stix2 Documentation

Release 0.5.1

OASIS

Apr 11, 2018

Contents:

CHAPTER 1

Overview

1.1 Goals

High level goals/principles of the python-stix2 library:

1. It should be as easy as possible (but no easier!) to perform common tasks of producing, consuming, and processing STIX 2 content.
2. It should be hard, if not impossible, to emit invalid STIX 2.
3. The library should default to doing "the right thing", complying with both the STIX 2.0 spec, as well as associated best practices. The library should make it hard to do "the wrong thing".

1.2 Design Decisions

To accomplish these goals, and to incorporate lessons learned while developing python-stix (for STIX 1.x), several decisions influenced the design of python-stix2:

1. All data structures are immutable by default. In contrast to python-stix, where users would create an object and then assign attributes to it, in python-stix2 all properties must be provided when creating the object.
2. Where necessary, library objects should act like `dict`'s. When treated as a `str`, the JSON reprentation of the object should be used.
3. Core Python data types (including numeric types, `datetime`) should be used when appropriate, and serialized to the correct format in JSON as specified in the STIX 2.0 spec.

1.3 Architecture

The *stix2* library APIs are divided into three logical layers, representing different levels of abstraction useful in different types of scripts and larger applications. It is possible to combine multiple layers in the same program, and the higher levels build on the layers below.

1.3.1 Object Layer

The lowest layer, **Object Layer**, is where Python objects representing STIX 2 data types (such as SDOs, SROs, and Cyber Observable Objects, as well as non-top-level objects like External References, Kill Chain phases, and Cyber Observable extensions) are created, and can be serialized and deserialized to and from JSON representation.

This layer is appropriate for stand-alone scripts that produce or consume STIX 2 content, or can serve as a low-level data API for larger applications that need to represent STIX objects as Python classes.

At this level, non-embedded reference properties (those ending in `_ref`, such as the links from a Relationship object to its source and target objects) are not implemented as references between the Python objects themselves, but by simply having the same values in `id` and reference properties. There is no referential integrity maintained by the `stix2` library.

This layer is mostly complete.

1.3.2 Environment Layer

The **Environment Layer** adds several components that make it easier to handle STIX 2 data as part of a larger application and as part of a larger cyber threat intelligence ecosystem.

- `Data Source`s represent locations from which STIX data can be retrieved, such as a TAXII server, database, or local filesystem. The Data Source API abstracts differences between these storage location, giving a common API to get objects by ID or query by various properties, as well as allowing federated operations over multiple data sources.
- Similarly, `Data Sink` objects represent destinations for sending STIX data.
- An `Object Factory` provides a way to add common properties to all created objects (such as the same `created_by_ref`, or a `StatementMarking` with copyright information or terms of use for the STIX data).

Each of these components can be used individually, or combined as part of an `Environment`. These `Environment` objects allow different settings to be used by different users of a multi-user application (such as a web application).

This layer is mostly complete.

1.3.3 Workbench Layer

The highest layer of the `stix2` APIs is the **Workbench Layer**, designed for a single user in a highly-interactive analytical environment (such as a Jupyter Notebook). It builds on the lower layers of the API, while hiding most of their complexity. Unlike the other layers, this layer is designed to be used directly by end users. For users who are comfortable with, Python, the Workbench Layer makes it easy to quickly interact with STIX data from a variety of sources without needing to write and run one-off Python scripts.

This layer is currently being developed.

CHAPTER 2

User's Guide

2.1 Creating STIX Content

2.1.1 Creating STIX Domain Objects

To create a STIX object, provide keyword arguments to the type's constructor:

```
In [3]: from stix2 import Indicator

        indicator = Indicator(name="File hash for malware variant",
                              labels=["malicious-activity"],
                              pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']")
        print(indicator)
```

```
Out[3]: <IPython.core.display.HTML object>
```

Certain required attributes of all objects will be set automatically if not provided as keyword arguments:

- If not provided, `type` will be set automatically to the correct type. You can also provide the type explicitly, but this is not necessary:

```
In [4]: indicator2 = Indicator(type='indicator',
                               labels=["malicious-activity"],
                               pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']")
```

Passing a value for `type` that does not match the class being constructed will cause an error:

```
In [5]: indicator3 = Indicator(type='xxx',
                               labels=["malicious-activity"],
                               pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']")
```

```
InvalidValueError: Invalid value for Indicator 'type': must equal 'indicator'.
```

- If not provided, `id` will be generated randomly. If you provide an `id` argument, it must begin with the correct prefix:

```
In [6]: indicator4 = Indicator(id="campaign--63ce9068-b5ab-47fa-a2cf-a602ea01f21a",
                               labels=["malicious-activity"],
                               pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']")

InvalidValueError: Invalid value for Indicator 'id': must start with 'indicator--'.
```

For indicators, `labels` and `pattern` are required and cannot be set automatically. Trying to create an indicator that is missing one of these properties will result in an error:

```
In [7]: indicator = Indicator()

MissingPropertiesError: No values for required properties for Indicator: (labels, pattern).
```

However, the required `valid_from` attribute on Indicators will be set to the current time if not provided as a keyword argument.

Once created, the object acts like a frozen dictionary. Properties can be accessed using the standard Python dictionary syntax:

```
In [8]: indicator['name']

Out[8]: u'File hash for malware variant'
```

Or access properties using the standard Python attribute syntax:

```
In [9]: indicator.name

Out[9]: u'File hash for malware variant'
```

Attempting to modify any attributes will raise an error:

```
In [10]: indicator['name'] = "This is a revised name"

TypeError: 'Indicator' object does not support item assignment


In [11]: indicator.name = "This is a revised name"

ImmutableError: Cannot modify 'name' property in 'Indicator' after creation.
```

To update the properties of an object, see the Versioning section.

Creating a Malware object follows the same pattern:

```
In [12]: from stix2 import Malware

         malware = Malware(name="Poison Ivy",
                           labels=['remote-access-trojan'])
         print(malware)

Out[12]: <IPython.core.display.HTML object>
```

As with indicators, the `type`, `id`, `created`, and `modified` properties will be set automatically if not provided. For Malware objects, the `labels` and `name` properties must be provided.

You can see the full list of SDO classes *here*.

2.1.2 Creating Relationships

STIX 2 Relationships are separate objects, not properties of the object on either side of the relationship. They are constructed similarly to other STIX objects. The `type`, `id`, `created`, and `modified` properties are added automatically if not provided. Callers must provide the `relationship_type`, `source_ref`, and `target_ref` properties.

```
In [13]: from stix2 import Relationship

         relationship = Relationship(relationship_type='indicates',
                                     source_ref=indicator.id,
                                     target_ref=malware.id)
         print(relationship)
```

```
Out[13]: <IPython.core.display.HTML object>
```

The `source_ref` and `target_ref` properties can be either the ID's of other STIX objects, or the STIX objects themselves. For readability, Relationship objects can also be constructed with the `source_ref`, `relationship_type`, and `target_ref` as positional (non-keyword) arguments:

```
In [14]: relationship2 = Relationship(indicator, 'indicates', malware)
         print(relationship2)
```

```
Out[14]: <IPython.core.display.HTML object>
```

