

THE MITRE CORPORATION

STIX™ 1.1.1

DEFAULT EXTENSIONS SPECIFICATION

MAY 27, 2015

The Structured Threat Information eXpression (STIX™) framework defines eight 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 default extensions for the STIX framework.

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 the STIX Default Extensions Specification. Please send any 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	3
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	Extending STIX	7
3	STIX Default Extension Data Models	10
3.1	Addresses: STIX-CIQ Address Data Model v1.1.1	10
3.1.1	CIQAddress3.0InstanceType Class	10
3.2	Attack Patterns: STIX-CAPEC Data Model v1.0.1	11
3.2.1	CAPEC2.7InstanceType Class	11
3.3	Identities: STIX-CIQ Identity Data Model v1.1.1	12
3.3.1	CIQIdentity3.0InstanceType Class	12
3.3.2	STIXCIQIdentity3.0Type Class	13
3.4	Malware: STIX-MAEC Data Model v1.0.1	14
3.4.1	MAEC4.1InstanceType Class	14
3.5	Marking Data Models	15
3.5.1	Simple Data Marking Data Model v1.1.1	16
3.5.1.1	SimpleMarkingStructureType Class	16
3.5.2	Terms of Use Data Marking Data Model v1.0.1	16
3.5.2.1	TermsOfUseMarkingStructureType Class	17
3.5.3	Traffic Light Protocol Data Marking Data Model v1.1.1	17
3.5.3.1	TLPMarkingStructureType Class	17
3.5.3.2	TLPColorEnum Enumeration	18
3.6	Generic Structured COA Data Model v1.1.1	18
3.6.1	GenericStructuredCOAType	19
3.7	Test Mechanism Data Models	20
3.7.1	Generic Test Mechanism Data Model v1.1.1	21
3.7.1.1	GenericTestMechanismType Class	21
3.7.2	OpenIOC Test Mechanism Data Model v1.1.1	23
3.7.2.1	OpenIOC2010TestMechanismType Class	23
3.7.3	OVAL Test Mechanism Data Model v1.1.1	24
3.7.3.1	OVAL5.10TestMechanismType Class	24

3.7.4	Snort Test Mechanism Data Model v1.1.1	25
3.7.4.1	SnortTestMechanismType Class	26
3.7.5	Yara Test Mechanism Data Model v1.1.1	27
3.7.5.1	YaraTestMechanismType Class	27
3.8	Vulnerabilities: STIX-CVRF Data Model v1.1.1	28
3.8.1	CVRF1.1InstanceType Class	29
References		30

1 Introduction

The Structured Threat Information eXpression (STIX™) framework defines eight top-level component data models: Observable³, Indicator, Incident, TTP, ExploitTarget, CourseOfAction, Campaign, and ThreatActor. In addition, it defines various default extension data models for leveraging other data models and standards specifications that have been defined outside of STIX. This specification document does not define those non-STIX data models, but discusses the extension points available in STIX and defines a corresponding set of default extension data models. The default extensions currently available are those related to addresses, identity, malware, attack patterns, test mechanisms, exploits, data markings and courses of action. Each default extension data model is versioned separately. This specification covers default extensions that are relevant to STIX v1.1.1.

In Section 1.1 we discuss STIX specification documents, and we provide document conventions in Section 1.2. In Section 2, we give background information to help the reader better understand the specification details that are provided later in the document and provide a high level list of the default extensions. We present the specification details for the default extension data models 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 [STIX₀], which in addition to the eight top-level 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. [STIX₀] 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 default 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 [STIX₀] for details). This STIX Extensions 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 [STIX₀].

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

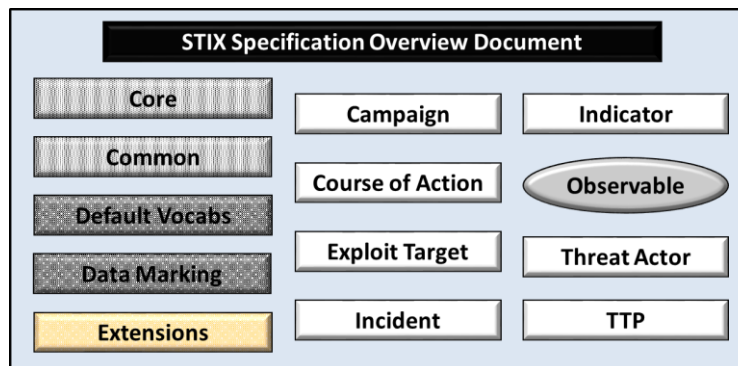


Figure 1-1. STIX Language v1.1.1 specification documents

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 [STIX₀].

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, with single quotes'* font is used for noting explicit values for STIX Language properties.

Example: *'STIX Default Package Intent Vocabulary'*

1.2.3 UML Package References

Each STIX data model is captured in a different UML package (e.g., Core package, Campaign package, etc.). To refer to a particular class of a specific package, we use the format `package_prefix:class`, where `package_prefix` corresponds to the appropriate UML package. Each default extension data model is in its own package, therefore, to avoid confusion, we will use a fully qualified UML name for all UML references.

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. 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. In particular, we will always capture properties of more simple 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

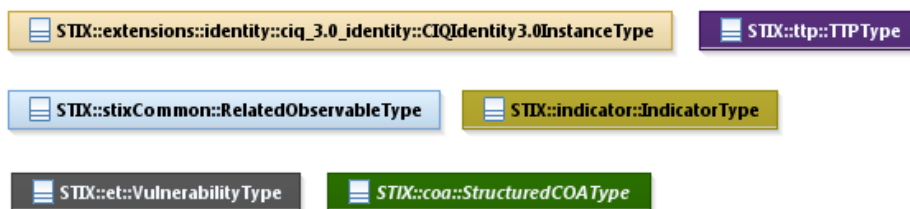
Diagram icons are used in a UML diagram to indicate whether a shape is a class, enumeration or data type, and decorative icons are used to indicate whether an element is an attribute of a class or an enumeration literal. In addition, two different arrow styles indicate either a directed association relationship (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 this specification are illustrated in **Error! Reference source not found.**

**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 all classes.

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>	<p>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>
	<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 that is necessary to fully understand the extension data models specification details given in Section 0.

