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Contributor** | ContributorType | 1..\* | The Contributor property characterizes the identity, resources and timing of involvement for a single contributor. |

## TimeType Class

The TimeType class specifies various time properties for this construct.

Table 3‑4. Properties of the TimeType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Start\_Time** | DateTimeWithPrecisionType | 0..1 | The Start\_Time property specifies the starting time for this property. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date and time, the Start property MAY also capture a precision property to specify the granularity with which the time should be considered, as specified by the DateTypePrecisionEnum enumeration (e.g., '*hour*,' '*minute*'). |
| **End\_Time** | DateTimeWithPrecisionType | 0..1 | The End\_Time property specifies the ending time for this property. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date and time, the Start property MAY also capture a precision property to specify the granularity with which the time should be considered, as specified by the DateTypePrecisionEnum enumeration (e.g., '*hour*,' '*minute*'). |
| **Produced\_Time** | DateTimeWithPrecisionType | 0..1 | The Produced\_Time property specifies the time that this property was produced. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date and time, the Start property MAY also capture a precision property to specify the granularity with which the time should be considered, as specified by the DateTypePrecisionEnum enumeration (e.g., '*hour*,' '*minute*'). |
| **Received\_Time** | DateTimeWithPrecisionType | 0..1 | The Received\_Time property specifies the time that this property was received. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date and time, the Start property MAY also capture a precision property to specify the granularity with which the time should be considered, as specified by the DateTypePrecisionEnum enumeration (e.g., '*hour*,' '*minute*'). |

## ToolSpecificDataType Class

The ToolSpecificDataType class is an Abstract type placeholder within the CybOX schema enabling the inclusion of metadata for a specific type of tool through the use of a custom type defined as an extension of this base Abstract type.

Properties of the ToolSpecificDataType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## ToolsInformationType Class

The ToolsInformationType class represents a description of a set of automated tools.

Properties of the ToolsInformationType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Tool** | ToolInformationType | 1..\* | The Tool property characterizes a single tool utilized for this cyber observation source. |

## ToolInformationType Class

The ToolInformationType class is intended to characterize the properties of a hardware or software tool, including those related to instances of its use.

Table 3‑5. Properties of the ToolInformationType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **id** | basicDataTypes:  QualifiedName | 0..1 | The id property specifies a globally unique identifier for the ToolInformation. |
| **idref** | basicDataTypes:  QualifiedName | 0..1 | The idref property specifies an identifier reference to a ToolInformation instance specified elsewhere. When the idref property is used, no other property should be specified. |
| **Name** | basicDataTypes:  BasicString | 0..1 | The Name property captures the name of the tool leveraged. |
| **Type** | VocabularyStringType | 0..\* | The Type property specifies the type of the tool. Examples of potential types are *NIDS*, *asset scanner*, and *malware analysis* (these specific values are only provided to help explain the property: they are neither recommended values nor necessarily part of any existing vocabulary). The content creator may choose any arbitrary value or may constrain the set of possible values by referencing an externally-defined vocabulary or leveraging a formally defined vocabulary extending from the stixCommon:ControlledVocabularyStringType class. The STIX default vocabulary class for use in the property is *‘ToolTypeVocab-1.1’*. |
| **Description** | StructuredTextType | 0..1 | The Description property captures a technical description of the ToolInformation. Any length is permitted. Optional formatting is supported via the structuring\_format property of the StructuredTextType class. |
| **References** | ToolReferencesType | 0..1 | The References property captures references to instances or additional information for this tool. |
| **Vendor** | basicDataTypes:  BasicString | 0..1 | The Vendor property captures information identifying the vendor organization for this tool. |
| **Version** | basicDataTypes:  BasicString | 0..1 | The Version property captures an appropriate version descriptor of this tool. |
| **Service\_Pack** | basicDataTypes:  BasicString | 0..1 | The Service\_Pack property captures an appropriate service pack descriptor for this tool. |
| **Tool\_Specific\_Data** | ToolSpecificDataType | 0..1 | The Tool\_Specific\_Data property characterizes tool-specific data to be included. |
| **Tool\_Hashes** | HashListType | 0..1 | The Tool\_Hashes property captures a hash value computed on the tool file content in order to verify its integrity. |
| **Tool\_Configuration** | ToolConfigurationType | 0..1 | The Tool\_Configuation property characterizes the configuration and usage of the tool. |
| **Execution\_Environment** | ExecutionEnvironmentType | 0..1 | The Execution\_Environment property characterizes the execution environment of the tool. |
| **Errors** | ErrorsType | 0..1 | The Errors property captures any errors generated during the run of the tool. |
| **Metadata** | MetadataType | 0..\* | The Metadata property captures other relevant metadata including tool-specific properties. |
| **Compensation\_Model** | CompensationModelType | 0..1 | The Compensation\_Model property captures the name of the compensation model used for the tool. |

## CompensationModelType Class

The CompensationModelType class characterizes the compensation model for a tool.

Properties of the CompensationModelType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## ToolReferencesType Class

Used to indicate one or more references to tool instances and information.

Table 3‑6. Properties of the ToolReferencesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Reference** | ToolReferenceType | 1..\* | The Reference property specifies one reference to information or instances of a given tool. |

## ToolReferenceType Class

Contains one reference to information or instances of a given tool.

Table 3‑7. Properties of the ToolReferenceType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **reference\_type** | ToolReferenceTypeEnum | 0..1 | The reference\_type property specifies the nature of the referenced material (documentation, source, executable, etc.). |

## ToolConfigurationType Class

The ToolConfigurationType class characterizes the configuration for a tool used as a cyber observation source.

Table 3‑8. Properties of the ToolConfigurationType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Configuration\_Settings** | ConfigurationSettingsType | 0..1 | The Configuration\_Settings property characterizes the configuration settings of this tool instance. |
| **Dependencies** | DependenciesType | 0..1 | The Dependencies property characterizes the relevant dependencies for this tool. |
| **Usage\_Context\_Assumptions** | UsageContextAssumptionsType | 0..1 | The Usage\_Context\_Assumptions property characterizes the various relevant usage context assumptions for this tool. |
| **Internationalization\_Settings** | Internationalization  SettingsType | 0..1 | The Internationalization\_Settings property characterizes the relevant internationalization setting for this tool. |
| **Build\_Information** | BuildInformationType | 0..1 | The Build\_information property characterizes how this tool was built. |

## ConfigurationSettingsType Class

The ConfigurationSettingsType class is a modularized data type used to provide a consistent approach to describing configuration settings for a tool, application or other cyber object.

Table 3‑9. Properties of the ConfigurationSettingsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Configuration\_Setting** | ConfigurationSettingType | 1..\* | The Configuration\_Setting property specifies a single configuration setting instance. |

## ConfigurationSettingType Class

The ConfigurationSettingType class is a modularized data type used to provide a consistent approach to describing a particular configuration setting for a tool, application or other cyber object.

Table 3‑10. Properties of the ConfigurationSettingType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Item\_Name** | basicDataTypes:  BasicString | 1 | The Item\_Name property captures the name of the configuration item referenced by this configuration setting instance. |
| **Item\_Value** | basicDataTypes:  BasicString | 1 | The Item\_Value property captures the value of this configuration setting instance. |
| **Item\_Type** | basicDataTypes:  BasicString | 0..1 | The Item\_Type property captures the type of the configuration item referenced in this configuration setting instance. |
| **Item\_Description** | basicDataTypes:  BasicString | 0..1 | The Item\_Description property captures a description of the configuration item referenced in this configuration setting instance. |

## DependenciesType Class

The DependenciesType class contains information describing a set of dependencies for this tool.

Table 3‑11. Properties of the DependenciesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Dependency** | DependencyType | 1..\* | The Dependency property characterizes a single dependency for this tool. |

## DependencyType Class

The DependencyType class contains information describing a single dependency for this tool.

Table 3‑12. Properties of the DependencyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Dependency\_Type** | basicDataTypes:  BasicString | 0..1 | The Dependency\_Type property captures the type of this dependency instance. |
| **Dependency\_Description** | StructuredTextType | 1 | The Dependency\_Description property captures a description of this dependency instance. |

## UsageContextAssumptionsType Class

The UsageContextAssumptionsType class contains descriptions of the various relevant usage context assumptions for this tool.

Table 3‑13. Properties of the UsageContextAssumptionsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Usage\_Context\_Assumption** | StructuredTextType | 1..\* | The Usage\_Context\_Assumption property captures a single usage context assumption for this tool. |

## InternationalizationSettingsType Class

The InternationalizationSettingsType class contains information describing relevant internationalization setting for this tool.

Table 3‑14. Properties of the InternationalizationSettingsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Internal\_Strings** | InternalStringsType | 1..\* | The Internal\_Strings property captures a single internal string instance for this internationalization setting instance. |

## InternalStringsType Class

The InternalStringsType class contains a single internal string instance for this internationalization setting instance.

Table 3‑15. Properties of the InternalStringsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Key** | basicDataTypes:  BasicString | 1 | The Key property captures the actual key of this internal string instance. |
| **Content** | basicDataTypes:  BasicString | 1 | The Content property captures the actual content of this internal string instance. |

## BuildInformationType Class

The BuildInformationType class contains information describing how this tool was built.

