THE MITRE CORPORATION

STIX™ 1.1.1
SPECIFICATION OVERVIEW

MAY 1, 2015

The Structured Threat Information eXpression (STIX™) is a collaborative, community-driven effort to define and develop a framework for expressing cyber threat information to enable cyber threat information sharing and cyber threat analysis. The STIX framework comprises a collection of extensible component specifications along with an overarching core specification and supporting specifications. This document serves as an overview of those specifications and defines how they are used within the broader STIX framework.

Acknowledgements

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

Trademark Information

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 1.1.1 Specification Overview document. Please send comments, questions, or suggestions to stix@mitre.org.[2]

[1] For detailed information see [TOU].

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

Table of Contents

1 Introduction

The objective of the Structured Threat Information eXpression (STIX™) effort is to specify, characterize, and capture cyber threat information. STIX addresses a full range of cyber threat use cases – including threat analysis, capture and specification of indicators, management of response activities, and information sharing – to improve consistency, efficiency, interoperability, and overall situational awareness.

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 individual data models that compose the full STIX UML model, which in addition to the eight top-level component data models (Observable[3], Indicator, Incident, TTP, ExploitTarget, CourseOfAction, Campaign, and ThreatActor), includes a core data model, a common data model, a default extension data model, a data marking data model, and a set of default controlled vocabularies.

As illustrated in Figure 1-1, this STIX specification overview document (shown in yellow) serves as a unifying document for the full set of STIX specification documents. As such, it discusses the modularity of STIX (Section 2), outlines general STIX data model conventions that is necessary as background information to fully understand the the set of STIX specification documents (Section 3), and summarizes the relationship of STIX to other languages (Section 4).

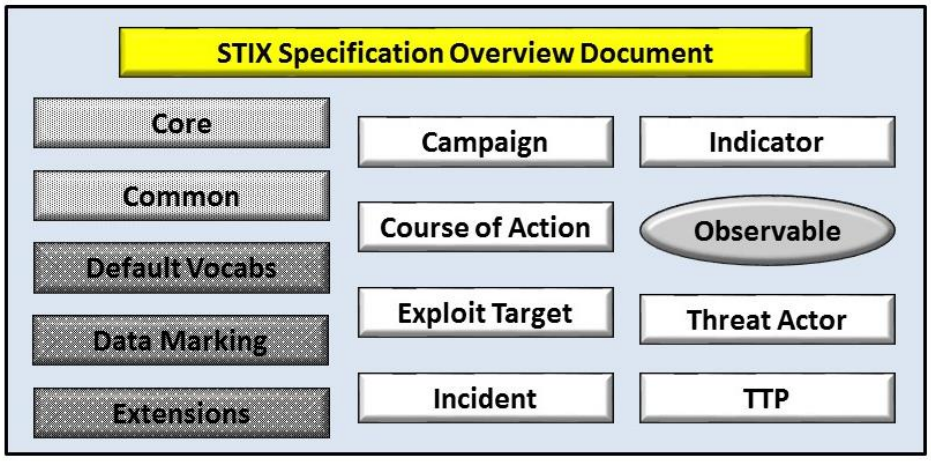

Figure 1-1. STIX Language v1.1.1 documents

Regarding Figure 1-1, 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

[3] The CybOX Observable data model is actually defined in the CybOX Language, not in STIX; but it is included in the list because it is referenced often from STIX.

Observable component data model is shown as an oval shape to indicate that it is defined as a CybOX specification (see [$STIX_O$] for details).

For completeness in terms of describing the document overview, note that we provide document conventions in Section 1.1 and references are provided in the final section.

1.1 Document Conventions

The following conventions are used in this document.

1.1.1 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].

1.1.2 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 Section 2.3.

  Examples: Indicator, Course of Action, Threat Actor

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

  Examples: `RelatedIndicatorsType`, `stixCommon:StatementType`

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

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

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

1.1.3 UML Package References

Each STIX data model is captured in a different UML package (e.g., Core package, Campaign package, etc.) where the packages together compose the full STIX UML model. To refer to a particular class of a specific package, we use the format `package_prefix:class`, where `package_prefix` corresponds to the appropriate UML package. Table 3-1 contains a list of the packages used by the STIX data models, along with the associated prefix notation, a description, and an example.

1.1.4 UML Diagrams

This overview document 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 this or the other specification documents. Typically, diagrams are included where the visualization of its relationships between classes is useful for illustration purposes. 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.