2.1 Extending STIX

In any UML model, an arbitrary class can usually be extended, but in general, extending a data model is antithetical to the concept behind a standardized data model used for sharing information. However, many of the concepts that need to be represented in STIX already are defined in established data models outside of STIX. Additionally, there are concepts where one single consensus data model may not exist but rather different ones exist for different contexts. To support the inclusion of those data models into STIX, a number of extension point classes have been identified. The number of extension points is not fixed, and others might be added in the future, if the need arises.

This document defines the default extension data models and their associated classes, which are specializations of the extension point classes. These default extension classes compose the currently available extension data models. The extensions defined in this document are defaults – others can be used. Note that some extension point classes do not have a corresponding default data model externally defined. Additionally, some extension point classes have no corresponding extension class defined in the STIX extension data models.

Table 2-1 shows the relationship between the extension point classes and the default extension classes.

Table 2-1. Extension points classes

Extension Point Class	Abstract?	Contains Properties?	Externally Defined Data Model?	Default Extension Classes
stixCommon: ActivityType	Y	Y	N	<i>none</i>
coa: StructuredCOAType	Y	Y	N	genericStructuredCOA: GenericStructuredCOAType
stixCommon: AbstractAddressType	Y	N	Y	stix-ciaddress: CIQAddress3.0InstanceType
indicator: TestMechanismType	Y	Y	Y	genericTM: GenericTestMechanismType stix-openioc: OpenIOC2010TestMechanismType

STIX™ 1.1.1: DEFAULT EXTENSIONS SPECIFICATION

				stix-oval:OVAL5.10TestMechanismType snortTM:SnortTestMechanismType yaraTM:YaraTestMechanismType
ttp:AttackPatternType	N	Y	Y	stix-capec:CAPEC2.7InstanceType
stixCommon:IdentityType	N	Y	Y	stix_ciqidentity: CIQIdentity3.0InstanceType
ttp:MalwareInstanceType	N	Y	Y	stix-maec:MAEC4.1InstanceType
marking:MarkingType	N	Y	Y	simpleMarking: SimpleMarkingStructureType TOUMarking: TermsOfUseMarkingStructureType tlpMarking:TLPMarkingStructureType
et:VulnerabilityType	N	Y	Y	stix-cvrf:CVRF1.1InstanceType
ttp:ExploitType	N	Y	N	<i>none</i>

From a UML package perspective, Table 2-2 shows the relationships between the various UML packages that exist to support a modular approach to creating extensions to the STIX data models. Each extension data model has its own package. The primary class of each of those packages specializes an extension point class that is contained in one of the main packages of the STIX model. The extension classes generally have one or more properties to support the connection between the STIX and the externally defined data models. Those properties are either associated with a class from the corresponding external package or contain a text specification in the native format of the external data model. In the former case, we provide the name of the external defined package in the table. If a text specification is used, then the package name is not applicable, because there is no formally defined UML package.

Table 2-2. Packages Associated with the Extensions Data Models

Extension Class Package	Extension Point Class Package	External Data Model Package
stix-ciqaddress	stixCommon	a
stix-ciqidentity	stixCommon	ciq
genericStructuredCOA	coa	<i>n/a</i>
genericTM	indicator	<i>n/a</i>
stix-openioc	indicator	ioc
stix-oval	indicator	oval-def; oval-var

STIX™ 1.1.1: DEFAULT EXTENSIONS SPECIFICATION

snortTM	indicator	<i>n/a</i>
yaraTM	indicator	<i>n/a</i>
stix-capec	ttp	capec
stix-maec	ttp	maec
simpleMarking	marking	<i>n/a</i>
TOUMarking	marking	<i>n/a</i>
tlpMarking	marking	tlp_marking
stix-cvrf	et	cvrf

3 STIX Default Extension Data Models

Each STIX extension data model contains a primary class, called the extension class that extends a class in one or more other STIX data models. In sections 3.1 through 3.8 we define the classes of each extension data model, listed in alphabetical order (except for the cases when one class defines a property of another class, in which case the higher level class is defined first). Externally defined data models are contained in a UML package named `external`. The names of the packages used in this document for the external data models are often aliases (e.g., the package `a` is an alias for `urn:oasis:names:tc:ciq:xal` from the external data model).

3.1 Addresses: STIX-CIQ Address Data Model v1.1.1

The default extension class for expressing geographic address information in STIX v1.1.1 is the `CIQAddress3.0InstanceType` class defined below. The underlying data model being referenced is the structured characterization of addresses of the OASIS Customer Information Quailty (CIQ) Specification as defined in [OASIS-CIQ].

3.1.1 CIQAddress3.0InstanceType Class

The `CIQAddress3.0InstanceType` class is defined in STIX v1.1.1 as the default subclass to extend the STIX Common `AddressAbstractType` abstract superclass and belongs to the `stix-ciqaddress` package. As shown in Figure 3-1, the `CIQAddress3.0InstanceType` class imports and leverages version 3.0 of the OASIS CIQ-PIL schema for structured characterization of addresses.

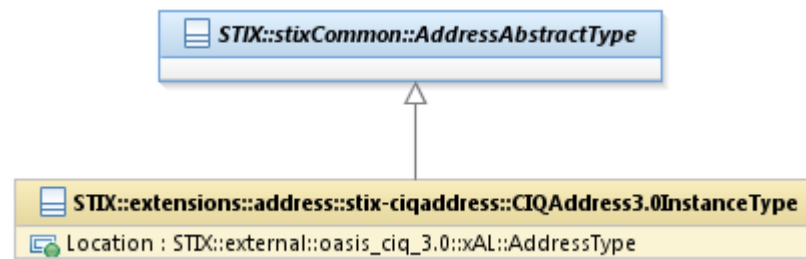


Figure 3-1. UML diagram of the `CIQAddress3.0InstanceType` class

The property table for the `CIQAddress3.0InstanceType` class is given in Table 3-1.

Table 3-1. Properties of the `CIQAddress3.0InstanceType` class

Name	Type	Multiplicity	Description
Location	<code>a:AddressType</code>	1	The <code>Location</code> property specifies a potentially long set of address-related information including address type (e.g., business, rural), country, administrative area, locality, postcode, and geolocation.