Table 3‑16. Properties of the BuildInformationType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Build\_ID** | basicDataTypes:  BasicString | 0..1 | The Build\_ID property captures an externally defined unique identifier of this build of this application instance. |
| **Build\_Project** | basicDataTypes:  BasicString | 0..1 | The Build\_Project property captures the project name of this build of this application instance. |
| **Build\_Utility** | BuildUtilityType | 0..1 | The Build\_Utility property characterizes the utility used to build this application. |
| **Build\_Version** | basicDataTypes:  BasicString | 0..1 | The Build\_Version property captures the appropriate version descriptor of this build of this application instance. |
| **Build\_Label** | basicDataTypes:  BasicString | 0..1 | The Build\_Label property captures any relevant label for this build of this application instance. |
| **Compilers** | CompilersType | 0..1 | The Compilers property characterizes compilers utilized during this build of this application. |
| **Compilation\_Date** | DateTimeWithPrecisionType | 0..1 | The Completion\_Date property specifies the compilation date for the build of the tool. In order to avoid ambiguity, it is strongly suggest that all timestamps in this field include a specification of the timezone if it is known. |
| **Build\_Configuration** | BuildConfigurationType | 0..1 | The Build\_Configuration property characterizes how the build utility was configured for this build of this application. |
| **Build\_Script** | basicDataTypes:  BasicString | 0..1 | The Build\_Script property captures the actual build script for this build of this application instance. |
| **Libraries** | LibrariesType | 0..1 | The Libraries property characterizes the libraries incorporated into the build of the tool. |
| **Build\_Output\_Log** | basicDataTypes:  BasicString | 0..1 | The Build\_Output\_Log property captures the output log of the build process. |

## BuildUtilityType Class

The BuildUtilityType class contains information identifying the utility used to build this application.

Table 3‑17. Properties of the BuildUtilityType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Build\_Utility\_Name** | basicDataTypes:  BasicString | 1 | The Build\_Utility\_Name property captures the informally defined name of the utility used to build this application instance. |
| **Build\_Utility\_Platform\_Specification** | PlatformSpecificationType | 1 | The Build\_Utility\_Platform\_Specification property characterizes the build utility used to build this application. |

## CompilersType Class

The CompilersType class describes the compilers utilized during this build of this application.

Table 3‑18. Properties of the CompilersType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Compiler** | CompilerType | 1..\* | The Compiler property characterizes a single compiler utilized during this build of this application. |

## CompilerType Class

The CompilerType class describes a single compiler utilized during this build of this application.

Table 3‑19. Properties of the CompilerType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Compiler\_Informal\_Description** | CompilerInformalDescriptionType | 0..1 | The Compiler\_Informal\_Description property characterizes this compiler instance. |
| **Compiler\_Platform\_Specification** | PlatformSpecificationType | 0..1 | The Compiler\_Platform\_Specification property characterizes this compiler instance. |

## CompilerInformalDescriptionType Class

The CompilerInformalDescriptionType class contains the informal description of this compiler instance.

Table 3‑20. Properties of the CompilerInformalDescriptionType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Compiler\_Name** | basicDataTypes:  BasicString | 1 | The Compiler\_Name property captures the name of the compiler. |
| **Compiler\_Version** | basicDataTypes:  BasicString | 0..1 | The Compiler\_Version property captures the version of the compiler. |

## BuildConfigurationType Class

The BuildConfigurationType class describes how the build utility was configured for this build of this application.

Table 3‑21. Properties of the BuildConfigurationType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Configuration\_Setting\_Description** | basicDataTypes:  BasicString | 0..1 | The Configuration\_Setting\_Description property captures the configuration settings for this build of this application instance. |
| **Configuration\_Settings** | ConfigurationSettingsType | 1 | The Configuration\_Settings property characterizes the configuration settings for this build of this application instance. |

## LibrariesType Class

The LibrariesType class identifies the libraries incorporated into the build of the tool.

Table 3‑22. Properties of the LibrariesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Library** | LibraryType | 0..1 | The Library property characterizes a library incorporated into the build of the tool. |

## LibraryType Class

The LibraryType class identifies a single library incorporated into the build of the tool.

Table 3‑23. Properties of the LibraryType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **name** | basicDataTypes:  BasicString | 0..1 | The name property captures the name of the library. |
| **version** | basicDataTypes:  BasicString | 0..1 | The version property captures the version of the library. |

## ExecutionEnvironmentType Class

The ExecutionEnvironmentType class contains information describing the execution environment of the tool.

Table 3‑24. Properties of the ExecutionEnvironmentType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **System** | ObjectPropertiesType | 0..1 | The System property characterizes the system on which the tool was executed. This property should be of class SystemObj:SystemObjectType. |
| **User\_Account\_Info** | ObjectPropertiesType | 0..1 | The User\_Account\_info property characterizes the user account that executed the tool. This property should be of class UserAccountObj:UserAccountObjectType. |
| **Command\_Line** | basicDataTypes:  BasicString | 0..1 | The Command\_Line property captures the command line string used to run the tool. |
| **Start\_Time** | DateTimeWithPrecisionType | 0..1 | The Start\_Time property specifies when the tool was run. In order to avoid ambiguity, it is strongly suggest that all timestamps in this field include a specification of the timezone if it is known. |

## ErrorsType Class

The ErrorsType class captures any errors generated during the run of the tool.

Table 3‑25. Properties of the ErrorsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Error** | ErrorType | 1..\* | The Error property captures a single type of error generated during the run of the tool. |

## ErrorType Class

The ErrorType class captures a single error generated during the run of the tool.

Table 3‑26. Properties of the ErrorType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Error\_Type** | basicDataTypes:  BasicString | 1 | The Error\_Type property captures the type for this tool run error. |
| **Error\_Count** | basicDataTypes:  PositiveInteger | 0..1 | The Error\_Count property specifies the count of instances for this error in the tool run. |
| **Error\_Instances** | ErrorInstancesType | 0..1 | The Error\_Instances property captures the actual error output for each instance of this type of error. |

## ErrorInstancesType Class

The ErrorInstancesType class captures the actual error output for each instance of this type of error.

Table 3‑27. Properties of the ErrorInstancesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Error\_Instance** | basicDataTypes:  BasicString | 1..\* | The Error\_Instance property captures the actual error output for a single instance of this type of error. |

## ObjectPropertiesType Class

The ObjectPropertiesType class is an abstract class within the CybOX schema enabling the inclusion of contextually varying object properties descriptions. This Abstract type is leveraged as the extension base for all predefined CybOX object properties schemas. Through this extension mechanism any object instance data based on an object properties schema extended from ObjectPropertiesType (e.g. File\_Object, Address\_Object, etc.) can be directly integrated into any instance document where a property is defined as ObjectPropertiesType. For flexibility and extensibility purposes any user of CybOX can specify their own externally defined object properties schemas (outside of or derived from the set of predefined objects) extended from ObjectPropertiesType class and utilize them as part of their CybOX content.

Table 3‑28. Properties of the ObjectPropertiesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **object\_reference** | basicDataTypes:  QualifiedName | 0..1 | The object\_reference property specifies a unique ID reference to an Object defined elsewhere. This property allows for the re-use of the defined Properties of one Object within another, without the need to embed the full Object in the location from which it is being referenced. Thus, this ID reference is intended to resolve to the Properties of the Object that it points to. |
| **Custom\_Properties** | CustomPropertiesType | 0..1 | The Custom\_Properties property characterizes a set of custom Object Properties that may not be defined in existing Properties schemas. |

## CustomPropertiesType Class

The CustomPropertiesType class enables the specification of a set of custom Object Properties that may not be defined in existing Properties schemas.

Table 3‑29. Properties of the CustomPropertiesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Property** | PropertyType | 1..\* | The Property property characterizes a single Object Property. |

## PropertyType Class

The PropertyType class is a type representing the specification of a single Object Property.

