python-stix Documentation

Release 1.1.1.12

The MITRE Corporation

June 14, 2018

Contents

Version: 1.1.1.12

The **python-stix** library provides an API for developing and consuming *Structured Threat Information eXpression* (**STIX**) content. Developers can leverage the API to develop applications that create, consume, translate, or otherwise process STIX content. This page should help new developers get started with using this library. For more information about STIX, please refer to the STIX website.

Note: These docs provide standard reference for this Python library. For documentation on *idiomatic* usage and *common patterns*, as well as various STIX-related information and utilities, please visit the STIXProject at GitHub.

CHAPTER 1

Versions

Each version of python-stix is designed to work with a single version of the STIX Language. The table below shows the latest version the library for each version of STIX.

STIX Version	python-stix Version
1.1.1	1.1.1.12 (PyPI) (GitHub)
1.1.0	1.1.0.6 (PyPI) (GitHub)
1.0.1	1.0.1.1 (PyPI) (GitHub)
1.0	1.0.0a7 (PyPI) (GitHub)

Users and developers working with multiple versions of STIX content may want to take a look at stix-ramrod, which is a library designed to update STIX and CybOX content.

Check out the Working with python-stix section for examples on how to integrate **stix-ramrod** and **python-stix**.

CHAPTER 2

Contents

Version: 1.1.1.12

Installation

The installation of python-stix can be accomplished through a few different workflows.

Recommended Installation

Use pypi and pip:

```
$ pip install stix
```

You might also want to consider using a virtualenv. Please refer to the pip installation instructions for details regarding the installation of pip.

Dependencies

The python-stix library relies on some non-standard Python libraries for the processing of STIX content. Revisions of python-stix may depend on particular versions of dependencies to function correctly. These versions are detailed within the distutils setup.py installation script.

The following libraries are required to use python-stix:

- lxml - A Pythonic binding for the C libraries **libxml2** and **libxslt**.
- python-cybox - A library for consuming and producing CybOX content.
- python-dateutil - A library for parsing datetime information.

Each of these can be installed with `pip` or by manually downloading packages from PyPI. On Windows, you will probably have the most luck using pre-compiled binaries for `lxml`. On Ubuntu (12.04 or 14.04), you should make sure the following packages are installed before attempting to compile `lxml` from source:

- libxml2-dev
- libxslt1-dev
- zlib1g-dev

Warning: Users have encountered errors with versions of libxml2 (a dependency of lxml) prior to version 2.9.1. The default version of libxml2 provided on Ubuntu 12.04 is currently 2.7.8. Users are encouraged to upgrade libxml2 manually if they have any issues. Ubuntu 14.04 provides libxml2 version 2.9.1.

Manual Installation

If you are unable to use pip, you can also install python-stix with setuptools. If you don't already have setuptools installed, please install it before continuing.

1. Download and install the *dependencies* above. Although setuptools will generally install dependencies automatically, installing the dependencies manually beforehand helps distinguish errors in dependency installation from errors in stix installation. Make sure you check to ensure the versions you install are compatible with the version of stix you plan to install.
2. Download the desired version of stix from PyPI or the GitHub releases page. The steps below assume you are using the 1.1.1.12 release.
3. Extract the downloaded file. This will leave you with a directory named stix-1.1.1.12.

```
$ tar -zxf stix-1.1.1.12.tar.gz
$ ls
stix-1.1.1.12 stix-1.1.1.12.tar.gz
```

OR

```
$ unzip stix-1.1.1.12.zip
$ ls
stix-1.1.1.12 stix-1.1.1.12.zip
```

4. Run the installation script.

```
$ cd stix-1.1.1.12
$ python setup.py install
```

5. Test the installation.

```
$ python
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import stix
>>>
```

If you don't see an `ImportError`, the installation was successful.

Further Information

If you're new to installing Python packages, you can learn more at the Python Packaging User Guide, specifically the Installing Python Packages section.

Version: 1.1.1.12

Getting Started

This page gives an introduction to **python-stix** and how to use it.

Note: This page is being actively worked on; feedback is always welcome.

Prerequisites

The python-stix library provides an API for creating or processing STIX content. As such, it is a developer tool that can be leveraged by those who know Python 2.7/3.3+ and are familiar with object-oriented programming practices, Python package layouts, and are comfortable with the installation of Python libraries. To contribute code to the python-stix repository, users must be familiar with git and GitHub pull request methodologies. Understanding XML, XML Schema, and the STIX language is also incredibly helpful when using python-stix in an application.

Your First STIX Application

Once you have installed python-stix, you can begin writing Python applications that consume or create STIX content!

Note: The *python-stix* library provides **bindings** and **APIs**, both of which can be used to parse and write STIX XML files. For in-depth description of the *APIs, bindings, and the differences between the two*, please refer to APIs or bindings?

Creating a STIX Package

```
from stix.core import STIXPackage, STIXHeader    # Import the STIX Package and STIX Header APIs

stix_package = STIXPackage()                     # Create an instance of STIXPackage
stix_header = STIXHeader()                       # Create an instance of STIXHeader
stix_header.description = "Getting Started!"     # Set the description
stix_package.stix_header = stix_header           # Link the STIX Head to our STIX Package

print(stix_package.to_xml())                     # print the XML for this STIX Package
```

Parsing STIX XML

```
from stix.core import STIXPackage          # Import the STIX Package API

fn = 'stix_content.xml'                    # The STIX content filename
stix_package = STIXPackage.from_xml(fn)    # Parse using the from_xml() method
```

Examples

The python-stix GitHub repository contains several example scripts that help illustrate the capabilities of the APIs. These examples can be found here. Accompanying walkthrough slides are available. These scripts are simple command line utilities that can be executed by passing the name of the script to a Python interpreter.

```
Example:
$ python ex_01.py
```

Note: You must install python-stix before running these example scripts.

Version: 1.1.1.12

Examples

This page includes some basic examples of creating and parsing STIX content.

There are a couple things we do in these examples for purposes of demonstration that shouldn't be done in production code:

- When calling `to_xml()`, we use `include_namespaces=False`. This is to make the example output easier to read, but means the resulting output cannot be successfully parsed. The XML parser doesn't know what namespaces to use if they aren't included. In production code, you should explicitly set `include_namespaces` to `True` or omit it entirely (`True` is the default).
- In some examples, we use `set_id_method(IDGenerator.METHOD_INT)` to make IDs for STIX constructs easier to read and cross-reference within the XML document. In production code, you should omit this statement, which causes random UUIDs to be created instead, or create explicit IDs yourself for STIX constructs.

See the STIX Idioms documentation for more great examples of how to use **python-stix**.

Creating a STIX Package

```
from stix.core import STIXPackage, STIXHeader
from stix.utils import IDGenerator, set_id_method

set_id_method(IDGenerator.METHOD_INT) # For testing and demonstration only!

stix_package = STIXPackage()
stix_header = STIXHeader()
stix_header.description = "Getting Started!"
stix_package.stix_header = stix_header

print stix_package.to_xml(include_namespaces=False)
```

Which outputs:

```
<stix:STIX_Package id="example:Package-1" version="1.1.1" timestamp="2014-08-12T18:03:44.240457+00:0
    <stix:STIX_Header>
        <stix:Description>Getting Started!</stix:Description>
    </stix:STIX_Header>
</stix:STIX_Package>
```

ID Namespaces

By default, **python-stix** sets the default ID namespace to `http://example.com` with an alias of `example`. This results in STIX id declarations that look like `id="example:Package-2813128d-f45e-41f7-b10a-20a5656e3785"`.

To change this, use the `stix.utils.set_id_namespace()` method which takes a dictionary as a parameter.

```
from stix.core import STIXPackage
from stix.utils import set_id_namespace

NAMESPACE = {"http://MY-NAMESPACE.com" : "myNS"}
set_id_namespace(NAMESPACE) # new ids will be prefixed by "myNS"

stix_package = STIXPackage() # id will be created automatically
print stix_package.to_xml()
```

Which outputs:

```
<stix:STIX_Package
    xmlns:myNS="http://MY-NAMESPACE.com"
    xmlns:stixCommon="http://stix.mitre.org/common-1"
    xmlns:stixVocabs="http://stix.mitre.org/default_vocabularies-1"
    xmlns:stix="http://stix.mitre.org/stix-1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
    http://stix.mitre.org/common-1 http://stix.mitre.org/XMLSchema/common/1.1.1/stix_common.xsd
    http://stix.mitre.org/default_vocabularies-1 http://stix.mitre.org/XMLSchema/default_vocabularies
    http://stix.mitre.org/stix-1 http://stix.mitre.org/XMLSchema/core/1.1.1/stix_core.xsd"
    id="myNS:Package-b2039368-9476-4a5b-8c1d-0ef5d1b37e06" version="1.1.1" timestamp="2014-08-12T18:1
```

Success! The `xmlns:myNS="http://MY-NAMESPACE.com"` matches our `NAMESPACE` dictionary and the `id` attribute includes the `myNS` namespace alias.

Working With CybOX

If you are creating CybOX entities such as `Observables`, you'll want to set the ID namespace for `python-cybox` as well.

Note that **python-stix** and `python-cybox` treat namespaces slightly differently (for now anyway). Where **python-stix** uses Python dictionaries, `python-cybox` uses the `cybox.utils.Namespace` class to represent a namespace.

```
from cybox.utils import set_id_namespace, Namespace
from cybox.core import Observable

NAMESPACE = Namespace("http://MY-NAMESPACE.com", "myNS")
set_id_namespace(NAMESPACE)

obs = Observable()
print obs.to_xml()
```

Which outputs:

```
<cybox:ObservableType
    xmlns:myNS="http://MY-NAMESPACE.com"
    xmlns:cybox="http://cybox.mitre.org/cybox-2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://cybox.mitre.org/cybox-2 http://cybox.mitre.org/XMLSchema/core/2.1/cybo
    id="myNS:Observable-7e6191d3-25e9-4283-a80c-867e175224ae">
</cybox:ObservableType>
```

Success (again)! The `xmlns:myNS="http://MY-NAMESPACE.com"` matches our `Namespace` object and the `id` attribute includes the `myNS` namespace alias.

Controlled Vocabularies: VocabString

Many fields in STIX leverage the `stixCommon:ControlledVocabularyStringType`, which acts as a base type for controlled vocabulary implementations. The STIX language defines a set of default controlled vocabularies which are found in the `stix_default_vocabs.xsd` XML Schema file.

The **python-stix** library contains a `stix.common.vocabs` module, which defines the `VocabString` class implementation of the schema `ControlledVocabularyStringType` as well as `VocabString` implementations which correspond to default controlled vocabularies.

For example, the `stix_default_vocabularies.xsd` schema defines a controlled vocabulary for STIX Package Intents: `PackageIntentVocab-1.0`. The `stix.common.vocabs` module contains an analogous `PackageIntent` class, which acts as a derivation of `VocabString`.

Each `VocabString` implementation contains:

- A static list of class-level term attributes, each beginning with `TERM_` (e.g., ``TERM_INDICATORS`)
- A tuple containing all allowed vocabulary terms: `ALLOWED_VALUES`, which is use for input validation
- Methods found on `stix.Entity`, such as `to_xml()`, `to_dict()`, `from_dict()`, etc.

Interacting With VocabString Fields

The following sections define ways of interacting with VocabString fields.

Default Vocabulary Terms

The STIX Language often suggested a default controlled vocabulary type for a given controlled vocabulary field. Each controlled vocabulary contains an enumeration of allowed terms.

Each `VocabString` implementation found in the `stix.common.vocabs` module contains static class-level attributes for each vocabulary term. When setting controlled vocabulary field values, it is recommended that users take advantage of these class-level attributes.

The following demonstrates setting the `Package_Intent` field with a default vocabulary term. Note that the `STIXHeader.package_intents` property returns a list. As such, we use the `append()` method to add terms. Other STIX controlled vocabulary fields may only allow one value rather than a list of values.

```
from stix.core import STIXHeader
from stix.common.vocabs import PackageIntent

header = STIXHeader()
header.package_intents.append(PackageIntent.TERM_INDICATORS)

print header.to_xml(include_namespaces=False)
```

Which outputs:

```
<stix:STIXHeaderType>
    <stix:Package_Intent xsi:type="stixVocabs:PackageIntentVocab-1.0">Indicators</stix:Package_Intent
</stix:STIXHeaderType>
```

Non-Default Vocabulary Terms

Though it is suggested, STIX content authors are not required to use the default controlled vocabulary for a given field. As such, **python-stix** allows users to pass in non-default values for controlled vocabulary fields.

To set a controlled vocabulary to a non-default vocabulary term, pass a `VocabString` instance into a controlled vocabulary field.

A raw `VocabString` field will contain no `xsi:type` information or `ALLOWED_VALUES` members, which removes the input and schema validation requirements.

```
from stix.core import STIXHeader
from stix.common.vocabs import VocabString, PackageIntent

header = STIXHeader()
non_default_term = VocabString("NON-DEFAULT VOCABULARY TERM")
header.package_intents.append(non_default_term)

print header.to_xml(include_namespaces=False)
```

Which outputs:

```
<stix:STIXHeaderType>
    <stix:Package_Intent>NON-DEFAULT VOCABULARY TERM</stix:Package_Intent>
</stix:STIXHeaderType>
```

Notice that the `<stix:Package_Intent>` field does not have an `xsi:type` attribute. As such, this field can contain any string value and is not bound by a controlled vocabulary enumeration of terms.

