

THE MITRE CORPORATION

STIX™ 1.2

INDICATOR SPECIFICATION (v2.2)

AUGUST 6, 2015

The Structured Threat Information eXpression (STIX™) framework defines nine core constructs and the relationships between them for the purposes of modeling cyber threat information and enabling cyber threat information analysis and sharing. This specification document defines the Indicator construct, which conveys specific Observable patterns combined with contextual information intended to represent artifacts and/or behaviors of interest within a cyber security context.

Acknowledgements

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

Trademark Information

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

Warnings

MITRE PROVIDES STIX "AS IS" AND MAKES NO WARRANTY, EXPRESS OR IMPLIED, AS TO THE ACCURACY, CAPABILITY, EFFICIENCY, MERCHANTABILITY, OR FUNCTIONING OF STIX. IN NO EVENT WILL MITRE BE LIABLE FOR ANY GENERAL, CONSEQUENTIAL, INDIRECT, INCIDENTAL, EXEMPLARY, OR SPECIAL DAMAGES, RELATED TO STIX 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

Feedback

The STIX development team welcomes any feedback regarding this document. Please send comments, questions, or suggestions to stix@mitre.org.²

¹ For detailed information see [TOU].

² For more information about the STIX Language, please visit [STIX].

Table of Contents

1	Introduction	1
1.1	STIX Specification Documents	1
1.2	Document Conventions.....	2
1.2.1	Keywords.....	2
1.2.2	Fonts.....	2
1.2.3	UML Package References.....	3
1.2.4	UML Diagrams.....	3
1.2.4.1	Class Properties	3
1.2.4.2	Diagram Icons and Arrow Types	4
1.2.4.3	Color Coding	4
1.2.5	Property Table Notation	4
1.2.6	Property and Class Descriptions	5
2	Background Information	7
2.1	Indicator-Related Component Data Models	7
2.2	Indicator Patterns.....	9
2.2.1	CybOX Observables.....	9
2.3	Simple and Composite Indicators	10
3	STIX Indicator Data Model.....	12
3.1	IndicatorVersionType Enumeration	18
3.2	ValidTimeType Class.....	19
3.3	CompositeIndicatorExpressionType Class	20
3.3.1	OperatorTypeEnum Enumeration	21
3.4	TestMechanismsType Class.....	21
3.4.1	TestMechanismType Class	22
3.5	SuggestedCOAsType Class.....	25
3.6	SightingsType Class	26
3.6.1	SightingType Class.....	27
3.6.1.1	RelatedObservablesType Class	29
3.7	RelatedIndicatorsType Class	30
3.8	RelatedCampaignReferencesType Class	32
	References	34

1 Introduction

The Structured Threat Information eXpression (STIX™) framework defines nine top-level component data models: Observable³, Indicator, Incident, TTP, ExploitTarget, CourseOfAction, Campaign, ThreatActor, and Report. This document serves as the specification for the STIX Indicator Version 2.2 data model.

As defined within the STIX language, an Indicator construct is inherently a mapping between a specific set of observable conditions (the “observable pattern”) and some sort of adversary modus operandi (TTP). More specifically, an Indicator consists of an observable pattern typically mapped to a related TTP context and potentially associated with other relevant metadata such as valid time windows, likely impact, related Campaigns, suggested Courses of Action, and source information.

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

1.1 STIX Specification Documents

The STIX 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 STIX UML model.

The STIX specification overview document provides a comprehensive overview of the full set of STIX data models [STIX_O], which in addition to the nine top-level component data models mentioned in the Introduction, includes a core data model, a common data model, a cross-cutting data marking data model, various extension data models, and a set of default controlled vocabularies. [STIX_O] also summarizes the relationship of STIX to other languages, and outlines general STIX data model conventions.

Figure 1-1 illustrates the set of specification documents that are available. The color black is used to indicate the specification overview document, altered shading differentiates the overarching Core and Common data models from the supporting data models (default vocabularies, data marking, and extensions), and the color white indicates the component data models. The Observable component data model is shown as an oval shape to indicate that it is defined as a CybOX specification (see [STIX_O] for details). This Indicator

³ The CybOX Observable data model is actually defined in the CybOX Language, not in STIX.

specification document is highlighted in its associated color (see Section 1.2.4.3). For a list of all STIX documents and related information sources, please see [STIX_o].

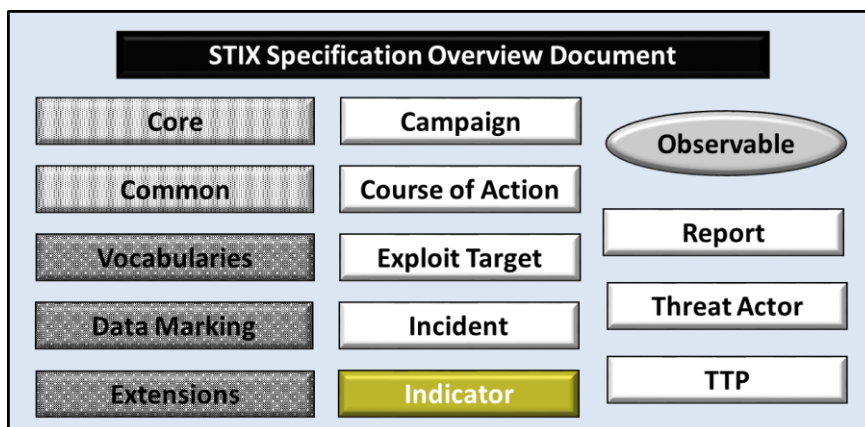


Figure 1-1. STIX Language v1.2 specification documents

All specification documents can be found on this STIX Website [STIX-SPECS].

1.2 Document Conventions

The following conventions are used in this document.

1.2.1 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].

1.2.2 Fonts

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

- Capitalization is used for STIX high level concepts, which are defined in the STIX Specification Overview [STIX_o].

Examples: Indicator, Course of Action, Threat Actor

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

Examples: `RelatedIndicatorsType`, `stixCommon:StatementType`

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

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

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

1.2.3 UML Package References

Each STIX data model is captured in a different UML package (e.g., Core package, Campaign package, etc.) where the packages together compose the full STIX 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. STIX™ 1.2 Specification Overview document [STIX₀] contains a list of the packages used by the Indicator data model, along with the associated prefix notations, descriptions, examples.

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

1.2.4 UML Diagrams

This specification makes use of UML diagrams to visually depict relationships between STIX Language constructs. Note that the diagrams have been extracted directly from the full UML model for STIX; 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 STIX Common data model. Other diagrams that are included correspond to classes that specialize a superclass and abstract or generalized classes that are extended by one or more subclasses.

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








1.2.4.1 Class Properties

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

1.2.4.2 Diagram Icons and Arrow Types

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

Table 1-1. UML diagram icons

Icon	Description
	This diagram icon indicates a class. If the name is in italics, it is an abstract class.
	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.

1.2.4.3 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 Indicator specification are illustrated via exemplars in Figure 1-2.



Figure 1-2. Data model color coding

1.2.5 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 Indicator 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_Call_A, Code_B) and single logic expression in the Multiplicity column. For example, if there is a choice of property API_Call_A and Code_B, 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.

1.2.6 Property and Class Descriptions

Each class and property defined in STIX is described using the format, “The X property verb Y.” For example, in the specification for the STIX Indicator, we write, “The id property specifies a globally unique identifier for the kill chain instance.” 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 STIX.