Table 3‑30. Properties of the PropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **name** | basicDataTypes:  BasicString | 0..1 | The name property captures the name for this property. |
| **description** | basicDataTypes:  BasicString | 0..1 | The description property captures a description of what this property represents. |

## BaseObjectPropertyType Class

The BaseObjectPropertyType class is a type representing a common typing foundation for the specification of a single Object Property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Table 3‑31. Properties of the BaseObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **id** | basicDataTypes:  QualifiedName | 0..1 | The id property specifies a globally unique identifier for the BaseObjectProperty. |
| **idref** | basicDataTypes:  QualifiedName | 0..1 | The idref property specifies an identifier reference to a BaseObjectProperty instance specified elsewhere. When the idref property is used, no other property should be specified. |
| **datatype** | DatatypeEnum | 0..1 | The datatype property specifies the expected type for the value of the specified property. |
| **appears\_random** | basicDataTypes:  Boolean | 0..1 | The appear\_random property specifies whether the associated object property value appears to somewhat random in nature. An object property with this property set to TRUE need not provide any further information including a value. If more is known about the particular variation of randomness, a regex value could be provided to outline what is known of the structure. |
| **is\_obfuscated** | basicDataTypes:  Boolean | 0..1 | The is\_obfuscated property specifies whether the associated Object property has been obfuscated. |
| **obfuscation\_algorithm\_ref** | basicDataType:URI | 0..1 | The obfuscation\_algorithm\_ref property specifies a reference to a description of the algorithm used to obfuscate this Object property. |
| **is\_defanged** | basicDataTypes:  Boolean | 0..1 | The is\_defanged property specifies whether the associated Object property has been defanged (representation changed to prevent malicious effects of handling/processing). |
| **defanging\_algorithm\_ref** | basicDataType:URI | 0..1 | The defanging\_algorithm\_ref property specifies a reference to a description of the algorithm used to defang (representation changed to prevent malicious effects of handling/processing) this Object property. |
| **refanging\_transform\_type** | basicDataTypes:  BasicString | 0..1 | The refanging\_transform\_type property specifies the type (e.g. RegEx) of refanging transform specified in the optional accompanying refangingTransform property. |
| **refanging\_transform** | basicDataTypes:  BasicString | 0..1 | The refanging\_transform property captures an automated transform that can be applied to the Object property content in order to refang it to its original format. |
| **observed\_encoding** | basicDataTypes:  BasicString | 0..1 | The observed\_encoding property captures the encoding of the string when it is/was observed. This may be different from the encoding used to represent the string within this element. It is strongly recommended that character set names should be taken from the IANA character set registry (https://www.iana.org/assignments/character-sets/character-sets.xhtml). This property is intended to be applicable only to properties which contain string values. |
| **condition** | ConditionTypeEnum | 0..1 | The condition property specifies the relevant condition to apply to the value. |
| **is\_case\_sensitive** | basicDataTypes:  Boolean | 0..1 | The is\_case\_sensitive property specifies the case-sensitivity of a pattern which uses an Equals, DoesNotEqual, Contains, DoesNotContain, StartsWith, EndsWith, or FitsPattern condition. The default value for this property is "true" which indicates that pattern evaluations are to be considered case-sensitive. |
| **apply\_condition** | ConditionApplicationEnum | 0..1 | The apply\_condition property specifies how a condition should be applied when the field body contains a list of values. (Its value is moot if the field value contains only a single value - both possible values for this field would have the same behavior.) If this property is set to ANY, then a pattern is considered to be matched if the provided condition successfully evaluates for any of the values in the field body. If the property is set to ALL, then the pattern only matches if the provided condition successfully evaluates for every value in the field body. |
| **delimiter** | basicDataTypes:  BasicString | 0..1 | The delimiter property captures the delimiter used when defining lists of values. The default value is "##comma##". |
| **bit\_mask** | basicDataType:  HexBinary | 0..1 | The bit\_mask property specifies a bit\_mask in conjunction with one of the defined binary conditions (bitwiseAnd, bitwiseOr, and bitwiseXor). This bitmask is then uses as one operand in the indicated bitwise computation. |
| **pattern\_type** | PatternTypeEnum | 0..1 | The pattern\_type property specifies the type of pattern used if one is specified for the field value. This is applicable only if the Condition property is set to 'FitsPattern'. |
| **regex\_syntax** | basicDataTypes:  BasicString | 0..1 | The regex\_syntax property captures the syntax format used for a regular expression, if one is specified for the property value. This is applicable only if the Condition property is set to 'FitsPattern'. Setting this attribute with an empty value (e.g., "") or omitting it entirely notifies CybOX consumers and pattern evaluators that the corresponding regular expression utilizes capabilities, character classes, escapes, and other lexical tokens defined by the CybOX Language Specification. Setting this attribute with a non-empty value notifies CybOX consumers and pattern evaluators that the corresponding regular expression utilizes capabilities not defined by the CybOX Language Specification. The regular expression must be evaluated through a compatible regular expression engine in this case. |
| **has\_changed** | basicDataTypes:  Boolean | 0..1 | The has\_changed property specifies whether a targeted observation pattern of the associated property value has changed. This property would be leveraged within a pattern observable triggering on whether the value of a single property value has changed. |
| **trend** | basicDataTypes:  Boolean | 0..1 | The trend property specifies whether a targeted observation pattern of the nature of any trend in the associated property value. This property would be leveraged within a pattern observable triggering on the matching of a specified trend in the value of a single specified property. |

## IntegerObjectPropertyType Class

The IntegerObjectPropertyType class is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type Int. This type will be assigned to any property of a CybOX object that should contain content of type Integer and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the IntegerObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## StringObjectPropertyType Class

The StringObjectPropertyType class is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type String. This type will be assigned to any property of a CybOX object that should contain content of type String and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the StringObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## NameObjectPropertyType Class

The NameObjectPropertyType class is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type Name. This type will be assigned to any property of a CybOX object that should contain content of type Name and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the NameObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## DateObjectPropertyRestrictionType Class

The DateObjectPropertyRestrictionType class is a type is an intermediate type to allow for the addition of the precision attribute to DateObjectPropertyType. It should not be used directly.

Properties of the DateObjectPropertyRestrictionType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## DateObjectPropertyType Class

The DateObjectPropertyType is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type Date. This type will be assigned to any property of a CybOX object that should contain content of type Date and enables the use of relevant metadata for the property. In order to avoid ambiguity, it is strongly suggested that any date representation in this property include a timezone if it is known. As with the rest of the property, this should be formatted per the xs:date specification. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space. For properties of this type using CybOX patterning, it is strongly suggested that the condition (pattern type) is limited to one of Equals, DoesNotEqual, GreaterThan, LessThan, GreaterThanOrEqual, LessThanOrEqual, ExclusiveBetween, or InclusiveBetween. The use of other conditions may lead to ambiguity or unexpected results. When evaluating data against a pattern, the evaluator should take into account the precision of the property (as given by the precision attribute) and any timezone information that is available to perform a data-aware comparison. The usage of simple string comparisons is discouraged due to ambiguities in how precision and timezone information is processed.

Table 3‑32. Properties of the DateObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **precision** | DatePrecisionEnum | 0..1 | The precision property of the associated time. If omitted, the default is "day", meaning the full property value. Digits in the date that are required by the xs:date datatype but are beyond the specified precision should be zeroed out. When used in conjunction with CybOX patterning, the pattern should only be evaluated against the target up to the given precision. |

## DateTimeObjectPropertyRestrictionType Class

The DateTimeObjectPropertyRestrictionType class is type is an intermediate type to allow for the addition of the precision attribute to DateTimeObjectPropertyType. It should not be used directly.

Properties of the DateTimeObjectPropertyRestrictionType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## DateTimeObjectPropertyType Class

The DateTimeObjectPropertyType is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type DateTime. This type will be assigned to any property of a CybOX object that should contain content of type DateTime and enables the use of relevant metadata for the property. In order to avoid ambiguity, it is strongly suggested that any dateTime representation in this property include a timezone. As with the rest of the property, this should be formatted per the xs:dateTime specification. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space. For properties of this type using CybOX patterning, it is strongly suggested that the condition (pattern type) is limited to one of Equals, DoesNotEqual, GreaterThan, LessThan, GreaterThanOrEqual, LessThanOrEqual, ExclusiveBetween, or InclusiveBetween. The use of other conditions may lead to ambiguity or unexpected results. When evaluating data against a pattern, the evaluator should take into account the precision of the property (as given by the precision attribute) and any timezone information that is available to perform a data-aware comparison. The usage of simple string comparisons is discouraged due to ambiguities in how precision and timezone information is processed.

Table 3‑33. Properties of the DateTimeObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Precision** | DateTimePrecisionEnum | 0..1 | The precision property specifies the granularity with which the time property should be considered, as specified by the DateTimePrecisionEnum enumeration (e.g., *hour*, *minute*). If omitted, the default precision is *second*. Digits in a timestamp that are beyond the specified precision should be zeroed out. When used in conjunction with CybOX patterning, the pattern should only be evaluated against the target up to the given precision. |

## FloatObjectPropertyType Class

The FloatObjectPropertyType class is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type Float. This type will be assigned to any property of a CybOX object that should contain content of type Float and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the FloatObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## DoubleObjectPropertyType Class

The DoubleObjectPropertyType is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type Double. This type will be assigned to any property of a CybOX object that should contain content of type Double and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the DoubleObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## UnsignedLongObjectPropertyType Class

The UnsignedLongObjectPropertyType is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type UnsignedLong. This type will be assigned to any property of a CybOX object that should contain content of type UnsignedLong and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the UnsignedLongObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## UnsignedIntegerObjectPropertyType Class

The UnsignedIntegerObjectPropertyType is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type UnsignedInt. This type will be assigned to any property of a CybOX object that should contain content of type UnsignedInteger and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the UnsignedIntegerObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## PositiveIntegerObjectPropertyType Class

The PositiveIntegerObjectPropertyType is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type PositveInteger. This type will be assigned to any property of a CybOX object that should contain content of type PositiveInteger and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the PositiveIntegerObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## HexBinaryObjectPropertyType Class

The HexBinaryObjectPropertyType is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type HexBinary. This type will be assigned to any property of a CybOX object that should contain content of type HexBinary and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the HexBinaryObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## LongObjectPropertyType Class

The LongObjectPropertyType class is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type Long. This type will be assigned to any property of a CybOX object that should contain content of type Long and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the LongObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## NonNegativeIntegerObjectPropertyType Class

The NonNegativeIntegerObjectPropertyType is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type nonNegativeInteger. This type will be assigned to any property of a CybOX object that should contain content of type NonNegativeInteger and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the NonNegativeIntegerObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## AnyURIObjectPropertyType Class

The AnyURIObjectPropertyType is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type anyURI. This type will be assigned to any property of a CybOX object that should contain content of type AnyURI and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the AnyURIObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## DurationObjectPropertyType Class

The DurationObjectPropertyType is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type duration. This type will be assigned to any property of a CybOX object that should contain content of type Duration and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the DurationObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## TimeObjectPropertyRestrictionType Class

The DateTimeObjectPropertyRestrictionType class is a type is an intermediate type to allow for the addition of the precision attribute to TimeObjectPropertyType. It should not be used directly.