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

1.1.4.1 Class Properties

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

1.1.4.2 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 on page 4.

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.

1.1.4.3 Color Coding

The shapes of the UML diagrams are color coded to indicate the data model associated with a class. The colors used in the collection of specification documents are illustrated in Figure 1-2.

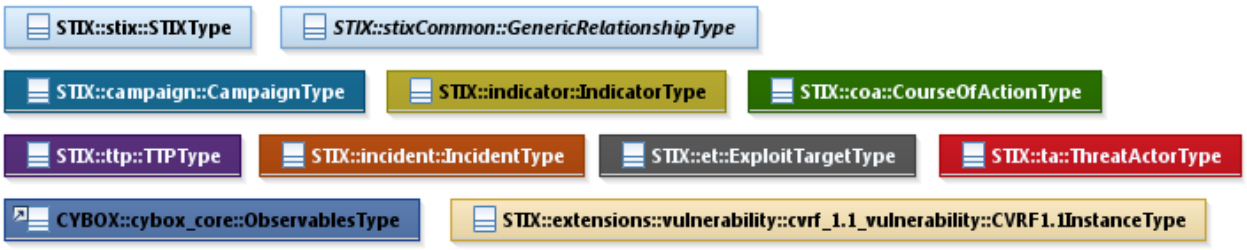

Figure 1-2. Data model color coding

2 Language Modularity

The data models of the STIX language were developed in a modular fashion to facilitate flexibility. As shown in Figure 2-1, the STIX core and common data models (see Sections 2.1 and 2.2) provide the overarching framework and common characteristics to support eight component data models (see Section 2.3), a cross-cutting data marking data model (see Section 2.4), and a set of default controlled vocabularies (see Section 2.6). Furthermore, the extensibility of the STIX design enables the use of external data models as appropriate (see Section 2.5).

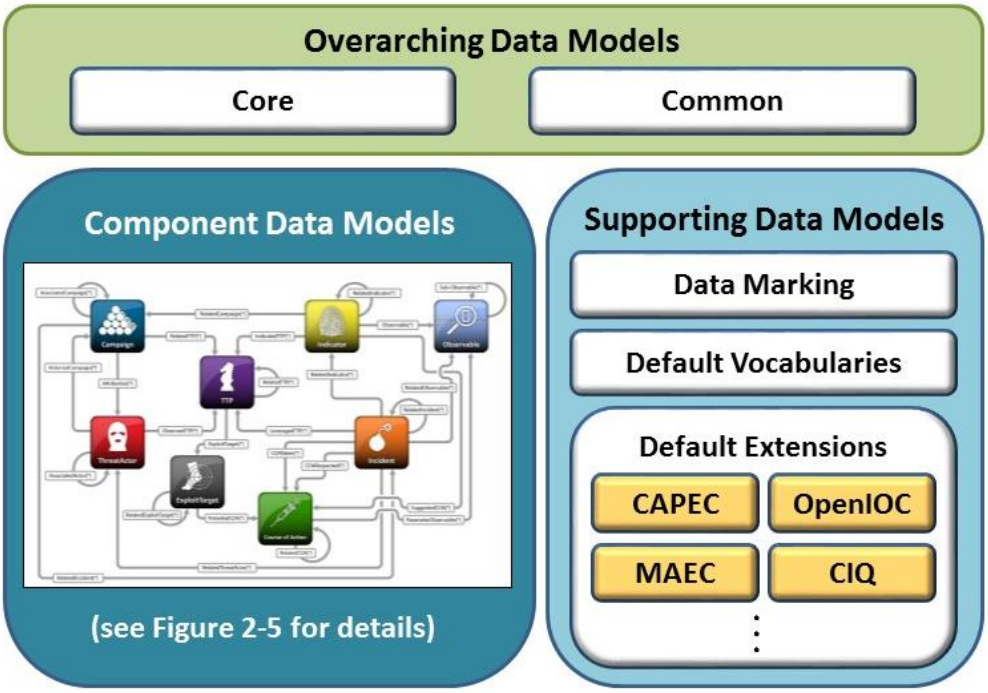

Figure 2-1. The STIX architecture

Each data model depicted in Figure 2-1 is described in a subsection below.

