STIX™ Version 1.2.1. Part 12: Default Extensions

Committee Specification 01

05 May 2016

Specification URIs

This version:

http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part12-extensions/stix-v1.2.1-cs01-part12-extensions.docx (Authoritative)
http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part12-extensions/stix-v1.2.1-cs01-part12-extensions.html
http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part12-extensions/stix-v1.2.1-cs01-part12-extensions.pdf

Previous version:

http://docs.oasis-open.org/cti/stix/v1.2.1/csprd01/part12-extensions/stix-v1.2.1-csprd01-part12-extensions.docx (Authoritative)
http://docs.oasis-open.org/cti/stix/v1.2.1/csprd01/part12-extensions/stix-v1.2.1-csprd01-part12-extensions.html
http://docs.oasis-open.org/cti/stix/v1.2.1/csprd01/part12-extensions/stix-v1.2.1-csprd01-part12-extensions.pdf

Latest version:

http://docs.oasis-open.org/cti/stix/v1.2.1/stix-v1.2.1-part12-extensions.docx (Authoritative)
http://docs.oasis-open.org/cti/stix/v1.2.1/stix-v1.2.1-part12-extensions.html
http://docs.oasis-open.org/cti/stix/v1.2.1/stix-v1.2.1-part12-extensions.pdf

Technical Committee:

OASIS Cyber Threat Intelligence (CTI) TC

Chair:

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

Editors:

Sean Barnum (sbarnum@mitre.org), MITRE Corporation
Desiree Beck (dbeck@mitre.org), MITRE Corporation
Aharon Chernin (achernin@soltra.com), Soltra
Rich Piazza (rpiazza@mitre.org), MITRE Corporation

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

- *STIX Version 1.2.1. Part 1: Overview*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part1-overview/stix-v1.2.1-cs01-part1-overview.html
- *STIX Version 1.2.1. Part 2: Common*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part2-common/stix-v1.2.1-cs01-part2-common.html
- *STIX Version 1.2.1. Part 3: Core*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part3-core/stix-v1.2.1-cs01-part3-core.html
- *STIX Version 1.2.1. Part 4: Indicator*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part4-indicator/stix-v1.2.1-cs01-part4-indicator.html
- *STIX Version 1.2.1. Part 5: TTP*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part5-ttp/stix-v1.2.1-cs01-part5-ttp.html

- *STIX Version 1.2.1. Part 6: Incident*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part6-incident/stix-v1.2.1-cs01-part6-incident.html
- *STIX Version 1.2.1. Part 7: Threat Actor*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part7-threat-actor/stix-v1.2.1-cs01-part7-threat-actor.html
- *STIX Version 1.2.1. Part 8: Campaign*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part8-campaign/stix-v1.2.1-cs01-part8-campaign.html
- *STIX Version 1.2.1. Part 9: Course of Action*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part9-coa/stix-v1.2.1-cs01-part9-coa.html
- *STIX Version 1.2.1. Part 10: Exploit Target*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part10-exploit-target/stix-v1.2.1-cs01-part10-exploit-target.html
- *STIX Version 1.2.1. Part 11: Report*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part11-report/stix-v1.2.1-cs01-part11-report.html
- *STIX Version 1.2.1. Part 12: Default Extensions* (this document). http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part12-extensions/stix-v1.2.1-cs01-part12-extensions.html
- *STIX Version 1.2.1. Part 13: Data Marking*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part13-data-marking/stix-v1.2.1-cs01-part13-data-marking.html
- *STIX Version 1.2.1. Part 14: Vocabularies*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part14-vocabularies/stix-v1.2.1-cs01-part14-vocabularies.html
- *STIX Version 1.2.1. Part 15: UML Model*. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part15-uml-model/stix-v1.2.1-cs01-part15-uml-model.html
- UML Model Serialization: http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/uml-model/

Related work:

This specification replaces or supersedes:

- *STIX™ 1.2 Default Extensions Specification* https://github.com/STIXProject/specifications/blob/version1.2/documents/pdf%20versions/STIX_Extensions_Draft.pdf

This specification is related to:

- *CybOX™ Version 2.1.1.* Work in progress. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=cti-cybox
- *CybOX™ 2.1.* https://cyboxproject.github.io/

Abstract:

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 default extensions for the STIX framework.

Status:

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

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

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

Citation format:

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

[STIX-v1.2.1-Extensions]

STIX[TM] Version 1.2.1. Part 12: Default Extensions. Edited by Sean Barnum, Desiree Beck, Aharon Chernin, and Rich Piazza. 05 May 2016. OASIS Committee Specification 01. http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part12-extensions/stix-v1.2.1-cs01-part12-extensions.html. Latest version: http://docs.oasis-open.org/cti/stix/v1.2.1/stix-v1.2.1-part12-extensions.html.