3.2 Attack Patterns: STIX-CAPEC Data Model v1.0.1

The default extension class for representing attack patterns in STIX v1.1.1 is the `CAPEC2.7InstanceType` class defined below. The underlying data model being referenced is the Common Attack Pattern Enumeration and Classification (CAPEC) specification as defined in [CAPEC].

3.2.1 CAPEC2.7InstanceType Class

The `CAPEC2.7InstanceType` class provides an extension to the STIX TTP `AttackPatternType` class and belongs to the `stix-capec` package. It imports and leverages the CAPEC 2.7 schema for a structured characterization of attack patterns.

The UML diagram for the `CAPEC2.7InstanceType` class is shown in Figure 3-2.

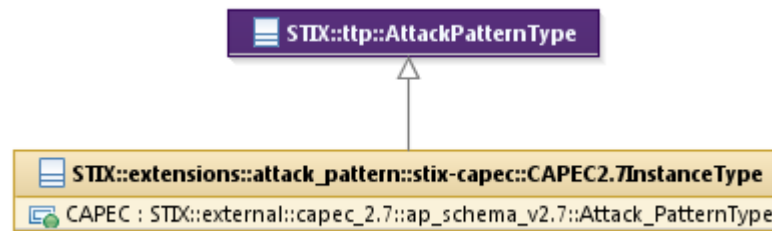


Figure 3-2. UML diagram of the `CAPEC2.7InstanceType` class

The property table for the `CAPEC2.7InstanceType` class is given in Table 3-2.

Table 3-2. Properties of the `CAPEC2.7InstanceType` class

Name	Type	Multiplicity	Description
CAPEC	<code>capec:Attack_PatternType</code>	1	The CAPEC property specifies the structured specification of an attack pattern utilizing the CAPEC schema.

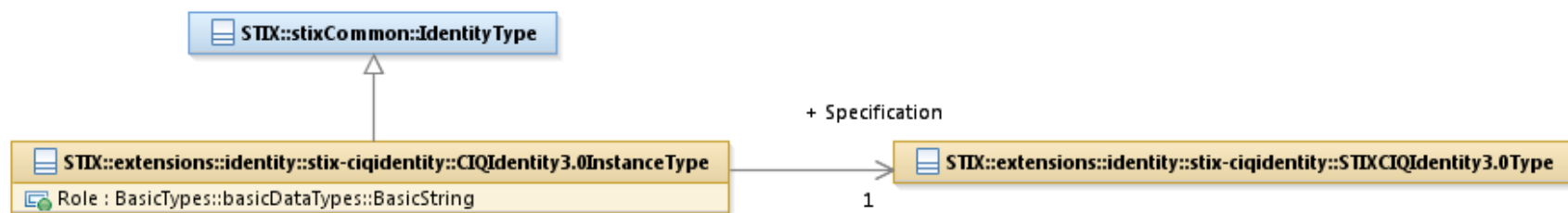
3.3 Identities: STIX-CIQ Identity Data Model v1.1.1

The default extension class for expressing identity information in STIX v1.1.1 is the `CIQIdentity3.0InstanceType` class defined below. The underlying data model being referenced is the structured characterization of identity information of the OASIS Customer Information Quality (CIQ) Specification as defined in [OASIS-CIQ].

3.3.1 `CIQIdentity3.0InstanceType` Class

The `CIQIdentity3.0InstanceType` class extends the `stixCommon:IdentityType` class and belongs to the `stix-ciqidentity` package. It imports and leverages version 3.0 of the OASIS CIQ-PIL schema for structured characterization of identity information (e.g. threat actors, victims, and sources of information).

The UML diagram for the `CIQIdentity3.0InstanceType` class is shown in Figure 3-3.

**Figure 3-3.** UML diagram of the `CIQIdentity3.0InstanceType` class

The properties of the `CIQIdentity3.0InstanceType` class are listed in Table 3-3.

Table 3-3. Properties of the `CIQIdentity3.0InstanceType` class

Name	Type	Multiplicity	Description
Specification	<code>stix-ciqidentity:STIXCIQIdentity3.0Type</code>	1	The <code>Specification</code> property specifies the structured characterization of an identity utilizing the CIQ-PIL schema.
Role	<code>basicDateTypes:BasicString</code>	0..*	The <code>Role</code> property specifies a relevant role played by the entity with the corresponding identity.

3.3.2 STIXCIQIdentity3.0Type Class

The `STIXCIQIdentityType` class provides a restriction and minor extension⁴ of the imported OASIS CIQ-PIL Party concept for use in characterizing STIX identities and belongs to the `stix-ciqidentity` package. Unlike the other extension classes described in Section 3, the `CIQIdentity3.0InstanceType` class does not contain a property that encompasses the whole data model for party identities from the OASIS CIQ-PIL definition. Instead, it selects certain properties from that data model, which are aggregated in the `STIXCIQIdentityType` class.

The allowed properties are restricted to the following subset:

Accounts	Favourites	Occupations	Relationships
Addresses	FreeTextLines	OrganisationInfo	Revenues
BirthInfo	Habits	PartyName	Stocks
ContactNumbers	Hobbies	PartyType	Vehicles
CountriesOfResidence	Identifiers	PersonInfo	Visas
Documents	Languages	PhysicalInfo	
ElectronicAddressIdentifiers	Memberships	Preferences	
Events	Nationalities	Qualifications	

⁴ The property `LanguageCode`.

3.4 Malware: STIX-MAEC Data Model v1.0.1

The default extension class for representing malware in STIX v1.1.1 is the `MAEC4.1InstanceType` class defined below. The underlying data model being referenced is the structured characterization of malware as defined in the Malware Attribute Enumeration and Characterization (MAEC) specification as defined in [MAEC].

3.4.1 MAEC4.1InstanceType Class

The `MAEC4.1InstanceType` class provides an extension to the STIX TTP `MalwareInstanceType` class and belongs to the `stix-maec` package. It imports and leverages the MAEC 4.1 schema for structured characterization of malware.

The UML diagram for the `MAEC4.1InstanceType` class is shown in Figure 3-4.

