python-stix Documentation

Release 1.1.1.5

The MITRE Corporation

April 28, 2015

Contents

Version: 1.1.1.5

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	1.1.1.5 (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: 1.1.1.5

2.1 Installation

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

2.1.1 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.

2.1.2 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.

2.1.3 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 1.1.1.5 release.
3. Extract the downloaded file. This will leave you with a directory named stix-1.1.1.5.

```
$ tar -zxf stix-1.1.1.5.tar.gz
$ ls
stix-1.1.1.5 stix-1.1.1.5.tar.gz
```

OR

```
$ unzip stix-1.1.1.5.zip
$ ls
stix-1.1.1.5 stix-1.1.1.5.zip
```

4. Run the installation script.

```
$ cd stix-1.1.1.5
$ 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.

2.1.4 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: 1.1.1.5

2.2 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.

2.2.1 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 2.6/2.7 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.

2.2.2 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
```

2.2.3 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: 1.1.1.5

2.3 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**.

2.3.1 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+00:00
    <stix:STIX_Header>
        <stix:Description>Getting Started!</stix:Description>
    </stix:STIX_Header>
</stix:STIX_Package>
```

2.3.2 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.

2.3.3 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: 1.1.1.5

2.4 APIs or bindings?

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

2.4.1 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.

2.4.2 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.

2.4.3 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.

2.4.4 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)
```

2.4.5 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: 1.1.1.5

3.1 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.

3.1.1 STIX

Modules located in the base stix package

Version: 1.1.1.5

`stix.base` Module

Classes

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

Base class for all classes in the STIX API.

classmethod `dict_from_object`(*entity_obj*)

Convert from object representation to dict representation.

`find`(*id_*)

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

classmethod `from_dict`(*d*, *return_obj=None*)

Convert from dict representation to object representation. This should be overriden by a subclass

classmethod `from_json`(*json_doc*)

Parses the JSON document *json_doc* and returns a STIX `Entity` implementation instance.

Parameters **json_doc** – Input JSON representation of a STIX entity. This can be a readable object or a JSON string.

Returns *An implementation of* – class:.*Entity* (e.g., `STIXPackage`).

classmethod **from_obj**(*obj*, *return_obj=None*)
Create an object from a binding object

classmethod **object_from_dict**(*entity_dict*)
Convert from dict representation to object representation.

to_dict()
Converts a STIX Entity implementation into a Python dictionary. This may be overridden by derived classes.

to_obj(*return_obj=None*, *ns_info=None*)
Converts an *Entity* into a binding object.

Note: This needs to be overridden by derived classes.

---

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 unicode string 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 unicode string is returned.

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

class stix.base.**EntityList**(**args*)
Bases: _abcoll.MutableSequence, stix.base.Entity

Version: 1.1.1.5

stix.data_marking Module

Classes

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

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*)

Constants

stix.data_marking.**_EXTENSION_MAP = {}**
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)

3.1.2 STIX Campaign

Modules located in the stix.campaign package

Version: 1.1.1.5

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

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

3.1.3 STIX Common

Modules located in the stix.common package

Version: 1.1.1.5

stix.common Module

Classes

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

Version: 1.1.1.5

stix.common.activity Module

Classes

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

Version: 1.1.1.5

stix.common.confidence Module

Classes

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

Version: 1.1.1.5

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: 1.1.1.5

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*)

Constants

stix.common.identity.**_EXTENSION_MAP = {}**
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: 1.1.1.5

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: 1.1.1.5

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: 1.1.1.5

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

class stix.common.related.**RelatedPackageRef**(***kwargs*)
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.

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: 1.1.1.5

stix.common.statement Module

Classes

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

Version: 1.1.1.5

stix.common.structured_text Module

Classes

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

Version: 1.1.1.5

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: 1.1.1.5

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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Functions

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

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

Constants

stix.common.vocabs.**_VOCAB_MAP = {'stixVocabs:LocationClassVocab-1.0': <class 'stix.common.vocabs.LocationClass'>**
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)

3.1.4 STIX Core

Modules located in the stix.core package

Version: 1.1.1.5

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

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

short_description
The short_description property for this entity.

Default Value: None

Note: If set to a value that is not an instance of stix.common.structured_text.StructuredText, an attempt to will be made to convert the value into an instance of stix.common.structured_text.StructuredText.

Returns An instance of stix.common.structured_text.StructuredText

Version: 1.1.1.5

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*)

Bases: stix.base.Entity

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.