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

Verb	STIX Definition
<u>captures</u>	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.
	<p><i>Examples:</i></p> <p>The Source property characterizes the source of the sighting information. Examples of details <u>captured</u> include identifying characteristics, time-related attributes, and a list of the tools used to collect the information.</p> <p>The Description property <u>captures</u> a textual description of the Indicator.</p>

<u>characterizes</u>	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.
	<p><i>Examples:</i></p> <p>The <code>Confidence</code> property <u>characterizes</u> the level of confidence in the accuracy of the overall content captured in the Incident.</p> <p>The <code>ActivityType</code> class <u>characterizes</u> basic information about an activity a defender might use in response to a Campaign.</p>
<u>specifies</u>	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.
	<p><i>Example:</i></p> <p>The <code>version</code> property <u>specifies</u> the version identifier of the STIX Campaign data model used to capture the information associated with the Campaign.</p>

2 Background Information

In this section, we provide high level information about the Indicator data model that is necessary to fully understand the Indicator data model specification details given in Section 3.

2.1 Indicator-Related Component Data Models

As will be explicitly detailed in Section 2.2, a STIX Indicator leverages four other top-level STIX component constructs, namely Campaign, Course of Action, Observable (as defined with the CybOX Language; see Section 2.2.1), and TTP (as indicated by the outward-oriented arrows). Figure 2-1 illustrates the relationship between the Indicator and the other core constructs. As stated in Section 1.1, each of these components is defined in a separate specification document.

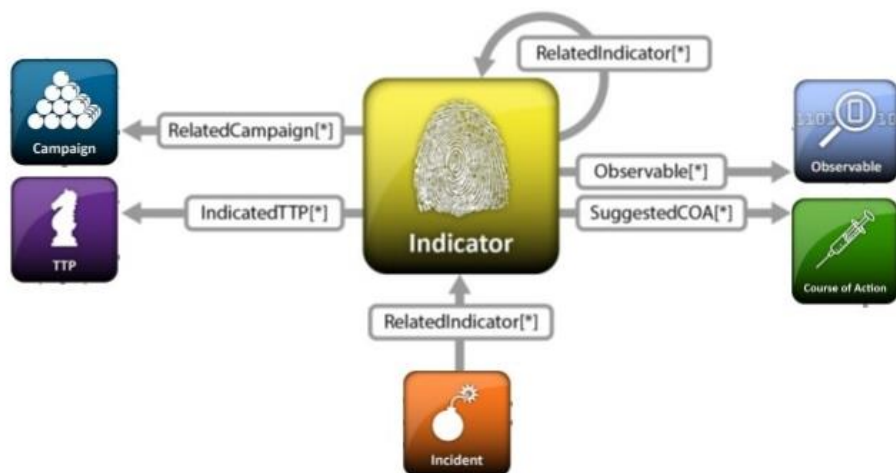


Figure 2-1. High level view of the Indicator data model

In this section, we give a high level summary of the relationship between the Indicator data model and the other components to which an Indicator may refer. We also make note of the fact that the Indicator data model can be self-referential. Other relationships shown in the diagram are defined in the specification of the component that they originate from.

- **Campaign**

A STIX Campaign represents a set of TTPs, Incidents, or Threat Actors that together express a common intent or desired effect. For example, an adversary using a particular set of TTPs (malware and tools) to target an industry sector with a specific intent may constitute a Campaign. In the STIX data model, a Campaign represents both that intent itself and, perhaps more importantly, acts as a meta-construct to capture the associated TTPs, incidents, and Threat Actors that are part of that Campaign. Please see the STIX Campaign data model specification [STIX_{CAM}] for details.

The Indicator data model references the Campaign data model as a means to identify Campaigns for which the Indicator may be relevant.

- **Course of Action**

A STIX Course of Action (COA) is used to convey information about courses of action that may be taken either in response to an attack or as a preventative measure prior to an attack. A Course of Action component captures a variety of information such as the Course of Action's objective, likely impact, efficacy, and cost. Please see the STIX Course of Action data model specification [STIX_{COA}] for details.

The Indicator data model references the Course of Action data model as a means to identify Courses of Action that may be appropriate for responding to the Indicator.

- **Observable**

A STIX Observable (as defined with the CybOX Language⁴) represents stateful properties or measurable events pertinent to the operation of computers and networks. Implicit in this is a practical need for descriptive capability of two forms of observables: "observable instances" and "observable patterns." Observable instances represent actual specific observations that took place in the cyber domain. The property details of this observation are specific and unambiguous. Observable patterns represent conditions for a potential observation that may occur in the future or may have already occurred and exists in a body of observable instances. These conditions may be anything from very specific concrete patterns that would match very specific observable instances to more abstract generalized patterns that have the potential to match against a broad range of potential observable instances.

The Indicator data model leverages the Observable data model to specify the observable pattern that forms the core of a STIX Indicator.

- **Indicator**

The Indicator data model is self-referential, enabling one Indicator to reference other Indicators that are asserted to be related. Self-referential relationships between Indicators may indicate general associativity or can be used to indicate relationships between different versions of the same Indicator.

⁴ CybOX specification documents will be created after STIX specification documents are completed.

- **Tactics, Techniques and Procedures (TTP)**

A STIX Tactics, Techniques, and Procedures (TTP) is used to represent the behavior or modus operandi of cyber adversaries. Please see the STIX TTP data model specification [STIX_{TTP}] for details.

The Indicator data model references the TTP data model as a means to identify sets of specific TTPs that the presence of the Indicator observable pattern may indicate.

2.2 Indicator Patterns

Each STIX Indicator contains an observable pattern that can be matched upon (“triggered”) where the pattern can be defined using the STIX Observable component (specified as a CybOX pattern, which corresponds to an Observable belonging to the CybOX `ObservableType` class; see Section 2.2.1) and/or through a provided extension point for other pattern definition formats (non-CybOX pattern, which corresponds to a test mechanism belonging to the STIX Indicator `TestMechanismType` class; see Section 3.4.1).

It is valid for an Indicator to include both a CybOX Observable-based pattern as well as one or more test mechanisms, which provide proprietary representations of the pattern (e.g., a YARA rule and/or an OVAL rule as an extension to the `TestMechanismType` class). In this case, the patterns defined by the test mechanism MUST semantically match the pattern defined by the Observable to the extent possible given the test mechanism language and the CybOX pattern language, however enforcement of such matching is outside the scope of STIX.

2.2.1 CybOX Observables

We will not give specification information for the CybOX `ObservableType` class, and instead we refer the reader to the appropriate CybOX specification documents. However, for a full understanding of the STIX Indicator data model, it is necessary to explain the role of the CybOX `Observable` property, which is used to specify a relevant cyber observable pattern for an Indicator.

A CybOX Observable pattern can either be on a CybOX Object with type corresponding to the CybOX `ObjectType` class (e.g., a File with name X), on a CybOX Event with type corresponding to the CybOX `EventType` class (an Event is typically one or more actions taken against one or more Objects; e.g., “delete the File with name X”), or as an Observable Composition with type corresponding to the CybOX `ObservableCompositionType` class.

The `ObservableCompositionType` class enables a content creator to define a composite Observable pattern expression through the specification of a single Boolean operator (the operator of the expression) and a list of simple (non-composition) Observable patterns (the operands of the expression).

More complex Observable compositions (of the `ObservableCompositionType` class) can be created using multiple simple Observable patterns and/or other Observable compositions. For example, it may be desired to express the Observable Composition, $OC_X = O_A \text{ AND } (E_A \text{ OR } E_B)$. The Observable (O_A) and each of the two Events (E_A and E_B) would be created individually, and then one Observable Composition would be defined as $OC_A = E_A \text{ OR } E_B$. This permits OC_X to be rewritten as $OC_X = O_A \text{ AND } OC_A$. Note that this example shows just one or several possible constructions that generate OC_X .

The evaluation of a CybOX observable pattern can be found in the CybOX specification documents [CybOX_{COR}].