Properties of the TimeObjectPropertyRestrictionType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## TimeObjectPropertyType Class

The TimeObjectPropertyType is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type time. This type will be assigned to any property of a CybOX object that should contain content of type Time and enables the use of relevant metadata for the property. In order to avoid ambiguity, it is strongly suggested that any time representation in this property include a specification of the timezone if it is known. As with the rest of the property, this should be formatted per the xs:time specification. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space. For properties of this type using CybOX patterning, it is strongly suggested that the condition (pattern type) is limited to one of Equals, DoesNotEqual, GreaterThan, LessThan, GreaterThanOrEqual, LessThanOrEqual, ExclusiveBetween, or InclusiveBetween. The use of other conditions may lead to ambiguity or unexpected results. When evaluating data against a pattern, the evaluator should take into account the precision of the property (as given by the precision attribute) and any timezone information that is available to perform a data-aware comparison. The usage of simple string comparisons is discouraged due to ambiguities in how precision and timezone information is processed.

Table 3‑34. Properties of the TimeObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **precision** | TimePrecisionEnum | 0..1 | The precision property specifies the granularity with which a timestamp should be considered as specified by the TypePrecsionEnum enumeration (e.g., '*hour*,' '*minute*'). If omitted, the default precision is '*second*.' Digits in a timestamp that are beyond a specified precision SHOULD be zeroed out.When used in conjuction with CybOX patterning, the pattern should only be evaluated against the target up to the given precision |

## Base64BinaryObjectPropertyType Class

The Base64BinaryObjectPropertyType is a type (extended from BaseObjectPropertyType) representing the specification of a single Object property whose core value is of type base64Binary. This type will be assigned to any property of a CybOX object that should contain content of type Base64Binary and enables the use of relevant metadata for the property. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the Base64BinaryObjectPropertyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## BaseObjectPropertyGroup Class

The ObjectPropertyGroup is a simple field group aggregating a set of fields for Object Properties.

Properties of the BaseObjectPropertyGroup class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **id** | basicDataTypes:  QualifiedName | 0..1 | The id property specifies a globally unique identifier for the BaseObjectPropertyGroup. |
| **idref** | basicDataTypes:  QualifiedName | 0..1 | The idref property specifies an identifier reference to a BaseObjectPropertyGroup instance specified elsewhere. When the idref property is used, no other property should be specified. |
| **datatype** | DatatypeEnum | 0..1 | The datatype property This attribute specifies the expected type for the value of the specified property. |
| **appears\_random** | basicDataTypes:  Boolean | 0..1 | The appears\_random property conveys whether the associated object property value appears to somewhat random in nature. An object property with this field set to TRUE need not provide any further information including a value. If more is known about the particular variation of randomness, a regex value could be provided to outline what is known of the structure. |
| **is\_obfuscated** | basicDataTypes:  Boolean | 0..1 | The is\_obfuscated property conveys whether the associated Object property has been obfuscated. |
| **obfuscation\_algorithm\_ref** | xs:anyURI | 0..1 | The obfuscation\_algorithm\_ref property conveys a reference to a description of the algorithm used to obfuscate this Object property. |
| **is\_defanged** | basicDataTypes:  Boolean | 0..1 | This property conveys whether the associated Object property has been defanged (representation changed to prevent malicious effects of handling/processing). |
| **defanging\_algorithm\_ref** | xs:anyURI | 0..1 | This property conveys a reference to a description of the algorithm used to defang (representation changed to prevent malicious effects of handling/processing) this Object property. |
| **refanging\_transform\_type** | basicDataTypes:BasicString | 0..1 | This property specifies the type (e.g. RegEx) of refanging transform specified in the optional accompanying refangingTransform property. |
| **refanging\_transform** | basicDataTypes:BasicString | 0..1 | This property specifies an automated transform that can be applied to the Object property content in order to refang it to its original format. |
| **observed\_encoding** | basicDataTypes:BasicString | 0..1 | This property specifies the encoding of the string when it is/was observed. This may be different from the encoding used to represent the string within this element. It is strongly recommended that character set names should be taken from the IANA character set registry (https://www.iana.org/assignments/character-sets/character-sets.xhtml). This field is intended to be applicable only to fields which contain string values. |

## PatternFieldGroup Class

The PatternFieldGroup is a simple field group aggregating a set of fields for application of patterns.

Properties of the PatternFieldGroup class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **condition** | ConditionTypeEnum | 0..1 | This property defines the relevant condition to apply to the value. |
| **is\_case\_sensitive** | basicDataTypes:Boolean | 0..1 | The is\_case\_sensitive property should be used when specifying the case-sensitivity of a pattern which uses an Equals, DoesNotEqual, Contains, DoesNotContain, StartsWith, EndsWith, or FitsPattern condition. The default value for this property is "true" which indicates that pattern evaluations are to be considered case-sensitive. |
| **apply\_condition** | ConditionApplicationEnum | 0..1 | This property indicates how a condition should be applied when the field body contains a list of values. (Its value is moot if the field value contains only a single value - both possible values for this property would have the same behavior.) If this property is set to ANY, then a pattern is considered to be matched if the provided condition successfully evaluates for any of the values in the field body. If the property is set to ALL, then the patern only matches if the provided condition successfully evaluates for every value in the field body. |
| **delimiter** | basicDataTypes:BasicString | 0..1 | The delimiter property specifies the delimiter used when defining lists of values. The default value is "##comma##". |
| **bit\_mask** | xs:hexBinary | 0..1 | The bit\_mask property Used to specify a bit\_mask in conjunction with one of the defined binary conditions (bitwiseAnd, bitwiseOr, and bitwiseXor). This bitmask is then uses as one operand in the indicated bitwise computation. |
| **pattern\_type** | PatternTypeEnum | 0..1 | This property defines the type of pattern used if one is specified for the field value. This is applicable only if the Condition property is set to 'FitsPattern'. |
| **regex\_syntax** | basicDataTypes:BasicString | 0..1 | This property defines the syntax format used for a regular expression, if one is specified for the property value. This is applicable only if the Condition property is set to 'FitsPattern'. Setting this attribute with an empty value (e.g., "") or omitting it entirely notifies CybOX consumers and pattern evaluators that the corresponding regular expression utilizes capabilities, character classes, escapes, and other lexical tokens defined by the CybOX Language Specification. Setting this attribute with a non-empty value notifies CybOX consumers and pattern evaluators that the corresponding regular expression utilizes capabilities not defined by the CybOX Language Specification. The regular expression must be evaluated through a compatible regular expression engine in this case. |
| **has\_changed** | basicDataTypes:Boolean | 0..1 | This property conveys a targeted observation pattern of whether the associated property value has changed. This property would be leveraged within a pattern observable triggering on whether the value of a single property value has changed. |
| **trend** | basicDataTypes:Boolean | 0..1 | This property conveys a targeted observation pattern of the nature of any trend in the associated property value. This property would be leveraged within a pattern observable triggering on the matching of a specified trend in the value of a single specified property. |

## LocationType Class

The LocationType class is used to express geographic location information. This type is extended through the xsi:type mechanism. The default type is CIQAddress3.0InstanceType in the http://cybox.mitre.org/extensions/Address#CIQAddress3.0-1 namespace. This type is defined in the extensions/location/ciq\_address\_3.0.xsd file or at the URL http://cybox.mitre.org/XMLSchema/extensions/location/ciq\_address\_3.0/1.0/ciq\_address\_3.0.xsd. Those who wish to express a simple name may also do so by not specifying an xsi:type and using the Name property of this type.

Table 3‑35. Properties of the LocationType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **id** | basicDataTypes:  QualifiedName | 0..1 | The id property specifies a globally unique identifier for the Location. |
| **idref** | basicDataTypes:  QualifiedName | 0..1 | The idref property specifies an identifier reference to a Location instance specified elsewhere. When the idref property is used, no other property should be specified. |
| **Name** | basicDataTypes:  BasicString | 0..1 | The Name property captures a location through a simple name. |

## ExtractedFeaturesType Class

The ExtractedFeaturesType class is a type representing a description of features extracted from an object such as a file.