2.1.3 Creating Bundles

STIX Bundles can be created by passing objects as arguments to the Bundle constructor. All required properties (`type`, `id`, and `spec_version`) will be set automatically if not provided, or can be provided as keyword arguments:

```
In [15]: from stix2 import Bundle

         bundle = Bundle(indicator, malware, relationship)
         print(bundle)
```

```
Out[15]: <IPython.core.display.HTML object>
```

2.2 Custom STIX Content

2.2.1 Custom Properties

Attempting to create a STIX object with properties not defined by the specification will result in an error. Try creating an `Identity` object with a custom `x_foo` property:

```
In [4]: from stix2 import Identity

        Identity(name="John Smith",
                 identity_class="individual",
                 x_foo="bar")
```

```
ExtraPropertiesError: Unexpected properties for Identity: (x_foo).
```

To create a STIX object with one or more custom properties, pass them in as a dictionary parameter called `custom_properties`:

```
In [2]: from stix2 import Identity

        identity = Identity(name="John Smith",
                            identity_class="individual",
                            custom_properties={
                                "x_foo": "bar"
                            })
        print(identity)
```

```
Out[2]: <IPython.core.display.HTML object>
```

Alternatively, setting `allow_custom` to `True` will allow custom properties without requiring a `custom_properties` dictionary.

```
In [6]: identity2 = Identity(name="John Smith",
                             identity_class="individual",
                             x_foo="bar",
                             allow_custom=True)
        print(identity2)
```

```
Out[6]: <IPython.core.display.HTML object>
```

Likewise, when parsing STIX content with custom properties, pass `allow_custom=True` to *parse()*:

```
In [7]: from stix2 import parse

        input_string = """{
            "type": "identity",
            "id": "identity--311b2d2d-f010-5473-83ec-1edf84858f4c",
            "created": "2015-12-21T19:59:11Z",
            "modified": "2015-12-21T19:59:11Z",
            "name": "John Smith",
            "identity_class": "individual",
            "x_foo": "bar"
        }"""
        identity3 = parse(input_string, allow_custom=True)
        print(identity3.x_foo)
```

```
bar
```

2.2.2 Custom STIX Object Types

To create a custom STIX object type, define a class with the @*CustomObject* decorator. It takes the type name and a list of property tuples, each tuple consisting of the property name and a property instance. Any special validation of the properties can be added by supplying an `__init__` function.

Let's say zoo animals have become a serious cyber threat and we want to model them in STIX using a custom object type. Let's use a `species` property to store the kind of animal, and make that property required. We also want a property to store the class of animal, such as "mammal" or "bird" but only want to allow specific values in it. We can add some logic to validate this property in `__init__`.

```
In [8]: from stix2 import CustomObject, properties

        @CustomObject('x-animal', [
            ('species', properties.StringProperty(required=True)),
            ('animal_class', properties.StringProperty()),
        ])
        class Animal(object):
            def __init__(self, animal_class=None, **kwargs):
                if animal_class and animal_class not in ['mammal', 'bird', 'fish', 'reptile']:
                    raise ValueError("'%s' is not a recognized class of animal." % animal_class)
```

Now we can create an instance of our custom `Animal` type.

```
In [9]: animal = Animal(species="lion",
                        animal_class="mammal")
        print(animal)
```

```
Out[9]: <IPython.core.display.HTML object>
```

Trying to create an `Animal` instance with an `animal_class` that's not in the list will result in an error:

```
In [10]: Animal(species="xenomorph",
                animal_class="alien")
```

```
ValueError: 'alien' is not a recognized class of animal.
```

Parsing custom object types that you have already defined is simple and no different from parsing any other STIX object.

```
In [11]: input_string2 = """{
             "type": "x-animal",
             "id": "x-animal--941f1471-6815-456b-89b8-7051ddf13e4b",
             "created": "2015-12-21T19:59:11Z",
             "modified": "2015-12-21T19:59:11Z",
             "species": "shark",
             "animal_class": "fish"
         }"""
         animal2 = parse(input_string2)
         print(animal2.species)
```

```
shark
```

However, parsing custom object types which you have not defined will result in an error:

```
In [12]: input_string3 = """{
             "type": "x-foobar",
             "id": "x-foobar--d362beb5-a04e-4e6b-a030-b6935122c3f9",
             "created": "2015-12-21T19:59:11Z",
             "modified": "2015-12-21T19:59:11Z",
             "bar": 1,
             "baz": "frob"
         }"""
         parse(input_string3)
```

```
ParseError: Can't parse unknown object type 'x-foobar'! For custom types, use the CustomObject decora
```

2.2.3 Custom Cyber Observable Types

Similar to custom STIX object types, use a decorator to create *custom Cyber Observable* types. Just as before, `__init__()` can hold additional validation, but it is not necessary.

```
In [13]: from stix2 import CustomObservable

         @CustomObservable('x-new-observable', [
             ('a_property', properties.StringProperty(required=True)),
             ('property_2', properties.IntegerProperty()),
         ])
         class NewObservable():
             pass

         new_observable = NewObservable(a_property="something",
                                        property_2=10)
         print(new_observable)
```

```
Out[13]: <IPython.core.display.HTML object>
```

Likewise, after the custom Cyber Observable type has been defined, it can be parsed.

```
In [14]: from stix2 import ObservedData

         input_string4 = """{
```

```
            "type": "observed-data",
            "id": "observed-data--b67d30ff-02ac-498a-92f9-32f845f448cf",
            "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
            "created": "2016-04-06T19:58:16.000Z",
            "modified": "2016-04-06T19:58:16.000Z",
            "first_observed": "2015-12-21T19:00:00Z",
            "last_observed": "2015-12-21T19:00:00Z",
            "number_observed": 50,
            "objects": {
                "0": {
                    "type": "x-new-observable",
                    "a_property": "foobaz",
                    "property_2": 5
                }
            }
        }"""
        obs_data = parse(input_string4)
        print(obs_data.objects["0"].a_property)
        print(obs_data.objects["0"].property_2)
```

```
foobaz
5
```

2.2.4 Custom Cyber Observable Extensions

Finally, custom extensions to existing Cyber Observable types can also be created. Just use the *@CustomExtension* decorator. Note that you must provide the Cyber Observable class to which the extension applies. Again, any extra validation of the properties can be implemented by providing an `__init__()` but it is not required. Let's say we want to make an extension to the `File` Cyber Observable Object:

```
In [15]: from stix2 import File, CustomExtension

         @CustomExtension(File, 'x-new-ext', [
             ('property1', properties.StringProperty(required=True)),
             ('property2', properties.IntegerProperty()),
         ])
         class NewExtension():
             pass

         new_ext = NewExtension(property1="something",
                                property2=10)
         print(new_ext)
```

```
Out[15]: <IPython.core.display.HTML object>
```

Once the custom Cyber Observable extension has been defined, it can be parsed.

```
In [16]: input_string5 = """{
             "type": "observed-data",
             "id": "observed-data--b67d30ff-02ac-498a-92f9-32f845f448cf",
             "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
             "created": "2016-04-06T19:58:16.000Z",
             "modified": "2016-04-06T19:58:16.000Z",
             "first_observed": "2015-12-21T19:00:00Z",
             "last_observed": "2015-12-21T19:00:00Z",
             "number_observed": 50,
             "objects": {
                 "0": {
                     "type": "file",
```

```
                    "name": "foo.bar",
                    "hashes": {
                        "SHA-256": "35a01331e9ad96f751278b891b6ea09699806faedfa237d40513d92ad1b7100f
                    },
                    "extensions": {
                        "x-new-ext": {
                            "property1": "bla",
                            "property2": 50
                        }
                    }
                }
            }
        }"""
        obs_data2 = parse(input_string5)
        print(obs_data2.objects["0"].extensions["x-new-ext"].property1)
        print(obs_data2.objects["0"].extensions["x-new-ext"].property2)
```

```
bla
50
```

```
In [3]: # without this configuration, only last print() call is outputted in cells
        from IPython.core.interactiveshell import InteractiveShell
        InteractiveShell.ast_node_interactivity = "all"
```

2.3 DataStore API

CTI Python STIX2 features a new interface for pulling and pushing STIX2 content. The new interface consists of *DataStore*, *DataSource* and *DataSink* constructs: a *DataSource* for pulling STIX2 content, a *DataSink* for pushing STIX2 content, and a *DataStore* for both pulling and pushing.