2.1 Core Data Model

The STIX Core data model defines the STIX Package (not to be confused with a UML package), which corresponds to the primary structure for bundling of information characterized in STIX. A STIX Package is used to bundle the various objects of the other STIX data models and has two main parts: a set of instantial content conformant to any of the eight top-level components (STIX Package content) and a STIX header (provides context for the content). Please see [$STIX_{COR}$] for complete information on the STIX Core data model.

2.2 Common Data Model

The STIX Common data model defines object classes that are shared across the various STIX data models. At a high level, the STIX Common data model provides base classes, relationship-oriented classes, content aggregation classes, and shared classes. Please see [$STIX_{COM}$] for complete information on the STIX Common data model.

2.3 Component Data Models

Individual component data models define objects specific to each top-level STIX component construct: Observable (defined in the CybOX Core data model); Indicator; Incident; Tactics, Techniques, and Procedures (TTPs); Exploit Target; Course of Action; Campaign; and Threat Actor. These data models each provide the capability to fully express information about their targeted conceptual area. In the STIX framework, they are all optional and may be used separately or in concert, as appropriate, using whichever components and architectural relationships that are relevant for a given use case.

The architecture diagram shown in Figure 2-2 on page 7 illustrates the interrelationships of the component data models that are found in a STIX Package. To overly simplify a relationship depicted between two components, we might say that a directed arrow from component “A” to component “B” indicates that either component “B” is an optional property of component “A” or that the *relationship* between components “A” and “B” is an optional property of component “A.” For example, the arrow going from Indicator to TTP denotes that an Indicator may specify a set of one or more relevant TTPs indicated by the Indicator (the TTPs are optional properties of the Indicator). To further illustrate, the arrow going from Indicator to Campaign denotes that the Indicator may specify a set of one or more relationships to a Campaign (the relationships are optional properties of the Indicator).

The bracketed asterisk on each of the arrow labels implies that such a property or relationship MAY exist zero to many times. Note that the interrelationships are not bidirectional. A further discussion of the STIX architecture as it relates to components is given in the STIX white paper [$STIX_{W}$].

In the subsections below, a brief description is given for each component data model as well as a reference to the data model’s individual specification document.