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* – class:*STIXPackage*.

class stix.core.stix_package.**RelatedPackages**(*scope=None*, **args*)

Bases: stix.common.related.GenericRelationshipList

Version: 1.1.1.5

stix.core.ttps Module

Classes

class stix.core.ttps.**TTPs**(*ttps=None*)

Bases: stix.base.EntityList

3.1.5 STIX Course of Action (COA)

Modules located in the stix.coa package

Version: 1.1.1.5

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

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

Version: 1.1.1.5

stix.coa.objective Module

Classes

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

3.1.6 STIX Exploit Target

Modules located in the stix.exploit_target package

Version: 1.1.1.5

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 ExploitTarget.

Parameters

- **id_** (*optional*) – An identifier. If `None`, a value will be generated via `stix.utils.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(*v*)
Adds a configuration to the `configurations` list property.

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

Parameters **v** – A configuration value.

Raises: ValueError if the `v` param is of type `stix.exploit_target.configuration`

add_vulnerability(*v*)
Adds a vulnerability to the `vulnerabilies` list property.

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

Parameters **v** – A Vulnerability value.

Raises: ValueError if the v param is of type `stix.exploit_target.vulnerability`

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

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

Parameters **v** – A weakness value.

Raises: ValueError if the `v` param is of type `stix.exploit_target.weakness`

configuration
A list of `Configuration` objects

Default Value: `None`

Returns A list of `stix.exploit_target.configuration`

Raises `ValueError` – If set to a value that is not `None` and not an instance of `stix.exploit_target.configuration`

vulnerabilities
A list of `Vulnerability` objects

Default Value: `None`

Returns A list of `stix.exploit_target.vulnerability`

Raises ValueError – If set to a value that is not None and not an instance of stix.exploit_target.vulnerability

weaknesses
A list of Weakness objects

Default Value: None

Returns A list of stix.exploit_target.weakness

Raises ValueError – If set to a value that is not None and not an instance of stix.exploit_target.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: 1.1.1.5

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.

cce_id
Common Configuration Enumeration value for this Configuration.

Default Value: None

Returns A string representing the CCE ID

description
The description property for this Configuration.

Default Value: None

Note: If set to a value that is not an instance of `stix.common.structured_text.StructuredText`, an attempt to will be made to convert the value into an instance of `stix.common.structured_text.StructuredText`.

Returns An instance of `stix.common.structured_text.StructuredText`

Version: 1.1.1.5

`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.

`description`

The `description` property for this `Vulnerability`.

Default Value: `None`

Note: If set to a value that is not an instance of `stix.common.structured_text.StructuredText`, an attempt to will be made to convert the value into an instance of `stix.common.structured_text.StructuredText`.

Returns An instance of `stix.common.structured_text.StructuredText`

`discovered_datetime`

Returns The time this vulnerability was discovered, represented as class:*DateTimeWithPrecision*

`short_description`

The `short_description` property for this `Vulnerability`.

Default Value: `None`

Note: If set to a value that is not an instance of `stix.common.structured_text.StructuredText`, an attempt to will be made to convert the value into an instance of `stix.common.structured_text.StructuredText`.

Returns An instance of stix.common.structured_text.StructuredText

title
String representing the Vulnerability Title

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: 1.1.1.5

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.

cwe_id
Common Weakness Enumeration value as a string

description
The description property for this Weakness.

Default Value: None

Note: If set to a value that is not an instance of stix.common.structured_text.StructuredText, an attempt to will be made to convert the value into an instance of stix.common.structured_text.StructuredText.

Returns An instance of stix.common.structured_text.StructuredText

3.1.7 STIX Extensions

Modules located in the stix.extensions package

Version: 1.1.1.5

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*)
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: 1.1.1.5

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

Version: 1.1.1.5

stix.extensions.marking.simple_marking Module

Classes

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

Version: 1.1.1.5

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: 1.1.1.5

stix.extensions.marking.tlp Module

Classes

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

Version: 1.1.1.5

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: 1.1.1.5

stix.extensions.test_mechanism.open_ioc_2010_test_mechanism Module

Classes

class stix.extensions.test_mechanism.open_ioc_2010_test_mechanism.**OpenIOCTestMechanism**(*id_=None*, *idref=None*)

Bases: stix.indicator.test_mechanism._BaseTestMechanism

Version: 1.1.1.5

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: 1.1.1.5

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

3.1.8 STIX Incident

Modules located in the stix.incident package

Version: 1.1.1.5

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

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 wil 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 **indicator** – 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`

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: 1.1.1.5

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: 1.1.1.5

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: 1.1.1.5

stix.incident.contributors Module

Classes

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

Version: 1.1.1.5

stix.incident.direct_impact_summary Module

Classes

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

Version: 1.1.1.5

stix.incident.external_id Module

Classes

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

Version: 1.1.1.5

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: 1.1.1.5

stix.incident.impact_assessment Module

Classes

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

Version: 1.1.1.5

stix.incident.indirect_impact_summary Module

Classes

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

Version: 1.1.1.5

stix.incident.loss_estimation Module

Classes

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

Version: 1.1.1.5

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: 1.1.1.5

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: 1.1.1.5

stix.incident.total_loss_estimation Module

Classes

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

Bases: stix.base.Entity

3.1.9 STIX Indicator

Modules located in the stix.indicator package

Version: 1.1.1.5

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 IndicatorType.