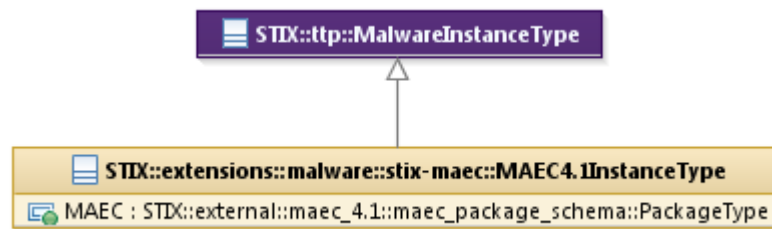


Figure 3-4. UML diagram of the `MAEC4.1InstanceType` class

The properties of the `MAEC4.1InstanceType` class are listed in Table 3-4.

Table 3-4. Properties of the `MAEC4.1InstanceType` class

Name	Type	Multiplicity	Description
MAEC	<code>maec:PackageType</code>	1	The MAEC property specifies the structured characterization of malware instances using the MAEC Package data model.

3.5 Marking Data Models

The default classes for providing data marking information in STIX v1.1.1 are defined below. Each of the classes extends the `MarkingStructureType` class from the Marking data model [STIX_{MAR}] as illustrated in Figure 3-5.

Three default extensions are provided for the `MarkingStructureType` class, which correspond to different popular data marking schemes:

- Simple data marking
- Terms of Use data marking
- Traffic Light Protocol (TLP) data marking

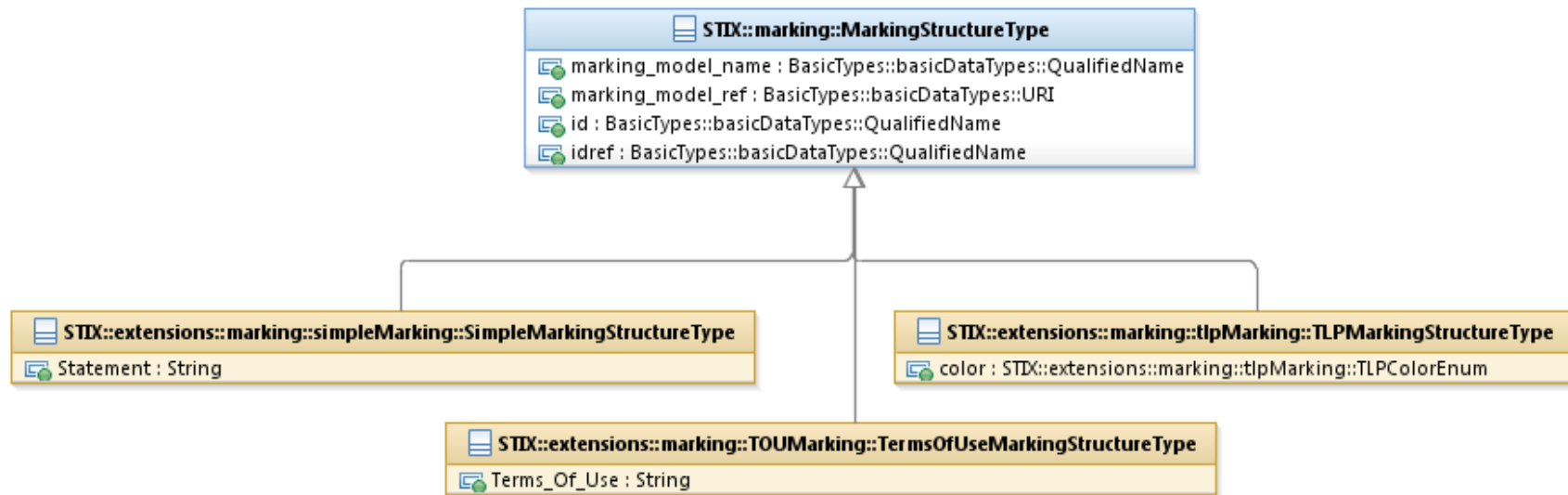


Figure 3-5. UML diagram of extensions to the Data Marking `MarkingStructureType` class

3.5.1 Simple Data Marking Data Model v1.1.1

The default extension class for representing simple data markings in STIX v1.1.1 is the `SimpleMarkingStructureType` class defined below.

3.5.1.1 SimpleMarkingStructureType Class

The `SimpleMarkingStructureType` class extends the `Data Marking MarkingStructureType` class and is a basic implementation of the Data Marking data model that allows for a string statement to be associated with the data being marked. It is contained in the `simpleMarking` package. One example is the application of a copyright statement to some data set.

Nodes may be marked by multiple Simple Marking statements. When this occurs, all of the multiple Simple Marking statements apply. It is up to the organization adding an additional Simple Marking statement to ensure that the addition does not conflict with any previously applied Simple Marking statements.

The property table for the `SimpleMarkingStructureType` class is given in

Table 3-5.

Table 3-5. Properties of the `SimpleMarkingStructureType` class

Name	Type	Multiplicity	Description
Statement	<code>basicDateTypes:BasicString</code>	1	The <code>Statement</code> property specifies the statement to apply to the structure for which the marking is to be applied.

3.5.2 Terms of Use Data Marking Data Model v1.0.1

The default extension class for representing Terms of Use Markings in STIX v1.1.1 is the `TermsOfUseMarkingStructureType` class defined below.

3.5.2.1 TermsOfUseMarkingStructureType Class

The `TermsOfUseMarkingStructureType` class extends the `Data Marking MarkingStructureType` class and is a basic implementation of the Data Marking data model that allows for a string statement describing the Terms of Use to be associated with the data being marked. It is contained in the `TOUMarking` package.

Nodes may be marked by multiple Terms of Use Marking statements. When this occurs, all of the multiple Terms of Use Marking statements apply. It is up to the organization adding an additional Terms of Use Marking statement to ensure that the addition does not conflict with any previously applied Terms of Use Marking statements.

The property table for the `SimpleMarkingStructureType` class is given in Table 3-6.

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

Name	Type	Multiplicity	Description
Terms_Of_Use	<code>basicDateTypes:BasicString</code>	1	The <code>Terms_Of_Use</code> property specifies the terms of use statement to apply to the structure for which the marking is to be applied.

3.5.3 Traffic Light Protocol Data Marking Data Model v1.1.1

The default extension class for representing Traffic Light Protocol Markings in STIX v1.1.1 is the `TLPMarkingStructureType` class defined below.