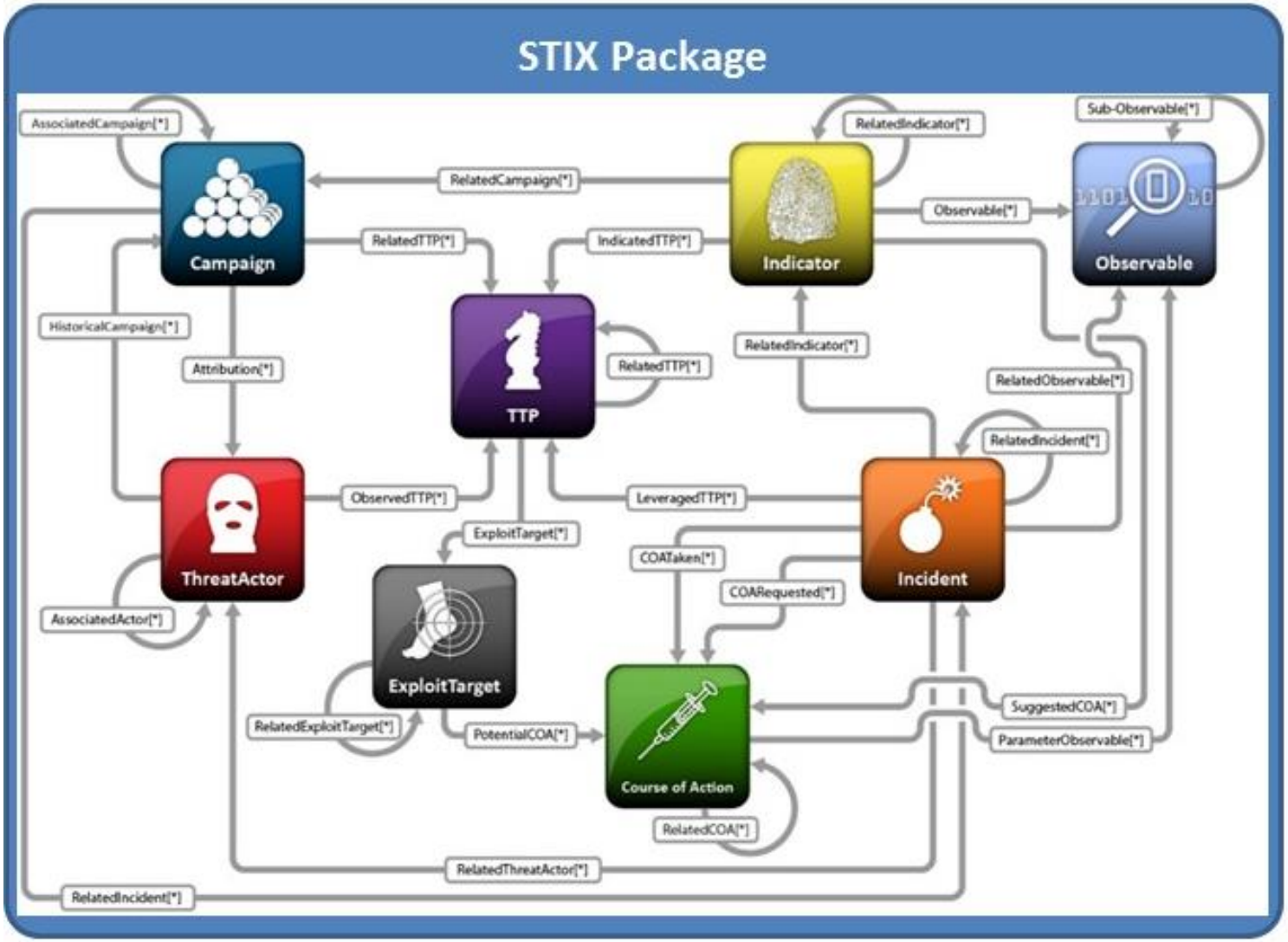

Figure 2-2. A STIX Package encompasses the STIX individual component data models

2.3.1 Observable

A STIX Observable (as defined with the CybOX Language[4]) represents stateful properties or measurable events pertinent to the operation of computers and networks. Implicit in this is a practical need for descriptive capability of two forms of observables: "observable instances" and "observable patterns." Observable instances represent actual specific observations that took place in the cyber domain. The property details of this observation are specific and unambiguous. Observable patterns represent conditions for a potential observation that may occur in the future or may have already occurred and exists in a body of observable instances. These conditions may be anything from very specific concrete patterns that would match very specific observable instances to more abstract generalized patterns that have the potential to match against a broad range of potential observable instances. Please see Section 4.4 for details on the relationship between CybOX and STIX.

2.3.2 Indicator

A STIX Indicator conveys specific Observable patterns combined with contextual information intended to represent artifacts and/or behaviors of interest within a cyber security context. Please see [$STIX_{IND}$] for complete information on the STIX Indicator data model.

2.3.3 Incident

An STIX Incident corresponds to sets of related security events affecting an organization, along with information discovered or decided during an incident response investigation. Please see [$STIX_{INC}$] for complete information on the STIX Incident data model.

2.3.4 Tactics, Techniques and Procedures (TTP)

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

2.3.5 Campaign

A STIX Campaign represents a set of TTPs, Incidents, or Threat Actors that together express a common intent or desired effect. Please see [$STIX_{CAM}$] for complete information on the STIX Campaign data model.

2.3.6 Threat Actor

A STIX Threat Actor is a characterization of a malicious actor (or adversary) representing a cyber attack threat including presumed intent and historically observed behavior. Please see [$STIX_{TA}$] for complete information on the STIX Threat Actor data model.

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

2.3.7 Exploit Target

A STIX Exploit Target conveys information about a vulnerability, weakness, or misconfiguration in software, systems, networks, or configurations that may be targeted for exploitation by an adversary. Please see [$STIX_{ET}$] for complete information on the STIX Exploit Target data model.

2.3.8 Course of Action (COA)

A STIX Course of Action (COA) is used to convey information about courses of action that may be taken either in response to an attack or as a preventative measure prior to an attack. Please see [$STIX_{COA}$] for complete information on the STIX Course of Action data model.

2.4 Data Marking Data Model

The STIX data marking data model enables flexible specification of data markings (e.g., handling restrictions) on any STIX content using any number of default or custom-defined data marking models. Please see [$STIX_{DM}$] for complete information on the STIX Data Marking data model.