The DataStore, *DataSource*, *DataSink* (collectively referred to as the "DataStore suite") APIs are not referenced directly by a user but are used as base classes, which are then subclassed by real DataStore suites. CTI Python STIX2 provides the DataStore suites of *FileSystem*, *Memory*, and *TAXII*. Users are also encouraged to subclass the base classes and create their own custom DataStore suites.

2.3.1 CompositeDataSource

CompositeDataSource is an available controller that can be used as a single interface to a set of defined *DataSources*. The purpose of this controller is allow for the grouping of *DataSources* and making `get()`/`query()` calls to a set of DataSources in one API call. *CompositeDataSources* can be used to organize/group *DataSources*, federate `get()`/`all_versions()`/`query()` calls, and reduce user code.

CompositeDataSource is just a wrapper around a set of defined *DataSources* (e.g. *FileSystemSource*) that federates `get()`/`all_versions()`/`query()` calls individually to each of the attached *DataSources* , collects the results from each *DataSource* and returns them.

Filters can be attached to *CompositeDataSources* just as they can be done to *DataStores* and *DataSources*. When `get()`/`all_versions()`/`query()` calls are made to the *CompositeDataSource*, it will pass along any query filters from the call and any of its own filters to the attached *DataSources*. In addition, those *DataSources* may have their own attached filters as well. The effect is that all the filters are eventually combined when the `get()`/`all_versions()`/`query()` call is actually executed within a *DataSource*.

A *CompositeDataSource* can also be attached to a *CompositeDataSource* for multiple layers of grouped *DataSources*.

CompositeDataSource API

CompositeDataSource Examples

```
In [4]: from taxii2client import Collection
        from stix2 import CompositeDataSource, FileSystemSource, TAXIICollectionSource

        # create FileSystemStore
        fs = FileSystemSource("/home/michael/cti-python-stix2/stix2/test/stix2_data/")

        # create TAXIICollectionSource
        colxn = Collection('https://test.freetaxii.com:8000/osint/collections/a9c22eaf-0f3e-482c-8bb4
        ts = TAXIICollectionSource(colxn)

        # add them both to the CompositeDataSource
        cs = CompositeDataSource()
        cs.add_data_sources([fs,ts])

        # get an object that is only in the filesystem
        intrusion_set = cs.get('intrusion-set--f3bdec95-3d62-42d9-a840-29630f6cdc1a')
        print(intrusion_set)

        # get an object that is only in the TAXII collection
        ind = cs.get('indicator--02b90f02-a96a-43ee-88f1-1e87297941f2')
        print(ind)

Out[4]: <IPython.core.display.HTML object>

Out[4]: <IPython.core.display.HTML object>
```

2.3.2 Filters

The CTI Python STIX2 DataStore suites - *FileSystem*, *Memory*, and *TAXII* - all use the *Filters* module to allow for the querying of STIX content. The basic functionality is that filters can be created and supplied everytime to calls to `query()`, and/or attached to a *DataStore* so that every future query placed to that *DataStore* is evaluated against the attached filters, supplemented with any further filters supplied with the query call. Attached filters can also be removed from *DataStores*.

Filters are very simple, as they consist of a field name, comparison operator and an object property value (i.e. value to compare to). All properties of STIX2 objects can be filtered on. In addition, TAXII2 Filtering parameters for fields can also be used in filters.

TAXII2 filter fields:

- added_after
- match[id]
- match[type]
- match[version]

Supported operators:

- =
- !=
- in

-
- <
- >=
- <=

Value types of the property values must be one of these (Python) types:

- bool
- dict
- float
- int
- list
- str
- tuple

Filter Examples

```
In [5]: import sys
        from stix2 import Filter

        # create filter for STIX objects that have external references to MITRE ATT&CK framework
        f = Filter("external_references.source_name", "=", "mitre-attack")

        # create filter for STIX objects that are not of SDO type Attack-Pattnern
        f1 = Filter("type", "!=", "attack-pattern")

        # create filter for STIX objects that have the "threat-report" label
        f2 = Filter("labels", "in", "threat-report")

        # create filter for STIX objects that have been modified past the timestamp
        f3 = Filter("modified", ">=", "2017-01-28T21:33:10.772474Z")

        # create filter for STIX objects that have been revoked
        f4 = Filter("revoked", "=", True)
```

For Filters to be applied to a query, they must be either supplied with the query call or attached to a *DataStore*, more specifically to a *DataSource* whether that *DataSource* is a part of a *DataStore* or stands by itself.

```
In [6]: from stix2 import MemoryStore, FileSystemStore, FileSystemSource

        fs = FileSystemStore("/home/michael/Desktop/sample_stix2_data")
        fs_source = FileSystemSource("/home/michael/Desktop/sample_stix2_data")

        # attach filter to FileSystemStore
        fs.source.filters.add(f)

        # attach multiple filters to FileSystemStore
        fs.source.filters.update([f1,f2])

        # can also attach filters to a Source
        # attach multiple filters to FileSystemSource
        fs_source.filters.update([f3, f4])
```

```
mem = MemoryStore()

# As it is impractical to only use MemorySink or MemorySource,
# attach a filter to a MemoryStore
mem.source.filters.add(f)

# attach multiple filters to a MemoryStore
mem.source.filters.update([f1,f2])
```

2.3.3 De-Referencing Relationships

Given a STIX object, there are several ways to find other STIX objects related to it. To illustrate this, let's first create a *DataStore* and add some objects and relationships.

```
In [13]: from stix2 import Campaign, Identity, Indicator, Malware, Relationship

         mem = MemoryStore()
         cam = Campaign(name='Charge', description='Attack!')
         idy = Identity(name='John Doe', identity_class="individual")
         ind = Indicator(labels=['malicious-activity'], pattern="[file:hashes.MD5 = 'd41d8cd98f00b204
         mal = Malware(labels=['ransomware'], name="Cryptolocker", created_by_ref=idy)
         rel1 = Relationship(ind, 'indicates', mal,)
         rel2 = Relationship(mal, 'targets', idy)
         rel3 = Relationship(cam, 'uses', mal)
         mem.add([cam, idy, ind, mal, rel1, rel2, rel3])
```

If a STIX object has a `created_by_ref` property, you can use the *creator_of()* method to retrieve the *Identity* object that created it.

```
In [14]: print(mem.creator_of(mal))

Out[14]: <IPython.core.display.HTML object>
```