Notices

WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR FREEDOM FROM INFRINGEMENT, ANY WARRANTY THAT THE STANDARDS OR THEIR COMPONENT PARTS WILL BE ERROR FREE, OR ANY WARRANTY THAT THE DOCUMENTATION, IF PROVIDED, WILL CONFORM TO THE STANDARDS OR THEIR COMPONENT PARTS. IN NO EVENT SHALL THE UNITED STATES GOVERNMENT OR ITS CONTRACTORS OR SUBCONTRACTORS BE LIABLE FOR ANY DAMAGES, INCLUDING, BUT NOT LIMITED TO, DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES, ARISING OUT OF, RESULTING FROM, OR IN ANY WAY CONNECTED WITH THESE STANDARDS OR THEIR COMPONENT PARTS OR ANY PROVIDED DOCUMENTATION, WHETHER OR NOT BASED UPON WARRANTY, CONTRACT, TORT, OR OTHERWISE, WHETHER OR NOT INJURY WAS SUSTAINED BY PERSONS OR PROPERTY OR OTHERWISE, AND WHETHER OR NOT LOSS WAS SUSTAINED FROM, OR AROSE OUT OF THE RESULTS OF, OR USE OF, THE STANDARDS, THEIR COMPONENT PARTS, AND ANY PROVIDED DOCUMENTATION. THE UNITED STATES GOVERNMENT DISCLAIMS ALL WARRANTIES AND LIABILITIES REGARDING THE STANDARDS OR THEIR COMPONENT PARTS ATTRIBUTABLE TO ANY THIRD PARTY, IF PRESENT IN THE STANDARDS OR THEIR COMPONENT PARTS AND DISTRIBUTES IT OR THEM "AS IS."

Table of Contents

1 Introduction

[All text is normative unless otherwise labeled]

The Structured Threat Information Expression (STIX™) framework defines nine top-level component data models: Observable[1], Indicator, Incident, TTP, ExploitTarget, CourseOfAction, Campaign, ThreatActor, and Report. 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 v1.2.1.

In Section **1.1** we discuss additional specification documents, in Section **1.2** we provide document conventions, and in Section **1.3** we provide terminology. References are given in Sections **1.4** and **1.5**. In Section **2**, we give background information to help the reader better understand the specification details that are provided later in the document. We present the specification details for the default extension data models in Section **3** and conformance information in Section **4**.

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 Version 1.2.1 Part 1: Overview* document provides a comprehensive overview of the full set of STIX data models, which in addition to the nine 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. *STIX Version 1.2.1 Part 1: Overview* 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 (vocabularies, data marking, and default extensions), and the color white indicates the component data models. The solid grey color denotes the overall STIX Language UML model. This STIX Default Extensions specification document is highlighted in its associated color (see Section **1.2.3.3**). For a list of all STIX documents and related information sources, please see *STIX Version 1.2.1 Part 1: Overview*.

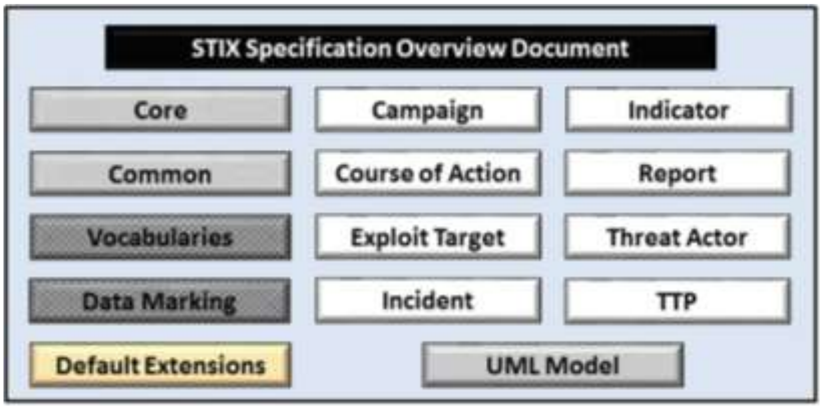


Figure 1-1. STIX[TM] Language v1.2.1 specification documents

1.2 Document Conventions

The following conventions are used in this document.

1.2.1 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 *STIX Version 1.2.1 Part 1: Overview*.

  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'*

1.2.2 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 models is in it own package, therefore, to avoid confusion, we will use a fully qualified UML names for all UML references.

1.2.3 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.

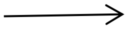
1.2.3.1 Class Properties

Generally, a class property can be shown in a UML diagram as either an attribute or an association (i.e., the distinction between attributes and associations is somewhat subjective). In order to make the size of UML diagrams is 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.

1.2.3.2 Diagram Icons and Arrow Types

Diagram icons are used in a UML diagram to indicate whether a shape is a class, enumeration or 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.

1.2.3.3 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 Default Extensions specification are illustrated via exemplars in **Figure 1-2**.