2.3 Simple and Composite Indicators

Similarly to how CybOX defines simple and composite Observable patterns, the STIX Indicator data model defines both simple and composite Indicator expressions. More specifically, a “simple” Indicator is defined as an Indicator with a single observable pattern defined (either via the CybOX-based Observable structure or via the Test Mechanism extension structure) and a single set of contextual information, and a “composite” Indicator is defined as an aggregate pattern (with its own contextual information) logically composed of other Indicators expressions (each with its own contextual information) using Boolean logic.

As the following example illustrates, more complex Indicator compositions can be created using multiple simple and/or other composite Indicators. Say we want to express the composite Indicator, $CI_X = [I_A \text{ AND } (I_B \text{ OR } I_C)] \text{ OR } (I_D \text{ AND } I_E \text{ AND } I_F)$. Each of the six Indicators (I_A through I_F) would be created individually (i.e., as simple Indicators), and then two composite Indicators could be created where $CI_1 = (I_B \text{ OR } I_C)$ and $CI_2 = (I_D \text{ AND } I_E \text{ AND } I_F)$. This permits CI_X to be rewritten as $CI_X = [I_A \text{ AND } CI_1] \text{ OR } CI_2$, which can be captured using one additional composite indicator: $CI_3 = I_A \text{ AND } CI_1$. Finally, $CI_X = CI_3 \text{ OR } CI_2$. Note that this example shows just one or several possible constructions that generate in CI_X .

As illustrated in our example, an Indicator composition may be built directly to support a desired use case. Alternatively, an Indicator composition can be built using other preexisting Indicators, which were possibly created by different producers. The particular circumstances will likely dictate the depth and variety of other content that is captured in the individual Indicators.

The need for composite Indicators is motivated by the fact that a pattern associated with an Indicator is either matched or not matched; a single, composite observable pattern will not provide partial information. For example, assume that we have defined an Indicator that uses a CybOX Observable Composition that corresponds to “Filename = X AND Mutex = Y.” It is necessary for both conditions to be true for the Indicator to trigger; the Indicator will not reveal whether one or the other or neither of the conditions is true. However, if each of the two conditions is captured as a separate Indicator, and those Indicators are

combined to create a composite Indicator, then each of the Indicators can trigger individually and finer-grained results are achieved.

3 STIX Indicator Data Model

The primary class of the STIX Indicator package is the `IndicatorType` class, which characterizes a cyber threat indicator by capturing an asserted relationship between a pattern identifying certain observable conditions and a particular TTP likely in play if those observable conditions are seen, as well as contextual information about how and when it should be acted on and how it relates to other Indicators, Campaigns, and TTPs. Similar to the primary classes of all of the component data models in STIX, the `IndicatorType` class extends a base class defined in the STIX Common data model; more specifically, it extends the `IndicatorBaseType` base class, which provides the essential identifier (`id`) and identifier reference (`idref`) properties.

The relationship between the `IndicatorType` class and the `IndicatorBaseType` base class, as well as the properties of the `IndicatorType` class, are illustrated in the UML diagram given in Figure 3-1.

Note that while all properties of the `IndicatorType` class are optional, the observable pattern (`Indicator_Expression` property) and the `Indicated_TTP` property SHOULD be present except in very rare and odd circumstances when their absence is justifiable.

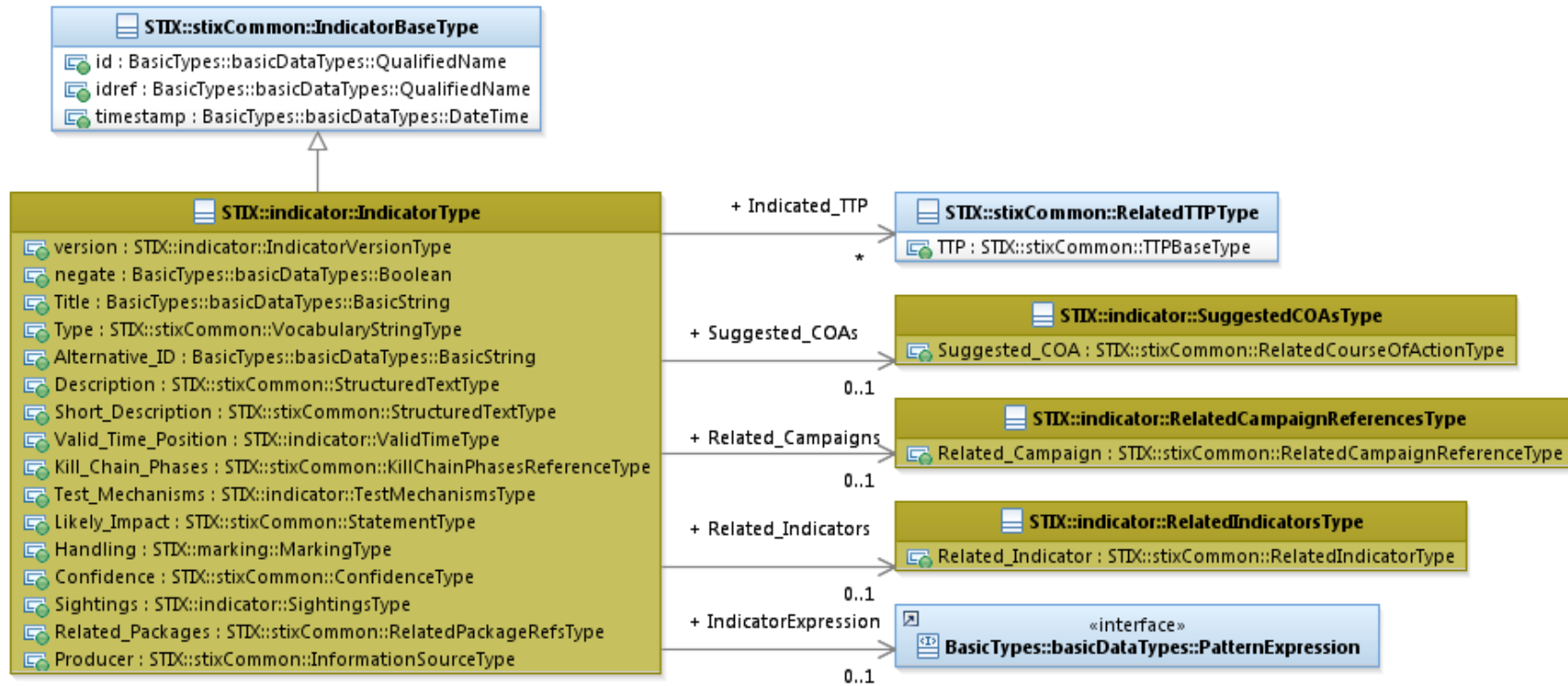


Figure 3-1. UML diagram of the `IndicatorType` class

The property table, which includes property descriptions and corresponds to the UML diagram above, is given in Table 3-1.

All classes defined in the Indicator data model are described in detail in Sections 3.1 through Section 3.8. Details are not provided for classes defined in non-Indicator data models; instead, the reader is referred to the corresponding data model specification as indicated by the package prefix specified in the Type column of the table.