Working With Custom Controlled Vocabularies

STIX allows content authors and developers to extend the `ControlledVocabularyStringType` schema type for the definition of new controlled vocabularies. The **python-stix** library allows developers to create and register Python types which mirror the custom XML Schema vocabulary types.

XSD Example The following XML Schema example shows the definition of a a new custom controlled vocabulary schema type. Instances of this schema type could be used wherever a `ControlledVocabularyStringType` instance is expected (e.g., the `STIX_Header/Package_Intent` field).

```
Filename: customVocabs.xsd

<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:customVocabs="http://customvocabs.com/vocabs-1"
    xmlns:stixVocabs="http://stix.mitre.org/default_vocabularies-1"
    xmlns:stixCommon="http://stix.mitre.org/common-1"
    targetNamespace="http://customvocabs.com/vocabs-1"
    elementFormDefault="qualified"
    version="1.1.1"
    xml:lang="English">
    <xs:import namespace="http://stix.mitre.org/common-1" schemaLocation="http://stix.mitre.org/XMLSc
    <xs:complexType name="CustomVocab-1.0">
        <xs:simpleContent>
            <xs:restriction base="stixCommon:ControlledVocabularyStringType">
                <xs:simpleType>
                    <xs:union memberTypes="customVocabs:CustomEnum-1.0"/>
                </xs:simpleType>
                <xs:attribute name="vocab_name" type="xs:string" use="optional" fixed="Test Vocab"/>
                <xs:attribute name="vocab_reference" type="xs:anyURI" use="optional" fixed="http://ex
            </xs:restriction>
        </xs:simpleContent>
    </xs:complexType>
```

```
    <xs:simpleType name="CustomEnum-1.0">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FOO"/>
            <xs:enumeration value="BAR"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

XML Instance Sample The following STIX XML instance document shows a potential use of this field. Note the `xsi:type=customVocabs:CustomVocab-1.0` on the `Package_Intent` field.

```
Filename: customVocabs.xml

<stix:STIX_Package
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:stixExample="http://stix.mitre.org/example"
    xmlns:stix="http://stix.mitre.org/stix-1"
    xmlns:customVocabs="http://customvocabs.com/vocabs-1"
    xsi:schemaLocation="
    http://stix.mitre.org/stix-1 /path/to/stix_core.xsd
    http://customvocabs.com/vocabs-1 /path/to/customVocabs.xsd"
    id="stixExample:STIXPackage-33fe3b22-0201-47cf-85d0-97c02164528d"
    timestamp="2014-05-08T09:00:00.000000Z"
    version="1.1.1">
    <stix:STIX_Header>
        <stix:Package_Intent xsi:type="customVocabs:CustomVocab-1.0">FOO</stix:Package_Intent>
    </stix:STIX_Header>
</stix:STIX_Package>
```

Python Code To parse content which uses custom controlled vocabularies, Python developers don't have to do anything special–you just call `STIXPackage.from_xml()` on the input and all the namespaces, `xsi:types`, etc. are attached to each instance of `VocabString`. When serializing the document, the input namespaces and `xsi:type` attributes are retained!

However, to *create* new content which utilizes a schema defined and enforced custom controlled vocabulary, developers must create a *`VocabString`* implementation which mirrors the schema definition.

For our `CustomVocab-1.0` schema type, the Python would look like this:

```
from stix.common import vocabs

# Create a custom vocabulary type
class CustomVocab(vocabs.VocabString):
    _namespace = 'http://customvocabs.com/vocabs-1'
    _XSI_TYPE = 'customVocabs:CustomVocab-1.0'
    _ALLOWED_VALUES = ('FOO', 'BAR')

# Register the type as a VocabString
vocabs.add_vocab(CustomVocab)
```

As you can see, we can express a lot of the same information found in the XML Schema definition, just with a lot less typing!

- `_namespace`: The `targetNamespace` for our custom vocabulary
- **`_XSI_TYPE`: The `xsi:type` attribute value to write out for instances** of this vocabulary.
- `_ALLOWED_VALUES`: A `tuple` of allowable values for this vocabulary.

Note: The call to `add_vocab()` registers the class and its `xsi:type` as a `VocabString` implementation so **python-stix** will know to build instances of `CustomVocab` when parsed content contains `CustomVocab-1.0` content. You must call `add_vocab()` to register your class prior to parsing content if you want the parser to build instances of your custom vocabulary class!

```
# builtin
from StringIO import StringIO

# python-stix modules
from stix.core import STIXPackage
from stix.common import vocabs

XML = \
"""
<stix:STIX_Package
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:stix="http://stix.mitre.org/stix-1"
    xmlns:customVocabs="http://customvocabs.com/vocabs-1"
    xmlns:example="http://example.com/"
    xsi:schemaLocation="
    http://stix.mitre.org/stix-1 /path/to/stix_core.xsd
    http://customvocabs.com/vocabs-1 /path/to/customVocabs.xsd"
    id="example:STIXPackage-33fe3b22-0201-47cf-85d0-97c02164528d"
    timestamp="2014-05-08T09:00:00.000000Z"
    version="1.1.1">
    <stix:STIX_Header>
        <stix:Package_Intent xsi:type="customVocabs:CustomVocab-1.0">FOO</stix:Package_Intent>
    </stix:STIX_Header>
</stix:STIX_Package>
"""

# Create a VocabString class for our CustomVocab-1.0 vocabulary which
class CustomVocab(vocabs.VocabString):
    _namespace = 'http://customvocabs.com/vocabs-1'
    _XSI_TYPE = 'customVocabs:CustomVocab-1.0'
    _ALLOWED_VALUES = ('FOO', 'BAR')

# Register our Custom Vocabulary class so parsing builds instances of
# CustomVocab
vocabs.add_vocab(CustomVocab)

# Parse the input document
sio = StringIO(XML)
package = STIXPackage.from_xml(sio)

# Retrieve the first (and only) Package_Intent entry
package_intent = package.stix_header.package_intents[0]

# Print information about the input Package_Intent
print type(package_intent), package_intent.xsi_type, package_intent

# Add another Package Intent
bar = CustomVocab('BAR')
package.stix_header.add_package_intent(bar)

# This will include the 'BAR' CustomVocab entry
```

```
print package.to_xml()
```

Version: 1.1.1.12

APIs or bindings?

This page describes both the **APIs** and the **bindings** provided by the *python-stix* library.

Overview

The python-stix library provides APIs and utilities that aid in the creation, consumption, and processing of Structured Threat Information eXpression (STIX) content. The APIs that drive much of the functionality of python-stix sit on top of a binding layer that acts as a direct connection between Python and the STIX XML. Because both the APIs and the bindings allow for the creation and development of STIX content, developers that are new to python-stix may not understand the differences between the two. This document aims to identify the purpose and uses of the APIs and bindings.

Bindings

The python-stix library leverages machine generated XML-to-Python bindings for the creation and processing of STIX content. These bindings are created using the generateDS utility and can be found under stix.bindings within the package hierarchy.

The STIX bindings allow for a direct, complete mapping between Python classes and STIX XML Schema data structures. That being said, it is possible (though not advised) to use only the STIX bindings to create STIX documents. However, because the code is generated from XML Schema without contextual knowledge of relationships or broader organizational/developmental schemes, it is often a cumbersome and laborious task to create even the simplest of STIX documents.

Developers within the python-stix team felt that the binding code did not lend itself to rapid development or natural navigation of data, and so it was decided that a higher-level API should be created.

APIs

The python-stix APIs are classes and utilities that leverage the STIX bindings for the creation and processing of STIX content. The APIs are designed to behave more naturally when working with STIX content, allowing developers to conceptualize and interact with STIX documents as pure Python objects and not XML Schema objects.

The APIs provide validation of inputs, multiple input and output formats, more Pythonic access of data structure internals and interaction with classes, and better interpretation of a developers intent through datatype coercion and implicit instantiation.

Note: The python-stix APIs are under constant development. Our goal is to provide full API coverage of the STIX data structures, but not all structures are exposed via the APIs yet. Please refer to the API Reference for API coverage details.

Brevity Wins

The two code examples show the difference in creating and printing a simple STIX document consisting of only a STIX Package and a STIX Header with a description and produced time using the python-stix and python-cybox bindings. Both examples will produce the same STIX XML!

API Example

```
from datetime import datetime
from stix.core import STIXPackage, STIXHeader
from stix.common import InformationSource
from cybox.common import Time

# Create the STIX Package and STIX Header objects
stix_package = STIXPackage()
stix_header = STIXHeader()

# Set the description
stix_header.description = 'APIs vs. Bindings Wiki Example'

# Set the produced time to now
stix_header.information_source = InformationSource()
stix_header.information_source.time = Time()
stix_header.information_source.time.produced_time = datetime.now()

# Build document
stix_package.stix_header = stix_header

# Print the document to stdout
print(stix_package.to_xml())
```

Binding Example

```
import sys
from datetime import datetime

import stix.bindings.stix_core as stix_core_binding
import stix.bindings.stix_common as stix_common_binding
import cybox.bindings.cybox_common as cybox_common_binding

# Create the STIX Package and STIX Header objects
stix_package = stix_core_binding.STIXType()
stix_header = stix_core_binding.STIXHeaderType()

# Set the description
stix_header_description = stix_common_binding.StructuredTextType()
stix_header_description.set_valueOf_('APIs vs. Bindings Wiki Example')

# Set the produced time to now
stix_header_time = cybox_common_binding.TimeType()
stix_header_time.set_Produced_Time(datetime.now())

# Bind the time to the STIX Header's Information Source element
stix_header_info_source = stix_common_binding.InformationSourceType()
stix_header_info_source.set_Time(stix_header_time)

# Build the document
stix_header.set_Description(stix_header_description)
stix_header.set_Information_Source(stix_header_info_source)
```

```
stix_package.set_STIX_Header(stix_header)