Table 3‑36. Properties of the ExtractedFeaturesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Strings** | ExtractedStringsType | 0..1 | The Strings property characterizes a set of static strings extracted from a raw cyber object. |
| **Imports** | ImportsType | 0..1 | The Imports property characterizes a set of references to external resources imported by a raw cyber object. |
| **Functions** | FunctionsType | 0..1 | The Functions property characterizes a set of references to functions called by a raw cyber object. |
| **Code\_Snippets** | CodeSnippetsType | 0..1 | The Code\_Snippets property characterizes a set of code snippets extracted from a raw cyber object. |

## ExtractedStringsType Class

The ExtractedStringsType class is intended as container for strings extracted from CybOX objects.

Table 3‑37. Properties of the ExtractedStringsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **String** | ExtractedStringType | 1..\* | The String property characterizes a single static string extracted from a raw cyber object. |

## ExtractedStringType Class

The ExtractedStringType class is intended as container a single string extracted from a CybOX object.

Table 3‑38. Properties of the ExtractedStringType class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | | **Multiplicity** | **Description** |
| **Encoding** | | VocabularyStringType | 0..1 | The Encoding property specifies the character encoding used for the String\_Value property. Examples of potential *ASCII, UTF-8, Windows-1250* (these specific values are only provided to help explain the property: they are neither recommended values nor necessarily part of any existing vocabulary). The content creator may choose any arbitrary value or may constrain the set of possible values by referencing an externally-defined vocabulary or leveraging a formally defined vocabulary extending from the stixCommon:ControlledVocabularyStringType class. The STIX default vocabulary class for use in the property is ‘*CharacterEncodingEnum-1.0*’. |
| **String\_Value** | | StringObjectPropertyType | 0..1 | The String\_Value property characterizes the actual value of the string extracted from the CybOX object, if it is capable of being represented in the encoding scheme used in the document (most commonly UTF-8). |
| **Byte\_String\_Value** | | HexBinaryObjectPropertyType | 0..1 | The Byte\_String\_Value property characterizes the raw, byte-string representation of the string extracted from the CybOX object, in hexadecimal format. |
| **Hashes** | | HashListType | 0..1 | The Hashes property specifies any hash values computed using the string extracted from the CybOX object as input. |
| **Address** | | HexBinaryObjectPropertyType | 0..1 | The Address property characterizes the location or offset of the specified string in the CybOX objects. |
| **Length** | | PositiveIntegerObjectPropertyType | 0..1 | The Length property characterizes the length, in characters, of the string extracted from the CybOX object. |
| **Language** | | StringObjectPropertyType | 0..1 | The Language property characterizes the language the string is written in, e.g. English. For consistency, we strongly recommend using the ISO 639-2 language code, if available. Please see http://www.loc.gov/standards/iso639-2/php/code\_list.php for a list of ISO 639-2 codes. |
| **English\_Translation** | | StringObjectPropertyType | 0..1 | The English\_Translation property characterizes the English translation of the string, if it is not written in English. |

## ImportsType Class

The ImportsType class is intended to represent an extracted list of imports specified within a CybOX object.

Table 3‑39. Properties of the ImportsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Import** | StringObjectPropertyType | 1..\* | The Import property characterizes a single reference to an external resource imported by a raw cyber object. |

## FunctionsType Class

The FunctionsType class is intended to represent an extracted list of functions leveraged within a CybOX object.

Table 3‑40. Properties of the FunctionsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Function** | StringObjectPropertyType | 1..\* | The Function property characterizes a single reference to a function called by a raw cyber object. |

## CodeSnippetsType Class

The CodeSnippetsType class is intended to represent a set of code snippets extracted from within a CybOX object.

Table 3‑41. Properties of the CodeSnippetsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Code\_Snippet** | ObjectPropertiesType | 1..\* | The Code\_Snippet property characterizes a single code snippet extracted from a raw cyber object. This property should be of class CodeObj:CodeObjectType. |

## ByteRunsType Class

The ByteRunsType class is used for representing a list of byte runs from within a raw object.

Table 3‑42. Properties of the ByteRunsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Byte\_Run** | ByteRunType | 1..\* | The Byte\_Run property characterizes a single byte run from the raw object. |

## ByteRunType Class

The ByteRunType class is used for representing a single byte run from within a raw object.

Table 3‑43. Properties of the ByteRunType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Offset** | IntegerObjectPropertyType | 0..1 | The Offset property characterizes the offset of the beginning of the byte run as measured from the beginning of the object. |
| **Byte\_Order** | EndiannessType | 0..1 | The Byte\_Order property characterizes the endianness of the unpacked (e.g., unencoded, unencrypted, etc.) data contained within the Byte\_Run\_Data property. |
| **File\_System\_Offset** | IntegerObjectPropertyType | 0..1 | The File\_System\_Offset property characterizes the offset of the beginning of the byte run as measured from the beginning of the relevant file system. It is relevant only for byte runs of files in forensic analysis. |
| **Image\_Offset** | IntegerObjectPropertyType | 0..1 | The Image\_Offset property characterizes the offset of the beginning of the byte run as measured from the beginning of the relevant forensic image. It is provided for forensic analysis purposes. |
| **Length** | IntegerObjectPropertyType | 0..1 | The Length property characterizes the number of bytes in the byte run. |
| **Hashes** | HashListType | 0..1 | The Hashes property specifies computed hash values for this the data in this byte run. |
| **Byte\_Run\_Data** |  | 0..1 | The Byte\_Run\_Data property captures a raw dump of the byte run data. |

## HashListType Class

The HashListType class is used for representing a list of hash values.

Table 3‑44. Properties of the HashListType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Hash** | HashType | 1..\* | The Hash property specifies a single calculated hash value. |

## HashValueType Class

The HashValueType class is used for specifying the resulting value from a hash calculation.

Table 3‑45. Properties of the HashValueType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Simple\_Hash\_Value** | SimpleHashValueType | 0..1 | The Simple\_Hash\_Value property characterizes a single result value of a basic cryptograhic hash function outputting a single hexbinary hash value. |
| **Fuzzy\_Hash\_Value** | FuzzyHashValueType | 0..1 | The Fuzzy\_Hash\_Value property characterizes a single result value of a cryptograhic fuzzy hash function outputting a single complex string based hash value. (e.g., SSDEEP's Block1hash:Block2hash format). |

## SimpleHashValueType Class

The SimpleHashValueType class is used for characterizing the output of basic cryptographic hash functions outputting a single hexbinary hash value.

Properties of the SimpleHashValueType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## FuzzyHashValueType Class

The FuzzyHashValueType class is used for characterizing the output of cryptographic fuzzy hash functions outputting a single complex string based hash value.

Properties of the FuzzyHashValueType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## FuzzyHashStructureType Class

The FuzzyHashStructureType class is used for characterizing the internal components of a cryptographic fuzzy hash algorithmic calculation.

Table 3‑46. Properties of the FuzzyHashStructureType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Block\_Size** | IntegerObjectPropertyType | 0..1 | The Block\_Size property characterizes the calculated block size for this fuzzy hash calculation. |
| **Block\_Hash** | FuzzyHashBlockType | 0..1 | The Block\_Hash property characterizes specification of the elemental components utilized for a fuzzy hash calculation on the hashed object utilizing the Block\_Size property to calculate trigger points. |

## FuzzyHashBlockType Class

The FuzzyHashBlockType class is used for characterizing the internal components of a single block in a cryptographic fuzzy hash algorithmic calculation.

Table 3‑47. Properties of the FuzzyHashBlockType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Block\_Hash\_Value** | HashValueType | 0..1 | The Block\_Hash\_Value property characterizes a fuzzy hash calculation result value for this Block. |
| **Segment\_Count** | IntegerObjectPropertyType | 0..1 | The Segment\_Count property characterizes the number of segments identified and utilized within this fuzzy hash calculation. |
| **Segments** | HashSegmentsType | 0..1 | The Segments property characterizes the set of segments identified and utilized within this fuzzy hash calculation. |

## HashSegmentsType Class

The HashSegmentsType class is used for characterizing the internal components of a set of trigger point-delimited segments in a cryptographic fuzzy hash algorithmic calculation.

Table 3‑48. Properties of the HashSegmentsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Segment** | HashSegmentType | 1..\* | The Segment property characterizes a single segment identified and utilized within this fuzzy hash calculation. |

## HashSegmentType Class

The HashSegmentType class is used for characterizing the internal components of a single trigger point-delimited segment in a cryptographic fuzzy hash algorithmic calculation.

Table 3‑49. Properties of the HashSegmentType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Trigger\_Point** | HexBinaryObjectPropertyType | 0..1 | The Trigger\_Point property characterizes the offset within the hashed object of the trigger point for this segment. |
| **Segment\_Hash** | HashValueType | 0..1 | The Segment\_Hash property characterizes a calculated hash value for this segment. |
| **Raw\_Segment\_Content** |  | 0..1 | The Raw\_Segment\_Content property captures the raw content of this segment of the hashed object. |

## HashType Class

The HashType class is intended to characterize hash values.