3.5.3.1 TLPMarkingStructureType Class

The `TLPMarkingStructureType` class extends the `Data Marking MarkingStructureType` class and is a basic implementation of the Data Marking data model that allows for a Traffic Light Protocol designation [TLP] to be attached to an identified structure. It is contained in the `tlpMarking` package.

STIX objects may be marked by multiple TLP Marking statements. When this occurs, the object should be considered marked at the most restrictive TLP Marking of all TLP Markings that were applied to it. For example, if an object is marked both GREEN and AMBER, the object should be considered AMBER.

The property table for the `TLPMarkingStructureType` class is given in Table 3-7.

Table 3-7. Properties of the `TLPMarkingStructureType` class

Name	Type	Multiplicity	Description
color	<code>tlp_marking:TLPColorEnum</code>	0..1	The <code>color</code> property specifies the TLP color designation of the marked structure.

3.5.3.2 TLPColorEnum Enumeration

The `TLPColorEnum` enumeration is an inventory of all possible Traffic Light Protocol color designations of the marked structure. It is contained in the `tlpMarking` package.

Table 3-8. Values of the `TLPColorEnum` enumeration

Enumeration Literal	Description
RED	The <code>RED</code> value specifies that information cannot be effectively acted upon by additional parties, and could lead to impacts on a party's privacy, reputation, or operations if misused.
AMBER	The <code>AMBER</code> value specifies that information requires support to be effectively acted upon, but carries risks to privacy, reputation, or operations if shared outside of the organizations involved.
GREEN	The <code>GREEN</code> value specifies that information is useful for the awareness of all participating organizations as well as with peers within the broader community or sector.
WHITE	The <code>WHITE</code> value specifies that information carries minimal or no foreseeable risk of misuse, in accordance with applicable rules and procedures for public release.

3.6 Generic Structured COA Data Model v1.1.1

The default class for expressing Course of Action (COA) information in STIX v1.1.1 is the `GenericStructuredCOAType` class defined below.

The `coa:StructuredCOAType` abstract class is intended to be extended to allow for the expression of a variety of structured COA types. The STIX default extension uses a generic structured COA to allow for the passing of proprietary or externally defined structured courses of action in their native format.

This implementation is captured in the Generic Structured COA extension, which provides the `GenericStructuredCOAType` class.

3.6.1 GenericStructuredCOAType

The `GenericStructuredCOAType` class extends the `Course of Action StructuredCOAType` class and belongs to the `genericStructuredCOA` package. It specifies an instantial extension from the abstract `StructuredCOAType` class intended to support the generic inclusion of any COA content. The UML diagram corresponding to the `GenericStructuredCOAType` class is shown in Figure 3-6.

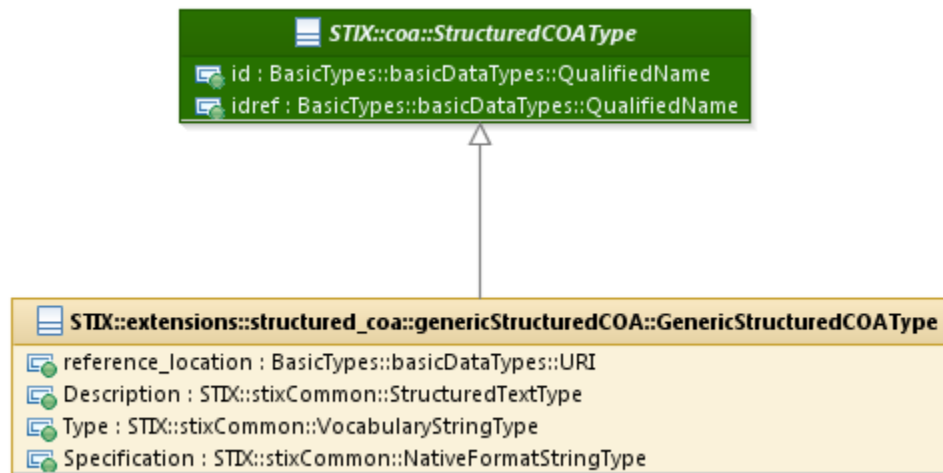


Figure 3-6. UML diagram of `GenericStructuredCOAType` class

The property table for the `GenericStructuredCOAType` class is given in Table 3-9.

Table 3-9. Properties of the `GenericStructuredCOAType` class

Name	Type	Multiplicity	Description
reference_location	<code>basicDataTypes:URI</code>	0..1	The <code>reference_location</code> property specifies a reference URI for the location of the data model definition used for the generic structured COA.
Description	<code>stixCommon:StructuredTextType</code>	1	The <code>Description</code> property captures a textual description of the generic Course of Action. Any length is permitted. Optional formatting is supported via the <code>structuring_format</code> property of the <code>StructuredTextType</code> class.
Type	<code>stixCommon:VocabularyStringType</code>	1	The <code>Type</code> property specifies the type of generic structured COA. No default vocabulary class for use in the property has been defined for STIX 1.1.1.
Specification	<code>stixCommon:NativeFormatStringType</code>	1	The <code>Specification</code> property specifies any Course of Action specification in its native format. The specification should be encoded so that it is compliant with the chosen structured course of action formalism, however this is not a requirement of the STIX specification.

3.7 Test Mechanism Data Models

The default classes for providing test mechanism information in STIX v1.1.1 are defined below. Each of the classes extend the `IndicatorTestMechanismType` class as illustrated in Figure 3-7.