Use the *relationships()* method to retrieve all the relationship objects that reference a STIX object.

```
In [15]: rels = mem.relationships(mal)
         len(rels)

Out[15]: 3
```

You can limit it to only specific relationship types:

```
In [27]: mem.relationships(mal, relationship_type='indicates')

Out[27]: [Relationship(type='relationship', id='relationship--bd6fd399-c907-4feb-b1da-b90f15942f1d',
```

You can limit it to only relationships where the given object is the source:

```
In [28]: mem.relationships(mal, source_only=True)

Out[28]: [Relationship(type='relationship', id='relationship--7eb7f5cd-8bf2-4f7c-8756-84c0b5693b9a',
```

And you can limit it to only relationships where the given object is the target:

```
In [30]: mem.relationships(mal, target_only=True)

Out[30]: [Relationship(type='relationship', id='relationship--bd6fd399-c907-4feb-b1da-b90f15942f1d',
          Relationship(type='relationship', id='relationship--3c759d40-c92a-430e-aab6-77d5c5763302',
```

Finally, you can retrieve all STIX objects related to a given STIX object using *related_to()*. This calls *relationships()* but then performs the extra step of getting the objects that these Relationships point to. *related_to()* takes all the same arguments that *relationships()* does.

```
In [42]: mem.related_to(mal, target_only=True, relationship_type='uses')
```

```
Out[42]: [Campaign(type='campaign', id='campaign--82ab7aa4-d13b-4e99-8a09-ebcba30668a7', created='201
```

2.4 Using Environments

An *Environment* object makes it easier to use STIX 2 content as part of a larger application or ecosystem. It allows you to abstract away the nasty details of sending and receiving STIX data, and to create STIX objects with default values for common properties.

2.4.1 Storing and Retrieving STIX Content

An *Environment* can be set up with a *DataStore* if you want to store and retrieve STIX content from the same place.

```
In [3]: from stix2 import Environment, MemoryStore

        env = Environment(store=MemoryStore())
```

If desired, you can instead set up an *Environment* with different data sources and sinks. In the following example we set up an environment that retrieves objects from *memory* and a directory on the *filesystem*, and stores objects in a different directory on the filesystem.

```
In [4]: from stix2 import CompositeDataSource, FileSystemSink, FileSystemSource, MemorySource

        src = CompositeDataSource()
        src.add_data_sources([MemorySource(), FileSystemSource("/tmp/stix_source")])
        env2 = Environment(source=src,
                           sink=FileSystemSink("/tmp/stix_sink"))
```

Once you have an *Environment* you can store some STIX content in its *DataSinks* with *add()*:

```
In [5]: from stix2 import Indicator

        indicator = Indicator(id="indicator--01234567-89ab-cdef-0123-456789abcdef",
                              labels=["malicious-activity"],
                              pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']")
        env.add(indicator)
```

You can retrieve STIX objects from the *DataSources* in the *Environment* with *get()*, *query()*, *all_versions()*, *creator_of()*, *related_to()*, and *relationships()* just as you would for a *DataSource*.

```
In [6]: print(env.get("indicator--01234567-89ab-cdef-0123-456789abcdef"))

Out[6]: <IPython.core.display.HTML object>
```

2.4.2 Creating STIX Objects With Defaults

To create STIX objects with default values for certain properties, use an *ObjectFactory*. For instance, say we want all objects we create to have a `created_by_ref` property pointing to the `Identity` object representing our organization.

```
In [7]: from stix2 import Indicator, ObjectFactory

        factory = ObjectFactory(created_by_ref="identity--311b2d2d-f010-5473-83ec-1edf84858f4c")
```

Once you've set up the *ObjectFactory*, use its *create()* method, passing in the class for the type of object you wish to create, followed by the other properties and their values for the object.

```
In [8]: ind = factory.create(Indicator,
                             labels=["malicious-activity"],
                             pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']")
        print(ind)
```

```
Out[8]: <IPython.core.display.HTML object>
```

All objects we create with that *ObjectFactory* will automatically get the default value for `created_by_ref`. These are the properties for which defaults can be set:

- `created_by_ref`
- `created`
- `external_references`
- `object_marking_refs`

These defaults can be bypassed. For example, say you have an *Environment* with multiple default values but want to create an object with a different value for `created_by_ref`, or none at all.

```
In [9]: factory2 = ObjectFactory(created_by_ref="identity--311b2d2d-f010-5473-83ec-1edf84858f4c",
                                 created="2017-09-25T18:07:46.255472Z")
        env2 = Environment(factory=factory2)

        ind2 = env2.create(Indicator,
                           created_by_ref=None,
                           labels=["malicious-activity"],
                           pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']")
        print(ind2)
```

```
Out[9]: <IPython.core.display.HTML object>
```

```
In [10]: ind3 = env2.create(Indicator,
                            created_by_ref="identity--962cabe5-f7f3-438a-9169-585a8c971d12",
                            labels=["malicious-activity"],
                            pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']")
         print(ind3)
```

```
Out[10]: <IPython.core.display.HTML object>
```

For the full power of the Environment layer, create an *Environment* with both a *DataStore*/*Source*/*Sink* and an *ObjectFactory*:

```
In [11]: environ = Environment(ObjectFactory(created_by_ref="identity--311b2d2d-f010-5473-83ec-1edf84
                               MemoryStore())

         i = environ.create(Indicator,
                            labels=["malicious-activity"],
                            pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']")
         environ.add(i)
         print(environ.get(i.id))
```

```
Out[11]: <IPython.core.display.HTML object>
```

2.5 FileSystem

The FileSystem suite contains *FileSystemStore*, *FileSystemSource* and *FileSystemSink*. Under the hood, all FileSystem objects point to a file directory (on disk) that contains STIX2 content.

The directory and file structure of the intended STIX2 content should be:

```
stix2_content/
    /STIX2 Domain Object type
        STIX2 Domain Object
        STIX2 Domain Object
            .
            .
            .
    /STIX2 Domain Object type
        STIX2 Domain Object
        STIX2 Domain Object
            .
            .
            .
        .
        .
        .
    /STIX2 Domain Object type
```

The master STIX2 content directory contains subdirectories, each of which aligns to a STIX2 domain object type (i.e. "attack-pattern", "campaign", "malware", etc.). Within each STIX2 domain object subdirectory are JSON files that are STIX2 domain objects of the specified type. The name of the json files correspond to the ID of the STIX2 domain object found within that file. A real example of the FileSystem directory structure:

```
stix2_content/
    /attack-pattern
        attack-pattern--00d0b012-8a03-410e-95de-5826bf542de6.json
        attack-pattern--0a3ead4e-6d47-4ccb-854c-a6a4f9d96b22.json
        attack-pattern--1b7ba276-eedc-4951-a762-0ceea2c030ec.json
    /campaign
    /course-of-action
        course-of-action--2a8de25c-f743-4348-b101-3ee33ab5871b.json
        course-of-action--2c3ce852-06a2-40ee-8fe6-086f6402a739.json
    /identity
        identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5.json
    /indicator
    /intrusion-set
    /malware
        malware--1d808f62-cf63-4063-9727-ff6132514c22.json
        malware--2eb9b131-d333-4a48-9eb4-d8dec46c19ee.json
    /observed-data
    /report
    /threat-actor
    /vulnerability
```

FileSystemStore is intended for use cases where STIX2 content is retrieved and pushed to the same file directory. As *FileSystemStore* is just a wrapper around a paired *FileSystemSource* and *FileSystemSink* that point the same file directory.