2.5 Default Extensions Data Model

A primary design principle of STIX is to avoid duplicating data models that already exist for capturing cyber threat information. Therefore, in addition to the direct import of classes defined in CybOX (see Section 4.4), STIX leverages a number of other structured languages and identifiers through the use of default extensions.

More precisely, the STIX Default Extensions data model provides loose-coupling mechanisms and default extensions for leveraging constituent data models – such as Malware Attribute Enumeration and Characterization [MAEC], Common Vulnerabilities and Exposures [CVE], Common Weakness Enumeration [CWE], and Common Platform Enumeration [CPE] – to capture such information as specific vulnerabilities, weaknesses, and platforms targeted for exploitation by a malware instance.

High level summary information is given in Section 4. Please see [$STIX_{EXT}$] for complete information on the STIX Default Extensions data model.

2.6 Default Vocabularies

For some properties captured in STIX, a content creator may choose to constrain the set of possible values by referencing an externally-defined vocabulary or by leveraging a default vocabulary class defined within STIX. Alternatively, the content creator may use an arbitrary value without specifying any vocabulary. Please see [$STIX_{VOC}$] for more information about the default vocabularies defined in STIX.

2.7 Basic Data Types

The Basic Data Types data model defines UML data types used in STIX and CybOX. As stated in the UML 2.4.1 specification, UML data types are similar to UML classes, but also different:

> "A data type is a special kind of classifier, similar to a class. It differs from a class in that instances of a data type are identified only by their value. All copies of an instance of a data type and any instances of that data type with the same value are considered to be equal instances. Instances of a data type that have attributes (i.e., is a structured data type) are considered to be equal if the structure is the same and the values of the corresponding attributes are equal. If a data type has attributes, then instances of that data type will contain attribute values matching the attributes."

Although four of the requisite primitive data types (`Boolean`, `Integer`, `String`, `UnlimitedNatural`) are defined in UML, the need for a broader set in STIX drove the decision to define a complete set of basic data types in a separate, stand-alone UML package (the Basic Data Types data model). We explicitly define the data types in the Basic Data Types data model in Sections 2.7.1 and 2.7.2.

2.7.1 Common Basic Data Types

Common data types, such as string and integer, are defined in the Basic DataTypes data model and adhere to the following definitions shown in Table 2-1. These definitions are based on the specification of the corresponding data types found in [W3-DT].

Table 2-1. Common basic data types

Data Type	Definition
`BasicString`	The `BasicString` data type is a sequence of characters. Currently, characters are defined using the UTF-8 character encoding. The number of characters allowed is finite, but unbounded.
`Boolean`	The `Boolean` data type is defined with two possible literals: '*true*' and '*false*'.
`Float`	The `Float` data type is a sequence of decimal digits, with perhaps an interveaning decimal point, ".". The number of digits on either side of the decimal point is finite, but unbounded. Often used to express currency amounts.
`Integer`	The `Integer` data type is a sequence of decimal digits, with perhaps a leading minus sign "-". The number of decimal digits allowed is finite, but unbounded.

NonNegativeInteger	The NonNegativeInteger data type is a restriction on the Integer data type such that the leading minus sign is not allowed.
PositiveInteger	The PositiveInteger data type is a restriction on the NonNegativeInteger data type that disallows zero (0).

2.7.2 Specializations of the BasicString Data Type

The data types in Table 2-2 correspond to strings that have sematics associated with them. Because of this, they usually are restricted to a certain pattern, defined via a regular expression, and/or more formally defined in a standardization document.

Table 2-2. Specializations of the BasicString Data Type

Data Type	Definition
CAPEC_ID	The CAPEC_ID data type is a restriction on the BasicString data type, such that it adheres to the regular expression "CAPEC-\d+". The CAPEC_ID values should correspond to those defined at http://capec.mitre.org.
CCE_ID	The CCE_ID data type is a restriction on the BasicString data type such that it adheres to the regular expression "CCE-\d+\d". The CCE_ID values should correspond to those defined at http://cce.mitre.org.
CVE_ID	The CVE_ID data type is a restriction on the BasicString data type such that it adheres to the regular expression "CVE-\d\d\d\d+\d+". The CVE_ID values should correspond to those defined at http://cve.mitre.org.
CWE_ID	The CWE_ID data type is a restriction on the BasicString data type such that it adheres to the regular expression "CWE-\d+". The CWE_ID values should correspond to those defined at http://cwe.mitre.org.
DateTime	The DateTime data type is a restriction on the BasicString data type such that it adheres to the standard defined in ISO-8601.
HexBinary	The HexBinary data type is a restriction on the BasicString data type such that it adheres to the regular expression [0-9A-Fa-f]*. The number of characters allowed is finite but unbounded. The number of digits must be even in length.