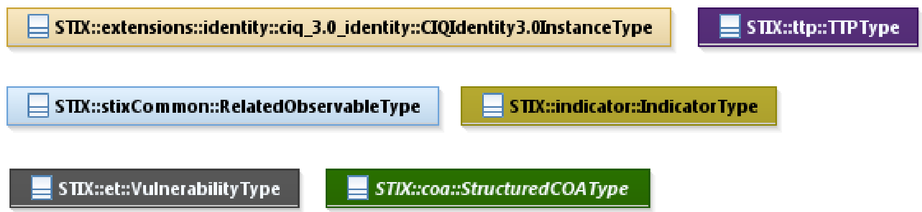


Figure 1-2. Data model color coding

1.2.4 Property Table Notation

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

1.2.5 Property and Class Descriptions

Each class and property defined in STIX is described using the format, “The X property <u>verb</u> Y.” For example, in the specification for the STIX Indicator, we write, “The `id` property <u>specifies</u> 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.
	Examples: The `Source` property characterizes the source of the sighting information. Examples of details <u>captured</u> include identitifying characteristics, time-related attributes, and a list of the tools used to collect the information. The `Description` property <u>captures</u> a textual description of the Indicator.

characterizes	Describes the distinctive nature or features of a class or property. Often used to describe classes and properties that themselves comprise one or more other properties.
	Examples: The `Confidence` property characterizes the level of confidence in the accuracy of the overall content captured in the Incident. The `ActivityType` class characterizes basic information about an activity a defender might use in response to a Campaign.
specifies	Used to clearly and precisely identify particular instances or values associated with a property. Often used for properties that are defined by a controlled vocabulary or enumeration; typically used for properties that take on only a single value.
	Example: The `version` property specifies the version identifier of the STIX Campaign data model used to capture the information associated with the Campaign.

1.3 Terminology

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

1.4 Normative References

[CAPEC] Common Attack Pattern Enumeration and Classification (CAPEC). (2014, Nov. 7). The MITRE Corporation. [Online]. Available: http://capec.mitre.org.

[CIQ] *Customer Information Quality (CIQ) Specifications Version 3.0*. Edited by Ram Kumar. 8 April 2008. OASIS Public Review Draft 03. Available: http://docs.oasis-open.org/ciq/v3.0/specs/ciq-specs-v3.html.

[CVRF] Common Vulnerabilites Reporting Framework (CVRF). (n.d.). The Industry Consortium for Advancement of Security on the Internet (ICASI). [Online]. Available: http://www.icasi.org/cvrf/. Accessed Aug. 22, 2015.

[MAEC] Malware Attribute Enumeration and Characterization (MAEC). (2015, Apr. 14). The MITRE Corporation. [Online]. Available: http://maec.mitre.org.

[OpenIOC] The OpenIOC Framework. (n.d.). Mandiant Corporation. [Online]. Available: http://openioc.org/. Accessed Aug. 23, 2015.

[OVAL] Open Vulnerability and Assessment Language (OVAL). (2015, Jul. 9). The MITRE Corporation. [Onlne]. Available: http://oval.mitre.org.

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[W3CDATA] **“**Extensible Markup Language (XML) 1.0 (Fifth Edition),” W3C Recommendation, 26 November 2008. Available: http://www.w3.org/TR/2008/REC-xml-20081126/#sec-cdata-sect

1.5 Non-Normative References

[Snort] Snort. (n.d.). Cisco. [Online]. Available: https://snort.org. Accessed Sep. 2, 2015.

[TLP] Traffic Light Protocol (TLP) Matrix and Frequently Asked Questions. (n.d.). US-CERT. [Online]. http://www.us-cert.gov/tlp. Accessed Sep. 2, 2015.

[YARA] “YARA – The pattern matching swiss knife for malware researchers.” (n.d.). [Online]. Available: http://plusvic.github.io/yara/. Accessed Sep. 2, 2015.

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 3.

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	*none*
`coa: StructuredCOAType`	Y	Y	N	`genericStructuredCOA: GenericStructuredCOAType`
`stixCommon: AbstractAddressType`	Y	N	Y	`stix-ciqaddress: CIQAddress3.0InstanceType`

`indicator: TestMechanismType`	Y	Y	Y	`genericTM:GenericTestMechanismType`
				
				`stix-openioc: OpenIOC2010TestMechanismType`
				
				`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	*none*

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 Default Extension Data Models

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

stix-capec	ttp	capec
stix-maec	ttp	maec
simpleMarking	marking	*n/a*
TOUMarking	marking	*n/a*
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 v1.2

The default extension class for expressing geographic address information in STIX v1.2.1 is the `CIQAddress3.0InstanceType` class defined below. The underlying data model being referenced is the structured characterization of addresses of the OASIS Customer Information Quaility (CIQ) Specification as defined in **[CIQ]**.

3.1.1 CIQAddress3.0InstanceType Class

The `CIQAddress3.0InstanceType` class is defined in STIX v1.2.1 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.