# Print the document to stdout
stix_package.export(sys.stdout, 0, stix_core_binding.DEFAULT_XML_NS_MAP)
```

Feedback

If there is a problem with the APIs or bindings, or if there is functionality missing from the APIs that forces the use of the bindings, let us know in the python-stix issue tracker

CHAPTER 3

API Reference

Version: 1.1.1.12

API Reference

The *python-stix* APIs are the recommended tools for reading, writing, and manipulating STIX XML documents.

Note: The python-stix APIs are currently under development. As such, API coverage of STIX data constructs is incomplete; please bear with us as we work toward complete coverage. This documentation also serves to outline current API coverage.

STIX

Modules located in the base stix package

Version: 1.1.1.12

`stix` Module

Classes

`stix.`**`supported_stix_version`**`()`
Returns a tuple of STIX version strings that this version of python-stix supports (i.e., can parse).

`stix.`**`register_extension`**`(`*`cls`*`)`
Class decorator for registering a stix.Entity class as an implementation of an xml type.

Classes must have an `_XSI_TYPE` class attributes to be registered.

Note: This was designed for internal use.

`stix.`**`lookup_extension`**`(`*`typeinfo`*`, `*`default=None`*`)`
Returns a stix.Entity class for that has been registered for the *typeinfo* value.

Note: This is for internal use only.

Parameters

- **typeinfo** – An object or string containing type information. This can be either an xsi:type attribute value or a stix.bindings object.
- **default** – Return class if typeinfo is None or contains no xml type information.

Returns A stix.Entity implementation class for the *xsi_type*.

Raises `ValueError` – If no class has been registered for the *xsi_type*.

`stix.`**add_extension**(*cls*)

Registers a stix.Entity class as an implementation of an xml type.

Classes must have an `_XSI_TYPE` class attributes to be registered. The value of this attribute must be a valid xsi:type.

Note: This was designed for internal use.

Version: 1.1.1.12

stix.base Module

Classes

class `stix.base.`**Entity**

Base class for all classes in the STIX API.

find(*id_*)

Searches the children of a *Entity* implementation for an object with an `id_` property that matches *id_*.

to_xml(*include_namespaces=True*, *include_schemalocs=False*, *ns_dict=None*, *schemaloc_dict=None*, *pretty=True*, *auto_namespace=True*, *encoding='utf-8'*)

Serializes a *Entity* instance to an XML string.

The default character encoding is `utf-8` and can be set via the *encoding* parameter. If *encoding* is `None`, a string (unicode in Python 2, str in Python 3) is returned.

Parameters

- **auto_namespace** – Automatically discover and export XML namespaces for a STIX *Entity* instance.
- **include_namespaces** – Export namespace definitions in the output XML. Default is `True`.
- **include_schemalocs** – Export `xsi:schemaLocation` attribute in the output document. This will attempt to associate namespaces declared in the STIX document with schema locations. If a namespace cannot be resolved to a schemaLocation, a Python warning will be raised. Schemalocations will only be exported if *include_namespaces* is also `True`.
- **ns_dict** – Dictionary of XML definitions (namespace is key, alias is value) to include in the exported document. This must be passed in if *auto_namespace* is `False`.

- **schemaloc_dict** – Dictionary of XML `namespace: schema location` mappings to include in the exported document. These will only be included if *auto_namespace* is `False`.
- **pretty** – Pretty-print the XML.
- **encoding** – The output character encoding. Default is `utf-8`. If *encoding* is set to `None`, a string (unicode in Python 2, str in Python 3) is returned.

Returns An XML string for this *Entity* instance. Default character encoding is `utf-8`.

class `stix.base.`**EntityList**(**args*)
Bases: `mixbox.entities.EntityList`, *stix.base.Entity*

Version: 1.1.1.12

stix.data_marking Module

Classes

class `stix.data_marking.`**Marking**(*markings=None*)
Bases: *stix.base.EntityList*

class `stix.data_marking.`**MarkingSpecification**(*controlled_structure=None*, *marking_structures=None*)
Bases: *stix.base.Entity*

class `stix.data_marking.`**MarkingStructure**
Bases: *stix.base.Entity*

Functions

`stix.data_marking.`**add_extension**(*cls*)
Registers a stix.Entity class as an implementation of an xml type.

Classes must have an `_XSI_TYPE` class attributes to be registered. The value of this attribute must be a valid xsi:type.

Note: This was designed for internal use.

STIX Campaign

Modules located in the stix.campaign package

Version: 1.1.1.12

stix.campaign Module

Classes

class `stix.campaign.`**Campaign**(*id_=None*, *idref=None*, *timestamp=None*, *title=None*, *description=None*, *short_description=None*)
Bases: `stix.base.BaseCoreComponent`

Implementation of the STIX Campaign.

Parameters

- **id** (*optional*) – An identifier. If `None`, a value will be generated via `mixbox.idgen.create_id()`. If set, this will unset the `idref` property.
- **idref** (*optional*) – An identifier reference. If set this will unset the `id_` property.
- **timestamp** (*optional*) – A timestamp value. Can be an instance of `datetime.datetime` or `str`.
- **description** – A description of the purpose or intent of this object.
- **short_description** – A short description of the intent or purpose of this object.
- **title** – The title of this object.

add_activity(*value*)
Adds an *Activity* object to the `activity` collection.

class stix.campaign.**AssociatedCampaigns**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

class stix.campaign.**Attribution**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

class stix.campaign.**Names**(**args*)
Bases: *stix.base.EntityList*

class stix.campaign.**RelatedIncidents**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

class stix.campaign.**RelatedIndicators**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

class stix.campaign.**RelatedTTPs**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

STIX Common

Modules located in the stix.common package

Version: 1.1.1.12

stix.common Module

Classes

class stix.common.**EncodedCDATA**(*value=None*, *encoded=None*)
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.common.activity Module

Classes

class stix.common.activity.**Activity**
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.common.confidence Module

Classes

class stix.common.confidence.**Confidence**(*value=None*, *timestamp=None*, *description=None*, *source=None*)
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.common.datetimewithprecision Module

Classes

class stix.common.datetimewithprecision.**DateTimeWithPrecision**(*value=None*, *precision='second'*)
Bases: *stix.base.Entity*

Constants

stix.common.datetimewithprecision.**DATE_PRECISION_VALUES = ('year', 'month', 'day')**
tuple() -> empty tuple tuple(iterable) -> tuple initialized from iterable's items

If the argument is a tuple, the return value is the same object.

stix.common.datetimewithprecision.**TIME_PRECISION_VALUES = ('hour', 'minute', 'second')**
tuple() -> empty tuple tuple(iterable) -> tuple initialized from iterable's items

If the argument is a tuple, the return value is the same object.

stix.common.datetimewithprecision.**DATETIME_PRECISION_VALUES = ('year', 'month', 'day', 'hour', 'minute', '**
tuple() -> empty tuple tuple(iterable) -> tuple initialized from iterable's items

If the argument is a tuple, the return value is the same object.

Version: 1.1.1.12

stix.common.identity Module

Classes

class stix.common.identity.**Identity**(*id_=None*, *idref=None*, *name=None*, *related_identities=None*)
Bases: *stix.base.Entity*

class stix.common.identity.**RelatedIdentities**(**args*)
Bases: *stix.base.EntityList*

Functions

stix.common.identity.**add_extension**(*cls*)
Registers a stix.Entity class as an implementation of an xml type.

Classes must have an _XSI_TYPE class attributes to be registered. The value of this attribute must be a valid xsi:type.

Note: This was designed for internal use.

Version: 1.1.1.12

stix.common.information_source Module

Classes

class stix.common.information_source.**InformationSource**(*description=None*, *identity=None*, *time=None*, *tools=None*, *contributing_sources=None*, *references=None*)
Bases: *stix.base.Entity*

class stix.common.information_source.**ContributingSources**(**args*)
Bases: *stix.base.EntityList*

Version: 1.1.1.12

stix.common.kill_chains Module

Classes

class stix.common.kill_chains.**KillChain**(*id_=None*, *name=None*, *definer=None*, *reference=None*)
Bases: *stix.base.Entity*

class stix.common.kill_chains.**KillChains**(**args*)
Bases: *stix.base.EntityList*

class stix.common.kill_chains.**KillChainPhase**(*phase_id=None*, *name=None*, *ordinality=None*)
Bases: *stix.base.Entity*

class stix.common.kill_chains.**KillChainPhaseReference**(*phase_id=None*, *name=None*, *ordinality=None*, *kill_chain_id=None*, *kill_chain_name=None*)
Bases: *stix.common.kill_chains.KillChainPhase*

class stix.common.kill_chains.**KillChainPhasesReference**(**args*)
Bases: *stix.base.EntityList*

Lockheed Martin Kill Chain

There is a shortcuts for adding kill chain phases from the Lockheed Martin Cyber Kill Chain to indicators:

```
from stix.common.kill_chains.lmco import PHASE_RECONNAISSANCE
from stix.indicator import Indicator
i = Indicator()
i.add_kill_chain_phase(PHASE_RECONNAISSANCE)
print i.to_xml(include_namespaces=False)
```