Table 3-1. Properties of the `IndicatorType` class

Name	Type	Multiplicity	Description
version	<code>IndicatorVersionType</code>	0..1	The <code>version</code> property specifies the version identifier of the STIX Indicator data model used to capture the information associated with the Indicator.
negate	<code>basicDataTypes:</code> <code>Boolean</code>	0..1	The <code>negate</code> property specifies whether the absence (true) or presence (false) of the pattern constitutes the Indicator. More explicitly, if the <code>negate</code> property is <i>'false'</i> , then if the Indicator pattern is matched, the Indicator is true. However, if the <code>negate</code> property is <i>'true'</i> , then if the Indicator pattern is <i>not</i> matched, the Indicator is true. Note that this property applies to the entire indicator, not to a specific part of the Indicator pattern. The default value is <i>'false'</i> .
Title	<code>basicDataTypes:</code> <code>BasicString</code>	0..1	The <code>Title</code> property captures a title for the Indicator and reflects what the content producer thinks the Indicator as a whole should be called. The <code>Title</code> property is typically used by humans to reference a particular Indicator; however, it is not suggested for correlation.
Type	<code>stixCommon:</code> <code>VocabularyStringType</code>	0..*	The <code>Type</code> property characterizes the type of the Indicator. Examples of potential values include <i>malicious e-mail</i> , <i>URL watchlist</i> , and <i>malware artifacts</i> (these specific values are only provided to help explain the <code>Type</code> 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 <code>stixCommon:ControlledVocabularyStringType</code> class. The STIX default vocabulary class for use in the property is <i>'IndicatorTypeVocab-1.1.'</i>

Alternative_ID	<code>basicDataTypes:</code> <code>BasicString</code>	0..*	The <code>Alternative_ID</code> property specifies an alternative identifier or alias for the Indicator. The <code>Alternative_ID</code> property is not intended to capture a STIX identifier; instead, it should be used for capturing identifiers from external systems (e.g., an incident ID from an organization's Remedy system or the rule ID of a Snort rule in the Snort community repository).
Description	<code>stixCommon:</code> <code>StructuredTextType</code>	0..*	The <code>Description</code> property captures a textual description of the Indicator. Any length is permitted. Optional formatting is supported via the <code>structuring_format</code> property of the <code>StructuredTextType</code> class.
Short_Description	<code>stixCommon:</code> <code>StructuredTextType</code>	0..*	The <code>Short_Description</code> property captures a short textual description of the Indicator. This property is secondary and should only be used if the <code>Description</code> property is already populated and another, shorter description is available.
Valid_Time_Position	<code>ValidTimeType</code>	0..*	The <code>Valid_Time_Position</code> property specifies the time window for which this Indicator is valid.
IndicatorExpression	<code>basicDataTypes:</code> <code>PatternExpression</code>	0..1	The <code>IndicatorExpression</code> property characterizes an observable pattern associated with the Indicator. The <code>basicDataTypes:PatternExpression</code> class is a UML interface, which can be realized using either the <code>cybox:ObservableType</code> class (see [CybOX _{COR}]) or the <code>CompositeIndicatorExpressionType</code> class (see below for more details).
Indicated_TTP	<code>stixCommon:RelatedTTPType</code>	0..*	The <code>Indicated_TTP</code> property specifies a TTP indicated by the presence of the observable pattern within this Indicator and characterizes the relationship between the TTP and the Indicator by capturing information such as the level of confidence that the observable pattern indicates the TTP, the source of the relationship information, and the type of relationship.
Kill_Chain_Phases	<code>stixCommon:</code> <code>KillChainPhasesReferenceType</code>	0..1	The <code>Kill_Chain_Phases</code> property specifies a set of one or more kill chain phases (from one or more kill chains defined

			elsewhere) relevant to the Indicator. The kill chain property is further defined in the STIX Common specification document. For more details on the cyber kill chain see [STIX _{COM}].
Test_Mechanisms	TestMechanismsType	0..1	The <code>Test_Mechanisms</code> property specifies a set of one or more test mechanisms effective at identifying the cyber Observable patterns characterized in the Indicator.
Likely_Impact	stixCommon:StatementType	0..1	The <code>Likely_Impact</code> property characterizes the probable impact if the TTP indicated by the presence of the indicator pattern were to occur, which includes a <code>Value</code> property that specifies the impact. Examples of potential impacts include <i>none</i> , <i>minor</i> , and <i>moderate</i> (these specific values are only provided to help explain the <code>Value</code> property: they are neither recommended impacts nor necessarily part of any existing vocabulary). The content creator may choose any arbitrary impact or may constrain the set of possible impacts by referencing an externally-defined vocabulary. The STIX default vocabulary class for use in the <code>Value</code> property is ' <i>ImpactRatingEnum-1.0.</i> '
Suggested_COAs	SuggestedCOAsType	0..1	The <code>Suggested_COA</code> property specifies a Course of Action suggested for this Indicator and characterizes the relationship between the Course of Action and the Indicator by capturing information such as the level of confidence that the Course of Action and the Indicator are related, the source of the relationship information, and the type of relationship.
Handling	marking:MarkingType	0..1	The <code>Handling</code> property specifies data handling markings for the properties of this Indicator. The marking scope is limited to the Indicator and the content it contains. Note that data handling markings can also be specified at a higher level.
Confidence	stixCommon:ConfidenceType	0..1	The <code>Confidence</code> property characterizes the level of confidence that an observed instance of the defined observable pattern actually indicates the presence of the TTP. (Recall that

			an Indicator asserts a relationship between an observable pattern and a TTP.)
Sightings	<code>SightingsType</code>	0..1	The <code>Sightings</code> property characterizes sightings associated with the Indicator (observed instance). Information captured includes a sightings count and a set of zero or more sighting reports.
Related_Indicators	<code>RelatedIndicatorsType</code>	0..1	The <code>Related_Indicators</code> property specifies a set of one or more other Indicators related to this Indicator.
Related_Campaigns	<code>RelatedCampaignReferencesType</code>	0..1	The <code>Related_Campaigns</code> property specifies a set of one or more Campaigns for which the Indicator may be relevant.
Related_Packages	<code>stixCommon: RelatedPackagesRefsType</code>	0..1	The <code>Related_Packages</code> property specifies a set of one or more Packages for which the Indicator may be relevant.
Producer	<code>stixCommon: InformationSourceType</code>	0..1	The <code>Producer</code> property characterizes the source of the Indicator information. Examples of details captured include identifying characteristics, time-related attributes, and a list of the tools used to collect the information.

As discussed in Section 3.3, the observable pattern associated with an Indicator can be expressed as a combination of other Indicators. This is characterized using the `CompositeIndicatorExpressionType` class, which models a compound Indicator expression through the specification of a single operator (the operator of the expression) and the capture of one or more Indicators (the operands of the expression). Eventually, these operands are simple observable patterns that can be represented using the `cybox:ObservableType` class. The content captured for the Observable pattern includes a cyber-relevant object, event, or observable composition, as well as a title, description, keywords, source information, and fidelity information. Please see the CybOX Core Specification [CybOX_{cor}] for details.

The modeling of Indicator pattern expressions is illustrated in the UML diagram in Figure 3-2.