LanguageCode	The LanguageCode data type is a restriction on the BasicString data type, such that it adheres to the standard defined in [RFC5646].
QualifiedName	The QualifiedName data type is a restriction on the BasicString data type such that it adheres to the requirements specified in [W3Name].
NoEmbeddedQuoteString	The NoEmbeddedQuoteString data type is a restriction on the BasicString data type such that it does not include any double quote characters. This data type captures properties that were attributes in the XML model.
URI	The URI data type is a restriction on the BasicString data type such that it adheres to the standard defined at http://tools.ietf.org/html/rfc 3986.

3 Data Model Conventions

The following general information and conventions are used to define the individual data models in UML. It should be noted that the STIX data models actually evolved as XML schemas, and as a consequence, our UML model follows some conventions so as to be compatible with the preexisting XML implementation. However, we have abstracted away from the XML implementation as much as possible.

3.1 UML Packages

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. Table 3-1 lists the packages used throughout the STIX data model specification documents, along with the prefix notation and an example. Descriptions of the packages are provided in Section 2.

Table 3-1. Package prefixes used by the STIX Language

Package	STIX Core v1.1.1
Prefix	**stix**
Description	The STIX Core data model defines a STIX Package that encompasses all other objects of STIX.
Example	`stix:TTPsType`
Package	**STIX Common v1.1.1**
Prefix	**stixCommon**
Description	The STIX Common data model defines classes that are shared across the various STIX data models.
Example	`stixCommon:ConfidenceType`
Package	**STIX Data Marking v1.1.1**
Prefix	**marking**
Description	The STIX Data Marking data model enables data markings to be used.
Example	`marking:MarkingType`
Package	**STIX Default Vocabularies v1.1**
Prefix	**stixVocabs**
Description	The STIX default vocabularies define the classes for default controlled vocabularies used within STIX.
Example	`stixVocabs:MalwareTypeVocab`
Package	**Packages used in STIX Default Extensions**
Prefix	**a (ciq address); capec; ciq; stix-ciqidentity; maec; tlpMarking; cvrf; ioc; oval-def; oval-var**

Description	Various packages are used by STIX extentions. Details are given in [$STIX_{EXT}$].
Example	`capec:Attack_PatternType`
Package	STIX Basic Data Types v1.1.1
Prefix	**basicDataTypes**
Description	The STIX Basic Data Types data model defines the types used within STIX.
Example	`basicDataTypes:URI`
Package	STIX Indicator v2.1.1
Prefix	**indicator**
Description	The STIX Indicator data model conveys specific Observable patterns combined with contextual information intended to represent artifacts and/or behaviors of interest within a cyber security context.
Example	`indicator:TestMechanismType`
Package	STIX Incident v1.1.1
Prefix	**incident**
Description	The STIX Incident data model captures discrete instances of a specific set of observed events or properties affecting an organization.
Example	`incident:AffectedAssetType`
Package	STIX TTP v1.1.1
Prefix	**ttp**
Description	The STIX TTP data model captures the behavior or modus operandi of cyber adversaries.
Example	`ttp:AttackPatternType`
Package	STIX Campaign v1.1.1
Prefix	**campaign**
Description	The STIX Campaign data model encompasses one or more Threat Actors pursuing an Intended Effect as observed through sets of Incidents and/or TTP, potentially across organizations.
Example	`campaign:AttributionType`
Package	STIX Threat Actor v1.1.1
Prefix	**ta**
Description	The STIX Threat Actor data model captures characterizations of malicious actors (or adversaries) representing a cyber attack threat including presumed intent and historically observed behavior.
Example	`ta:ObservedTTPsType`

Package	STIX Exploit Target v1.1.1
Prefix	**et**
Description	The STIX Exploit Target data model conveys a vulnerability or weakness in software, systems, networks or configurations that is targeted for exploitation by the TTP of a Threat Actor.
Example	`et:ConfigurationType`

Package	STIX Course of Action v1.1.1
Prefix	**coa**
Description	The STIX Course of Action data model conveys specific measures to be taken to address threats whether they are corrective or preventative to address Exploit Targets, or responsive to counter or mitigate the potential impacts of Incidents.
Example	`coa:StructuredCOAType`

Package	CybOX Core v2.1
Prefix	**cybox**
Description	The CybOX core data model defines the core constructs used in CybOX.
Example	`cybox:ObservablesType`

3.2 Naming Conventions

The UML classes, enumerations, and properties defined in STIX follow the particular naming conventions outlined in Table 3-2[5].