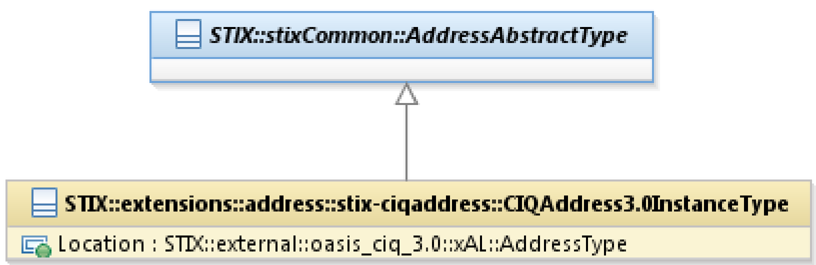


Figure 3-1. UML diagram of the `CIQAddress3.0InstanceType` class

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	a:AddressType	1	The Location 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 v1.1

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**.

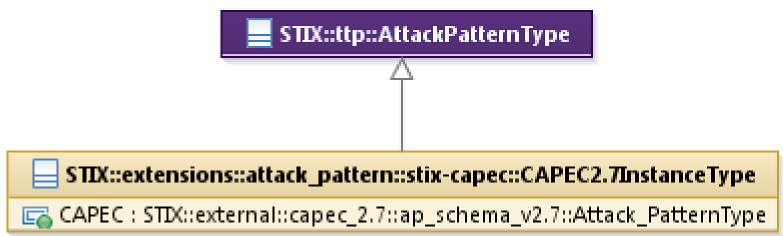


Figure 3-2. UML diagram of the CAPEC2.7InstanceType class

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	capec:Attack_PatternType	1	The CAPEC property specifies the structured specification of an attack pattern utilizing the CAPEC schema.

3.3 Identities: STIX-CIQ Identity Data Model v1.2

The default extension class for expressing identity information in STIX v1.2.1 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 Quaility (CIQ) Specification as defined in **[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**.

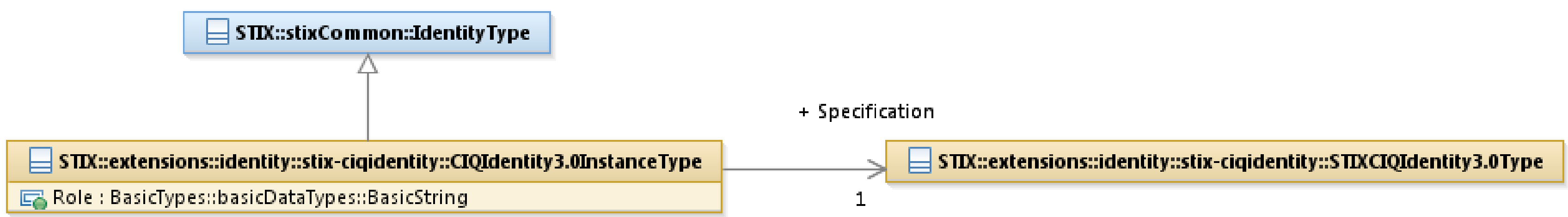


Figure 3-3. UML diagram of the `CIQIdentity3.0InstanceType` class

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	`stix-ciqidentity: STIXCIQIdentity3.0Type`	1	The `Specification` property specifies the structured characterization of an identity utilizing the CIQ-PIL schema.
Role	`basicDataTypes:BasicString`	0..*	The `Role` 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 extension[2] 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	

3.4 Malware: STIX-MAEC Data Model v1.1

The default extension class for representing malware in STIX v1.2.1 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**.

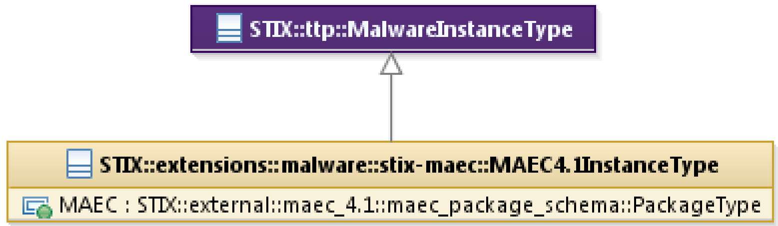


Figure 3-4. UML diagram of the `MAEC4.1InstanceType` class

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	`maec:PackageType`	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 v1.2.1 are defined below. Each of the classes extends the `MarkingStructureType` class from the Marking data model (see *STIX Version 1.2.1 Part 13: Data Marking*) 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