For use cases where STIX2 content will only be retrieved or pushed, then a *FileSystemSource* and *FileSystemSink* can be used individually. They can also be used individually when STIX2 content will be retrieved from one distinct file directory and pushed to another.

2.5.1 FileSystem API

A note on *get()*, *all_versions()*, and *query()*: The format of the STIX2 content targeted by the FileSystem suite is JSON files. When the *FileSystemStore* retrieves STIX2 content (in JSON) from disk, it will attempt to parse the content into

full-featured python-stix2 objects and returned as such.

A note on *add()*: When STIX content is added (pushed) to the file system, the STIX content can be supplied in the following forms: Python STIX objects, Python dictionaries (of valid STIX objects or Bundles), JSON-encoded strings (of valid STIX objects or Bundles), or a (Python) list of any of the previously listed types. Any of the previous STIX content forms will be converted to a STIX JSON object (in a STIX Bundle) and written to disk.

2.5.2 FileSystem Examples

FileSystemStore

```
In [10]: from stix2 import FileSystemStore

         """
         Working with the FileSystemStore, where STIX content can be retrieved and pushed to a file s
         """

         # create FileSystemStore
         fs = FileSystemStore("/home/michael/Desktop/sample_stix2_data")

         # retrieve STIX2 content from FileSystemStore
         ap = fs.get("attack-pattern--00d0b012-8a03-410e-95de-5826bf542de6")
         mal = fs.get("malware--00c3bfcb-99bd-4767-8c03-b08f585f5c8a")

         # for visual purposes
         print(mal)
```


```
{
    "type": "malware",
    "id": "malware--00c3bfcb-99bd-4767-8c03-b08f585f5c8a",
    "created_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5",
    "created": "2017-05-31T21:33:19.746Z",
    "modified": "2017-05-31T21:33:19.746Z",
    "name": "PowerDuke",
    "description": "PowerDuke is a backdoor that was used by APT29 in 2016. It has primarily been del
    "labels": [
        "malware"
    ],
    "external_references": [
        {
            "source_name": "mitre-attack",
            "url": "https://attack.mitre.org/wiki/Software/S0139",
            "external_id": "S0139"
        },
        {
            "source_name": "Volexity PowerDuke November 2016",
            "description": "Adair, S.. (2016, November 9). PowerDuke: Widespread Post-Election Spear
            "url": "https://www.volexity.com/blog/2016/11/09/powerduke-post-election-spear-phishing-c
        }
    ],
    "object_marking_refs": [
        "marking-definition--fa42a846-8d90-4e51-bc29-71d5b4802168"
    ]
}
```


```
In [2]: from stix2 import ThreatActor, Indicator

         # create new STIX threat-actor
```

```
        ta = ThreatActor(name="Adjective Bear",
                         labels=["nation-state"],
                         sophistication="innovator",
                         resource_level="government",
                         goals=[
                             "compromising media outlets",
                             "water-hole attacks geared towards political, military targets",
                             "intelligence collection"
                         ])

        # create new indicators
        ind = Indicator(description="Crusades C2 implant",
                         labels=["malicious-activity"],
                         pattern="[file:hashes.'SHA-256' = '54b7e05e39a59428743635242e4a867c932140a99

        ind1 = Indicator(description="Crusades C2 implant 2",
                          labels=["malicious-activity"],
                          pattern="[file:hashes.'SHA-256' = '64c7e05e40a59511743635242e4a867c932140a9

        # add STIX object (threat-actor) to FileSystemStore
        fs.add(ta)

        # can also add multiple STIX objects to FileSystemStore in one call
        fs.add([ind, ind1])
```