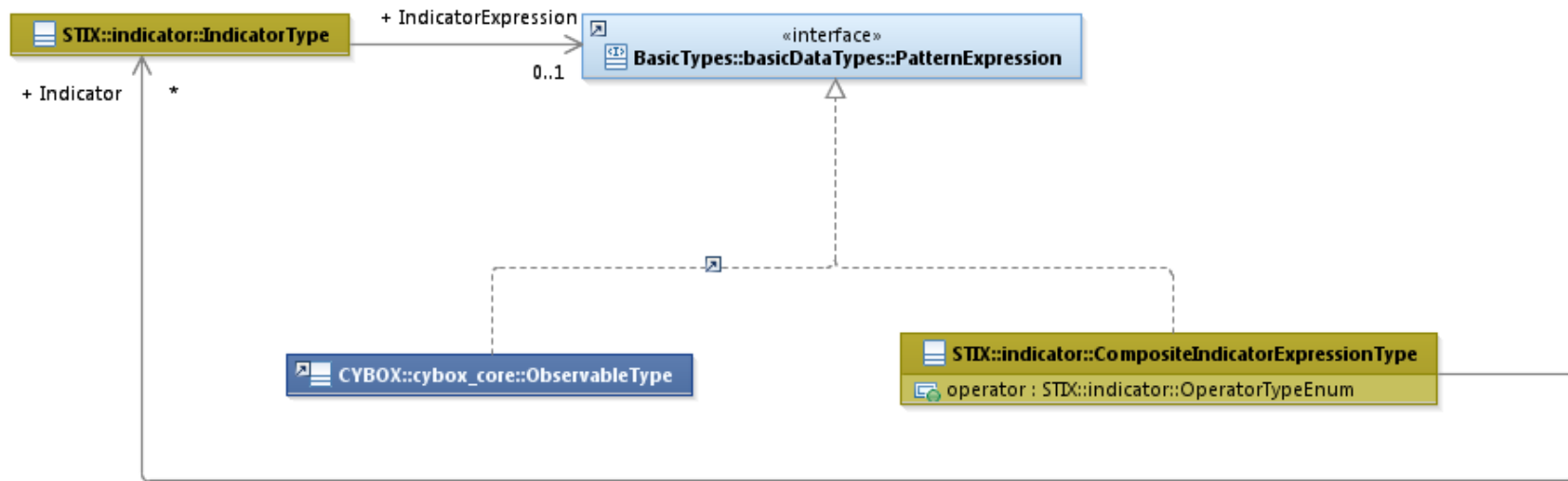


Figure 3-2. Modeling Indicator pattern expressions

3.1 IndicatorVersionType Enumeration

The `IndicatorVersionType` enumeration is an inventory of all versions of the Indicator data model that are valid in STIX Version 1.2. The enumeration literals are given in Table 3-2.

Table 3-2. Literals of the `IndicatorVersionType` enumeration

Enumeration Literal	Description
2.0	Indicator data model Version 2.0
2.0.1	Indicator data model Version 2.0.1
2.1	Indicator data model Version 2.1
2.1.1	Indicator data model Version 2.1.1
2.2	Indicator data model Version 2.2

3.2 ValidTimeType Class

The `ValidTimeType` class characterizes a temporal window during which the Indicator is able to accurately detect the TTP.

The properties of the `ValidTimeType` class are given in Table 3-3.

Table 3-3. Properties of the `ValidTimeType` class

Name	Type	Multiplicity	Description
Start_Time	<code>stixCommon:</code> <code>DateTimeWithPrecisionType</code>	0..1	The <code>Start_Time</code> property specifies the beginning of the temporal window in which an Indicator is able to accurately detect the TTP. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date and time, the <code>Start_Time</code> property MAY also capture a <code>precision</code> property to specify the granularity with which the time should be considered, as specified by the <code>DateTimePrecisionEnum</code> enumeration (e.g., <code>'hour'</code> , <code>'minute'</code>). If the <code>Start_Time</code> property is not present, then there is no constraint on the earliest time for which the

			Indicator is able to accurately detect the TTP (i.e., the temporal window is only bounded by the <code>End_Time</code> property, if present).
End_Time	<code>stixCommon:</code> <code>DateTimeWithPrecisionType</code>	0..1	The <code>End_Time</code> property specifies the end of the temporal window in which an Indicator is able to accurately detect the TTP. To avoid ambiguity, timestamps SHOULD include a specification of the time zone. In addition to capturing a date and time, the <code>End_Time</code> property MAY also capture a <code>precision</code> property to specify the granularity with which the time should be considered, as specified by the <code>DateTimePrecisionEnum</code> enumeration (e.g., <code>'hour'</code> , <code>'minute'</code>). If the <code>End_Time</code> property is not present, then there is no constraint on the latest time for which the Indicator is able to accurately detect the TTP (i.e., the temporal window is only bounded by the <code>Start_Time</code> property, if present).

3.3 CompositeIndicatorExpressionType Class

The `CompositeIndicatorExpressionType` class characterizes a composite Indicator expression through the specification of a single operator (the operator of the expression) and zero or more Indicators (the operands of the expression).

The properties of the `CompositeIndicatorExpressionType` class are given in Table 3-4.

It should be noted that each simple Indicator can “trigger” individually and independently of whether a related composite Indicator triggers. A composite Indicator will trigger only when the full specified logic on the set of individual Indicators that it comprises triggers.

Table 3-4. Properties of the `CompositeIndicatorExpressionType` class

Name	Type	Multiplicity	Description
operator	<code>OperatorTypeEnum</code>	1	The <code>operator</code> property specifies the logical composition operator for the composite Indicator expression. The enumeration that defines valid operators in the Indicator v2.2 data model contains the operators AND and OR.

Indicator	<code>IndicatorType</code>	0..*	The <code>Indicator</code> property characterizes a cyber threat indicator by capturing an asserted relationship between a pattern identifying certain observable conditions and a particular TTP likely in play if those observable conditions are seen, as well as contextual information about how and when it should be acted on and how it relates to other Indicators, Campaigns, and TTPs.
------------------	----------------------------	------	---

3.3.1 OperatorTypeEnum Enumeration

The `OperatorTypeEnum` enumeration is an inventory of valid operators for the `CompositeIndicatorExpressionType` class, which is used to define a compound Indicator expression. The enumeration literals are given in Table 3-5.

Table 3-5. Literals of the `OperatorTypeEnum` enumeration

Enumeration Literals	Description
AND	Logical AND operator
OR	Logical OR operator

3.4 TestMechanismsType Class

A test mechanism is an alternative (non-CybOX) method of expressing an observable pattern; examples include Snort rules, Yara rules, and OVAL definitions. Test mechanisms may be provided by content producers to support consumers who rely on the relevant underlying tools and methods in their operations. An Indicator may contain both a CybOX-based Observable pattern (or a Composite Indicator Expression) and one or more semantically equivalent test mechanisms where equivalency means that the test mechanisms trigger on the same conditions as the Observable (note that this equivalency is not enforced by STIX). Just as an Indicator may contain an Observable or Composite Indicator without a test mechanism, it is also possible for an Indicator to contain one or more test mechanisms without containing an Observable or Composite Indicator Expression, in which case the Indicator relies solely on non-CybOX pattern matching. Please see Sections 2.2 through 2.3 for additional information.

The `TestMechanismsType` class specifies a set of one or more test mechanisms effective at identifying the cyber observable pattern characterized in the Indicator. The property of the `TestMechanismsType` class is shown in Table 3-6.

Table 3-6. Properties of the `TestMechanismsType` class

Name	Type	Multiplicity	Description
Test_Mechanism	<code>TestMechanismType</code>	1..*	The <code>Test_Mechanism</code> property characterizes an alternative (non-CybOX) test mechanism representation for the desired observable pattern of the Indicator. Its underlying abstract class MUST be extended to enable the expression of a structured or unstructured test mechanism (e.g., Snort, OVAL).

3.4.1 TestMechanismType Class

The `TestMechanismType` class characterizes an alternative (non-CybOX) test mechanism effective for representing the desired observable pattern of the Indicator. It is an abstract class, so it **MUST** be extended via a subclass to enable the expression of any structured or unstructured test mechanism. STIX v1.2 provides five default subclasses; please refer to the “STIX Default Extensions Specification” document [STIX_{EXT}].

The `TestMechanismType` class is illustrated in Figure 3-3 (note that qualified names are not provided for the five default subclasses due to space considerations).