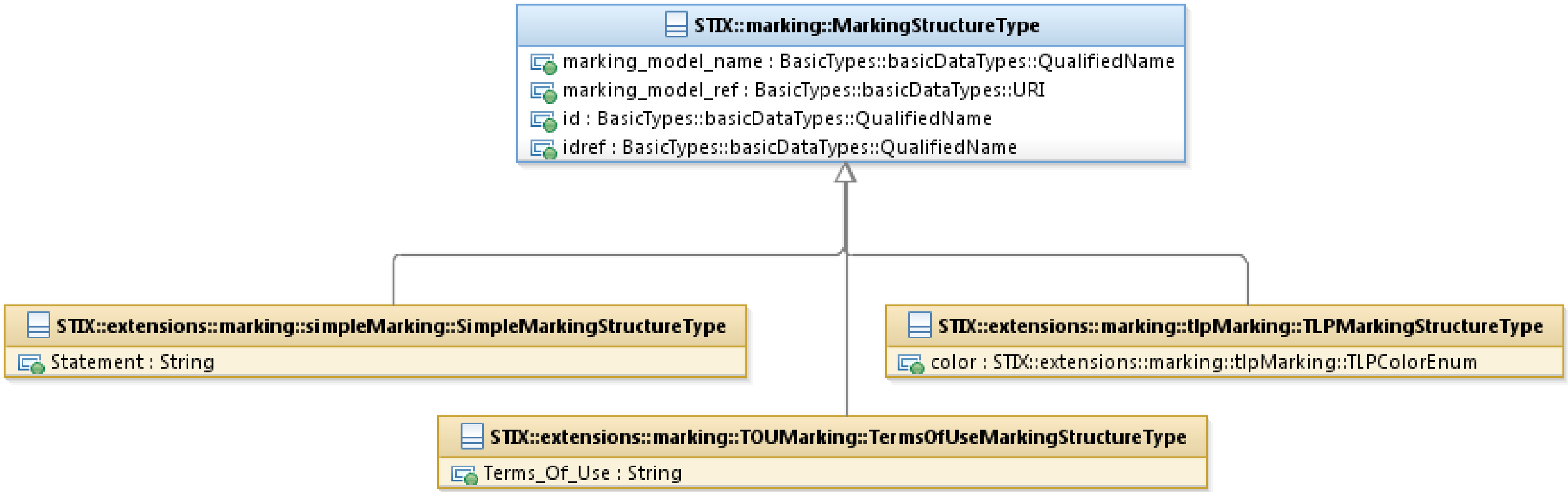


Figure 3-5. UML diagram of extensions to the Data Marking `MarkingStructureType` class

3.5.1 Simple Data Marking Data Model v1.2

The default extension class for representing simple data markings in STIX v1.2.1 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	basicDataTypes:BasicString	1	The Statement 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 v1.1

The default extension class for representing Terms of Use Markings in STIX v1.2.1 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	basicDataTypes:BasicString	1	The Terms_Of_Use 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 v1.2

The default extension class for representing Traffic Light Protocol Markings in STIX v1.2.1 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	`tlp_marking:TLPColorEnum`	0..1	The `color` 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 `RED` 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 AMBER 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 GREEN 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 WHITE 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 v1.2

The default class for expressing Course of Action (COA) information in STIX v1.2.1 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**.

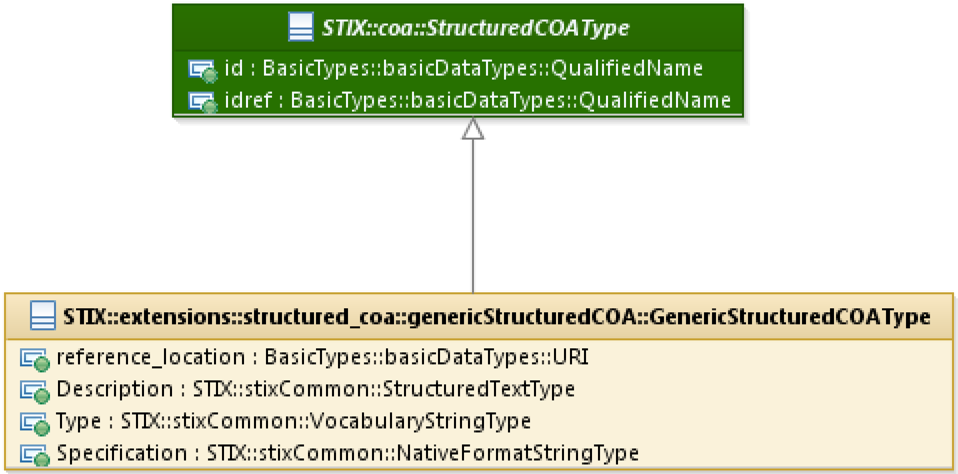


Figure 3-6. UML diagram of `GenericStructuredCOAType` class

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	`basicDataTypes:URI`	0..1	The `reference_location` property specifies a reference URI for the location of the data model definition used for the generic structured COA.
Description	`stixCommon:StructuredTextType`	0..*	The `Description` property captures a textual description of the generic Course of Action. Any length is permitted. Optional formatting is supported via the `structuring_format` property of the `StructuredTextType` class.
Type	`stixCommon: VocabularyStringType`	1	The `Type` property specifies the type of generic structured COA. No default vocabulary class for use in the property has been defined for STIX 1.2.
Specification	`stixCommon: NativeFormatStringType`	1	The `Specification` 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 couse 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 v1.2.1 are defined below. Each of the classes extend the Indicator `TestMechanismType` 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