Table 3‑50. Properties of the HashType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Type** | VocabularyStringType | 0..1 | The Type property specifies the type of hash algorithm used to create the hash value. Examples of potential types of hashes are *MD5, SHA1* and *SHA256* (these specific values are only provided to help explain the property: they are neither recommended values nor necessarily part of any existing vocabulary). The content creator may choose any arbitrary value or may constrain the set of possible values by referencing an externally-defined vocabulary or leveraging a formally defined vocabulary extending from the stixCommon:ControlledVocabularyStringType class. The STIX default vocabulary class for use in the property is ‘*HashNameEnum-1.0’*. |
| **Simple\_Hash\_Value** | SimpleHashValueType | 0..1 | The Simple\_Hash\_Value property specifies a single result value of a basic cryptograhic hash function outputting a single hexbinary hash value. |
| **Fuzzy\_Hash\_Value** | FuzzyHashValueType | 0..1 | The Fuzzy\_Hash\_Value property specifies a single result value of a cryptograhic fuzzy hash function outputting a single complex string based hash value. (e.g. SSDEEP's Block1hash:Block2hash format). |
| **Fuzzy\_Hash\_Structure** | FuzzyHashStructureType | 0..\* | The Fuzzy\_Hash\_Structure property enables the characterization of the key internal components of a fuzzy hash calculation with a given block size. |

## StructuredTextType Class

The StructuredTextType class is a type representing a generalized structure for capturing structured or unstructured textual information such as descriptions of things.

Table 3‑51. Properties of the StructuredTextType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **structuring\_format** | basicDataTypes:  BasicString | 0..1 | The structuring\_format property specifies a particular structuring format (e.g., HTML5) used within an instance of StructuredTextType. If this property is absent, then markup MUST NOT be used. |

## DataSegmentType Class

The DataSegmentType is intended to provide a relatively abstract way of characterizing data segments that may be written/read/transmitted or otherwise utilized in actions or behaviors.

Table 3‑52. Properties of the DataSegmentType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **id** | basicDataTypes:  QualifiedName | 0..1 | The id property specifies a globally unique identifier for the DataSegment. |
| **Data\_Format** | DataFormatEnum | 0..1 | The Data\_Format property characterizes the type of data contained in the Data\_Segment element. |
| **Data\_Size** | DataSizeType | 0..1 | The Data\_Size property characterizes the size of the data contained in this element. |
| **Byte\_Order** | EndiannessType | 0..1 | The Byte\_Order property characterizes the endianness of the unpacked (e.g., decoded, unencrypted, etc.) data stored within the Data\_Segment property. |
| **Data\_Segment** | StringObjectPropertyType | 0..1 | The Data\_Segment property characterizes the actual segment of data being characterized. |
| **Offset** | IntegerObjectPropertyType | 0..1 | The Offset property characterizes where to start searching for the specified data segment in an object, in bytes. |
| **Search\_Distance** | IntegerObjectPropertyType | 0..1 | The Search\_Distance property characterizes how far into an object should be ignored, in bytes, before starting to search for the specified data segment relative to the end of the previous data segment. |
| **Search\_Within** | IntegerObjectPropertyType | 0..1 | The Search\_Within property characterizes that at most N bytes are between data segments in related objects. |

## DataSizeType Class

The DataSizeType class specifies the size of the data segment.

Table 3‑53. Properties of the DataSizeType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **units** | DataSizeUnitsEnum | 0..1 | The units property specifies the Units used in the object size element. |

## PlatformSpecificationType Class

The PlatformSpecificationType class is a modularized data type intended for providing a consistent approach to uniquely specifying the identity of a specific platform. In addition to capturing basic information, this type is intended to be extended to enable the structured description of a platform instance using the XML Schema extension feature. The CybOX default extension uses the Common Platform Enumeration (CPE) Applicability Language schema to do so. The extension that defines this is captured in the CPE23PlatformSpecificationType in the http://cybox.mitre.org/extensions/platform#CPE2.3-1 namespace. This type is defined in the extensions/platform/cpe2.3.xsd file.

Table 3‑54. Properties of the PlatformSpecificationType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Description** | StructuredTextType | 0..1 | The Description property captures a technical description of the PlatformSpecification. Any length is permitted. Optional formatting is supported via the structuring\_format property of the StructuredTextType class. |
| **Identifier** | PlatformIdentifierType | 0..\* | The Identifier property characterizes a pre-defined name for the given platform using some naming scheme. For example, one could provide a CPE (Common Platform Enumeration) name using the CPE naming format. |

## PlatformIdentifierType Class

Used to specify a name for a platform using a particular naming system and also allowing a reference pointing to more information about that naming scheme. For example, one could provide a CPE (Common Platform Enumeration) name using the CPE naming format. In this case, the system value could be "CPE" while the system\_ref value could be "http://scap.nist.gov/specifications/cpe/".

Table 3‑55. Properties of the PlatformIdentifierType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **system** | basicDataTypes:  BasicString | 0..1 | The system property captures the naming system from which the indicated name was drawn. |
| **system-ref** | basicDataTypes:  URI | 0..1 | The system-ref property specifies a reference to information about the naming system from which the indicated name was drawn. |

## MetadataType Class

The MetadataType class is intended as mechanism to capture any non-context-specific metadata.

Table 3‑56. Properties of the MetadataType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **type** | basicDataTypes:  BasicString | 0..1 | The type property captures the type of name of a single metadata property. |
| **Value** | basicDataTypes:  BasicString | 0..1 | The Value property captures the value of name of a single metadata property. |
| **SubDatum** | MetadataType | 0..\* | The SubDatum property uses recursion of the MetadataType to characterize subdatum structures for this metadata property. |

## EnvironmentVariableListType Class

The EnvironmentVariableListType class is used for representing a list of environment variables.

Table 3‑57. Properties of the EnvironmentVariableListType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Environment\_Variable** | EnvironmentVariableType | 1..\* | The Environment\_Variable property is used for capturing environment variables using a name/value pair. |

## EnvironmentVariableType Class

The EnvironmentVariableType class is used for representing environment variables using a name/value pair.

Table 3‑58. Properties of the EnvironmentVariableType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Name** | StringObjectPropertyType | 1 | The Name property characterizes the name of the environment variable. |
| **Value** | StringObjectPropertyType | 0..1 | The Value property characterizes the value of the environment variable. |

## DigitalSignaturesType Class

The DigitalSignaturesType class is used for representing a list of digital signatures.

Table 3‑59. Properties of the DigitalSignaturesType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Digital\_Signature** | DigitalSignatureInfoType | 0..\* | The Digital\_Signature property characterizes a single digital signature for this Object. |

## DigitalSignatureInfoType Class

The DigitalSignatureInfoType class is used as a way to represent some of the basic information about a digital signature.

Properties of the DigitalSignatureInfoType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **signature\_exists** | basicDataTypes:  Boolean | 0..1 | The signature\_exists property specifies whether the digital signature exists. |
| **signature\_verified** | basicDataTypes:  Boolean | 0..1 | The signature\_verified property specifies if the digital signature is verified. |
| **Certificate\_Issuer** | StringObjectPropertyType | 0..1 | The Certificate\_Issuer property characterizes the certificate issuer of the digital signature. |
| **Certificate\_Subject** | StringObjectPropertyType | 0..1 | The Certificate\_Subject property characterizes the certificate subject of the digital signature. |
| **Signature\_Description** | StringObjectPropertyType | 0..1 | The Signature\_Description property characterizes a description of the digital signature. |

## PatternableFieldType Class

The PatternableFieldType class is a grouping of attributes applicable to defining patterns on a specific field.

Table 3‑60. Properties of the PatternableFieldType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **condition** | ConditionTypeEnum | 0..1 | The condition property specifies the relevant condition to apply to the value. |
| **is\_case\_sensitive** | basicDataTypes:  Boolean | 0..1 | The is\_case\_sensitive property should be used when specifying the case-sensitivity of a pattern which uses an Equals, DoesNotEqual, Contains, DoesNotContain, StartsWith, EndsWith, or FitsPattern condition. The default value for this property is ‘*true*’, which indicates that pattern evaluations are to be considered case-sensitive. |
| **apply\_condition** | ConditionApplicationEnum | 0..1 | The apply\_condition property indicates how a condition should be applied when the field body contains a list of values. (Its value is moot if the field value contains only a single value - both possible values for this property would have the same behavior.) If this property is set to ANY, then a pattern is considered to be matched if the provided condition successfully evaluates for any of the values in the field body. If the property is set to ALL, then the pattern only matches if the provided condition successfully evaluates for every value in the field body. |
| **delimiter** | basicDataTypes:  BasicString | 0..1 | The delimiter property specifies the delimiter used when defining lists of values. The default value is "##comma##". |
| **bit\_mask** | basicDataTypes:  hexBinary | 0..1 | The bit\_mask property Used to specify a bit\_mask in conjunction with one of the defined binary conditions (bitwiseAnd, bitwiseOr, and bitwiseXor). This bitmask is then uses as one operand in the indicated bitwise computation. |
| **pattern\_type** | PatternTypeEnum | 0..1 | The pattern\_type property defines the type of pattern used if one is specified for the property value. This is applicable only if the Condition property is set to 'FitsPattern'. |
| **regex\_syntax** | basicDataTypes:  BasicString | 0..1 | The regex\_syntax property captures the syntax format used for a regular expression, if one is specified for the property value. This is applicable only if the Condition property is set to 'FitsPattern'. Setting this attribute with an empty value (e.g., "") or omitting it entirely notifies CybOX consumers and pattern evaluators that the corresponding regular expression utilizes capabilities, character classes, escapes, and other lexical tokens defined by the CybOX Language Specification. Setting this attribute with a non-empty value notifies CybOX consumers and pattern evaluators that the corresponding regular expression utilizes capabilities not defined by the CybOX Language Specification. The regular expression must be evaluated through a compatible regular expression engine in this case. |
| **has\_changed** | basicDataTypes:  Boolean | 0..1 | The has\_changed property specifies a targeted observation pattern of whether the associated property value has changed. This property would be leveraged within a pattern observable triggering on whether the value of a single property value has changed. |
| **trend** | basicDataTypes:  Boolean | 0..1 | The trend property specifies a targeted observation pattern of the nature of any trend in the associated property value. This property would be leveraged within a pattern observable triggering on the matching of a specified trend in the value of a single specified property. |

## ControlledVocabularyStringType Class

The ControlledVocabularyStringType is used as the basis for defining controlled vocabularies.