FileSystemSource - (if STIX content is only to be retrieved from FileSystem)

```
In [4]: from stix2 import FileSystemSource
        """
        Working with FileSystemSource for retrieving STIX content.
        """

        # create FileSystemSource
        fs_source = FileSystemSource("/home/michael/Desktop/sample_stix2_data")

        # retrieve STIX 2 objects
        ap = fs_source.get("attack-pattern--00d0b012-8a03-410e-95de-5826bf542de6")

        # for visual purposes
        print(ap)
```


```
{
    "type": "attack-pattern",
    "id": "attack-pattern--00d0b012-8a03-410e-95de-5826bf542de6",
    "created_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5",
    "created": "2017-05-31T21:30:54.176Z",
    "modified": "2017-05-31T21:30:54.176Z",
    "name": "Indicator Removal from Tools",
    "description": "If a malicious...command-line parameters, Process monitoring",
    "kill_chain_phases": [
        {
            "kill_chain_name": "mitre-attack",
            "phase_name": "defense-evasion"
        }
    ],
    "external_references": [
        {
            "source_name": "mitre-attack",
```

```
            "url": "https://attack.mitre.org/wiki/Technique/T1066",
            "external_id": "T1066"
        }
    ],
    "object_marking_refs": [
        "marking-definition--fa42a846-8d90-4e51-bc29-71d5b4802168"
    ]
}
```

```
In [5]: from stix2 import Filter

        # create filter for type=malware
        query = [Filter("type", "=", "malware")]

        # query on the filter
        mals = fs_source.query(query)

        for mal in mals:
            print(mal)
```

```
{
    "type": "malware",
    "id": "malware--0f862b01-99da-47cc-9bdb-db4a86a95bb1",
    "created_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5",
    "created": "2017-05-31T21:32:54.772Z",
    "modified": "2017-05-31T21:32:54.772Z",
    "name": "Emissary",
    "description": "Emissary is a Trojan that has been used by Lotus Blossom. It shares code with El
    "labels": [
        "malware"
    ],
    "external_references": [
        {
            "source_name": "mitre-attack",
            "url": "https://attack.mitre.org/wiki/Software/S0082",
            "external_id": "S0082"
        },
        {
            "source_name": "Lotus Blossom Dec 2015",
            "description": "Falcone, R. and Miller-Osborn, J.. (2015, December 18). Attack on French
            "url": "http://researchcenter.paloaltonetworks.com/2015/12/attack-on-french-diplomat-lin
        }
    ],
    "object_marking_refs": [
        "marking-definition--fa42a846-8d90-4e51-bc29-71d5b4802168"
    ]
}
{
    "type": "malware",
    "id": "malware--2a6f4c7b-e690-4cc7-ab6b-1f821fb6b80b",
    "created_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5",
    "created": "2017-05-31T21:32:33.348Z",
    "modified": "2017-05-31T21:32:33.348Z",
    "name": "LOWBALL",
    "description": "LOWBALL is malware used by admin@338. It was used in August 2015 in email message
    "labels": [
        "malware"
    ],
    "external_references": [
        {
```

```
            "source_name": "mitre-attack",
            "url": "https://attack.mitre.org/wiki/Software/S0042",
            "external_id": "S0042"
        },
        {
            "source_name": "FireEye admin@338",
            "description": "FireEye Threat Intelligence. (2015, December 1). China-based Cyber Threat
            "url": "https://www.fireeye.com/blog/threat-research/2015/11/china-based-threat.html"
        }
    ],
    "object_marking_refs": [
        "marking-definition--fa42a846-8d90-4e51-bc29-71d5b4802168"
    ]
}
{
    "type": "malware",
    "id": "malware--00c3bfcb-99bd-4767-8c03-b08f585f5c8a",
    "created_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5",
    "created": "2017-05-31T21:33:19.746Z",
    "modified": "2017-05-31T21:33:19.746Z",
    "name": "PowerDuke",
    "description": "PowerDuke is a backdoor that was used by APT29 in 2016. It has primarily been del
    "labels": [
        "malware"
    ],
    "external_references": [
        {
            "source_name": "mitre-attack",
            "url": "https://attack.mitre.org/wiki/Software/S0139",
            "external_id": "S0139"
        },
        {
            "source_name": "Volexity PowerDuke November 2016",
            "description": "Adair, S.. (2016, November 9). PowerDuke: Widespread Post-Election Spear
            "url": "https://www.volexity.com/blog/2016/11/09/powerduke-post-election-spear-phishing-c
        }
    ],
    "object_marking_refs": [
        "marking-definition--fa42a846-8d90-4e51-bc29-71d5b4802168"
    ]
}
{
    "type": "malware",
    "id": "malware--0db09158-6e48-4e7c-8ce7-2b10b9c0c039",
    "created_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5",
    "created": "2017-05-31T21:32:55.126Z",
    "modified": "2017-05-31T21:32:55.126Z",
    "name": "Misdat",
    "description": "Misdat is a backdoor that was used by Dust Storm from 2010 to 2011.[[Citation: Cy
    "labels": [
        "malware"
    ],
    "external_references": [
        {
            "source_name": "mitre-attack",
            "url": "https://attack.mitre.org/wiki/Software/S0083",
            "external_id": "S0083"
        },
        {
```

```
            "source_name": "Cylance Dust Storm",
            "description": "Gross, J. (2016, February 23). Operation Dust Storm. Retrieved February 2
            "url": "https://www.cylance.com/hubfs/2015%20cylance%20website/assets/operation-dust-stor
        }
    ],
    "object_marking_refs": [
        "marking-definition--fa42a846-8d90-4e51-bc29-71d5b4802168"
    ]
}
{
    "type": "malware",
    "id": "malware--1d808f62-cf63-4063-9727-ff6132514c22",
    "created_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5",
    "created": "2017-05-31T21:33:06.433Z",
    "modified": "2017-05-31T21:33:06.433Z",
    "name": "WEBC2",
    "description": "WEBC2 is a backdoor used by APT1 to retrieve a Web page from a predetermined C2 s
    "labels": [
        "malware"
    ],
    "external_references": [
        {
            "source_name": "mitre-attack",
            "url": "https://attack.mitre.org/wiki/Software/S0109",
            "external_id": "S0109"
        },
        {
            "source_name": "Mandiant APT1 Appendix",
            "description": "Mandiant. (n.d.). Appendix C (Digital) - The Malware Arsenal. Retrieved J
            "url": "https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-repor
        }
    ],
    "object_marking_refs": [
        "marking-definition--fa42a846-8d90-4e51-bc29-71d5b4802168"
    ]
}
```

```
In [6]: # add more filters to the query
        query.append(Filter("modified", ">" , "2017-05-31T21:33:10.772474Z"))

        mals = fs_source.query(query)

        # for visual purposes
        for mal in mals:
            print(mal)
```

```
{
    "type": "malware",
    "id": "malware--00c3bfcb-99bd-4767-8c03-b08f585f5c8a",
    "created_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5",
    "created": "2017-05-31T21:33:19.746Z",
    "modified": "2017-05-31T21:33:19.746Z",
    "name": "PowerDuke",
    "description": "PowerDuke is a backdoor that was used by APT29 in 2016. It has primarily been del
    "labels": [
        "malware"
    ],
    "external_references": [
        {
            "source_name": "mitre-attack",
```

```
            "url": "https://attack.mitre.org/wiki/Software/S0139",
            "external_id": "S0139"
        },
        {
            "source_name": "Volexity PowerDuke November 2016",
            "description": "Adair, S.. (2016, November 9). PowerDuke: Widespread Post-Election Spear
            "url": "https://www.volexity.com/blog/2016/11/09/powerduke-post-election-spear-phishing-c
        }
    ],
    "object_marking_refs": [
        "marking-definition--fa42a846-8d90-4e51-bc29-71d5b4802168"
    ]
}
```

FileSystemSink - (if STIX content is only to be pushed to FileSystem)

```
In [7]: from stix2 import FileSystemSink, Campaign
        """
        Working with FileSystemSink for pushing STIX content.
        """
        # create FileSystemSink
        fs_sink = FileSystemSink("/home/michael/Desktop/sample_stix2_data")

        # create STIX objects and add to sink
        camp = Campaign(name="The Crusades",
                        objective="Infiltrating Israeli, Iranian and Palestinian digital infrastructu
                        aliases=["Desert Moon"])

        ind = Indicator(description="Crusades C2 implant",
                        labels=["malicious-activity"],
                        pattern="[file:hashes.'SHA-256' = '54b7e05e39a59428743635242e4a867c932140a99

        ind1 = Indicator(description="Crusades C2 implant",
                         labels=["malicious-activity"],
                         pattern="[file:hashes.'SHA-256' = '54b7e05e39a59428743635242e4a867c932140a9

        # add Campaign object to FileSystemSink
        fs_sink.add(camp)

        # can also add STIX objects to FileSystemSink in on call
        fs_sink.add([ind, ind1])
```

2.6 Data Markings

2.6.1 Creating Objects With Data Markings

To create an object with a (predefined) TLP marking to an object, just provide it as a keyword argument to the constructor. The TLP markings can easily be imported from python-stix2.

```
In [3]: from stix2 import Indicator, TLP_AMBER

        indicator = Indicator(labels=["malicious-activity"],
                              pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']",
                              object_marking_refs=TLP_AMBER)
        print(indicator)
```

```
Out[3]: <IPython.core.display.HTML object>
```

If you're creating your own marking (for example, a `Statement` marking), first create the statement marking:

```
In [7]: from stix2 import MarkingDefinition, StatementMarking

        marking_definition = MarkingDefinition(
            definition_type="statement",
            definition=StatementMarking(statement="Copyright 2017, Example Corp")
        )
        print(marking_definition)
```

```
Out[7]: <IPython.core.display.HTML object>
```

Then you can add it to an object as it's being created (passing either full object or the the ID as a keyword argument, like with relationships).

```
In [5]: indicator2 = Indicator(labels=["malicious-activity"],
                               pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']",
                               object_marking_refs=marking_definition)
        print(indicator2)
```

```
Out[5]: <IPython.core.display.HTML object>
```

```
In [6]: indicator3 = Indicator(labels=["malicious-activity"],
                               pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']",
                               object_marking_refs="marking-definition--f88d31f6-486f-44da-b317-01333
        print(indicator3)
```

```
Out[6]: <IPython.core.display.HTML object>
```

Granular markings work in the same way, except you also need to provide a full granular-marking object (including the selector).

```
In [8]: from stix2 import Malware, TLP_WHITE

        malware = Malware(name="Poison Ivy",
                          labels=['remote-access-trojan'],
                          description="A ransomware related to ...",
                          granular_markings=[
                              {
                                  "selectors": ["description"],
                                  "marking_ref": marking_definition
                              },
                              {
                                  "selectors": ["name"],
                                  "marking_ref": TLP_WHITE
                              }
                          ])
        print(malware)
```

```
Out[8]: <IPython.core.display.HTML object>
```

Make sure that the selector is a field that exists and is populated on the object, otherwise this will cause an error:

```
In [8]: Malware(name="Poison Ivy",
                labels=['remote-access-trojan'],
                description="A ransomware related to ...",
                granular_markings=[
                    {
                        "selectors": ["title"],
                        "marking_ref": marking_definition
                    }
                ])
```

```
InvalidSelectorError: Selector title in Malware is not valid!
```

2.6.2 Adding Data Markings To Existing Objects

Several functions exist to support working with data markings.