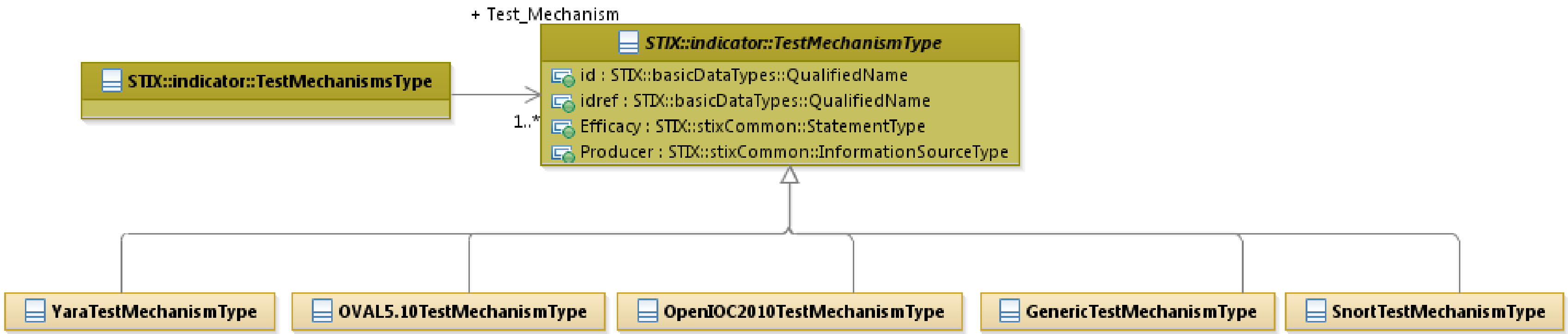


Figure 3-7. UML diagram of extensions to the `indicator:TestMechanismType` class

3.7.1 Generic Test Mechanism Data Model v1.2

The default extension class for representing generic test mechanisms in STIX v1.2.1 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**.

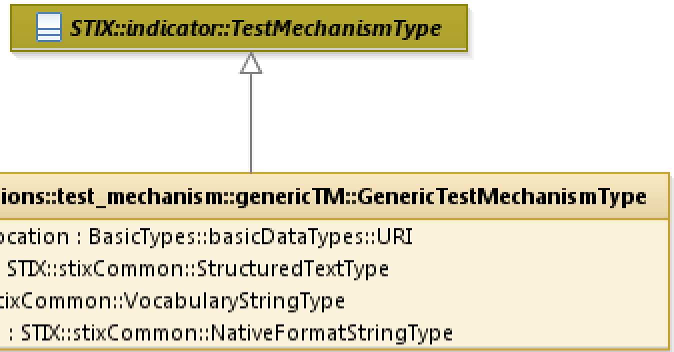


Figure 3-8. UML diagram of the `GenericTestMechanismType` class

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	`basicDataTypes:URI`	0..1	The `reference_location` property specifies a reference URI for the location of the data model definition used for the generic test mechanism.
Description	`stixCommon:StructuredTextType`	0..*	The `Description` property captures a textual description of the generic test mechanism.
Type	`stixCommon:VocabularyStringType`	0..1	The `Type` property specifies the type of the generic test mechanism. No default vocabulary has been defined for STIX v1.2.1.
Specification	`stixCommon:NativeFormatString`	0..1	The `Specification` 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 v1.2

The default extension class for representing OpenIOC test mechanisms in STIX v1.2.1 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 *STIX Version 1.2.1 Part 4: Indicator*. It is contained in the `stix-openioc` package.

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

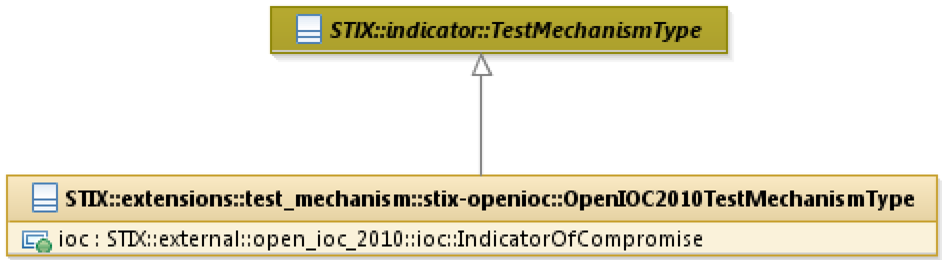


Figure 3-9. UML diagram for `OpenIOC2010TestMechanismType` class

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	`ioc:IndicatorOfCompromise`	1	The `ioc` 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 v1.2

The default extension class for representing OVAL test mechanisms in STIX v1.2.1 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 *STIX Version 1.2.1 Part 4: Indicator*. It is contained in the `stix-oval` package.