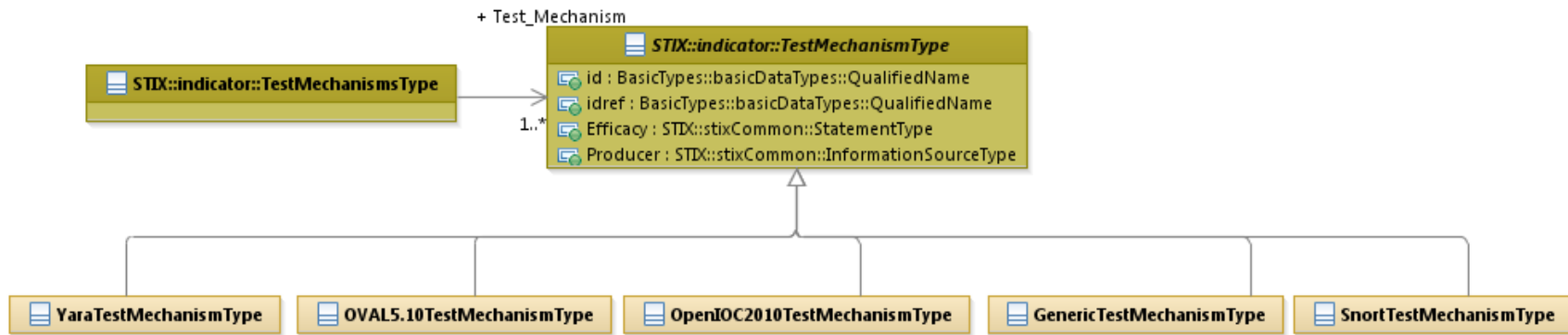


Figure 3-3. UML diagram of the TestMechanismType class

Test mechanisms may be different for different use cases. For example, one consumer may routinely rely on Snort signatures while another uses Yara rules and OVAL definitions. As stated previously, individual test mechanisms are intended to test for the same conditions as the Observable (CybOX) patterns; however, STIX does not enforce equivalency.

As listed in Table 3-7, the five default subclasses and their descriptions are defined as possible extensions to the TestMechanismType class. Additional formats can be used by either specifying the test mechanism in the Specification property of the GenericTestMechanismType class or by defining a new subclass of the TestMechanismType class. Each of the extensions is further defined in the STIX Version 1.2 Default Extensions Specification document [STIX_{EXT}].

Table 3-7. Default subclass extensions of the `TestMechanismType` class

Subclasses	Description
GenericTestMechanismType	The Generic test mechanism allows for the specification of any generic test mechanism through the use of a raw data section (via the <code>Specification</code> property).
OpenIOC2010TestMechanismType	The OpenIOC test mechanism allows for the specification of an OpenIOC test by importing the OpenIOC schema.
Data model	OpenIOC (http://www.openioc.org/#schema)
OVAL5.10TestMechanismType	The OVAL test mechanism allows for the specification of an OVAL definition through importing the OVAL schemas.
Data model	OVAL (http://oval.mitre.org/language/version5.10.1/)
SnortTestMechanismType	The Snort test mechanism allows for the specification of a snort signature through the use of raw data sections.
Signatures	https://www.snort.org/downloads/#rule-downloads
YaraTestMechanismType	The YARA test mechanism allows for the specification of a YARA test through the use of a raw data section (via the <code>Rule</code> property).
Signatures	https://github.com/plusvic/yara/releases/tag/v3.1.0

The properties of the `TestMechanismType` class are given in Table 3-8.

Table 3-8. Properties of the `TestMechanismType` class

Name	Type	Multiplicity	Description
id	<code>basicDataTypes: QualifiedName</code>	0..1	The <code>id</code> property specifies a globally unique identifier for the test mechanism instance.
idref	<code>basicDataTypes: QualifiedName</code>	0..1	The <code>idref</code> property specifies an identifier reference to a test mechanism instance specified elsewhere. When the <code>idref</code> property is used, the <code>id</code> property MUST NOT also be specified and the other properties of the <code>TestMechanismType</code> class MUST NOT hold any content.
Efficacy	<code>stixCommon: StatementType</code>	0..1	The <code>Efficacy</code> property characterizes a measure of the likely effectiveness of a <code>TestMechanismType</code> instance to detect the targeted cyber observable pattern, which includes a <code>Value</code> property that specifies the level of effectiveness. Examples of potential levels include <i>high</i> , <i>medium</i> , and <i>low</i> (these specific values are only provided to help explain the <code>Value</code> 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 levels by referencing an externally-defined vocabulary. The STIX default vocabulary class for use in the <code>Value</code> property is ' <i>HighMediumLowVocab-1.0</i> .'
Producer	<code>stixCommon: InformationSourceType</code>	0..1	The <code>Producer</code> property characterizes the source of the test mechanism information. Examples of details captured include identifying characteristics, time-related attributes, and a list of the tools used to collect the information.

3.5 SuggestedCOAsType Class

The `SuggestedCOAsType` class specifies one or more suggested Courses of Action that could be taken if the Indicator is sighted. It extends the `GenericRelationshipListType` superclass defined in the STIX Common data model, which specifies the scope (whether the elements of the set are related individually or as a group).

The UML diagram corresponding to the `SuggestedCOAsType` class is shown in Figure 3-4.

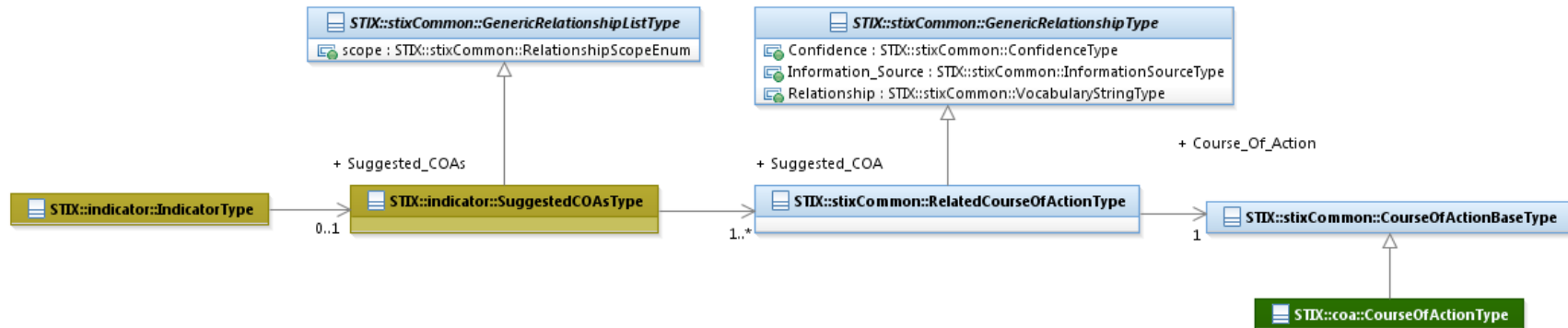


Figure 3-4. UML diagram of the SuggestedCOAsType class

The property table given in Table 3-9 corresponds to the UML diagram shown in Figure 3-4.

Table 3-9. Properties of the SuggestedCOAsType class

Name	Type	Multiplicity	Description
Suggested_COA	stixCommon: RelatedCourseOfActionType	1..*	The Suggested_COA property specifies a Course of Action suggested for this Indicator and characterizes the relationship between the Course of Action and the Indicator. The relationship between the Course of Action and the Indicator is characterized by capturing information such as the level of confidence that the Course of Action and the Indicator are related, the source of the relationship information, and the type of relationship.

3.6 SightingsType Class

The `SightingsType` class characterizes the sightings of the Indicator by capturing the number of sightings and a set of zero or more sighting reports (see Section 3.6.1 for details on what constitutes an Indicator sighting report). The properties of the

`SightingsType` class are shown in Table 3-10, and the classes defining the associated property types are discussed in Sections 3.6.1 and 3.6.1.1.