Five default extensions are provided for the `indicator:TestMechanismType` abstract class, which correspond to different popular indicator test mechanisms:

- Generic Test Mechanism
- OpenIOC test mechanism
- OVAL test mechanism
- Snort test mechanism
- YARA test mechanism

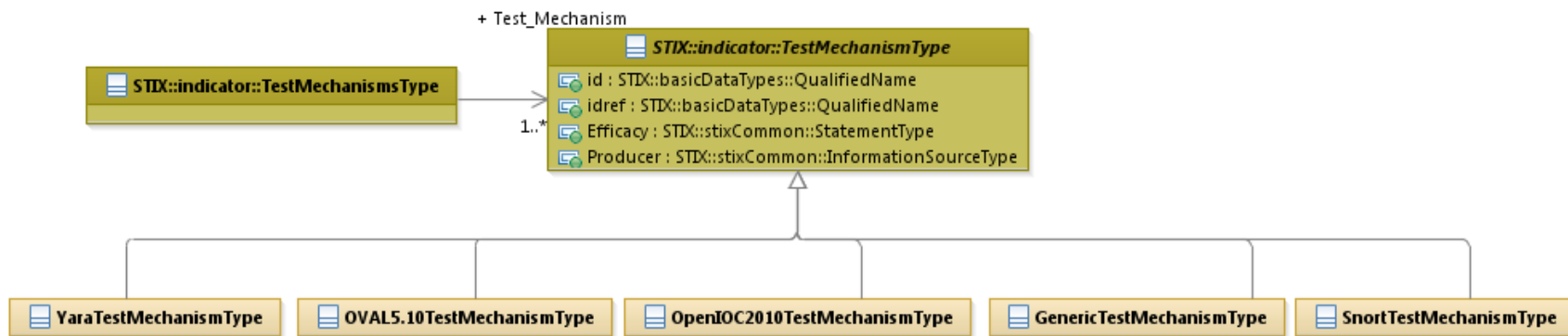


Figure 3-7. UML diagram of extensions to the `indicator:TestMechanismType` class

3.7.1 Generic Test Mechanism Data Model v1.1.1

The default extension class for representing generic test mechanisms in STIX v1.1.1 is the `GenericTestMechanismType` class defined below.

3.7.1.1 GenericTestMechanismType Class

The `GenericTestMechanismType` class enables any generic pattern or expression to be leveraged as a test mechanism in an Indicator. It is contained in the `genericTM` package.

The UML diagram corresponding to the `GenericTestMechanismType` class is shown in Figure 3-8.

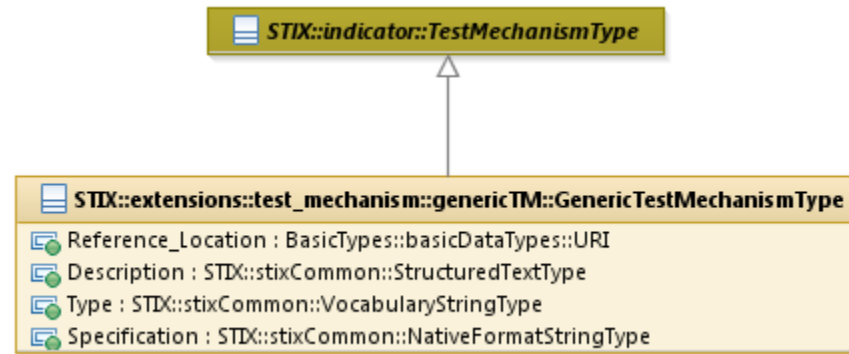


Figure 3-8. UML diagram of the `GenericTestMechanismType` class

The properties of the `GenericTestMechanismType` class specialization are listed in Table 3-10.

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

Name	Type	Multiplicity	Description
reference_location	<code>basicDataTypes:URI</code>	0..1	The <code>reference_location</code> property specifies a reference URI for the location of the data model definition used for the generic test mechanism.
Description	<code>stixCommon:StructuredTextType</code>	0..1	The <code>Description</code> property captures a textual description of the generic test mechanism.
Type	<code>stixCommon:VocabularyStringType</code>	0..1	The <code>Type</code> property specifies the type of the generic test mechanism. No default vocabulary has been defined for STIX v1.1.1.
Specification	<code>stixCommon:NativeFormatString</code>	0..1	The <code>Specification</code> property specifies a test mechanism specification in its native format. The specification should be

			encoded so that it is compliant with the chosen test mechanism formalism, however this is not a requirement of the STIX specification.
--	--	--	--

3.7.2 OpenIOC Test Mechanism Data Model v1.1.1

The default extension class for representing OpenIOC test mechanisms in STIX v1.1.1 is the `OpenIOC2010TestMechanismType` class defined below. The underlying data model being referenced is OpenIOC – An Open Framework for Sharing Threat Intelligence [OpenIOC].

3.7.2.1 OpenIOC2010TestMechanismType Class

The `OpenIOC2010TestMechanismType` class enables OpenIOC indicators of compromise, as defined in the 2010 Open IOC data model, to be leveraged as test mechanisms of an Indicator. The class is a specialization of the abstract `TestMechanismType` superclass defined in the Indicator specification document [STIX_{IND}]. It is contained in the `stix-openioc` package.

The UML diagram corresponding to the `OpenIOC2010TestMechanismType` class is shown in Figure 3-9.

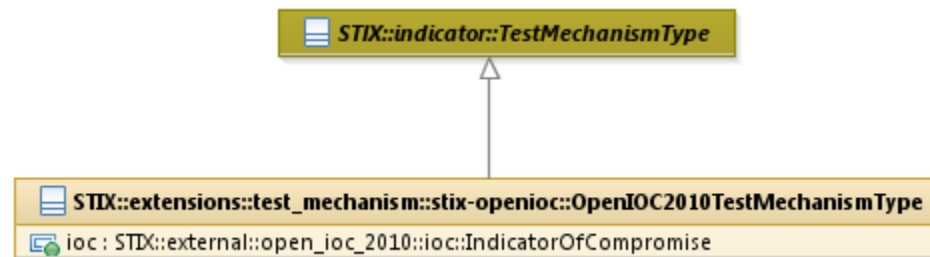


Figure 3-9. UML diagram for `OpenIOC2010TestMechanismType` class

The properties of the `OpenIOC2010TestMechanismType` class specialization is listed in Table 3-11.

Table 3-11. Properties of the `OpenIOC2010TestMechanismType` class

Name	Type	Multiplicity	Description
ioc	<code>ioc:IndicatorOfCompromise</code>	1	The <code>ioc</code> property specifies the structured specification of an OpenIOC test mechanism, which will typically be semantically equivalent to the Observables captured in the Indicator. An Indicator of Compromise (IOC) instance captures information such as a textual description of the indicator, keywords associated with the indicator, and author information, as well as the actual indicator definition pattern.

3.7.3 OVAL Test Mechanism Data Model v1.1.1

The default extension class for representing OVAL test mechanisms in STIX v1.1.1 is the `OVAL5.10TestMechanismType` class defined below. The underlying data model being referenced is OVAL - Open Vulnerability and Assessment Language [OVAL].