```
<indicator:Indicator id="example:indicator-2bb1c0ea-7dd8-40fb-af64-7199f00719c1"
        timestamp="2015-03-17T19:14:22.797675+00:00" xsi:type='indicator:IndicatorType'>
    <indicator:Kill_Chain_Phases>
        <stixCommon:Kill_Chain_Phase phase_id="stix:TTP-af1016d6-a744-4ed7-ac91-00fe2272185a"/>
    </indicator:Kill_Chain_Phases>
</indicator:Indicator>
```

Version: 1.1.1.12

stix.common.related Module

Classes

class stix.common.related.**GenericRelationship**(*confidence=None*, *information_source=None*, *relationship=None*)

Bases: *stix.base.Entity*

class stix.common.related.**GenericRelationshipList**(*scope=None*, **args*)

Bases: *stix.base.EntityList*

Base class for concrete GenericRelationshipList types.

Note: Subclasses must supply exactly one multiple TypedField.

---

class stix.common.related.**RelatedPackageRef**(*idref=None*, *timestamp=None*, *confidence=None*, *information_source=None*, *relationship=None*)

Bases: *stix.common.related.GenericRelationship*

class stix.common.related.**RelatedPackageRefs**(**args*)

Bases: *stix.base.EntityList*

class stix.common.related.**_BaseRelated**(*item=None*, *confidence=None*, *information_source=None*, *relationship=None*)

Bases: *stix.common.related.GenericRelationship*

A base class for related types.

This class is not a real STIX type and should not be directly instantiated.

Note: Subclasses must supply a TypedField named *item*!

---

class stix.common.related.**RelatedCampaign**(*item=None*, *confidence=None*, *information_source=None*, *relationship=None*)

Bases: *stix.common.related._BaseRelated*

class stix.common.related.**RelatedCOA**(*item=None*, *confidence=None*, *information_source=None*, *relationship=None*)
Bases: *stix.common.related._BaseRelated*

class stix.common.related.**RelatedExploitTarget**(*item=None*, *confidence=None*, *information_source=None*, *relationship=None*)
Bases: *stix.common.related._BaseRelated*

class stix.common.related.**RelatedIdentity**(*item=None*, *confidence=None*, *information_source=None*, *relationship=None*)
Bases: *stix.common.related._BaseRelated*

class stix.common.related.**RelatedIncident**(*item=None*, *confidence=None*, *information_source=None*, *relationship=None*)
Bases: *stix.common.related._BaseRelated*

class stix.common.related.**RelatedIndicator**(*item=None*, *confidence=None*, *information_source=None*, *relationship=None*)
Bases: *stix.common.related._BaseRelated*

class stix.common.related.**RelatedObservable**(*item=None*, *confidence=None*, *information_source=None*, *relationship=None*)
Bases: *stix.common.related._BaseRelated*

class stix.common.related.**RelatedThreatActor**(*item=None*, *confidence=None*, *information_source=None*, *relationship=None*)
Bases: *stix.common.related._BaseRelated*

class stix.common.related.**RelatedTTP**(*item=None*, *confidence=None*, *information_source=None*, *relationship=None*)
Bases: *stix.common.related._BaseRelated*

Version: 1.1.1.12

stix.common.statement Module

Classes

class stix.common.statement.**Statement**(*value=None*, *timestamp=None*, *description=None*, *source=None*)
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.common.structured_text Module

Classes

class stix.common.structured_text.**StructuredText**(*value=None*)
Bases: *stix.base.Entity*

Used for storing descriptive text elements.

id_
An id for the text element, typically used for controlled structure xpath selectors.

value
The text value of this object.

structuring_format
The format of the text. For example, `html5`.

__str__()
Returns a UTF-8 encoded string representation of the `value`.

__unicode__()
Returns a `unicode` string representation of the `value`.

to_dict()
Converts this object into a dictionary representation.

> **Note:** If no properies or attributes are set other than `value`, this will return a string.

Version: 1.1.1.12

stix.common.tools Module

Classes

class stix.common.tools.**ToolInformation**(*title=None*, *short_description=None*, *tool_name=None*, *tool_vendor=None*)
Bases: *stix.base.Entity*, cybox.common.tools.ToolInformation

Version: 1.1.1.12

stix.common.vocabs Module

Classes

class stix.common.vocabs.**VocabString**(*value=None*)
Bases: *stix.base.Entity*

is_plain()
Whether the VocabString can be represented as a single value.

stix.common.vocabs.**AssetType**
alias of AssetType_1_0

stix.common.vocabs.**AttackerInfrastructureType**
alias of AttackerInfrastructureType_1_0

stix.common.vocabs.**AttackerToolType**
alias of AttackerToolType_1_0

stix.common.vocabs.**AvailabilityLossType**
alias of AvailabilityLossType_1_1_1

stix.common.vocabs.**CampaignStatus**
alias of CampaignStatus_1_0

stix.common.vocabs.**COAStage**
alias of COAStage_1_0

stix.common.vocabs.**CourseOfActionType**
alias of CourseOfActionType_1_0

stix.common.vocabs.**DiscoveryMethod**
alias of DiscoveryMethod_2_0

stix.common.vocabs.**HighMediumLow**
alias of HighMediumLow_1_0

stix.common.vocabs.**ImpactQualification**
alias of ImpactQualification_1_0

stix.common.vocabs.**ImpactRating**
alias of ImpactRating_1_0

stix.common.vocabs.**IncidentCategory**
alias of IncidentCategory_1_0

stix.common.vocabs.**IncidentEffect**
alias of IncidentEffect_1_0

stix.common.vocabs.**IncidentStatus**
alias of IncidentStatus_1_0

stix.common.vocabs.**IndicatorType**
alias of IndicatorType_1_1

stix.common.vocabs.**InformationSourceRole**
alias of InformationSourceRole_1_0

stix.common.vocabs.**InformationType**
alias of InformationType_1_0

stix.common.vocabs.**IntendedEffect**
alias of IntendedEffect_1_0

stix.common.vocabs.**LocationClass**
alias of LocationClass_1_0

stix.common.vocabs.**LossDuration**
alias of LossDuration_1_0

stix.common.vocabs.**LossProperty**
alias of LossProperty_1_0

stix.common.vocabs.**MalwareType**
alias of MalwareType_1_0

stix.common.vocabs.**ManagementClass**
alias of ManagementClass_1_0

stix.common.vocabs.**Motivation**
alias of Motivation_1_1

stix.common.vocabs.**OwnershipClass**
alias of OwnershipClass_1_0

stix.common.vocabs.**PackageIntent**
alias of PackageIntent_1_0

stix.common.vocabs.**PlanningAndOperationalSupport**
alias of PlanningAndOperationalSupport_1_0_1

stix.common.vocabs.**SecurityCompromise**
alias of SecurityCompromise_1_0

stix.common.vocabs.**SystemType**
alias of SystemType_1_0

stix.common.vocabs.**ThreatActorSophistication**
alias of ThreatActorSophistication_1_0

stix.common.vocabs.**ThreatActorType**
alias of ThreatActorType_1_0

Functions

stix.common.vocabs.**add_vocab**(*cls*)
Registers a VocabString subclass.

Note: The *register_vocab()* class decorator has replaced this method.

stix.common.vocabs.**register_vocab**(*cls*)
Class decorator that registers a VocabString subclass.

Also, calculate all the permitted values for class being decorated by adding an _ALLOWED_VALUES tuple of all the values of class members beginning with TERM_.

STIX Core

Modules located in the stix.core package

Version: 1.1.1.12

stix.core.stix_header Module

Classes

class stix.core.stix_header.**STIXHeader**(*package_intents=None*, *description=None*, *handling=None*, *information_source=None*, *title=None*, *short_description=None*)
Bases: *stix.base.Entity*

The STIX Package Header.

Parameters

- **handling** – The data marking section of the Header.
- **information_source** – The *InformationSource* section of the Header.
- **package_intents** – A collection of *VocabString* defining the intent of the parent *STIXPackage*.
- **description** – A description of the intent or purpose of the parent *STIXPackage*.
- **short_description** – A short description of the intent or purpose of the parent *STIXPackage*.
- **title** – The title of the *STIXPackage*.

profiles
A collection of STIX Profiles the parent *STIXPackage* conforms to.

title
The title of the parent *STIXPackage*.

add_profile(*profile*)
Adds a profile to the STIX Header. A Profile is represented by a string URI.

Version: 1.1.1.12

stix.core.stix_package Module

Classes

class stix.core.stix_package.**STIXPackage**(*id_=None*, *idref=None*, *timestamp=None*, *stix_header=None*, *courses_of_action=None*, *exploit_targets=None*, *indicators=None*, *observables=None*, *incidents=None*, *threat_actors=None*, *ttps=None*, *campaigns=None*, *related_packages=None*)

Bases: *stix.base.Entity*

A STIX Package object.

Parameters

- **id** (*optional*) – An identifier. If `None`, a value will be generated via `mixbox.idgen.create_id()`. If set, this will unset the `idref` property.
- **idref** – An identifier reference. If set this will unset the `id_` property.
- **timestamp** – A timestamp value. Can be an instance of `datetime.datetime` or `str`.
- **stix_header** – A Report `Header` object.
- **campaigns** – A collection of *Campaign* objects.
- **courses_of_action** – A collection of *CourseOfAction* objects.
- **exploit_targets** – A collection of *ExploitTarget* objects.
- **incidents** – A collection of *Incident* objects.
- **indicators** – A collection of *Indicator* objects.
- **threat_actors** – A collection of *ThreatActor* objects.
- **ttps** – A collection of *TTP* objects.
- **related_packages** – A collection of `RelatedPackage` objects.

add(*entity*)
Adds *entity* to a top-level collection. For example, if *entity* is an Indicator object, the *entity* will be added to the `indicators` top-level collection.

add_campaign(*campaign*)
Adds a `Campaign` object to the `campaigns` collection.

add_course_of_action(*course_of_action*)
Adds an *CourseOfAction* object to the `courses_of_action` collection.

add_exploit_target(*exploit_target*)
Adds an *ExploitTarget* object to the `exploit_targets` collection.

add_incident(*incident*)
Adds an *Incident* object to the `incidents` collection.

add_indicator(*indicator*)
Adds an *Indicator* object to the `indicators` collection.

add_observable(*observable*)
Adds an `Observable` object to the `observables` collection.

If *observable* is not an `Observable` instance, an effort will be made to convert it to one.

add_related_package(*related_package*)
Adds a `RelatedPackage` object to the `related_packages` collection.

add_threat_actor(*threat_actor*)
Adds an *ThreatActor* object to the `threat_actors` collection.

add_ttp(*ttp*)
Adds an *TTP* object to the `ttps` collection.

classmethod from_xml(*xml_file*, *encoding=None*)
Parses the *xml_file* file-like object and returns a *STIXPackage* instance.

Parameters

- **xml_file** – A file, file-like object, etree._Element, or etree._ElementTree instance.
- **encoding** – The character encoding of the *xml_file* input. If `None`, an attempt will be made to determine the input character encoding. Default is `None`.

Returns An instance of *STIXPackage*.

class stix.core.stix_package.**RelatedPackages**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

Version: 1.1.1.12

stix.core.ttps Module

Classes

class stix.core.ttps.**TTPs**(*ttps=None*)
Bases: *stix.base.EntityList*

STIX Course of Action (COA)

Modules located in the stix.coa package

Version: 1.1.1.12

stix.coa Module

Classes

class stix.coa.**CourseOfAction**(*id_=None*, *idref=None*, *timestamp=None*, *title=None*, *description=None*, *short_description=None*)
Bases: `stix.base.BaseCoreComponent`

Implementation of the STIX Course of Action.

Parameters

- **id** (*optional*) – An identifier. If `None`, a value will be generated via `mixbox.idgen.create_id()`. If set, this will unset the `idref` property.
- **idref** (*optional*) – An identifier reference. If set this will unset the `id_` property.
- **timestamp** (*optional*) – A timestamp value. Can be an instance of `datetime.datetime` or `str`.
- **description** – A description of the purpose or intent of this object.
- **short_description** – A short description of the intent or purpose of this object.
- **title** – The title of this object.

class `stix.coa.`**RelatedCOAs**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

Version: 1.1.1.12

stix.coa.objective Module

Classes

class `stix.coa.objective.`**Objective**(*description=None*, *short_description=None*)
Bases: *stix.base.Entity*

STIX Exploit Target

Modules located in the stix.exploit_target package

Version: 1.1.1.12

stix.exploit_target Module

Overview

The *stix.exploit_target* module implements `ExploitTarget`. This denotes the specific vulnerability, weakness, or software configuration that creates a security risk.

Documentation Resources

- ExploitTarget Data Model
- ExploitTarget Idioms

Classes

class `stix.exploit_target.`**ExploitTarget**(*id_=None*, *idref=None*, *timestamp=None*, *title=None*, *description=None*, *short_description=None*)
Bases: `stix.base.BaseCoreComponent`

Implementation of STIX Exploit Target.

Parameters

- **id** (*optional*) – An identifier. If `None`, a value will be generated via `mixbox.idgen.create_id()`. If set, this will unset the `idref` property.
- **idref** (*optional*) – An identifier reference. If set this will unset the `id_` property.
- **title** (*optional*) – A string title.
- **timestamp** (*optional*) – A timestamp value. Can be an instance of `datetime.datetime` or `str`.
- **description** (*optional*) – A string description.
- **short_description** (*optional*) – A string short description.

add_configuration(*value*)
Adds a configuration to the `configurations` list property.

Note: If `None` is passed in no value is added

Parameters **value** – A configuration value.

Raises `ValueError` – If the *value* param is of type *Configuration*

add_vulnerability(*value*)
Adds a vulnerability to the `vulnerabilities` list property.

Note: If `None` is passed in no value is added

Parameters **value** – A *Vulnerability* object..

Raises `ValueError` – if the *value* param is of type *Vulnerability*

add_weakness(*value*)
Adds a weakness to the `weaknesses` list property.

Note: If `None` is passed in no value is added

Parameters **value** – A *Weakness* object.

Raises: ValueError if the *value* param is of type *Weakness*

class `stix.exploit_target.`**PotentialCOAs**(*coas=None*, *scope=None*)
Bases: *stix.common.related.GenericRelationshipList*

A list of `Potential_COA` objects, defaults to empty array

class `stix.exploit_target.`**RelatedExploitTargets**(*related_exploit_targets=None*, *scope=None*)
Bases: *stix.common.related.GenericRelationshipList*

A list of `RelatedExploitTargets` objects, defaults to empty array

Version: 1.1.1.12

stix.exploit_target.configuration Module

Overview

The *stix.exploit_target.configuration* module captures the software configuration that causes a vulnerability in a system.

Classes

class stix.exploit_target.configuration.**Configuration**(*description=None*, *short_description=None*, *cce_id=None*)

Bases: *stix.base.Entity*

Implementation of STIX `Configuration`.

Parameters

- **cce_id** (*optional*) – Common Configuration Enumeration value as a string
- **description** (*optional*) – A string description.
- **short_description** (*optional*) – A string short description.

Version: 1.1.1.12

stix.exploit_target.vulnerability Module

Overview

The *stix.exploit_target.vulnerability* module captures the software version and specific bug that causes an exploitable condition.

Classes

class stix.exploit_target.vulnerability.**Vulnerability**(*title=None*, *description=None*, *short_description=None*)

Bases: *stix.base.Entity*

Implementation of STIX `Vulnerability`.

Parameters

- **title** (*optional*) – A string title.
- **description** (*optional*) – A string description.
- **short_description** (*optional*) – A string short description.

class stix.exploit_target.vulnerability.**CVSSVector**

Bases: *stix.base.Entity*

Common Vulnerabilit Scoring System object, representing its component measures

class stix.exploit_target.vulnerability.**AffectedSoftware**(*scope=None*, **args*)

Bases: *stix.common.related.GenericRelationshipList*

Version: 1.1.1.12

stix.exploit_target.weakness Module

Overview

The *stix.exploit_target.weakness* module captures a given software weakness as enumerated by CWE

Classes

class stix.exploit_target.weakness.**Weakness**(*description=None*, *cwe_id=None*)
Bases: *stix.base.Entity*

Implementation of STIX Weakness.

Parameters

- **cwe_id** (*optional*) – Common Weakness Enumeration value as a string
- **description** (*optional*) – A string description.

STIX Extensions

Modules located in the stix.extensions package

Version: 1.1.1.12

stix.extensions.identity.ciq_identity_3_0 Module

Classes

class stix.extensions.identity.ciq_identity_3_0.**CIQIdentity3_0Instance**(*roles=None*, *specification=None*)
Bases: *stix.common.identity.Identity*

class stix.extensions.identity.ciq_identity_3_0.**STIXCIQIdentity3_0**(*party_name=None*, *languages=None*, *addresses=None*, *organisation_info=None*, *electronic_address_identifiers=None*, *free_text_lines=None*, *contact_numbers=None*, *nationalities=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**Address**(*free_text_address=None*, *country=None*, *administrative_area=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**AdministrativeArea**(*name_elements=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**_BaseNameElement**(*value=None*)
Bases: *stix.base.Entity*

Do not instantiate directly: use PersonNameElement or OrganisationNameElement

class stix.extensions.identity.ciq_identity_3_0.**ContactNumber**(*contact_number_elements=None*, *communication_media_type=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**ContactNumberElement**(*value=None*, *type_=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**Country**(*name_elements=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**ElectronicAddressIdentifier**(*value=None*, *type_=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**FreeTextAddress**(*address_lines=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**FreeTextLine**(*value=None*, *type_=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**Language**(*value=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**NameElement**(*value=None*, *name_type=None*, *name_code=None*, *name_code_type=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**NameLine**(*value=None*, *type_=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**OrganisationInfo**(*industry_type=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**OrganisationName**(*name_elements=None*, *subdivision_names=None*, *type_=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**OrganisationNameElement**(*value=None*, *element_type=None*)
Bases: *stix.extensions.identity.ciq_identity_3_0._BaseNameElement*

class stix.extensions.identity.ciq_identity_3_0.**PartyName**(*name_lines=None*, *person_names=None*, *organisation_names=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**PersonName**(*name_elements=None*)
Bases: *stix.base.Entity*

class stix.extensions.identity.ciq_identity_3_0.**PersonNameElement**(*value=None*, *element_type=None*)
Bases: *stix.extensions.identity.ciq_identity_3_0._BaseNameElement*

class stix.extensions.identity.ciq_identity_3_0.**SubDivisionName**(*value=None*, *type_=None*)
Bases: *stix.base.Entity*

Constants

stix.extensions.identity.ciq_identity_3_0.**XML_NS_XPIL = 'urn:oasis:names:tc:ciq:xpil:3'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

stix.extensions.identity.ciq_identity_3_0.**XML_NS_XNL = 'urn:oasis:names:tc:ciq:xnl:3'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

stix.extensions.identity.ciq_identity_3_0.**XML_NS_XAL = 'urn:oasis:names:tc:ciq:xal:3'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

stix.extensions.identity.ciq_identity_3_0.**XML_NS_STIX_EXT = 'http://stix.mitre.org/extensions/Identity#CI**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

Version: 1.1.1.12

stix.extensions.malware.maec_4_1_malware Module

Classes

class stix.extensions.malware.maec_4_1_malware.**MAECInstance**(*maec=None*)
Bases: *stix.ttp.malware_instance.MalwareInstance*

The MAECInstance object provides an extension to the MalwareInstanceType which imports and leverages the MAEC 4.1 schema for structured characterization of Malware.

This class extension is automatically registered by the MalwareInstanceFactory.

> **Warning:** Interacting with the `maec` field will fail if the maec library is not installed in your Python environment.

Version: 1.1.1.12

stix.extensions.marking.ais Module

STIX Extension for AIS Data Markings

Unlike the other marking extensions, the AIS marking extension is not loaded automatically, since AIS markings are not a part of the published STIX 1.x specifications. They are included in python-stix because they're common enough that it is not worth creating a separate package.

If you are writing code that needs to parse AIS markings, make sure that your program imports this module before beginning to parse any STIX documents:

```
import stix.extensions.marking.ais
```

Classes

class stix.extensions.marking.ais.**AISMarkingStructure**(*is_proprietary=None*, *not_proprietary=None*)

Bases: *stix.data_marking.MarkingStructure*

Functions

stix.extensions.marking.ais.**add_ais_marking**(*stix_package*, *proprietary*, *consent*, *color*, ***kwargs*)

This utility functions aids in the creation of an AIS marking and appends it to the provided STIX package.

Parameters

- **stix_package** – A stix.core.STIXPackage object.
- **proprietary** – True if marking uses IsProprietary, False for NotProprietary.
- **consent** – A string with one of the following values: "EVERYONE", "NONE" or "USG".
- **color** – A string that corresponds to TLP values: "WHITE", "GREEN" or "AMBER".
- ****kwargs** – Six required keyword arguments that are used to create a CIQ identity object. These are: country_name_code, country_name_code_type, admin_area_name_code, admin_area_name_code_type, organisation_name, industry_type.

Raises ValueError – When keyword arguments are missing. User did not supply correct values for: proprietary, color and consent.

Note: The following line is required to register the AIS extension:

```
>>> import stix.extensions.marking.ais
```

Any Markings under STIX Header will be removed. Please follow the guidelines for AIS.

The industry_type keyword argument accepts: a list of string based on defined sectors, a pipe-delimited string of sectors, or a single sector.

Examples

Applying AIS Markings

The STIX specification allows data markings to be applied to any combination of attributes and elements that can be described by XPath. That being said, the Automated Indicator Sharing (AIS) capability requires those markings controlled structure to select all nodes and attributes `//node() | //@*`. All required fields to create a valid AIS Markings are provided through the `add_ais_marking` function.