Both object markings and granular markings can be added to STIX objects which have already been created.

Note: Doing so will create a new version of the object (note the updated `modified` time).

```
In [21]: indicator4 = indicator.add_markings(marking_definition)
         print(indicator4)
```

```
Out[21]: <IPython.core.display.HTML object>
```

You can also remove specific markings from STIX objects. This will also create a new version of the object.

```
In [22]: indicator5 = indicator4.remove_markings(marking_definition)
         print(indicator5)
```

```
Out[22]: <IPython.core.display.HTML object>
```

The markings on an object can be replaced with a different set of markings:

```
In [23]: from stix2 import TLP_GREEN

         indicator6 = indicator5.set_markings([TLP_GREEN, marking_definition])
         print(indicator6)
```

```
Out[23]: <IPython.core.display.HTML object>
```

STIX objects can also be cleared of all markings with *clear_markings()*:

```
In [12]: indicator7 = indicator5.clear_markings()
         print(indicator7)
```

```
Out[12]: <IPython.core.display.HTML object>
```

All of these functions can be used for granular markings by passing in a list of selectors. Note that they will create new versions of the objects.

2.6.3 Evaluating Data Markings

You can get a list of the object markings on a STIX object:

```
In [19]: indicator6.get_markings()
```

```
Out[19]: ['marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da',
          'marking-definition--030bb5c6-c5eb-4e9c-8e7a-b9aab08ded53']
```

To get a list of the granular markings on an object, pass the object and a list of selectors to *get_markings()*:

```
In [9]: from stix2 import get_markings

        get_markings(malware, 'name')
```

```
Out[9]: ['marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9']
```

You can also call *get_markings()* as a method on the STIX object.

```
In [14]: malware.get_markings('name')
```

```
Out[14]: ['marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9']
```

Finally, you may also check if an object is marked by a specific markings. Again, for granular markings, pass in the selector or list of selectors.

```
In [16]: indicator.is_marked(TLP_AMBER.id)

Out[16]: True

In [17]: malware.is_marked(TLP_WHITE.id, 'name')

Out[17]: True

In [18]: malware.is_marked(TLP_WHITE.id, 'description')

Out[18]: False
```

2.7 Memory

The Memory suite consists of *MemoryStore*, *MemorySource*, and *MemorySink*. Under the hood, the Memory suite points to an in-memory dictionary. Similarly, the *MemoryStore* is a just a wrapper around a paired *MemorySource* and *MemorySink*; as there is quite limited uses for just a *MemorySource* or a *MemorySink*, it is recommended to always use *MemoryStore*. The *MemoryStore* is intended for retrieving/searching and pushing STIX content to memory. It is important to note that all STIX content in memory is not backed up on the file system (disk), as that functionality is encompassed within the *FileSystemStore*. However, the Memory suite does provide some utility methods for saving and loading STIX content to disk. *MemoryStore.save_to_file()* allows for saving all the STIX content that is in memory to a json file. *MemoryStore.load_from_file()* allows for loading STIX content from a JSON-formatted file.

2.7.1 Memory API

A note on adding and retreiving STIX content to the Memory suite: As mentioned, under the hood the Memory suite is an internal, in-memory dictionary. STIX content that is to be added can be in the following forms: python-stix2 objects, (Python) dictionaries (of valid STIX objects or Bundles), JSON-encoded strings (of valid STIX objects or Bundles), or a (Python) list of any of the previously listed types. *MemoryStore* actually stores STIX content either as python-stix2 objects or as (Python) dictionaries, reducing and converting any of the aforementioned types to one of those. Additionally, whatever form the STIX object is stored as, is how it will be returned when retrieved. python-stix2 objects, and json-encoded strings (of STIX content) are stored as python-stix2 objects, while (Python) dictionaries (of STIX objects) are stored as (Python) dictionaries.

A note on *load_from_file()*: For *load_from_file()*, STIX content is assumed to be in JSON form within the file, as an individual STIX object or in a Bundle. When the JSON is loaded, the STIX objects are parsed into python-stix2 objects before being stored in the in-memory dictionary.

A note on *save_to_file()*: This method dumps all STIX content that is in the *MemoryStore* to the specified file. The file format will be JSON, and the STIX content will be within a STIX Bundle. Note also that the output form will be a JSON STIX Bundle regardless of the form that the individual STIX objects are stored in (i.e. supplied to) the *MemoryStore*.

2.7.2 Memory Examples

MemoryStore

```
In [3]: from stix2 import MemoryStore, Indicator

        # create default MemoryStore
        mem = MemoryStore()
```

```
        # insert newly created indicator into memory
        ind = Indicator(description="Crusades C2 implant",
                        labels=["malicious-activity"],
                        pattern="[file:hashes.'SHA-256' = '54b7e05e39a59428743635242e4a867c932140a99

        mem.add(ind)

        # for visual purposes
        print(mem.get(ind.id))
```

Out[3]: <IPython.core.display.HTML object>

```
In [4]: from stix2 import Malware

        # add multiple STIX objects into memory
        ind2 = Indicator(description="Crusades stage 2 implant",
                         labels=["malicious-activity"],
                         pattern="[file:hashes.'SHA-256' = '70fa62fb218dd9d936ee570dbe531dfa4e7c128f
        ind3 = Indicator(description="Crusades stage 2 implant variant",
                         labels=["malicious-activity"],
                         pattern="[file:hashes.'SHA-256' = '31a45e777e4d58b97f4c43e38006f8cd6580ddab
        mal = Malware(labels=["rootkit"], name= "Alexios")

        mem.add([ind2,ind3, mal])

        # for visual purposes
        print(mem.get(ind3.id))
```

Out[4]: <IPython.core.display.HTML object>

```
In [5]: from stix2 import Filter

        mal = mem.query([Filter("labels","=", "rootkit")])[0]
        print(mal)
```

Out[5]: <IPython.core.display.HTML object>

```
In [6]: from stix2 import Filter

        # add json formatted string to MemoryStore
        # Again, would NOT manually create json-formatted string
        # but taken as an output form from another source
        report = '{"type": "report","id": "report--2add14d6-bbf3-4308-bb8e-226d314a08e4","labels": ["

        mem.add(report)

        print(mem.get("report--2add14d6-bbf3-4308-bb8e-226d314a08e4"))
```