Table 3-2. Naming formats of different object types

Object Type	Format	Example
Class	CamelCase ending with "Type"	IndicatorBaseType
Property (simple)	Lowercase with underscores between words	capec_id
Property (complex)	Capitalized with underscores between words	Associated_Actor
Enumeration	CamelCase ending with "Enum" or "Type	DateTimePrecisionEnum; IndicatorVersionType
Enumeration value	*varies*	Flash drive; Public Disclosure; Externally-Located
Data type	CamelCase or if the words are	PositiveInteger; CVE_ID

[5] These choices were made for the XML schema to differentiate XML attributes and elements. Although such distinctions are not made in the UML model, we kept the naming convention for consistency. We expect that eventually, the names of the UML model will be made uniform.

	acroynms, all capitalized with underscores between words.	

3.3 Identifiers

Optional identifiers (IDs) can be assigned to several STIX constructs so that the constructs can be unambiguously referenced. Technically, the decision to specify an ID on a given construct is optional based on the specifics of the usage context. As a general rule, specifying IDs on particular instances of constructs enables clear referencing, relating, and pivoting.

Assigning IDs supports several very common STIX use cases such as:

- Enabling individual portions of content to be externally referenced unambiguously (e.g., a report talking about a specific Campaign or Threat Actor)
- Enabling the sharing/resharing of portions of STIX content (e.g., PartyB resharing two of a set of 100 Indicators received from PartyA)
- Enabling versioning of content
- Enabling the specification of potentially complex webs of interconnection and correlation between portions of STIX content (e.g., connecting particular TTPs and Indicators to specific Campaigns over time)
- Enabling analysis pivoting on content with multiple contexts (e.g., the same IP Address seen in multiple Incidents and with connections to multiple TTPs and Indicators)

In STIX v1.1.1, each STIX ID is a fully qualified name, which consists of a producer namespace and a unique identifier. The producer namespace is a short-hand prefix, which is separated from the unique identifier by a colon (":"). For example:

```
[producer namespace]:[unique identifier]
```

This format provides high assurance that IDs will be both meaningful and unique. Meaning comes from producer namespace, which denotes who is producing it, and uniqueness comes from the unique identifier.

4 Relationships to Other Externally-defined Data Models

STIX Version 1.1.1 leverages several other externally-defined data models that are relevant to the cyber threat domain. However, the STIX specification documents do not define any classes that are part of a non-STIX data model (e.g., CybOX classes are not defined in STIX specification documents). An alphabetical listing of these other data models is given below.

Please see the STIX Version 1.1.1 default extensions specification document [$STIX_{EXT}$] for further information on all of the externally-defined data models STIX leverages by default (with the exception of CybOX, for which a different reference is given in Section 4.4).

4.1 Common Attack Pattern Enumeration and Classification (CAPEC)

Common Attack Pattern Enumeration and Classification (CAPEC) is a publicly available catalog of attack patterns along with a comprehensive schema and classification taxonomy. By extending the STIX TTP `AttackPatternType` class, STIX Version 1.1.1 uses CAPEC to enable the structured description of attack patterns.

4.2 Common Vulnerability Reporting Framework (CVRF)

The ICASI Common Vulnerability Reporting Framework (CVRF) is an XML-based language that enables different organizations to share critical security-related information in a single format. In addition to capturing basic information and referencing vulnerability registries, the STIX Exploit Target `VulnerabilityType` class is intended to be extended as appropriate to enable the structured description of a vulnerability using CVRF 1.1.

4.3 Customer Information Quality (CIQ)

The OASIS Customer Information Quality (CIQ) is a set of XML specifications for representing characteristic information about individuals and organizations. By extending the STIX Common `AddressAbstractType` and `IdentityType` classes, STIX Version 1.1.1 leverages CIQ Version 3.0 to capture geographic address information and identity information associated with Threat Actors, victims, and sources of information.