3.7.3.1 OVAL5.10TestMechanismType Class

The `OVAL5.10TestMechanismType` class enables OVAL definitions and variables, as defined in the OVAL 5.10 data model, to be leveraged as test mechanisms of an Indicator. The class is a specialization of the abstract `TestMechanismType` superclass defined in the Indicator specification document [STIX_{IND}]. It is contained in the `stix-oval` package.

The UML diagram corresponding to the `OpenIOC2010TestMechanismType` class is shown in Figure 3-10.

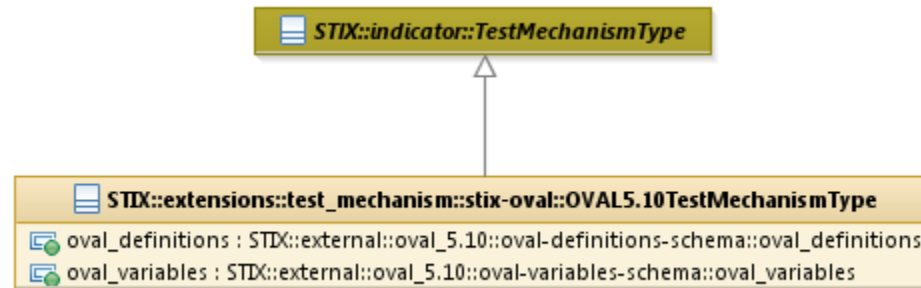


Figure 3-10. UML diagram of `OVAL5.10TestMechanismType` class

The properties of the `OVAL5.10TestMechanismType` class specialization are listed in Table 3-12.

Table 3-12. Properties of the `OVAL5.10TestMechanismType` class

Name	Type	Multiplicity	Description
oval_definitions	<code>oval:DefinitionsType</code>	1	The <code>oval_definitions</code> property specifies the structured specification of the OVAL test mechanism. When including OVAL definition documents it is expected that at least one valid OVAL definition is included.
oval_variables	<code>oval:VariablesType</code>	0..1	The <code>oval_variables</code> property specifies a valid OVAL Variables document and SHOULD only be used to supply external variable values needed by this OVAL Test Mechanism's OVAL definitions.

3.7.4 Snort Test Mechanism Data Model v1.1.1

The default extension class for representing Snort test mechanisms in STIX v1.1.1 is the `SnortTestMechanismType` class defined below. The underlying data model being referenced is described in more detail in [SNORT].

3.7.4.1 SnortTestMechanismType Class

The `SnortTestMechanismType` class enables a Snort signature be leveraged as a test mechanism in an Indicator. The class is a specialization of the abstract `TestMechanismType` superclass defined in the Indicator specification document [STIX_{IND}]. It is contained in the `snortTM` package.

The UML diagram corresponding to the `SnortTestMechanismType` class is shown in Figure 3-11.

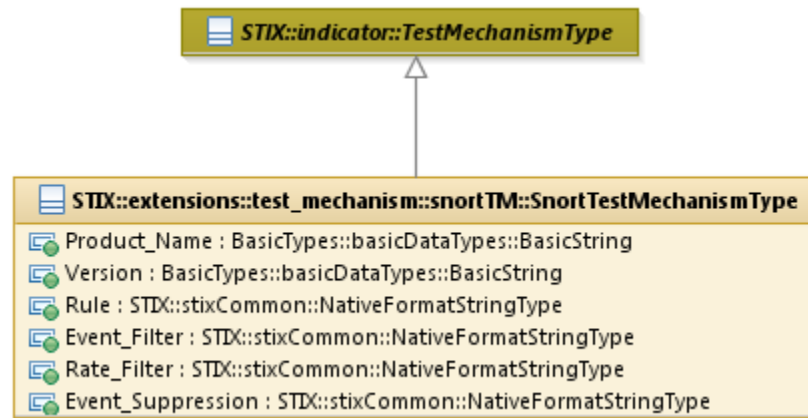


Figure 3-11. UML diagram of the `SnortTestMechanismType` class

The properties of the `SnortTestMechanismType` class specialization are listed in Table 3-13.

Table 3-13. Properties of the `SnortTestMechanismType` class

Name	Type	Multiplicity	Description
Product_Name	<code>basicDateTypes:BasicString</code>	0..1	The <code>Product_Name</code> property specifies the name of the Snort-compatible tool that the rules were written against. The Common Platform Enumeration (CPE) name of the tool SHOULD be used, if

			available. Otherwise, a simple name like "Snort", "Suricata", or "Sourcefire" MAY be used.
Version	<code>basicDateTypes:BasicString</code>	0..1	The <code>Version</code> property captures the version of the Snort or Snort-compatible tool that the Snort rules were written against.
Rule	<code>stixCommon: NativeFormatString</code>	0..*	The <code>Rule</code> property specifies a Snort rule in its native format. The specification should be encoded so that it is compliant with the Snort formalism, however this is not a requirement of the STIX specification.
Event_Filter	<code>stixCommon: NativeFormatString</code>	0..*	The <code>Event_Filter</code> property specifies a Snort event filter line in its native format. The filter should be encoded so that it is compliant with the Snort formalism, however this is not a requirement of the STIX specification.
Rate_Filter	<code>stixCommon: NativeFormatString</code>	0..*	The <code>Rate_Filter</code> property specifies a Snort rate filter line in its native format. The filter should be encoded so that it is compliant with Snort formalism, however this is not a requirement of the STIX specification.
Event_Suppression	<code>stixCommon: NativeFormatString</code>	0..*	The <code>Event_Suppression</code> property specifies a Snort event suppression line in its native format. The event suppression specification should be encoded so that it is compliant with the Snort formalism, however this is not a requirement of the STIX specification.

3.7.5 Yara Test Mechanism Data Model v1.1.1

The default extension class for representing Yara test mechanisms in STIX v1.1.1 is the `YaraTestMechanismType` class defined below. The underlying data model being referenced is described in more detail in [YARA].

3.7.5.1 YaraTestMechanismType Class

The `YaraTestMechanismType` class enables a Yara signature to be leveraged as a test mechanism in an Indicator. The class is a specialization of the abstract `TestMechanismType` superclass defined in the Indicator specification document [STIX_{IND}]. It is contained in the `yaraTM` package.

The UML diagram corresponding to the `SnortTestMechanismType` class is shown in Figure 3-12.