Parameters

- **id_** (*optional*) – An identifier. If None, a value will be generated via stix.utils.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 `observables` list property of the `Indicator`.

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

Note: The STIX Language dictates that an `Indicator` can have only one `Observable` under it. Because of this, the `to_xml()` method will convert the `observables` list into an `cybox.core.ObservableComposition` instance, in which each item in the `observables` list will be added to the composition. 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.

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_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`.

alternative_id

An alternative identifi er for this `Indicator`

This property can be set to a single string identifier or a list of identifiers. If set to a single object, the object will be inserted into an empty list internally.

Default Value: Empty `list`

Returns A list of alternative ids.

confidence

The confidence for this `Indicator`.

This property can be set to an instance of `str`, `stix.common.vocabs.VocabString`, or `stix.common.confidence.Confidence`.

Default Value: `None`

Note: If set to an instance of `str` or `stix.common.vocabs.VocabString`, that value will be wrapped in an instance of `stix.common.confidence.Confidence`.

Returns An instance of of `stix.common.confidence.Confidence`.

Raises `ValueError` – If set to a `str` value that cannot be converted into an instance of `stix.common.confidence.Confidence`.

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`.

indicator_types
A list of indicator types for this Indicator.

This property can be set to lists or single instances of str or stix.common.vocabs.VocabString or an instance of IndicatorTypes.

Note: If an instance of str is passed in (or a list containing str values) an attempt will be made to convert that string value to an instance of stix.common.vocabs.IndicatorType.

Default Value: An empty IndicatorTypes instance.

See also:

Documentation for IndicatorTypes.

Returns An instance of IndicatorTypes.

observable
A convenience property for accessing or setting the only cybox.core.Observable instance held by this Indicator.

Default Value: Empty list.

Setting this property results in the observables property being reinitialized to an empty list and appending the input value, resulting in a list containing one value.

Note: If the observables list contains more than one item, this property will only return the first item in the list.

Returns An instance of cybox.core.Observable.

Raises ValueError – If set to a value that cannot be converted to an instance of cybox.core.Observable.

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

Note: If the input value or values are not instance(s) of cybox.core.Observable, an attempt will be made to convert the value to an instance of cybox.core.Observable.

Default Value: Empty list

Returns A list of cybox.core.Observable instances.

Raises ValueError – If set to a value that cannot be converted to an instance of cybox.core.Observable.

producer
Contains information about the source of the Indicator.

Default Value: None

Returns An instance of stix.common.information_source.InformationSource

Raises ValueError – If set to a value that is not None and not an instance of stix.common.information_source.InformationSource

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`.

---

valid_time_positions
A list of valid time positions for this `Indicator`.

This property can be set to a single instance or a list of `stix.indicator.valid_time.ValidTime` instances. If set to a single instance, that object is converted into a list containing one item.

Default Value: Empty `list`

Returns A list of `stix.indicator.valid_time.ValidTime` instances.

class stix.indicator.indicator.**CompositeIndicatorExpression**(*operator='OR'*, **args*)
Bases: stix.base.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: 1.1.1.5

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: 1.1.1.5

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*)

Constants

stix.indicator.test_mechanism.**_EXTENSION_MAP = {'genericTM:GenericTestMechanismType': <class 'stix.extensi**
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: 1.1.1.5

stix.indicator.valid_time Module

Classes

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

3.1.10 STIX Threat Actor

Modules located in the stix.threat_actor package

Version: 1.1.1.5

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

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

3.1.11 STIX Tactics, Techniques, and Procedures (TTP)

Modules located in the stix.ttp package

Version: 1.1.1.5

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

Version: 1.1.1.5

stix.ttp.attack_pattern Module

Classes

class stix.ttp.attack_pattern.**AttackPattern**(*id_=None*, *title=None*, *description=None*, *short_description=None*)
Bases: stix.base.Entity

Version: 1.1.1.5

stix.ttp.behavior Module

Classes

class stix.ttp.behavior.**Behavior**(*malware_instances=None*, *attack_patterns=None*, *exploits=None*)

Bases: stix.base.Entity

Version: 1.1.1.5

stix.ttp.exploit Module

Classes

class stix.ttp.exploit.**Exploit**(*id_=None*, *title=None*, *description=None*, *short_description=None*)