Properties of the ControlledVocabularyStringType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **vocab\_name** | basicDataTypes:  BasicString | 0..1 | The vocab\_name property specifies the name of the controlled vocabulary. |
| **vocab\_reference** | basicDataTypes:URI | 0..1 | The vocab\_reference property specifies the URI to the location of where the controlled vocabulary is defined, e.g., in an externally located XML schema file. |

## DateWithPrecisionType Class

This type is used as a replacement for the standard xs:date type but allows for the representation of the precision of the date. If the precision is given, consumers must ignore the portions of this property that is more precise than the given precision. Producers should zero-out (fill with zeros) digits in the date that are required by the xs:date datatype but are beyond the specified precision. In order to avoid ambiguity, it is strongly suggested that all dates include a specification of the timezone if it is known.

Properties of the DateWithPrecisionType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **precision** | DatePrecisionEnum | 0..1 | The precision property of the associated date, specifies how precise the date is. If omitted, the default is "day", meaning the full property value. |

## DateTimeWithPrecisionType Class

This type is used as a replacement for the standard xs:dateTime type but allows for the representation of the precision of the dateTime. If the precision is given, consumers must ignore the portions of this property that is more precise than the given precision. Producers should zero-out (fill with zeros) digits in the dateTime that are required by the xs:dateTime datatype but are beyond the specified precision. In order to avoid ambiguity, it is strongly suggested that all dateTimes include a specification of the timezone if it is known.

Table 3‑61. Properties of the DateTimeWithPrecisionType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **precision** | DateTimePrecisionEnum | 0..1 | The precision of the associated dateTime. If omitted, the default is "second", meaning the full property value (including fractional seconds). |

## SIDType Class

SIDType specifies Windows Security ID (SID) types via a union of the SIDTypeEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectPropertyType, for permitting complex (i.e. regular-expression based) specifications. Properties that use this type can express multiple values by providing them using a delimiter-separated list. The default delimiter is '##comma##' (no quotes) but can be overridden through use of the delimiter property. Note that whitespace is preserved and so, when specifying a list of values, do not include a space following the delimiter in a list unless the first character of the next list item should, in fact, be a space.

Properties of the SIDType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## Layer4ProtocolType Class

Layer4ProtocolType specifies Layer 4 protocol types, via a union of the Layer4ProtocolEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectPropertyType, for permitting complex (i.e. regular-expression based) specifications.

Properties of the Layer4ProtocolType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## EndiannessType Class

The EndiannessType specifies names for byte ordering methods.

Properties of the EndiannessType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## CipherType Class

CipherType specifies encryption algorithms, via a union of the CipherEnum type and the atomic xs:string type. Its base type is the CybOX Core BaseObjectPropertyType, for permitting complex (i.e. regular-expression based) specifications.

Properties of the CipherType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## RegionalRegistryType Class

The RegionalRegistryType specifies a Regional Internet Registry (RIR) for a given WHOIS entry. RIRs defined by the RegionalRegistryTypeEnum may be used, as well as those specified by a free form text string.

Properties of the RegionalRegistryType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |

## SourceClassTypeEnum Class

Table 3‑62. Literals of the SourceClassTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **Network** | Describes a Network-based cyber observation. |
| **System** | Describes a System-based cyber observation. |
| **Software** | Describes a Software-based cyber observation. |

## SourceTypeEnum Class

Table 3‑63. Literals of the SourceTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **Tool** | Describes a cyber observation made using various tools, such as scanners, firewalls, gateways, protection systems, and detection systems. See ToolTypeEnum for a more complete list of tools that CybOX supports. |
| **Analysis** | Describes a cyber observation made from analysis methods, such as Static and Dynamic methods. See AnalysisMethodTypeEnum for a more complete list of methods that CybOX supports. |
| **Information Source** | Describes a cyber observation made using other information sources, such as logs, Device Driver APIs, and TPM output data. See InformationSourceTypeEnum for a more complete list of information sources that CybOX supports. |

## CompensationModelEnum Class

Table 3‑64. Literals of the CompensationModelEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **Freeware** | Specifies that the tool is available for use at no monetary cost as the compensation model. |
| **Shareware** | Specifies that the tool is proprietary and offers a limited use license as the compensation model. |
| **Commercial** | Specifies that the tool is produced for sale or serves commercial purposes as the compensation model. |
| **Adware** | Specifies that the tool uses automatically rendered advertisements as the compensation model. |

## ToolReferenceTypeEnum Class

Table 3‑65. Literals of the ToolReferenceTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **Documentation** | The reference is to documentation about the identified tool. |
| **Source** | The reference is to source code for the identified tool. |
| **Download** | The reference is to where an executable version of the tool can be downloaded. |
| **Execute** | The reference is to the tool implemented as an online service. |
| **Other** | The reference is to material about the tool not covered by other values in this enumeration. |

## ConditionTypeEnum Class

Table 3‑66. Literals of the ConditionTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **Equals** | Specifies the equality or = condition. |
| **DoesNotEqual** | Specifies the "does not equal" or != condition. |
| **Contains** | Specifies the "contains" condition. |
| **DoesNotContain** | Specifies the "does not contain" condition. |
| **StartsWith** | Specifies the "starts with" condition. |
| **EndsWith** | Specifies the "ends with" condition. |
| **GreaterThan** | Specifies the "greater than" condition. |
| **GreaterThanOrEqual** | Specifies the "greater than or equal to" condition. |
| **LessThan** | Specifies the "less than" condition. |
| **LessThanOrEqual** | Specifies the "less than or equal" condition. |
| **InclusiveBetween** | The pattern is met if the given value lies between the values indicated in the field value body, inclusive of the bounding values themselves. The field value body MUST contain at least 2 values to be valid. If the field value body contains more than 2 values, then only the greatest and least values are considered. (I.e., If the body contains "2,4,6", then an InclusiveBetween condition would be satisfied if the observed value fell between 2 and 6, inclusive. Since this is an inclusive range, an observed value of 2 or 6 would fit the pattern in this example.) As such, always treat the InclusiveBetween condition as applying to a single range for the purpose of evaluating the apply\_condition attribute. |
| **ExclusiveBetween** | The pattern is met if the given value lies between the values indicated in the field value body, exclusive of the bounding values themselves. The field value body MUST contain at least 2 values to be valid. If the field value body contains more than 2 values, then only the greatest and least values are considered. (I.e., If the body contains "2,4,6", then an InclusiveBetween condition would be satisfied if the observed value fell between 2 and 6, exclusive. Since this is an exclusive range, an observed value of 2 or 6 would not fit the pattern in this example.) As such, always treat the ExclusiveBetween condition as applying to a single range for the purpose of evaluating the apply\_condition attribute. |
| **FitsPattern** | Specifies the condition that a value fits a given pattern. |
| **BitwiseAnd** | Specifies the condition of bitwise AND. Specifically, when applying this pattern, a given value is bitwise-ANDed with the bit\_mask attribute value (which must be present). If the result is identical to the value provided in the body of this field value, the pattern is considered fulfilled. |
| **BitwiseOr** | Specifies the condition of bitwise OR. Specifically, when applying this pattern, a given value is bitwise-ORed with the bit\_mask attribute value (which must be present). If the result is identical to the value provided in the body of this field value, the pattern is considered fulfilled. |
| **BitwiseXor** | Specifies the condition of bitwise XOR. Specifically, when applying this pattern, a given value is bitwise-XORed with the bit\_mask attribute value (which must be present). If the result is identical to the value provided in the body of this field value, the pattern is considered fulfilled. |

## ConditionApplicationEnum Class

Table 3‑67. Literals of the ConditionApplicationEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **ANY** | Indicates that a pattern holds if the given condition can be successfully applied to any of the field values. |
| **ALL** | Indicates that a pattern holds only if the given condition can be successfully applied to all of the field values. |
| **NONE** | Indicates that a pattern holds only if the given condition can be successfully applied to none of the field values. |