4.4 Cyber Observable eXpression (CybOX)

STIX Version 1.1.1 uses the Cyber Observable eXpression (CybOX™) language Version 2.1 to describe cyber Observables. The CybOX data models are natively imported and used within STIX to characterize system and network events, characteristics, and behaviors observed within the operational domain. The reader is referred to [CYBOX] for the definitions of these classes, and in the cases where a STIX class (the subclass) is a

specialization of a CybOX class (the superclass), we will explicitly define the class extensions (i.e., new names and types) that have been made in the STIX subclass.

4.5 Malware Attribute Enumeration and Characterization (MAEC)

Malware Attribute Enumeration and Characterization (MAEC™) is a standardized language for sharing structured information about malware based upon attributes such as behaviors, artifacts, and attack patterns. By extending the STIX TTP `MalwareInstanceType` class, STIX Version 1.1.1 uses MAEC Version 4.1 to capture a structured description of a malware instance.

The Characterizing Malware with STIX and MAEC white paper [$STIX_{MAEC}$] provides more details on the relationship between MAEC and STIX and when each should be used in the context of malware characterization.

4.6 Open Indicators of Compromise (OpenIOC)

Open Indicators of Compromise (OpenIOC) is an extensible XML schema for the description of technical characteristics that identify a known threat, an attacker's methodology, or other evidence of compromise. By extending the STIX Indicator `GenericTestMechanismType` class, STIX Version 1.1.1 enables 2010 OpenIOC to be leveraged as a test mechanism of an Indicator.

4.7 Open Vulnerability and Assessment Language (OVAL)

The Open Vulnerability and Assessment Language (OVAL) is an information security community effort to standardize how to assess and report upon the machine state of computer systems. By extending the STIX Indicator `GenericTestMechanismType` class, STIX Version 1.1.1 enables OVAL 5.10 to be leveraged as a test mechanism of an Indicator.

References

The full set of STIX specification documents are provided below. Note that not all references are used in the STIX specification documents; however, they are included here for completeness.

A.1 STIX Specification Documents

[$STIX_{CAM}$] STIX Campaign Specification
http://stix.mitre.org/about/documents/STIX_Campaign.pdf

[$STIX_{ET}$] STIX Exploit Target Specification
http://stix.mitre.org/about/documents/STIX_ExploitTarget.pdf

[$STIX_{INC}$] STIX Incident Specification
http://stix.mitre.org/about/documents/STIX_Incident.pdf

[$STIX_{IND}$] STIX Indicator Specification
http://stix.mitre.org/about/documents/STIX_Indicator.pdf

[$STIX_{COA}$] STIX Course of Action Specification
http://stix.mitre.org/about/documents/STIX_CourseOfAction.pdf

[$STIX_{TTP}$] STIX TTP Specification
http://stix.mitre.org/about/documents/STIX_TTP.pdf

[$STIX_{TA}$] STIX Threat Actor Specification
http://stix.mitre.org/about/documents/STIX_ThreatActor.pdf

A.2 Other References

[CPE] Common Platform Enumeration (CPE)
http://cpe.mitre.org

[CVE] Common Vulnerabilities and Exposures (CVE)
http://cve.mitre.org

[CWE] Common Weakness Enumeration (CWE)
http://cwe.mitre.org

[CYBOX] Cyber Observable eXpression (CybOX)
http://cybox.mitre.org

[IOC] Open Indicators of Compromise (OpenIOC)
http://openioc.org/

[MAEC] Malware Attribute Enumeration and Characterization (MAEC)
http://maec.mitre.org/

[RFC2119] RFC 2119 – Key words for use in RFCs to Indicate Requirement Levels
http://www.ietf.org/rfc/rfc2119.txt

[RFC5646] RFC 5646 – Tags for Identifying Languages
http://tools.ietf.org/html/rfc5646

[$STIX_{MAEC}$] Characterizing Malware with MAEC and STIX
http://stix.mitre.org/about/documents/Characterizing_Malware_MAEC_and_STIX_v1.0.pdf

[UML] Unified Modeling Language (UML)
http://www.uml.org/
http://www.omg.org/spec/UML/2.0/

[W3Name] Namespaces in XML 1.0 (Third Edition)
http://www.w3.org/TR/REC-xml-names

[W3-DT] XML Schema Part 2: Datatypes Second Edition
http://www.w3.org/TR/xmlschema-2