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

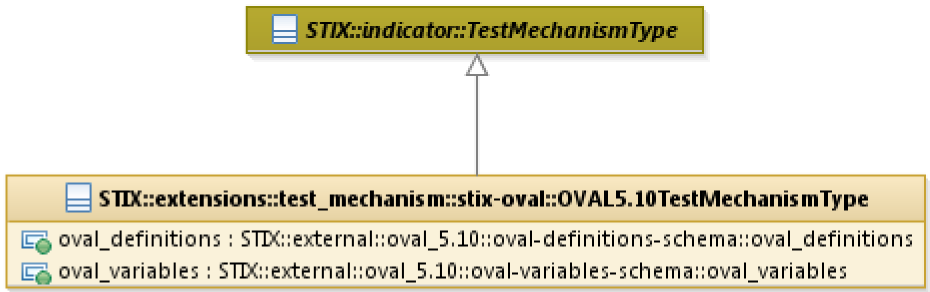


Figure 3-10. UML diagram of `OVAL5.10TestMechanismType` class

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	`oval:DefinitionsType`	1	The `oval_definitions` 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	`oval:VariablesType`	0..1	The `oval_variables` 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 v1.2

The default extension class for representing Snort test mechanisms in STIX v1.2.1 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 *STIX Version 1.2.1 Part 4: Indicator*. It is contained in the `snortTM` package.

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

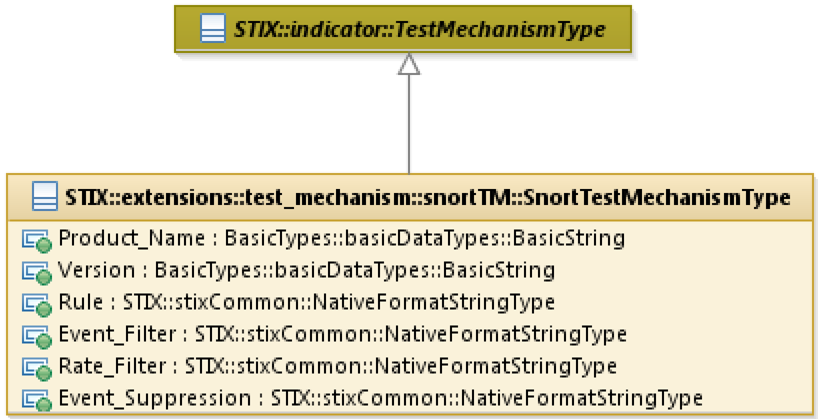


Figure 3-11. UML diagram of the `SnortTestMechanismType` class

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	`basicDataTypes:BasicString`	0..1	The `Product_Name` 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	`basicDataTypes:BasicString`	0..1	The `Version` property captures the version of the Snort or Snort-compatible tool that the Snort rules were written against.

Rule	stixCommon: NativeFormatString	0..*	The Rule 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	stixCommon: NativeFormatString	0..*	The Event_Filter 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	stixCommon: NativeFormatString	0..*	The Rate_Filter 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	stixCommon: NativeFormatString	0..*	The Event_Suppression 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 v1.2

The default extension class for representing Yara test mechanisms in STIX v1.2.1 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 *STIX Version 1.2.1 Part 4: Indicator*. It is contained in the `yaraTM` package.

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

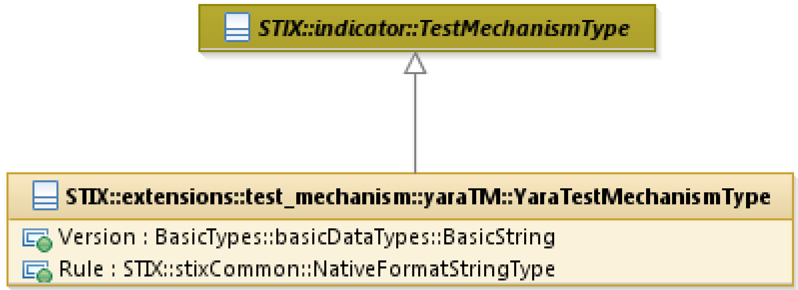


Figure 3-12. UML diagram of the `YaraTestMechanismType` class

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	`basicDataTypes:BasicString`	0..1	The `Version` property specifies the version of YARA that the rule was written against.
Rule	`stixCommon: NativeFormatString`	0..1	The `Rule` 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 v1.2

The default extension class for representing vulnerability details in STIX v1.2.1 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**.