```
# python-stix imports
import stix
from stix.core import STIXPackage
from stix.extensions.marking.ais import (add_ais_marking,
                                         COMMUNICATIONS_SECTOR,
                                         INFORMATION_TECHNOLOGY_SECTOR)
from stix.indicator import Indicator

# Create new STIX Package
stix_package = STIXPackage()

# Create new Indicator
indicator = Indicator(title='My Indicator Example',
                      description='Example using AIS')

# Add indicator to our STIX Package
stix_package.add_indicator(indicator)

# Create AIS Marking with CIQ Identity and attach it to STIX Header.
add_ais_marking(stix_package, False, 'EVERYONE', 'GREEN',
    country_name_code='US',
    country_name_code_type='ISO 3166-1 alpha-2',
    admin_area_name_code='US-VA',
    admin_area_name_code_type='ISO 3166-2',
    organisation_name='Example Corporation',
    industry_type=[INFORMATION_TECHNOLOGY_SECTOR, COMMUNICATIONS_SECTOR]
)

# Print the XML.
print stix_package.to_xml()

# Print the JSON.
print stix_package.to_json()
```

This corresponds to the XML result:

```
<stix:STIX_Package
    xmlns:AIS="http://www.us-cert.gov/STIXMarkingStructure#AISConsentMarking-2"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xpil="urn:oasis:names:tc:ciq:xpil:3"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xal="urn:oasis:names:tc:ciq:xal:3"
    xmlns:xnl="urn:oasis:names:tc:ciq:xnl:3"
    xmlns:stix="http://stix.mitre.org/stix-1"
    xmlns:indicator="http://stix.mitre.org/Indicator-2"
    xmlns:marking="http://data-marking.mitre.org/Marking-1"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:stixCommon="http://stix.mitre.org/common-1"
    xmlns:example="http://example.com"
    xmlns:stix-ciqidentity="http://stix.mitre.org/extensions/Identity#CIQIdentity3.0-1"
    xmlns:xlink="http://www.w3.org/1999/xlink"
     id="example:Package-73ac199c-9dd8-4d8d-a37e-8ac40fc65ccf" version="1.1.1">
    <stix:STIX_Header>
        <stix:Handling>
            <marking:Marking>
                <marking:Controlled_Structure>//node() | //@*</marking:Controlled_Structure>
                <marking:Marking_Structure xsi:type='AIS:AISMarkingStructure'>
                    <AIS:Not_Proprietary CISA_Proprietary="false">
```

```
                        <AIS:AISConsent consent="EVERYONE"/>
                        <AIS:TLPMarking color="GREEN"/>
                    </AIS:Not_Proprietary>
                </marking:Marking_Structure>
                <marking:Information_Source>
                    <stixCommon:Identity xsi:type="stix-ciqidentity:CIQIdentity3.0InstanceType">
                        <stix-ciqidentity:Specification xmlns:stix-ciqidentity="http://stix.mitre.org
                            <xpil:PartyName xmlns:xpil="urn:oasis:names:tc:ciq:xpil:3">
                                <xnl:OrganisationName xmlns:xnl="urn:oasis:names:tc:ciq:xnl:3">
                                    <xnl:NameElement>Example Corporation</xnl:NameElement>
                                </xnl:OrganisationName>
                            </xpil:PartyName>
                            <xpil:Addresses xmlns:xpil="urn:oasis:names:tc:ciq:xpil:3">
                                <xpil:Address>
                                    <xal:Country xmlns:xal="urn:oasis:names:tc:ciq:xal:3">
                                        <xal:NameElement xal:NameCode="US" xal:NameCodeType="ISO 3166
                                    </xal:Country>
                                    <xal:AdministrativeArea xmlns:xal="urn:oasis:names:tc:ciq:xal:3">
                                        <xal:NameElement xal:NameCode="US-VA" xal:NameCodeType="ISO 3
                                    </xal:AdministrativeArea>
                                </xpil:Address>
                            </xpil:Addresses>
                            <xpil:OrganisationInfo xmlns:xpil="urn:oasis:names:tc:ciq:xpil:3" xpil:In
                        </stix-ciqidentity:Specification>
                    </stixCommon:Identity>
                </marking:Information_Source>
            </marking:Marking>
        </stix:Handling>
    </stix:STIX_Header>
    <stix:Indicators>
        <stix:Indicator id="example:indicator-eab71e49-e982-4874-a057-e75e51a76009" timestamp="2017-0
            <indicator:Title>My Indicator Example</indicator:Title>
            <indicator:Description>Example using AIS</indicator:Description>
        </stix:Indicator>
    </stix:Indicators>
</stix:STIX_Package>
```

The following corresponds to the JSON result:

```
{
    "stix_header": {
        "handling": [
            {
                "controlled_structure": "//node() | //@*",
                "information_source": {
                    "identity": {
                        "xsi:type": "stix-ciqidentity:CIQIdentity3.0InstanceType",
                        "specification": {
                            "organisation_info": {
                                "industry_type": "Information Technology Sector|Communications Secto
                            },
                            "party_name": {
                                "organisation_names": [
                                    {
                                        "name_elements": [
                                            {
                                                "value": "Example Corporation"
                                            }
```

```
                                        ]
                                    }
                                ]
                            },
                            "addresses": [
                                {
                                    "country": {
                                        "name_elements": [
                                            {
                                                "name_code_type": "ISO 3166-1 alpha-2",
                                                "name_code": "US"
                                            }
                                        ]
                                    },
                                    "administrative_area": {
                                        "name_elements": [
                                            {
                                                "name_code_type": "ISO 3166-2",
                                                "name_code": "US-VA"
                                            }
                                        ]
                                    }
                                }
                            ]
                        }
                    }
                },
                "marking_structures": [
                    {
                        "xsi:type": "AIS:AISMarkingStructure",
                        "not_proprietary": {
                            "tlp_marking": {
                                "color": "GREEN"
                            },
                            "ais_consent": {
                                "consent": "EVERYONE"
                            },
                            "cisa_proprietary": "false"
                        }
                    }
                ]
            }
        ]
    },
    "version": "1.1.1",
    "indicators": [
        {
            "description": "Example using AIS",
            "title": "My Indicator Example",
            "timestamp": "2017-10-02T14:26:57.510000+00:00",
            "id": "example:indicator-81466b8d-4efb-460f-ba13-b072420b9540"
        }
    ],
    "id": "example:Package-a8c8135d-18d8-4384-903f-71285a02346e"
}
```

Parsing AIS Markings

Using the same example used for Applying AIS Markings. This would be how a consumer of AIS would parse the data.

```
# python-stix imports
import stix
from stix.core import STIXPackage
import stix.extensions.marking.ais  # Register the AIS markings

# Parse STIX Package
stix_package = STIXPackage.from_xml("stix_input.xml")
# stix_package = STIXPackage.from_json("stix_input.json")

# Print all indicators
for indicator in stix_package.indicators:
    print(indicator)

# Extract markings from STIX Header
markings = stix_package.stix_header.handling

# Print all markings contained in the STIX Header
for marking in markings:
    print(marking)
    print(marking.marking_structures)
    print("----------MARKING CONTENT----------")
    ais_struct = marking.marking_structures[0]
    print("OBJ: %s" % ais_struct)
    print("NotProprietary OBJ: %s" % ais_struct.not_proprietary)
    print("CISA_Proprietary: %s" % ais_struct.not_proprietary.cisa_proprietary)
    print("Consent: %s" % ais_struct.not_proprietary.ais_consent.consent)
    print("TLP color: %s" % ais_struct.not_proprietary.tlp_marking.color)

    print("----------INFORMATION SOURCE----------")
    identity = marking.information_source.identity.specification
    print("OBJ: %s" % identity)
    print("Organization Name: %s" % identity.party_name.organisation_names[0].name_elements[0].value)
    print("Country: %s" % identity.addresses[0].country.name_elements[0].name_code)
    print("Country code type: %s" % identity.addresses[0].country.name_elements[0].name_code_type)
    print("Administrative area: %s" % identity.addresses[0].administrative_area.name_elements[0].name
    print("Administrative area code type: %s" % identity.addresses[0].administrative_area.name_elemen
    print("Industry Type: %s" % identity.organisation_info.industry_type)


>>> <stix.indicator.indicator.Indicator object at 0x...>
>>> <stix.data_marking.MarkingSpecification object at 0x...>
>>> [<stix.extensions.marking.ais.AISMarkingStructure object at 0x...>, ...]
>>> ----------MARKING CONTENT----------
>>> OBJ: <stix.extensions.marking.ais.AISMarkingStructure object at 0x...>
>>> NotProprietary OBJ: <stix.extensions.marking.ais.NotProprietary object at 0x...>
>>> CISA_Proprietary: False
>>> Consent: EVERYONE
>>> TLP color: GREEN
>>> ----------INFORMATION SOURCE----------
>>> OBJ: <stix.extensions.identity.ciq_identity_3_0.STIXCIQIdentity3_0 object at 0x...>
>>> Organization Name: Example Corporation
>>> Country: US
>>> Country code type: ISO 3166-1 alpha-2
```