## DatatypeEnum Class

Table 3‑68. Literals of the DatatypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **string** | Specifies the string datatype as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#string for more information. |
| **int** | Specifies the int datatype as it applies to the W3C standard for int. See http://www.w3.org/TR/xmlschema-2/#int for more information. |
| **float** | Specifies the float datatype as it apples to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#float for more information. |
| **date** | Specifies a date, which is usually in the form yyyy-mm--dd as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#date for more information. |
| **positiveInteger** | Specifies a positive integer in the infinite set {1,2,...} as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#positiveInteger for more information. |
| **unsignedInt** | Specifies an unsigned integer, which is a nonnegative integer in the set {0,1,2,...,4294967295} as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#unsignedInt for more information. |
| **dateTime** | Specifies a date in full format including both date and time as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#dateTime for more information. |
| **time** | Specifies a time as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#time for more information. |
| **boolean** | Specifies a boolean value in the set {true,false,1,0} as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#boolean for more information. |
| **name** | Specifies a name (which represents XML Names) as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#Name and http://www.w3.org/TR/2000/WD-xml-2e-20000814#dt-name for more information. |
| **long** | Specifies a long integer, which is an integer whose maximum value is 9223372036854775807 and minimum value is -9223372036854775808 as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#long for more information. |
| **unsignedLong** | Specifies an unsigned long integer, which is an integer whose maximum value is 18446744073709551615 and minimum value is 0 as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#unsignedLong for more information. |
| **duration** | Specifies a length of time in the extended format PnYn MnDTnH nMnS, where nY represents the number of years, nM the number of months, nD the number of days, 'T' is the date/time separator, nH the number of hours, nM the number of minutes and nS the number of seconds, as it applies to the W3 standard. See http://www.w3.org/TR/xmlschema-2/#duration for more information. |
| **double** | Specifies a decimal of datatype double as it is patterned after the IEEE double-precision 64-bit floating point type (IEEE 754-1985) and as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#double for more information. |
| **nonNegativeInteger** | Specifies a non-negative integer in the infinite set {0,1,2,...} as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#nonNegativeInteger for more information. |
| **hexBinary** | Specifies arbitrary hex-encoded binary data as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#hexBinary for more information. |
| **anyURI** | Specifies a Uniform Resource Identifier Reference (URI) as it applies to the W3C standard and to RFC 2396, as amended by RFC 2732. See http://www.w3.org/TR/xmlschema-2/#anyURI for more information. |
| **base64Binary** | Specifies base64-encoded arbitrary binary data as it applies to the W3C standard. See http://www.w3.org/TR/xmlschema-2/#base64Binary for more information. |
| **IPv4 Address** | Specifies an IPV4 address in dotted decimal form. CIDR notation is also accepted. |
| **IPv6 Address** | Specifies an IPV6 address, which is represented by eight groups of 16-bit hexadecimal values separated by colons (:) in the form a:b:c:d:e:f:g:h. CIDR notation is also accepted. |
| **Host Name** | Specifies a host name. For compatibility reasons, this could be any string. Even so, it is best to use the proper notation for the given host type. For example, web hostnames should be written as fully qualified hostnames in practice. |
| **MAC Address** | Specifies a MAC address, which is represented by six groups of 2 hexdecimal digits, separated by hyphens (-) or colons (:) in transmission order. |
| **Domain Name** | Specifies a domain name, which is represented by a series of labels concatenated with dots conforming to the rules in RFC 1035, RFC 1123, and RFC 2181. |
| **URI** | Specifies a Uniform Resource Identifier, which identifies a name or resource and can act as a URL or URN. |
| **TimeZone** | Specifies a timezone in UTC notation (UTC+number). |
| **Octal** | Specifies arbitrary octal (base-8) encoded data. |
| **Binary** | Specifies arbitrary binary encoded data. |
| **BinHex** | Specifies arbitrary data encoded in the Mac OS-originated BinHex format. |
| **Subnet Mask** | Specifies a subnet mask in IPv4 or IPv6 notation. |
| **UUID/GUID** | Specifies a globally/universally unique ID represented as a 32-character hexadecimal string. See ISO/IEC 11578:1996 Information technology -- Open Systems Interconnection -- Remote Procedure Call - http://www.iso.ch/cate/d2229.html. |
| **Collection** | Specifies data represented as a container of multiple data of a shared elemental type. |
| **CVE ID** | Specifies a CVE ID, expressed as CVE- appended by a four-digit integer, a - and another four-digit integer, as in CVE-2012-1234. |
| **CWE ID** | Specifies a CWE ID, expressed as CWE- appended by an integer. |
| **CAPEC ID** | Specifies a CAPEC ID, expressed as CAPEC- appended by an integer. |
| **CCE ID** | Specifies a CCE ID, expressed as CCE- appended by an integer. |
| **CPE Name** | Specifies a CPE Name. See http://cpe.mitre.org/specification/archive/version2.0/cpe-specification\_2.0.pdf for more information. |

## PatternTypeEnum Class

Table 3‑69. Literals of the PatternTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **Regex** | Specifies the regular expression pattern type. |
| **Binary** | Specifies the binary (bit operations) pattern type. |
| **XPath** | Specifies the XPath 1.0 expression pattern type. |

## DataFormatEnum Class

Table 3‑70. Literals of the DataFormatEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **Binary** | Specifies binary data. |
| **Hexadecimal** | Specifies hexadecimal data. |
| **Text** | Specifies text. |
| **Other** | Specifies any other type of data from the ones listed. |

## DataSizeUnitsEnum Class

Table 3‑71. Literals of the DataSizeUnitsEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **Bytes** | Specifies an object size in Bytes. |
| **Kilobytes** | Specifies an object size in Kilobytes. |
| **Megabytes** | Specifies an object size in Megabytes. |

## DatePrecisionEnum Class

Table 3‑72. Literals of the DatePrecisionEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **year** | Date is precise to the given year. |
| **month** | Date is precise to the given month. |
| **day** | Date is precise to the given day. |

## TimePrecisionEnum Class

Table 3‑73. Literals of the TimePrecisionEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **hour** | Time is precise to the given hour. |
| **minute** | Time is precise to the given minute. |
| **second** | Time is precise to the given second (including fractional seconds). |

## SIDTypeEnum Class

Table 3‑74. Literals of the SIDTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **SidTypeUser** | Indicates a SID of type User. |
| **SidTypeGroup** | Indicates a SID of type Group. |
| **SidTypeDomain** | Indicates a SID of type Domain. |
| **SidTypeAlias** | Indicates a SID of type Alias. |
| **SidTypeWellKnownGroup** | Indicates a SID for a well-known group. |
| **SidTypeDeletedAccount** | Indicates a SID for a deleted account. |
| **SidTypeInvalid** | Indicates an invalid SID. |
| **SidTypeUnknown** | Indicates a SID of unknown type. |
| **SidTypeComputer** | Indicates a SID for a computer. |
| **SidTypeLabel** | Indicates a mandatory integrity label SID. |

## Layer4ProtocolEnum Class

Table 3‑75. Literals of the Layer4ProtocolEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **TCP** | Specifies the Transmission Control Protocol. |
| **UDP** | Specifies the User Datagram Protocol. |
| **AH** | Specifies the Authentication Header protocol. |
| **ESP** | Specifies the Encapsulating Security Payload protocol. |
| **GRE** | Specifies the Generic Routing Encapsulation protocol. |
| **IL** | Specifies the Internet Link protocol. |
| **SCTP** | Specifies the Stream Control Transmission Protocol. |
| **Sinec H1** | Specifies the Siemens Sinec H1 protocol. |
| **SPX** | Specifies the Sequenced Packet Exchange protocol. |
| **DCCP** | Specifies the Datagram Congestion Control Protocol. |

## EndiannessTypeEnum Class

Table 3‑76. Literals of the EndiannessTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **Big-endian** | The Big-endian value specifies a big-endian byte ordering. |
| **Little-endian** | The Little-endian value specifies a little-endian byte ordering. |
| **Middle-endian** | The Middle-endian value specifies a middle-endian byte ordering. |

## CipherEnum Class

Table 3‑77. Literals of the CipherEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **3DES** | Specifies the Triple Data Encryption Standard (DES) algorithm. |
| **AES** | Specifies the Advanced Encryption Standard (AES) algorithm. |
| **Blowfish** | Specifies the Blowfish algorithm. |
| **CAST-128** | Specifies the CAST-128 algorithm. |
| **CAST-256** | Specifies the CAST-256 algorithm. |
| **DES** | Specifies the Data Encryption Standard (DES) algorithm. |
| **IDEA** | Specifies the International Data Encryption Algorithm (IDEA). |
| **Rijndael** | Specifies the Rijndael algorithm. |
| **RC5** | Specifies the RC5 algorithm. |
| **Skipjack** | Specifies the Skipjack algorithm. |

## RegionalRegistryTypeEnum Class

Table 3‑78. Literals of the RegionalRegistryTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **AfriNIC** | AfriNIC stands for African Network Information Centre, and is the RIR for Africa. |
| **ARIN** | ARIN stands for American Registry for Internet Numbers, and is the RIR for the United States, Canada, several parts of the Caribbean Region, and Antarctica. |
| **APNIC** | APNIC stands for Asia-Pacific Network Information Centre, and is the RIR for Asia, Australia, New Zealand, and neighboring countries. |
| **LACNIC** | LACNIC stands for Latin American and Caribbean Network Information Centre, and is the RIR for Latin America and parts of the Caribbean region. |
| **RIPE NCC** | RIPE NCC stands for Réseaux IP Européens Network Coordination Centre, and is the RIR for Europe, Russia, the Middle East, and Central Asia. |

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