Table 3-10. Properties of the `SightingsType` class

Name	Type	Multiplicity	Description
sightings_count	<i>Integer</i>	0..1	The <code>sightings_count</code> property specifies the total number of times the Indicator was sighted.
Sighting	<code>SightingType</code>	0..*	The <code>Sighting</code> property characterizes a single sighting report for the Indicator. Note that this property can occur zero times because it's possible to just count the sightings, not capture any.

3.6.1 SightingType Class

The `SightingType` class characterizes a single sighting report for an Indicator. An Indicator sighting report may capture a variety of information associated with an observable instance that matches the Indicator's observable pattern including the time the sighting occurred, a textual description of the sighting, the set of Observables (instances) related to the sighting, the source of the sighting information (e.g., tool or an analyst name), and the level of confidence that the sighting information actually represents a sighting of that particular Indicator.

Properties of the class are given in Table 3-11. Please see Figure 3-5 in Section 3.6.1.1 for a diagram illustrating the role of the `SightingType` class.

Table 3-11. Properties of the `SightingType` class

Name	Type	Multiplicity	Description
timestamp	<code>basicDataTypes:DateTime</code>	0..1	The <code>timestamp</code> property specifies the date and time of the Indicator sighting. To avoid ambiguity, all timestamps SHOULD include a specification of the time zone.

timestamp_precision	<code>stixCommon:DateTimePrecisionEnum</code>	0..1	The <code>timestamp_precision</code> property specifies the granularity with which the <code>timestamp</code> property should be considered, as specified by the <code>DateTimePrecisionEnum</code> enumeration (e.g., <i>'hour,'</i> <i>'minute'</i>). If omitted, the default precision is <i>'second.'</i> Digits in a timestamp that are beyond the specified precision should be zeroed out.
Source	<code>stixCommon:InformationSourceType</code>	0..1	The <code>Source</code> property characterizes the the organization or tool that is the source of the sighting of the sighting information. Examples of details captured include identifying characteristics, time-related attributes, and a list of the tools used to collect the information.
Reference	<code>basicDataTypes:URI</code>	0..1	The <code>Reference</code> property specifies a formal reference to an external description of the sighting through the capture of a Uniform Resource Locator (URL).
Confidence	<code>stixCommon:ConfidenceType</code>	0..1	The <code>Confidence</code> property characterizes the level of confidence that the observed instance of the Sighting actually matches the Indicator pattern.
Description	<code>stixCommon:StructuredTextType</code>	0..*	The <code>Description</code> property captures a textual description of the Indicator sighting. Any length is permitted. Optional formatting is supported via the <code>structuring_format</code> property of the <code>StructuredTextType</code> class.
Related_Observables	<code>RelatedObservablesType</code>	0..1	The <code>Related_Observables</code> property specifies a set of one or more <code>Observable</code> instances that represent the observations for this Indicator sighting.

3.6.1.1 RelatedObservablesType Class

The `RelatedObservablesType` class specifies one or more CybOX Observables (instances) representing the actual observations that are believed to be a match for the Indicator observable pattern. It extends the `GenericRelationshipListType` superclass defined in the STIX Common data model, which specifies the scope (whether the elements of the set are related individually or as a group).

The UML diagram corresponding to the `RelatedObservablesType` class is shown in Figure 3-5.

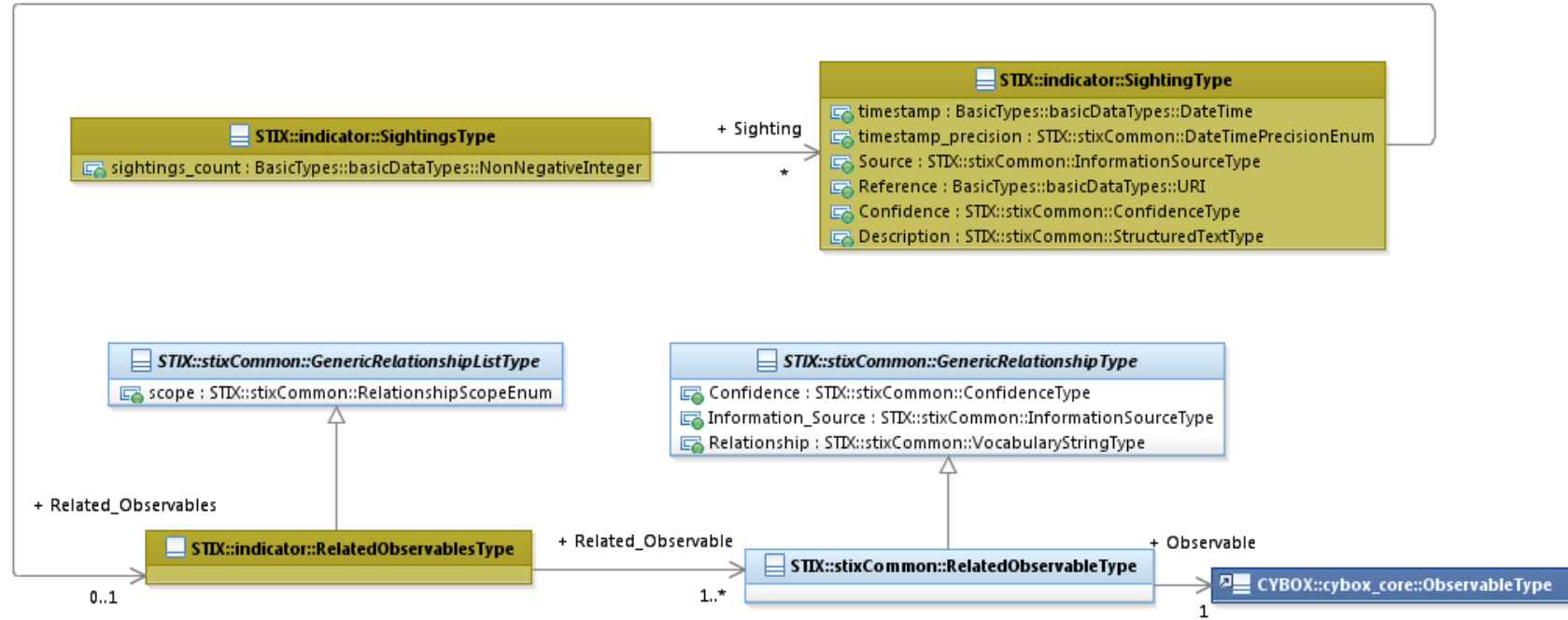


Figure 3-5. UML diagram of the `RelatedObservablesType` class

The property table given in Table 3-12 corresponds to the UML diagram shown in Figure 3-5.

Table 3-12. Properties of the `RelatedObservablesType` class

Name	Type	Multiplicity	Description
Related_Observable	<code>stixCommon:</code> <code>RelatedObservableType</code>	1..*	The <code>Related_Observable</code> property specifies an Observable instance that represents the observation for this Indicator sighting and characterizes the relationship between the Observable and the Indicator sighting by capturing information such as the level of confidence that the Observable accurately characterizes the observation for the Indicator sighting, the source of the relationship information, and the type of relationship.

3.7 RelatedIndicatorsType Class

The `RelatedIndicatorsType` class specifies a set of one or more other Indicators asserted to be related to this Indicator and therefore is a self-referential relationship. It extends the `GenericRelationshipListType` superclass defined in the STIX Common data model, which specifies the scope (whether the elements of the set are related individually or as a group).

The UML diagram corresponding to the `RelatedIndicatorsType` class is shown in Figure 3-6.