```
>>> Administrative area: US-VA
>>> Administrative area code type: ISO 3166-2
>>> Industry Type: Information Technology Sector|Communications Sector
```

Constants

The following constants can be used for the `industry_type` keyword argument to `add_ais_marking`:

`stix.extensions.marking.ais.`**`CHEMICAL_SECTOR = 'Chemical Sector'`**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`stix.extensions.marking.ais.`**`COMMERCIAL_FACILITIES_SECTOR = 'Commercial Facilities Sector'`**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`stix.extensions.marking.ais.`**`COMMUNICATIONS_SECTOR = 'Communications Sector'`**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`stix.extensions.marking.ais.`**`CRITICAL_MANUFACTURING_SECTOR = 'Critical Manufacturing Sector'`**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`stix.extensions.marking.ais.`**`DAMS_SECTOR = 'Dams Sector'`**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`stix.extensions.marking.ais.`**`DEFENSE_INDUSTRIAL_BASE_SECTOR = 'Defense Industrial Base Sector'`**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`stix.extensions.marking.ais.`**`EMERGENCY_SERVICES_SECTOR = 'Emergency Services Sector'`**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`stix.extensions.marking.ais.`**`ENERGY_SECTOR = 'Energy Sector'`**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`stix.extensions.marking.ais.`**`FINANCIAL_SERVICES_SECTOR = 'Financial Services Sector'`**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`stix.extensions.marking.ais.`**`FOOD_AND_AGRICULTURE_SECTOR = 'Food and Agriculture Sector'`**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`stix.extensions.marking.ais.`**`GOVERNMENT_FACILITIES_SECTOR = 'Government Facilities Sector'`**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

stix.extensions.marking.ais.**HEALTH_CARE_AND_PUBLIC_HEALTH_SECTOR = 'Healthcare and Public Health Sec**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

stix.extensions.marking.ais.**INFORMATION_TECHNOLOGY_SECTOR = 'Information Technology Sector'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

stix.extensions.marking.ais.**NUCLEAR_REACTORS_MATERIALS_AND_WASTE_SECTOR = 'Nuclear Reactors, Mate**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

stix.extensions.marking.ais.**OTHER = 'Other'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

stix.extensions.marking.ais.**TRANSPORTATION_SYSTEMS_SECTOR = 'Transportation Systems Sector'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

stix.extensions.marking.ais.**WATER_AND_WASTEWATER_SYSTEMS_SECTOR = 'Water and Wastewater Systems Sec**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

Version: 1.1.1.12

stix.extensions.marking.simple_marking Module

Classes

class stix.extensions.marking.simple_marking.**SimpleMarkingStructure**(*statement=None*)
Bases: *stix.data_marking.MarkingStructure*

Version: 1.1.1.12

stix.extensions.marking.terms_of_use_marking Module

Classes

class stix.extensions.marking.terms_of_use_marking.**TermsOfUseMarkingStructure**(*terms_of_use=None*)
Bases: *stix.data_marking.MarkingStructure*

Version: 1.1.1.12

stix.extensions.marking.tlp Module

Classes

class stix.extensions.marking.tlp.**TLPMarkingStructure**(*color=None*)
Bases: *stix.data_marking.MarkingStructure*

Version: 1.1.1.12

stix.extensions.test_mechanism.generic_test_mechanism Module

Classes

class stix.extensions.test_mechanism.generic_test_mechanism.**GenericTestMechanism**(*id_=None*, *idref=None*)

Bases: *stix.indicator.test_mechanism._BaseTestMechanism*

Version: 1.1.1.12

stix.extensions.test_mechanism.open_ioc_2010_test_mechanism Module

Classes

class stix.extensions.test_mechanism.open_ioc_2010_test_mechanism.**OpenIOCTestMechanism**(*id_=Non*
idref=N

Bases: *stix.indicator.test_mechanism._BaseTestMechanism*

Version: 1.1.1.12

stix.extensions.test_mechanism.snort_test_mechanism Module

Classes

class stix.extensions.test_mechanism.snort_test_mechanism.**SnortTestMechanism**(*id_=None*, *idref=None*)

Bases: *stix.indicator.test_mechanism._BaseTestMechanism*

Version: 1.1.1.12

stix.extensions.test_mechanism.yara_test_mechanism Module

Classes

class stix.extensions.test_mechanism.yara_test_mechanism.**YaraTestMechanism**(*id_=None*, *idref=None*)

Bases: *stix.indicator.test_mechanism._BaseTestMechanism*

STIX Incident

Modules located in the stix.incident package

Version: 1.1.1.12

stix.incident Module

Classes

class stix.incident.**Incident**(*id_=None*, *idref=None*, *timestamp=None*, *title=None*, *description=None*, *short_description=None*)

Bases: stix.base.BaseCoreComponent

Implementation of the STIX Incident.

Parameters

- **id** (*optional*) – An identifier. If `None`, a value will be generated via `mixbox.idgen.create_id()`. If set, this will unset the `idref` property.
- **idref** (*optional*) – An identifier reference. If set this will unset the `id_` property.
- **timestamp** (*optional*) – A timestamp value. Can be an instance of `datetime.datetime` or `str`.
- **description** – A description of the purpose or intent of this object.
- **short_description** – A short description of the intent or purpose of this object.
- **title** – The title of this object.

add_affected_asset(*v*)
Adds a *AffectedAsset* object to the `affected_assets` collection.

add_category(*category*)
Adds a *VocabString* object to the `categories` collection.

If *category* is a string, an attempt will be made to convert it into an instance of `IncidentCategory`.

add_coa_requested(*value*)
Adds a `COARequested` object to the `coas_requested` collection.

add_coa_taken(*value*)
Adds a *COATaken* object to the `coas_taken` collection.

add_coordinator(*value*)
Adds a *InformationSource* object to the `coordinators` collection.

add_discovery_method(*value*)
Adds a *VocabString* object to the `discovery_methods` collection.

If *value* is a string, an attempt will be made to convert it to an instance of `DiscoveryMethod`.

add_external_id(*value*)
Adds a *ExternalID* object to the `external_ids` collection.

add_intended_effect(*value*)
Adds a *Statement* object to the `intended_effects` collection.

If *value* is a string, an attempt will be made to convert it into an instance of *Statement*.

add_leveraged_ttps(*ttp*)
Adds a *RelatedTTP* value to the `leveraged_ttps` collection.

add_related_indicator(*value*)
Adds an Related Indicator to the `related_indicators` list property of this *Incident*.

The *indicator* parameter must be an instance of *RelatedIndicator* or `Indicator`.

If the *indicator* parameter is `None`, no item will be added to the `related_indicators` list property.

Calling this method is the same as calling `append()` on the `related_indicators` property.

See also:

The *RelatedIndicators* documentation.

Note: If the *indicator* parameter is not an instance of *RelatedIndicator* an attempt will be made to convert it to one.

Parameters **value** – An instance of Indicator or *RelatedIndicator*.

Raises ValueError – If the *indicator* parameter cannot be converted into an instance of *RelatedIndicator*

add_related_observable(*value*)

Adds a Related Observable to the related_observables list property of this *Incident*.

The *observable* parameter must be an instance of *RelatedObservable* or Observable.

If the *observable* parameter is None, no item will be added to the related_observables list property.

Calling this method is the same as calling append() on the related_observables property.

See also:

The *RelatedObservables* documentation.

Note: If the *observable* parameter is not an instance of *RelatedObservable* an attempt will be made to convert it to one.

Parameters **observable** – An instance of Observable or *RelatedObservable*.

Raises ValueError – If the *value* parameter cannot be converted into an instance of *RelatedObservable*

add_responder(*value*)

Adds a *InformationSource* object to the responders collection.

add_victim(*victim*)

Adds a IdentityType value to the victims collection.

class stix.incident.**AttributedThreatActors**(*scope=None*, **args*)

Bases: *stix.common.related.GenericRelationshipList*

class stix.incident.**LeveragedTTPs**(*scope=None*, **args*)

Bases: *stix.common.related.GenericRelationshipList*

class stix.incident.**RelatedIndicators**(*scope=None*, **args*)

Bases: *stix.common.related.GenericRelationshipList*

class stix.incident.**RelatedObservables**(*scope=None*, **args*)

Bases: *stix.common.related.GenericRelationshipList*

class stix.incident.**RelatedIncidents**(*scope=None*, **args*)

Bases: *stix.common.related.GenericRelationshipList*

Version: 1.1.1.12

stix.incident.affected_asset Module

Classes

class stix.incident.affected_asset.**AffectedAsset**
Bases: *stix.base.Entity*

class stix.incident.affected_asset.**AssetType**(*value=None*, *count_affected=None*)
Bases: *stix.common.vocabs.VocabString*

is_plain()
Override VocabString.is_plain()

Version: 1.1.1.12

stix.incident.coa Module

Classes

class stix.incident.coa.**COATaken**(*course_of_action=None*)
Bases: *stix.base.Entity*

class stix.incident.coa.**COATime**(*start=None*, *end=None*)
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.incident.contributors Module

Classes

class stix.incident.contributors.**Contributors**(**args*)
Bases: *stix.base.EntityList*

Version: 1.1.1.12

stix.incident.direct_impact_summary Module

Classes

class stix.incident.direct_impact_summary.**DirectImpactSummary**
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.incident.external_id Module

Classes

class stix.incident.external_id.**ExternalID**(*value=None*, *source=None*)
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.incident.history Module

Classes

class stix.incident.history.**History**(**args*)
Bases: *stix.base.EntityList*

class stix.incident.history.**HistoryItem**
Bases: *stix.base.Entity*

class stix.incident.history.**JournalEntry**(*value=None*)
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.incident.impact_assessment Module

Classes

class stix.incident.impact_assessment.**ImpactAssessment**
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.incident.indirect_impact_summary Module

Classes

class stix.incident.indirect_impact_summary.**IndirectImpactSummary**
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.incident.loss_estimation Module

Classes

class stix.incident.loss_estimation.**LossEstimation**
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.incident.property_affected Module

Classes

class stix.incident.property_affected.**PropertyAffected**
Bases: *stix.base.Entity*

class stix.incident.property_affected.**NonPublicDataCompromised**(*value=None*, *data_encrypted=None*)
Bases: *stix.common.vocabs.VocabString*

Version: 1.1.1.12

stix.incident.time Module

Classes

class stix.incident.time.**Time**(*first_malicious_action=None*, *initial_compromise=None*, *first_data_exfiltration=None*, *incident_discovery=None*, *incident_opened=None*, *containment_achieved=None*, *restoration_achieved=None*, *incident_reported=None*, *incident_closed=None*)

Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.incident.total_loss_estimation Module

Classes

class stix.incident.total_loss_estimation.**TotalLossEstimation**

Bases: *stix.base.Entity*

STIX Indicator

Modules located in the stix.indicator package

Version: 1.1.1.12

stix.indicator.indicator Module

Overview

The *stix.indicator.indicator* module implements IndicatorType STIX Language construct. The IndicatorType characterizes a cyber threat indicator made up of a pattern identifying certain observable conditions as well as contextual information about the patterns meaning, how and when it should be acted on, etc.

Documentation Resources

- Indicator Data Model
- Indicator Idioms

Classes

class stix.indicator.indicator.**Indicator**(*id_=None*, *idref=None*, *timestamp=None*, *title=None*, *description=None*, *short_description=None*)

Bases: stix.base.BaseCoreComponent

Implementation of the STIX Indicator.

Parameters

- **id** (*optional*) – An identifier. If None, a value will be generated via mixbox.idgen.create_id(). If set, this will unset the idref property.

- **idref** (*optional*) – An identifier reference. If set this will unset the `id_` property.
- **title** (*optional*) – A string title.
- **timestamp** (*optional*) – A timestamp value. Can be an instance of `datetime.datetime` or `str`.
- **description** (*optional*) – A string description.
- **short_description** (*optional*) – A string short description.

add_alternative_id(*value*)
Adds an alternative id to the `alternative_id` list property.

Note: If `None` is passed in no value is added to the `alternative_id` list property.

Parameters **value** – An identifier value.

add_indicated_ttp(*v*)
Adds an Indicated TTP to the `indicated_ttps` list property of this *Indicator*.

The *v* parameter must be an instance of *stix.common.related.RelatedTTP* or *stix.ttp.TTP*.

If the *v* parameter is `None`, no item wil be added to the `indicated_ttps` list property.

Note: If the *v* parameter is not an instance of *stix.common.related.RelatedTTP* an attempt will be made to convert it to one.

Parameters **v** – An instance of *stix.common.related.RelatedTTP* or *stix.ttp.TTP*.

Raises `ValueError` – If the *v* parameter cannot be converted into an instance of *stix.common.related.RelatedTTP*

add_indicator_type(*value*)
Adds a value to the `indicator_types` list property.

The *value* parameter can be a `str` or an instance of *stix.common.vocabs.VocabString*.

Note: If the *value* parameter is a `str` instance, an attempt will be made to convert it into an instance of *stix.common.vocabs.IndicatorType*

Parameters **value** – An instance of *stix.common.vocabs.VocabString* or `str`.

Raises `ValueError` – If the *value* param is a `str` instance that cannot be converted into an instance of *stix.common.vocabs.IndicatorType*.

add_kill_chain_phase(*value*)
Add a new Kill Chain Phase reference to this Indicator.

Parameters **value** – a *stix.common.kill_chains.KillChainPhase* or a *str* representing the phase_id of. Note that you if you are defining a custom Kill Chain, you need to add it to the STIX package separately.

add_object(*object_*)
Adds a python-cybox Object instance to the `observables` list property.

This is the same as calling `indicator.add_observable(object_)`.

Note: If the *object* param is not an instance of `cybox.core.Object` an attempt will be made to to convert it into one before wrapping it in an `cybox.core.Observable` layer.

Parameters **object** – An instance of `cybox.core.Object` or an object that can be converted into an instance of `cybox.core.Observable`

Raises `ValueError` – if the *object_* param cannot be converted to an instance of `cybox.core.Observable`.

add_observable(*observable*)
Adds an observable to the `observable` property of the *Indicator*.

If the *observable* parameter is `None`, no item will be added to the `observable` property.

Note: The STIX Language dictates that an *Indicator* can have only one `Observable` under it. Because of this, when a user adds another `Observable` a new, empty `Observable` will be crated and append the existing and new `observable` using the `ObservableComposition` property. To access the top level `Observable` can be achieved by the `observable` property .By default, the `operator` of the composition layer will be set to `"OR"`. The `operator` value can be changed via the `observable_composition_operator` property.

Setting `observable` or `observables` with re-initialize the property and lose all `Observable` in the composition layer.

Parameters **observable** – An instance of `cybox.core.Observable` or an object type that can be converted into one.

Raises `ValueError` – If the *observable* param cannot be converted into an instance of `cybox.core.Observable`.

add_related_campaign(*value*)
Adds a Related Campaign to this Indicator.

The *value* parameter must be an instance of `RelatedCampaignRef` or `CampaignRef`.

If the *value* parameter is `None`, no item wil be added to the `related_campaigns` collection.

Calling this method is the same as calling `append()` on the `related_campaigns` property.

See also:

The `RelatedCampaignRef` documentation.

Note: If the *value* parameter is not an instance of `RelatedCampaignRef` an attempt will be made to convert it to one.

Parameters **value** – An instance of `RelatedCampaignRef` or *Campaign*.

Raises `ValueError` – If the *value* parameter cannot be converted into an instance of `RelatedCampaignRef`

add_related_indicator(*indicator*)
Adds an Related Indicator to the `related_indicators` list property of this *Indicator*.

The *indicator* parameter must be an instance of *stix.common.related.RelatedIndicator* or *Indicator*.

If the *indicator* parameter is `None`, no item wil be added to the `related_indicators` list property.

Calling this method is the same as calling `append()` on the `related_indicators` proeprty.

See also:

The *RelatedIndicators* documentation.

Note: If the *tm* parameter is not an instance of *stix.common.related.RelatedIndicator* an attempt will be made to convert it to one.

Parameters **indicator** – An instance of *Indicator* or *stix.common.related.RelatedIndicator*.

Raises `ValueError` – If the *indicator* parameter cannot be converted into an instance of *stix.common.related.RelatedIndicator*

add_test_mechanism(*tm*)
Adds an Test Mechanism to the `test_mechanisms` list property of this *Indicator*.

The *tm* parameter must be an instance of a *stix.indicator.test_mechanism._BaseTestMechanism* implementation.

If the *tm* parameter is `None`, no item will be added to the `test_mechanisms` list property.

See also:

Test Mechanism implementations are found under the `stix.extensions.test_mechanism` package.

Parameters **tm** – An instance of a *stix.indicator.test_mechanism._BaseTestMechanism* implementation.

Raises `ValueError` – If the *tm* parameter is not an instance of *stix.indicator.test_mechanism._BaseTestMechanism*

add_valid_time_position(*value*)
Adds an valid time position to the `valid_time_positions` property list.

If *value* is `None`, no item is added to the `value_time_positions` list.

Parameters **value** – An instance of *stix.indicator.valid_time.ValidTime*.

Raises `ValueError` – If the *value* argument is not an instance of *stix.indicator.valid_time.ValidTime*.

get_produced_time()
Gets the produced time for this *Indicator*.

This is the same as calling `produced_time = indicator.producer.time.produced_time`.

Returns `None` or an instance of `cybox.common.DateTimeWithPrecision`.

get_received_time()
Gets the received time for this *Indicator*.

This is the same as calling `received_time = indicator.producer.time.received_time`.

Returns `None` or an instance of `cybox.common.DateTimeWithPrecision`.

observables
A list of `cybox.core.Observable` instances. This can be set to a single object instance or a list of objects.

Note: If only one Observable is set, this property will return a list with the `observable` property.

If multiple `cybox.core.Observable` this property will return Observables under the `cybox.core.ObservableComposition`.

Access to the top level `cybox.core.Observable` is made via `observable` property.

Default Value: Empty `list`.

Returns A `list` of `cybox.core.Observable` instances.

set_produced_time(*produced_time*)
Sets the `produced_time` property of the `producer` property instance fo *produced_time*.

This is the same as calling `indicator.producer.time.produced_time = produced_time`.

The *produced_time* parameter must be an instance of `str`, `datetime.datetime`, or `cybox.common.DateTimeWithPrecision`.

Note: If *produced_time* is a `str` or `datetime.datetime` instance an attempt will be made to convert it into an instance of `cybox.common.DateTimeWithPrecision`.

Parameters **produced_time** – An instance of `str`, `datetime.datetime`, or `cybox.common.DateTimeWithPrecision`.

set_producer_identity(*identity*)
Sets the name of the producer of this indicator.

This is the same as calling `indicator.producer.identity.name = identity`.

If the `producer` property is `None`, it will be initialized to an instance of *stix.common.information_source.InformationSource*.

If the `identity` property of the `producer` instance is `None`, it will be initialized to an instance of *stix.common.identity.Identity*.

Note: if the *identity* parameter is not an instance *stix.common.identity.Identity* an attempt will be made to convert it to one.

Parameters **identity** – An instance of `str` or `stix.common.identity.Identity`.

set_received_time(*received_time*)
Sets the received time for this *Indicator*.

This is the same as calling `indicator.producer.time.produced_time = produced_time`.

The *received_time* parameter must be an instance of `str`, `datetime.datetime`, or `cybox.common.DateTimeWithPrecision`.

Parameters **received_time** – An instance of `str`, `datetime.datetime`, or `cybox.common.DateTimeWithPrecision`.

Note: If *received_time* is a `str` or `datetime.datetime` instance an attempt will be made to convert it into an instance of `cybox.common.DateTimeWithPrecision`.

---

class `stix.indicator.indicator.`**CompositeIndicatorExpression**(*operator='OR', *args*)
Bases: `mixbox.entities.EntityList`

Implementation of the STIX `CompositeIndicatorExpressionType`.

The `CompositeIndicatorExpression` class implements methods found on `collections.MutableSequence` and as such can be interacted with as a `list` (e.g., `append()`).

Note: The `append()` method can only accept instances of *Indicator*.

Examples

Add a *Indicator* instance to an instance of *CompositeIndicatorExpression*:

```
>>> i = Indicator()
>>> comp = CompositeIndicatorExpression()
>>> comp.append(i)
```

Create a *CompositeIndicatorExpression* from a list of *Indicator* instances using `*args` argument list:

```
>>> list_indicators = [Indicator() for i in xrange(10)]
>>> comp = CompositeIndicatorExpression(CompositeIndicatorExpression.OP_OR, *list_indicators)
>>> len(comp)
10
```

Parameters

- **operator** (*str, optional*) – The logical composition operator. Must be `"AND"` or `"OR"`.
- ***args** – Variable length argument list of *Indicator* instances.

OP_AND
str

String `"AND"`

OP_OR
str

String `"OR"`

OPERATORS
tuple

Tuple of allowed `operator` values.

operator
str

The logical composition operator. Must be `"AND"` or `"OR"`.

class `stix.indicator.indicator.`**RelatedIndicators**(*related_indicators=None*, *scope=None*)
Bases: *stix.common.related.GenericRelationshipList*

The `RelatedIndicators` class provides functionality for adding *stix.common.related.RelatedIndicator* instances to an *Indicator* instance.

The `RelatedIndicators` class implements methods found on `collections.MutableSequence` and as such can be interacted with as a `list` (e.g., `append()`).

The `append()` method can accept instances of *stix.common.related.RelatedIndicator* or *Indicator* as an argument.

Note: Calling `append()` with an instance of *stix.coa.CourseOfAction* will wrap that instance in a *stix.common.related.RelatedIndicator* layer, with `item` set to the *Indicator* instance.

Examples

Append an instance of *Indicator* to the `Indicator.related_indicators` property. The instance of *Indicator* will be wrapped in an instance of *stix.common.related.RelatedIndicator*:

```
>>> related = Indicator()
>>> parent_indicator = Indicator()
>>> parent_indicator.related_indicators.append(related)
>>> print(type(indicator.related_indicators[0]))
<class 'stix.common.related.RelatedIndicator'>
```

Iterate over the `related_indicators` property of an *Indicator* instance and print the ids of each underlying `Indicator` instance:

```
>>> for related in indicator.related_indicators:
>>>     print(related.item.id_)
```

Parameters

- **related_indicators** (*list, optional*) – A list of *Indicator* or *stix.common.related.RelatedIndicator* instances.
- **scope** (*str, optional*) – The scope of the items. Can be set to `"inclusive"` or `"exclusive"`. See *stix.common.related.GenericRelationshipList* documentation for more information.

scope
str

The scope of the items. Can be set to `"inclusive"` or `"exclusive"`. See *stix.common.related.GenericRelationshipList* documentation for more information.

class stix.indicator.indicator.**SuggestedCOAs**(*suggested_coas=None*, *scope=None*)
Bases: *stix.common.related.GenericRelationshipList*

The SuggestedCOAs class provides functionality for adding *stix.common.related.RelatedCOA* instances to an *Indicator* instance.

The SuggestedCOAs class implements methods found on collections.MutableSequence and as such can be interacted with as a list (e.g., append()).

The append() method can accept instances of *stix.common.related.RelatedCOA* or *stix.coa.CourseOfAction* as an argument.

Note: Calling append() with an instance of *stix.coa.CourseOfAction* will wrap that instance in a *stix.common.related.RelatedCOA* layer, with the item set to the *stix.coa.CourseOfAction* instance.

Examples

Append an instance of *stix.coa.CourseOfAction* to the Indicator.suggested_coas property. The instance of *stix.coa.CourseOfAction* will be wrapped in an instance of *stix.common.related.RelatedCOA*.

```
>>> coa = CourseOfAction()
>>> indicator = Indicator()
>>> indicator.suggested_coas.append(coa)
>>> print(type(indicator.suggested_coas[0]))
<class 'stix.common.related.RelatedCOA'>
```

Iterate over the suggested_coas property of an *Indicator* instance and print the ids of each underlying *stix.coa.CourseOfAction* instance.

```
>>> for related_coa in indicator.suggested_coas:
>>>     print(related_coa.item.id_)
```

Parameters

- **suggested_coas** (*list*) – A list of *stix.coa.CourseOfAction* or *stix.common.related.RelatedCOA* instances.
- **scope** (*str*) – The scope of the items. Can be set to "inclusive" or "exclusive". See *stix.common.related.GenericRelationshipList* documentation for more information.

scope
str

The scope of the items. Can be set to "inclusive" or "exclusive". See *stix.common.related.GenericRelationshipList* documentation for more information.

class stix.indicator.indicator.**IndicatorTypes**(**args*)
Bases: stix.base.TypedList

A *stix.common.vocabs.VocabString* collection which defaults to *stix.common.vocabs.IndicatorType*. This class implements methods found on collections.MutableSequence and as such can be interacted with like a list.

Note: The `append()` method can accept `str` or *stix.common.vocabs.VocabString* instances. If a `str` instance is passed in, an attempt will be made to convert it to an instance of *stix.common.vocabs.IndicatorType*.

Examples

Add an instance of *stix.common.vocabs.IndicatorType*:

```
>>> from stix.common.vocabs import IndicatorType
>>> itypes = IndicatorTypes()
>>> type_ = IndicatorType(IndicatorType.TERM_IP_WATCHLIST)
>>> itypes.append(type_)
>>> print(len(itypes))
1
```

Add a string value:

```
>>> from stix.common.vocabs import IndicatorType
>>> itypes = IndicatorTypes()
>>> type(IndicatorType.TERM_IP_WATCHLIST)
<type 'str'>
>>> itypes.append(IndicatorType.TERM_IP_WATCHLIST)
>>> print(len(itypes))
1
```

Parameters ***args** – Variable length argument list of strings or *stix.common.vocabs.VocabString* instances.

Version: 1.1.1.12

stix.indicator.sightings Module

Classes

class stix.indicator.sightings.**Sighting**(*timestamp=None*, *timestamp_precision=None*, *description=None*)
Bases: *stix.base.Entity*

class stix.indicator.sightings.**Sightings**(*sightings_count=None*, **args*)
Bases: *stix.base.EntityList*

class stix.indicator.sightings.**RelatedObservables**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

Version: 1.1.1.12

stix.indicator.test_mechanism Module

Classes

class stix.indicator.test_mechanism.**_BaseTestMechanism**(*id_=None*, *idref=None*)
Bases: *stix.base.Entity*

Functions

stix.indicator.test_mechanism.**add_extension**(*cls*)
Registers a stix.Entity class as an implementation of an xml type.

Classes must have an _XSI_TYPE class attributes to be registered. The value of this attribute must be a valid xsi:type.

Note: This was designed for internal use.

Version: 1.1.1.12

stix.indicator.valid_time Module

Classes

class stix.indicator.valid_time.**ValidTime**(*start_time=None*, *end_time=None*)
Bases: mixbox.entities.Entity

STIX Threat Actor

Modules located in the stix.threat_actor package

Version: 1.1.1.12

stix.threat_actor Module

Classes

class stix.threat_actor.**ThreatActor**(*id_=None*, *idref=None*, *timestamp=None*, *title=None*, *description=None*, *short_description=None*)
Bases: stix.base.BaseCoreComponent

Implementation of the STIX Threat Actor.

Parameters

- **id** (*optional*) – An identifier. If None, a value will be generated via mixbox.idgen.create_id(). If set, this will unset the idref property.
- **idref** (*optional*) – An identifier reference. If set this will unset the id_ property.
- **timestamp** (*optional*) – A timestamp value. Can be an instance of datetime.datetime or str.
- **description** – A description of the purpose or intent of this object.
- **short_description** – A short description of the intent or purpose of this object.
- **title** – The title of this object.

add_intended_effect(*value*)
Adds a *Statement* object to the intended_effects collection.

If *value* is a string, an attempt will be made to convert it into an instance of *Statement*.

add_motivation(*value*)
Adds a `Motivation` object to the `motivations` collection.

add_planning_and_operational_support(*value*)
Adds a *VocabString* object to the `planning_and_operational_supports` collection.

If *value* is a string, an attempt will be made to convert it to an instance of `PlanningAndOperationalSupport`.

add_sophistication(*value*)
Adds a *VocabString* object to the `sophistications` collection.

If *value* is a string, an attempt will be made to convert it to an instance of `ThreatActorSophistication`.

add_type(*value*)
Adds a *VocabString* object to the `types` collection.

If set to a string, an attempt will be made to convert it into an instance of `ThreatActorType`.

class `stix.threat_actor.`**AssociatedActors**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

class `stix.threat_actor.`**AssociatedCampaigns**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

class `stix.threat_actor.`**ObservedTTPs**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

STIX Tactics, Techniques, and Procedures (TTP)

Modules located in the stix.ttp package

Version: 1.1.1.12

`stix.ttp` Module

Classes

class `stix.ttp.`**TTP**(*id_=None*, *idref=None*, *timestamp=None*, *title=None*, *description=None*, *short_description=None*)
Bases: `stix.base.BaseCoreComponent`

Implementation of the STIX TTP.

Parameters

- **id** (*optional*) – An identifier. If `None`, a value will be generated via `mixbox.idgen.create_id()`. If set, this will unset the `idref` property.
- **idref** (*optional*) – An identifier reference. If set this will unset the `id_` property.
- **timestamp** (*optional*) – A timestamp value. Can be an instance of `datetime.datetime` or `str`.
- **description** – A description of the purpose or intent of this object.
- **short_description** – A short description of the intent or purpose of this object.
- **title** – The title of this object.

add_exploit_target(*value*)
Adds a *ExploitTarget* object to the *exploit_targets* collection.

add_intended_effect(*value*)
Adds a *Statement* object to the intended_effects collection.

If *value* is a string, an attempt will be made to convert it into an instance of *Statement*.

add_kill_chain_phase(*value*)
Adds a *KillChainPhaseReference* to the kill_chain_phases collection.

Parameters **value** – A *KillChainPhase*, *KillChainPhaseReference* or a str representing the phase_id of. Note that you if you are defining a custom Kill Chain, you need to add it to the STIX package separately.

add_related_package(*value*)
Adds a *RelatedPackageRef* object to the related_packages collection.

Parameters **value** – A *RelatedPackageRef* or a *STIXPackage* object.

add_related_ttp(*value*)
Adds an Related TTP to the *related_ttps* list property of this *TTP*.

The *TTP* parameter must be an instance of *RelatedTTP* or *TTP*.

If the *TTP* parameter is None, no item wil be added to the related_ttps list property.

Calling this method is the same as calling append() on the related_ttps property.

See also:

The RelatedTTPs documentation.

Note: If the *TTP* parameter is not an instance of *RelatedTTP* an attempt will be made to convert it to one.

Parameters **value** – An instance of *TTP* or *RelatedTTP*.

Raises ValueError – If the *TTP* parameter cannot be converted into an instance of *RelatedTTP*

Version: 1.1.1.12

stix.ttp.attack_pattern Module

Classes

class stix.ttp.attack_pattern.**AttackPattern**(*id_=None*, *idref=None*, *title=None*, *description=None*, *short_description=None*)

Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.ttp.behavior Module

Classes

class stix.ttp.behavior.**Behavior**(*malware_instances=None*, *attack_patterns=None*, *exploits=None*)
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.ttp.exploit Module

Classes

class stix.ttp.exploit.**Exploit**(*id_=None*, *idref=None*, *title=None*, *description=None*, *short_description=None*)
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.ttp.exploit_targets Module

Classes

class stix.ttp.exploit_targets.**ExploitTargets**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

Version: 1.1.1.12

stix.ttp.infrastructure Module

Classes

class stix.ttp.infrastructure.**Infrastructure**(*id_=None*, *idref=None*, *title=None*, *description=None*, *short_description=None*)
Bases: *stix.base.Entity*

Version: 1.1.1.12

stix.ttp.malware_instance Module

Classes

class stix.ttp.malware_instance.**MalwareInstance**(*id_=None*, *idref=None*, *title=None*, *description=None*, *short_description=None*)
Bases: *stix.base.Entity*

Functions

stix.ttp.malware_instance.**add_extension**(*cls*)
Registers a stix.Entity class as an implementation of an xml type.

Classes must have an `_XSI_TYPE` class attributes to be registered. The value of this attribute must be a valid xsi:type.

Note: This was designed for internal use.

Version: 1.1.1.12

`stix.ttp.related_ttps` Module

Classes

class `stix.ttp.related_ttps.`**`RelatedTTPs`**(*scope=None*, **args*)
Bases: *stix.common.related.GenericRelationshipList*

Version: 1.1.1.12

`stix.ttp.resource` Module

Classes

class `stix.ttp.resource.`**`Resource`**(*tools=None*, *infrastructure=None*, *personas=None*)
Bases: *stix.base.Entity*

Version: 1.1.1.12

`stix.ttp.victim_targeting` Module

Classes

class `stix.ttp.victim_targeting.`**`VictimTargeting`**
Bases: *stix.base.Entity*

STIX Utils

Modules located in the stix.utils package

Version: 1.1.1.12

`stix.utils.dates` Module

Functions

`stix.utils.dates.`**`parse_value`**(*value*)
Attempts to parse *value* into an instance of `datetime.datetime`. If *value* is `None`, this function will return `None`.

Parameters **`value`** – A timestamp. This can be a string or datetime.datetime value.

stix.utils.dates.**serialize_value**(*value*)
Attempts to convert *value* into an ISO8601-compliant timestamp string. If *value* is `None`, `None` will be returned.

Parameters **value** – A datetime.datetime value.

Returns An ISO8601 formatted timestamp string.

Version: 1.1.1.12

stix.utils.idgen Module

Classes

class stix.utils.idgen.**IDGenerator**(*namespace=None*, *method=1*)
Bases: `object`

Utility class for generating STIX ids

create_id(*prefix='guid'*)
Create an ID.

Note that if *prefix* is not provided, it will be *quid*, even if the *method* is *METHOD_INT*.

class stix.utils.idgen.**InvalidMethodError**(*method*)
Bases: `exceptions.ValueError`

Functions

stix.utils.idgen.**_get_generator**()
Return the *stix.utils* module's generator object.

Only under rare circumstances should this function be called by external code. More likely, external code should initialize its own IDGenerator or use the *set_id_namespace*, *set_id_method*, or *create_id* functions of the *stix.utils* module.

stix.utils.idgen.**set_id_namespace**(*namespace*)
Set the namespace for the module-level ID Generator

stix.utils.idgen.**set_id_method**(*method*)
Set the method for the module-level ID Generator

stix.utils.idgen.**get_id_namespace**()
Return the namespace associated with generated ids

stix.utils.idgen.**get_id_namespace_alias**()
Returns the namespace alias assoicated with generated ids

stix.utils.idgen.**create_id**(*prefix=None*)
Create an ID using the module-level ID Generator

Constants

stix.utils.idgen.**__generator = None**

stix.utils.idgen.**EXAMPLE_NAMESPACE = {'http://example.com': 'example'}**
dict() -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's

(key, value) pairs

dict(iterable) -> new dictionary initialized as if via: d = {} for k, v in iterable:

d[k] = v

dict(kwargs) -> new dictionary initialized with the name=value pairs** in the keyword argument list. For example: dict(one=1, two=2)

Version: 1.1.1.12

stix.utils.nsparser Module

This module automatically registers all STIX namespaces into mixbox.

Version: 1.1.1.12

stix.utils.parser Module

Classes

class stix.utils.parser.**UnsupportedVersionError**(*message*, *expected=None*, *found=None*)
Bases: exceptions.Exception

A parsed document is a version unsupported by the parser.

class stix.utils.parser.**UnknownVersionError**
Bases: exceptions.Exception

A parsed document contains no version information.

stix.utils.parser.**UnsupportedRootElement**
alias of UnsupportedRootElementError

class stix.utils.parser.**EntityParser**
Bases: mixbox.parser.EntityParser

Version: 1.1.1.12

API Coverage

The *python-stix* APIs currently provide partial coverage of all STIX-defined constructs. Development is ongoing toward the goal of providing full STIX language support in the APIs. Until such time that full coverage is provided, an overview of which constructs are available in these APIs will be maintained below.

Note: Many STIX constructs can contain **CybOX** constructs. The **python-cybox** project provides its own APIs for interacting with the **CybOX** specification. Please see the CybOX API Documentation for information about CybOX API coverage.

STIX Core

STIX Construct	API Coverage	Documentation
STIX Package	Full	*stix.core.stix_package.STIXPackage*
STIX Header	Full	*stix.core.stix_header.STIXHeader*
Related Packages	Full	*stix.core.stix_package.RelatedPackages*

STIX Top-level Constructs

STIX Construct	API Coverage	Documentation
Campaign	Full	*stix.campaign.Campaign*
Course of Action	Full	*stix.coa.CourseOfAction*
Exploit Target	Full	*stix.exploit_target.ExploitTarget*
Incident	Partial	*stix.incident.Incident*
Indicator	Full	*stix.indicator.indicator.Indicator*
Observable	*Provided by* **CybOX**	
Threat Actor	Full	*stix.threat_actor.ThreatActor*
TTP	Partial	*stix.ttp.TTP*

STIX Features

STIX Construct	API Coverage	Documentation
Confidence	Partial	*stix.common.confidence.Confidence*
Handling	Full	*stix.data_marking.Marking*
Markup in Structured Text	× None	
Relationships	Full	

STIX Extensions

STIX Construct	API Coverage	Documentation
Address Extensions		
CIQ Address	× None	
Attack Pattern Extensions		
CAPEC 2.7	× None	
Identity Extensions		
CIQ Identity	Full	*stix.extensions.identity.ciq_identit*
Malware Extensions		
MAEC	Full	*stix.extensions.malware.maec_4_1_mal*
Marking Extensions		
Simple Marking	Full	*stix.extensions.marking.simple_marki*
TLP	Full	*stix.extensions.marking.tlp.TLPMarki*
Terms of Use	Full	*stix.extensions.marking.terms_of_use*
Structured COA Extensions		
Generic Structured COA	× None	
Test Mechanism Extensions		
Generic Test Mechanism	Full	*stix.extensions.test_mechanism.gener*
OVAL	× None	
OpenIOC	Full	*stix.extensions.test_mechanism.open_*
SNORT	Full	*stix.extensions.test_mechanism.snort*
YARA	Full	*stix.extensions.test_mechanism.yara_*
Vulnerability Extensions		
CVRF	× None	

STIX Vocabularies

STIX Construct	API Coverage	Documentation
AssetTypeVocab-1.0	Full	*stix.common.vocabs.AssetType*
AttackerInfrastructureTypeVocab-1.0	Full	*stix.common.vocabs.AttackerInfra*
AttackerToolTypeVocab-1.0	Full	*stix.common.vocabs.AttackerToolT*
AvailabilityLossTypeVocab-1.0	× None *(replaced by version 1.1.1)*	
AvailabilityLossTypeVocab-1.1.1	Full	*stix.common.vocabs.AvailabilityL*
COAStageVocab-1.0	Full	*stix.common.vocabs.COAStage*
CampaignStatusVocab-1.0	Full	*stix.common.vocabs.CampaignStatu*
CourseOfActionTypeVocab-1.0	Full	*stix.common.vocabs.CourseOfActio*
DiscoveryMethodVocab-1.0	Full	*stix.common.vocabs.DiscoveryMeth*
HighMediumLowVocab-1.0	Full	*stix.common.vocabs.HighMediumLow*
ImpactQualificationVocab-1.0	Full	*stix.common.vocabs.ImpactQualifi*
ImpactRatingVocab-1.0	Full	*stix.common.vocabs.ImpactRating*
IncidentCategoryVocab-1.0	Full	*stix.common.vocabs.IncidentCateg*
IncidentEffectVocab-1.0	Full	*stix.common.vocabs.IncidentEffec*
IncidentStatusVocab-1.0	Full	*stix.common.vocabs.IncidentStatu*
IndicatorTypeVocab-1.0	× None *(replaced by version 1.1)*	
IndicatorTypeVocab-1.1	Full	*stix.common.vocabs.IndicatorType*
InformationSourceRoleVocab-1.0	Full	*stix.common.vocabs.InformationSo*
InformationTypeVocab-1.0	Full	*stix.common.vocabs.InformationTy*
IntendedEffectVocab-1.0	Full	*stix.common.vocabs.IntendedEffec*
LocationClassVocab-1.0	Full	*stix.common.vocabs.LocationClass*
LossDurationVocab-1.0	Full	*stix.common.vocabs.LossDuration*
LossPropertyVocab-1.0	Full	*stix.common.vocabs.LossProperty*
MalwareTypeVocab-1.0	Full	*stix.common.vocabs.MalwareType*
ManagementClassVocab-1.0	Full	*stix.common.vocabs.ManagementCla*
MotivationVocab-1.0	× None *(replaced by version 1.0.1)*	
MotivationVocab-1.0.1	× None *(replaced by version 1.1)*	
MotivationVocab-1.1	Full	*stix.common.vocabs.Motivation*
OwnershipClassVocab-1.0	Full	*stix.common.vocabs.OwnershipClas*
PackageIntentVocab-1.0	Full	*stix.common.vocabs.PackageIntent*
PlanningAndOperationalSupportVocab-1.0	× None *(replaced by version 1.0.1)*	
PlanningAndOperationalSupportVocab-1.0.1	Full	*stix.common.vocabs.PlanningAndOp*
SecurityCompromiseVocab-1.0	Full	*stix.common.vocabs.SecurityCompr*
SystemTypeVocab-1.0	Full	*stix.common.vocabs.SystemType*
ThreatActorSophisticationVocab-1.0	Full	*stix.common.vocabs.ThreatActorSo*
ThreatActorTypeVocab-1.0	Full	*stix.common.vocabs.ThreatActorTy*

CHAPTER 4

FAQ

- **My RAM consumption rises when processing a large amount of files.** This problem is caused by a python-cybox caching mechanism that is enabled by default. To prevent this issue from happening use the `cybox.utils.caches.cache_clear()` method in your code/script to release the cached resources as appropriate. Refer to the `cybox` documentation for more details.

CHAPTER 5

Contributing

If a bug is found, a feature is missing, or something just isn't behaving the way you'd expect it to, please submit an issue to our tracker. If you'd like to contribute code to our repository, you can do so by issuing a pull request and we will work with you to try and integrate that code into our repository.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

S

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y