Bases: stix.base.Entity

Version: 1.1.1.5

stix.ttp.exploit_targets Module

Classes

class stix.ttp.exploit_targets.**ExploitTargets**(*scope=None*, **args*)

Bases: stix.common.related.GenericRelationshipList

Version: 1.1.1.5

stix.ttp.infrastructure Module

Classes

class stix.ttp.infrastructure.**Infrastructure**(*id_=None*, *title=None*, *description=None*, *short_description=None*)

Bases: stix.base.Entity

Version: 1.1.1.5

stix.ttp.malware_instance Module

Classes

class stix.ttp.malware_instance.**MalwareInstance**(*id_=None*, *title=None*, *description=None*, *short_description=None*)

Bases: stix.base.Entity

Functions

stix.ttp.malware_instance.**add_extension**(*cls*)

Constants

stix.ttp.malware_instance.**_EXTENSION_MAP = {'stix-maec:MAEC4.1InstanceType': <class 'stix.extensions.malware**
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: 1.1.1.5

stix.ttp.related_ttps Module

Classes

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

Version: 1.1.1.5

stix.ttp.resource Module

Classes

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

Version: 1.1.1.5

stix.ttp.victim_targeting Module

Classes

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

3.1.12 STIX Utils

Modules located in the stix.utils package

Version: 1.1.1.5

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: 1.1.1.5

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: 1.1.1.5

stix.utils.nsparser Module

Classes

class stix.utils.nsparser.**NamespaceParser**
Bases: object

Constants

stix.utils.nsparser.**XML_NAMESPACES** = {'http://www.w3.org/2000/09/xmldsig#': 'ds', 'http://www.w3.org/1999/xlink'
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)

stix.utils.nsparser.**STIX_NS_TO_SCHEMALOCATION** = {'http://stix.mitre.org/extensions/StructuredCOA#Generic-1':
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)

stix.utils.nsparser.**EXT_NS_TO_SCHEMALOCATION** = {'urn:oasis:names:tc:ciq:xpil:3': 'http://stix.mitre.org/XMLSc
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)

stix.utils.nsparser.**DEFAULT_STIX_NS_TO_PREFIX = {'http://stix.mitre.org/extensions/StructuredCOA#Generic-1'**
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)

stix.utils.nsparser.**DEFAULT_EXT_TO_PREFIX = {'http://capec.mitre.org/capec-2': 'capec', 'http://schemas.mandian**
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: 1.1.1.5

stix.utils.parser Module

Classes

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

Raised when a parsed STIX document contains a version that is not supported by this verison of python-stix.

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

Raised when a parsed STIX document contains no version information.

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

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

parse_xml(*xml_file*, *check_version=True*, *check_root=True*, *encoding=None*)
Creates a python-stix STIXPackage object from the supplied xml_file.

Parameters

- **xml_file** – A filename/path or a file-like object representing a STIX instance document

- **check_version** – Inspect the version before parsing.
- **check_root** – Inspect the root element before parsing.
- **encoding** – The character encoding of the input *xml_file*. If `None`, an attempt will be made to determine the input character encoding.

Raises

- `UnknownVersionError` – If *check_version* is `True` and *xml_file* does not contain STIX version information.
- `UnsupportedVersionError` – If *check_version* is `False` and *xml_file* contains an unsupported STIX version.
- `UnsupportedRootElement` – If *check_root* is `True` and *xml_file* contains an invalid root element.

parse_xml_to_obj(*xml_file*, *check_version=True*, *check_root=True*, *encoding=None*)

Creates a STIX binding object from the supplied xml file.

Parameters

- **xml_file** – A filename/path or a file-like object representing a STIX instance document
- **check_version** – Inspect the version before parsing.
- **check_root** – Inspect the root element before parsing.
- **encoding** – The character encoding of the input *xml_file*.

Raises

- `UnknownVersionError` – If *check_version* is `True` and *xml_file* does not contain STIX version information.
- `UnsupportedVersionError` – If *check_version* is `False` and *xml_file* contains an unsupported STIX version.
- `UnsupportedRootElement` – If *check_root* is `True` and *xml_file* contains an invalid root element.

Version: 1.1.1.5

3.2 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.

3.2.1 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`

3.2.2 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`

3.2.3 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	

3.2.4 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	

3.2.5 STIX Vocabularies

STIX Construct	API Coverage	Documentation
AssetTypeVocab-1.0	Full	stix.common.vocabs.AssetType
		Co

Table 3.1 – continued from previous page

STIX Construct	API Coverage	Documentation
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

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 5

Indices and tables

- *genindex*
- *modindex*
- *search*

S

Symbols

A

P

R

S

T

U

V

W

X

Y