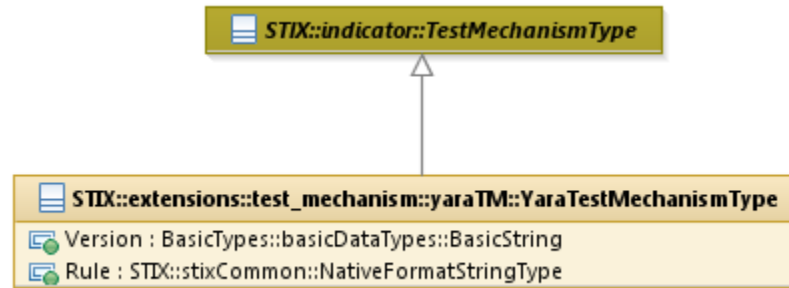


Figure 3-12. UML diagram of the `YaraTestMechanismType` class

The properties of the `YaraTestMechanismType` class specialization are listed in Table 3-14.

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

Name	Type	Multiplicity	Description
Version	<code>basicDateTypes:BasicString</code>	0..1	The <code>Version</code> property specifies the version of YARA that the rule was written against.
Rule	<code>stixCommon: NativeFormatString</code>	0..1	The <code>Rule</code> property specifies a YARA rule in its native format. The rule The specification should be encoded so that it is compliant with the chosen YARA formalism, however this is not a requirement of the STIX specification.

3.8 Vulnerabilities: STIX-CVRF Data Model v1.1.1

The default extension class for representing vulnerability details in STIX v1.1.1 is the `CVRF1.1InstanceType` class defined below. The underlying data model being referenced is ICASI's Common Vulnerability Reporting Framework (CVRF) specification as defined in [CVRF].

3.8.1 CVRF1.1InstanceType Class

The `CVRF1.1InstanceType` class provides an extension to the `Exploit Target VulnerabilityType` class and belongs to the `stix-cvrf` package.. It imports and leverages the CVRF schema for structured characterization of Vulnerabilities. This could include characterization of zero-days or other vulnerabilities that do not have a CVE or OSVDB ID (Open Sourced Vulnerability Database). The UML diagram corresponding to the `CVRF1.1InstanceType` class is shown in Figure 3-13.

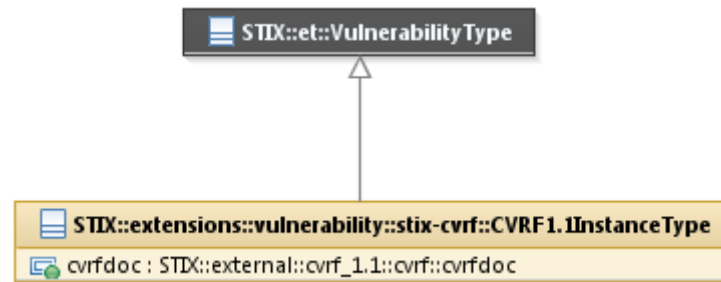


Figure 3-13. UML diagram of the `CVRF1.1InstanceType` class

The properties of the `CVRF1.1InstanceType` class specialization are listed in Table 3-15.

Table 3-15. Properties of the `CVRF1.1InstanceType` class

Name	Type	Multiplicity	Description
cvrf:cvrfdoc	<code>cvrf:cvrfdoc</code> ⁵	1	The CVRF property specifies the structured characterization of Vulnerabilities utilizing the CVRF schema.

⁵ This type is defined in [CVRF], and is provided for informative purposes only.

References

References made in this document are listed below.

- [CAPEC] CAPEC List Version 2.6
<https://capec.mitre.org/data/index.html#downloads>
- [CVRF] The Common Vulnerability Reporting Framework (CVRF)
<http://www.icasi.org/cvrf>
- [CybOXCOR] CybOX Core Specification (*not yet available*).
- [MAEC] MAEC Language — Version 4.1
<http://maec.mitre.org/language/version4.1/>
- [OASIS-CIQ] OASIS - Customer Information Quality Specifications Version 3.0
<http://docs.oasis-open.org/ciq/v3.0/prd03/specs/ciq-specs-v3-prd3.html>
- [OpenIOC] OpenIOC – An Open Framework for Sharing Threat Intelligence
<http://www.openioc.org/>
- [OVAL] OVAL - Open Vulnerability and Assessment Language Version 5.11
<https://oval.mitre.org/>
- [REL] STIX™ Indicator Model as implement in XSD
https://stix.mitre.org/language/version4.1/xxx_schema.xsd
- [RFC2119] RFC 2119 – Key words for use in RFCs to Indicate Requirement Levels
<http://www.ietf.org/rfc/rfc2119.txt>
- [SNORT] SNORT
<https://www.snort.org/>
- [STIX] STIX™ Web Site
<https://stix.mitre.org>
- [STIX-SPECS] STIX™ Project Github Site
<http://github.com/STIXProject/specifications>
- [STIXCAM] STIX™ 1.1.1 Campaign Specification (v1.1.1)
<http://stix.mitre.org/about/documents/XXXX.pdf>
- [STIXCOA] STIX™ 1.1.1 Course of Action Specification (v1.1.1)

	http://stix.mitre.org/about/documents/XXXX.pdf
[STIX _{IND}]	STIX™ 1.1.1 Indicator Specification (v2.1.1) http://stix.mitre.org/about/documents/XXXX.pdf
[STIX _{MAR}]	STIX™ 1.1.1 Marking Specification (v1.1.1) http://stix.mitre.org/about/documents/XXXX.pdf
[STIX _O]	STIX™ 1.1.1 Specification Overview http://stix.mitre.org/about/documents/XXXX.pdf
[STIX _{TTP}]	STIX™ 1.1.1 TTP Specification (v1.1.1) http://stix.mitre.org/about/documents/XXXX.pdf
[TLP]	Traffic Light Protocol (TLP) Matrix and Frequently Asked Questions http://www.us-cert.gov/tlp
[W3-CDATA]	Extensible Markup Language (XML) 1.0 (Fifth Edition) http://www.w3.org/TR/2008/REC-xml-20081126/#sec-cdata-sect
[YARA]	Yara – The Pattern Matching Swiss Knife for Malware Researchers http://plusvic.github.io/yara/