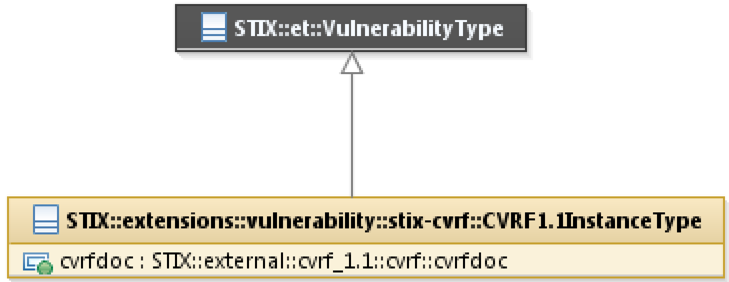


Figure 3-13. UML diagram of the CVRF1.1InstanceType class

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	`cvrf:cvrfdoc`[3]	1	The CVRF property specifies the structured characterization of Vulnerabilities utilizing the CVRF schema.

4 Conformance

Implementations have discretion over which parts (components, properties, extensions, controlled vocabularies, etc.) of STIX they implement (e.g., Indicator/Suggested_COAs).

[1] Conformant implementations must conform to all normative structural specifications of the UML model or additional normative statements within this document that apply to the portions of STIX they implement (e.g., Implementers of the entire TTP component must conform to all normative structural specifications of the UML model or additional normative statements within this document regarding the TTP component).

[2] Conformant implementations are free to ignore normative structural specifications of the UML model or additional normative statements within this document that do not apply to the portions of STIX they implement (e.g., Non-implementers of any particular properties of the TTP component are free to ignore all normative structural specifications of the UML model or additional normative statements within this document regarding those properties of the TTP component).

The conformance section of this document is intentionally broad and attempts to reiterate what already exists in this document. The STIX 1.2 Specifications, which this specification is based on, did not have a conformance section. Instead, the STIX 1.2 Specifications relied on normative statements and the non-mandatory implementation of STIX profiles. STIX 1.2.1 represents a minimal change from STIX 1.2, and in that spirit no requirements have been added, modified, or removed by this section.

Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Dean Thompson, Australia and New Zealand Banking Group (ANZ Bank)
Bret Jordan, Blue Coat Systems, Inc.
Adnan Baykal, Center for Internet Security (CIS)
Jyoti Verma, Cisco Systems
Liron Schiff, Comilion (mobile) Ltd.
Jane Ginn, Cyber Threat Intelligence Network, Inc. (CTIN)
Richard Struse, DHS Office of Cybersecurity and Communications (CS&C)
Marlon Taylor, DHS Office of Cybersecurity and Communications (CS&C)
David Eilken, Financial Services Information Sharing and Analysis Center (FS-ISAC)
Sarah Brown, Fox-IT
Ryusuke Masuoka, Fujitsu Limited
Eric Burger, Georgetown University
Jason Keirstead, IBM
Paul Martini, iboss, Inc.
Jerome Athias, Individual
Terry MacDonald, Individual
Alex Pinto, Individual
Patrick Maroney, Integrated Networking Technologies, Inc.
Wouter Bolsterlee, Intelworks BV
Joep Gommers, Intelworks BV
Sergey Polzunov, Intelworks BV
Rutger Prins, Intelworks BV
Andrei Sîrghi, Intelworks BV
Raymon van der Velde, Intelworks BV
Jonathan Baker, MITRE Corporation
Sean Barnum, MITRE Corporation
Desiree Beck, MITRE Corporation
Mark Davidson, MITRE Corporation
Ivan Kirillov, MITRE Corporation
Jon Salwen, MITRE Corporation
John Wunder, MITRE Corporation
Mike Boyle, National Security Agency
Jessica Fitzgerald-McKay, National Security Agency
Takahiro Kakumaru, NEC Corporation
John-Mark Gurney, New Context Services, Inc.
Christian Hunt, New Context Services, Inc.
Daniel Riedel, New Context Services, Inc.
Andrew Storms, New Context Services, Inc.
John Tolbert, Queralt, Inc.
Igor Baikalov, Securonix
Bernd Grobauer, Siemens AG
Jonathan Bush, Soltra
Aharon Chernin, Soltra
Trey Darley, Soltra
Paul Dion, Soltra
Ali Khan, Soltra
Natalie Suarez, Soltra
Cedric LeRoux, Splunk Inc.
Brian Luger, Splunk Inc.

Crystal Hayes, The Boeing Company
Brad Butts, U.S. Bank
Mona Magathan, U.S. Bank
Adam Cooper, United Kingdom Cabinet Office
Mike McLellan, United Kingdom Cabinet Office
Chris O'Brien, United Kingdom Cabinet Office
Julian White, United Kingdom Cabinet Office
Anthony Rutkowski, Yaana Technologies, LLC

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

Appendix B. Revision History

Revision	Date	Editor	Changes Made
wd01	21 August 2015	Sean Barnum Desiree Beck Aharon Chernin Rich Piazza	Initial transfer to OASIS template

Notes

[1] The CybOX Observable data model is actually defined in the CybOX Language, not in STIX.

[2] The property `LanguageCode`.

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