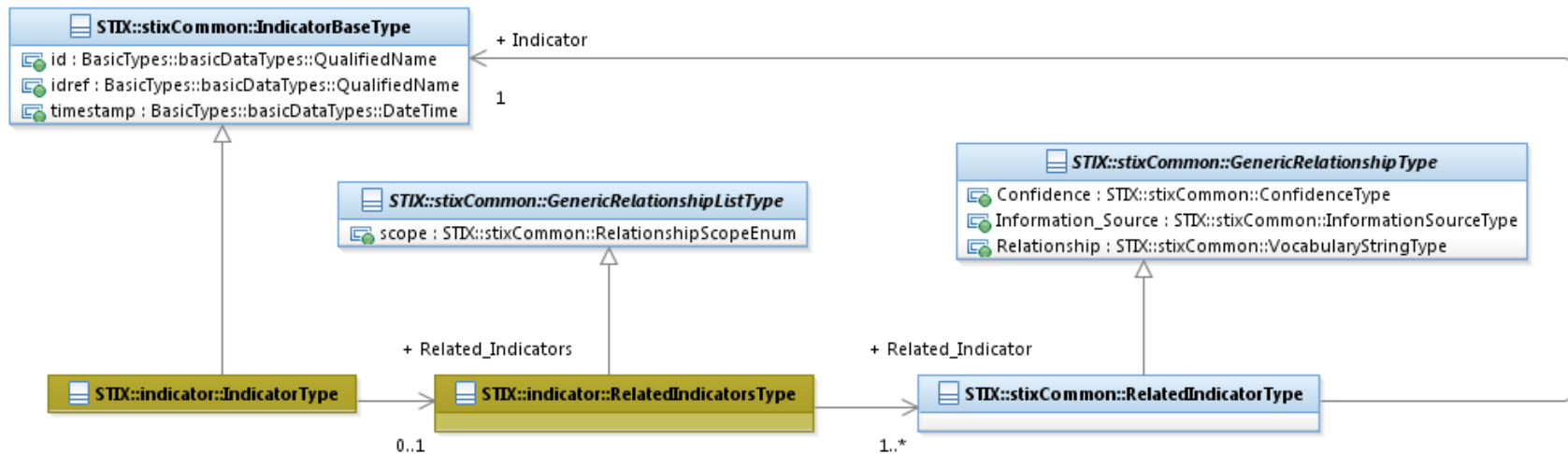


Figure 3-6. UML diagram of the RelatedIndicatorsType class

The property table given in Table 3-13 corresponds to the UML diagram given in Figure 3-6.

Table 3-13. Properties of the RelatedIndicatorsType class

Name	Type	Multiplicity	Description
Related_Indicator	stixCommon:RelatedIndicatorType	1..*	The <code>Related_Indicator</code> property specifies another Indicator associated with this Indicator and characterizes the relationship between the Indicators by capturing information such as the level of confidence that the Indicators are related, the source of the relationship information, and type of the relationship. A relationship between Indicators may represent assertions of general associativity or different versions of the same Indicator.

3.8 RelatedCampaignReferencesType Class

The `RelatedCampaignReferencesType` class specifies a set of one or more Campaigns for which the Indicator may be relevant. It extends the `GenericRelationshipListType` superclass defined in the STIX Common data model, which specifies the scope (whether the elements of the set are related individually or as a group).

The ability to define a direct relationship between an Indicator and a Campaign is provided for ease of use and may not always be appropriate. The more formal and always appropriate method for capturing such a relationship is to first define a relationship between the Indicator and a TTP and then to define a relationship from the TTP to the Campaign (which uses the TTP).

The UML diagram corresponding to the `RelatedCampaignReferencesType` class is shown in Figure 3-7.

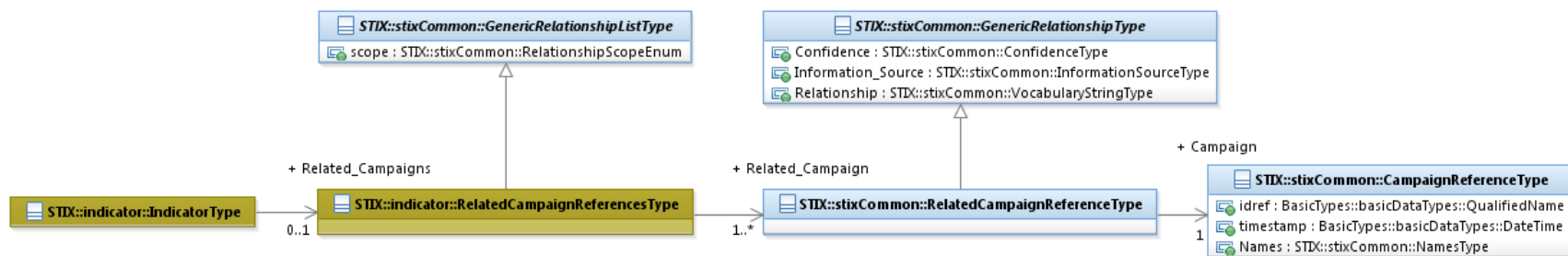


Figure 3-7. UML diagram of the `RelatedCampaignReferencesType` class

Note that the STIX Common `RelatedCampaignReferencesType` class is used instead of the STIX Common `RelatedCampaignType` class. As a result, an Indicator only references a Campaign by its Name property and/or its identifier; the Campaign is not directly embedded in the Indicator. (Compare this to how Related Indicators are captured; see Section 3.7.)

Table 3-14 shows the property of the `RelatedCampaignReferencesType` specialization and is associated with the UML diagram given in Figure 3-7.

Table 3-14. Properties of the `RelatedCampaignReferenceType` class

Name	Type	Multiplicity	Description
Related_Campaign	<code>stixCommon:</code> <code>RelatedCampaignReferenceType</code>	1..*	The <code>Related_Campaign</code> property specifies a Campaign for which the Indicator may be relevant. Unlike most other relationships that are defined in STIX, the underlying <code>RelatedCampaignReferenceType</code> class does not allow a Campaign to be embedded; an already-defined Campaign MUST be specified by its <code>Name</code> property and/or a reference to its identifier. The relationship between the Indicator and the Campaign is characterized by capturing information such as the level of confidence that the Indicator and the Campaign are related, the source of the relationship information, and the type of the relationship.

References

References made in this document are listed below.

- [CybOX_{COR}] CybOX™ Core Specification (*not yet available*).
- [RFC2119] RFC 2119 – Key words for use in RFCs to Indicate Requirement Levels
<http://www.ietf.org/rfc/rfc2119.txt>
- [STIX] STIX™ Web Site
<https://stix.mitre.org>
- [STIX-SPECS] STIX™ Project Github Site
<http://github.com/STIXProject/specifications>
- [STIX_{CAM}] STIX™ 1.2 Campaign Specification (v1.2)
<http://stix.mitre.org/about/documents/XXXX.pdf>
- [STIX_{COM}] STIX™ 1.2 Common Specification (v1.2)
<http://stix.mitre.org/about/documents/XXXX.pdf>
- [STIX_{COA}] STIX™ 1.2 Course of Action Specification (v1.2)
<http://stix.mitre.org/about/documents/XXXX.pdf>
- [STIX_{EXT}] STIX™ 1.2 Default Extensions Specification
<http://stix.mitre.org/about/documents/XXXX.pdf>
- [STIX_O] STIX™ 1.2 Specification Overview
<http://stix.mitre.org/about/documents/XXXX.pdf>
- [STIX_{TTP}] STIX™ 1.2 TTP Specification (v1.2)
<http://stix.mitre.org/about/documents/XXXX.pdf>
- [TOU] Terms of Use
<http://stix.mitre.org/about/termsfuse.html>