Out[6]: <IPython.core.display.HTML object>

2.7.3 load_from_file() and save_to_file()

```
In [8]: mem_2 = MemoryStore()

        # save (dump) all STIX content in MemoryStore to json file
        mem.save_to_file("path_to_target_file.json")

        # load(add) STIX content from json file into MemoryStore
        mem_2.load_from_file("path_to_target_file.json")
```

```
report = mem_2.get("report--2add14d6-bbf3-4308-bb8e-226d314a08e4")

# for visualpurposes
print(report)
```

```
Out[8]: <IPython.core.display.HTML object>
```

2.8 Parsing STIX Content

Parsing STIX content is as easy as calling the *parse()* function on a JSON string. It will automatically determine the type of the object. The STIX objects within `bundle` objects, and the cyber observables contained within `observed-data` objects will be parsed as well.

```
In [3]: from stix2 import parse

input_string = """{
    "type": "observed-data",
    "id": "observed-data--b67d30ff-02ac-498a-92f9-32f845f448cf",
    "created": "2016-04-06T19:58:16.000Z",
    "modified": "2016-04-06T19:58:16.000Z",
    "first_observed": "2015-12-21T19:00:00Z",
    "last_observed": "2015-12-21T19:00:00Z",
    "number_observed": 50,
    "objects": {
        "0": {
            "type": "file",
            "hashes": {
                "SHA-256": "0969de02ecf8a5f003e3f6d063d848c8a193aada092623f8ce408c15bcb5f038"
            }
        }
    }
}"""

obj = parse(input_string)
print(obj.type)
print(obj.objects["0"].hashes['SHA-256'])
```

```
observed-data
0969de02ecf8a5f003e3f6d063d848c8a193aada092623f8ce408c15bcb5f038
```

2.9 Serializing STIX Objects

The string representation of all STIX classes is a valid STIX JSON object.

```
In [2]: from stix2 import Indicator

indicator = Indicator(name="File hash for malware variant",
                      labels=["malicious-activity"],
                      pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']")

print(str(indicator))
```

```
Out[2]: <IPython.core.display.HTML object>
```

However, the string representation can be slow, as it sorts properties to be in a more readable order. If you need performance and don't care about the human-readability of the output, use the object's serialize() function:

```
In [6]: print(indicator.serialize())
```

```
Out[6]: <IPython.core.display.HTML object>
```

2.10 TAXIICollection

The TAXIICollection suite contains *TAXIICollectionStore*, *TAXIICollectionSource*, and *TAXIICollectionSink*. *TAXIICollectionStore* pushes and retrieves STIX content to local/remote TAXII Collection(s). *TAXIICollectionSource* retrieves STIX content from local/remote TAXII Collection(s). *TAXIICollectionSink* pushes STIX content to local/remote TAXII Collection(s). Each of the interfaces is designed to be bound to a Collection from the taxii2client library (taxii2client.Collection), where all *TAXIICollection* API calls will be executed through that Collection instance.

A note on TAXII2 searching/filtering of STIX content: TAXII2 server implementations natively support searching on the STIX2 object properties: id, type and version; API requests made to TAXII2 can contain filter arguments for those 3 properties. However, the *TAXIICollection* suite supports searching on all STIX2 common object properties (see *Filters* documentation for full listing). This works simply by augmenting the filtering that is done remotely at the TAXII2 server instance. *TAXIICollection* will seperate any supplied queries into TAXII supported filters and non-supported filters. During a *TAXIICollection* API call, TAXII2 supported filters get inserted into the TAXII2 server request (to be evaluated at the server). The rest of the filters are kept locally and then applied to the STIX2 content that is returned from the TAXII2 server, before being returned from the *TAXIICollection* API call.

2.10.1 TAXIICollection API

2.10.2 TAXIICollection Examples

TAXIICollectionSource

```
In [18]: from stix2 import TAXIICollectionSource
         from taxii2client import Collection

         # establish TAXII2 Collection instance
         collection = Collection("http://127.0.0.1:5000/trustgroup1/collections/91a7b528-80eb-42ed-a7
         # supply the TAXII2 collection to TAXIICollection
         tc_source = TAXIICollectionSource(collection)

         #retrieve STIX objects by id
         stix_obj = tc_source.get("malware--fdd60b30-b67c-11e3-b0b9-f01faf20d111")
         stix_obj_versions = tc_source.all_versions("indicator--a932fcc6-e032-176c-126f-cb970a5a1ade"

         #for visual purposes
         print(stix_obj)
         print("-------")
         for so in stix_obj_versions:
             print(so)
```

```
{
    "type": "malware",
    "id": "malware--fdd60b30-b67c-11e3-b0b9-f01faf20d111",
    "created": "2017-01-27T13:49:53.997Z",
    "modified": "2017-01-27T13:49:53.997Z",
    "name": "Poison Ivy",
    "description": "Poison Ivy",
    "labels": [
        "remote-access-trojan"
```

```
    ]
}
-------
{
    "type": "indicator",
    "id": "indicator--a932fcc6-e032-176c-126f-cb970a5a1ade",
    "created": "2014-05-08T09:00:00.000Z",
    "modified": "2014-05-08T09:00:00.000Z",
    "name": "File hash for Poison Ivy variant",
    "pattern": "[file:hashes.'SHA-256' = 'ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c863
    "valid_from": "2014-05-08T09:00:00Z",
    "labels": [
        "file-hash-watchlist"
    ]
}
```

```
In [20]: from stix2 import Filter

         # retrieve multiple object from TAXIICollectionSource
         # by using filters
         f1 = Filter("type","=", "indicator")

         indicators = tc_source.query([f1])

         #for visual purposes
         for indicator in indicators:
             print(indicator)
```

```
{
    "type": "indicator",
    "id": "indicator--a932fcc6-e032-176c-126f-cb970a5a1ade",
    "created": "2014-05-08T09:00:00.000Z",
    "modified": "2014-05-08T09:00:00.000Z",
    "name": "File hash for Poison Ivy variant",
    "pattern": "[file:hashes.'SHA-256' = 'ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c863
    "valid_from": "2014-05-08T09:00:00Z",
    "labels": [
        "file-hash-watchlist"
    ]
}
```

TAXIICollectionSink

```
In [ ]: from stix2 import TAXIICollectionSink, ThreatActor

        #create TAXIICollectionSINK and push STIX content to it
        tc_sink = TAXIICollectionSink(collection)

        # create new STIX threat-actor
        ta = ThreatActor(name="Teddy Bear",
                         labels=["nation-state"],
                         sophistication="innovator",
                         resource_level="government",
                         goals=[
                             "compromising environment NGOs",
                             "water-hole attacks geared towards energy sector",
                         ])

        tc_sink.add(ta)
```

TAXIICollectionStore

```
In [19]: from stix2 import TAXIICollectionStore

         # create TAXIICollectionStore - note the same collection instance can
         # be used for the store
         tc_store = TAXIICollectionStore(collection)

         # retrieve STIX object by id from TAXII Collection through
         # TAXIICollectionStore
         stix_obj2 = tc_source.get("malware--fdd60b30-b67c-11e3-b0b9-f01faf20d111")

         print(stix_obj2)
```

```
{
    "type": "malware",
    "id": "malware--fdd60b30-b67c-11e3-b0b9-f01faf20d111",
    "created": "2017-01-27T13:49:53.997Z",
    "modified": "2017-01-27T13:49:53.997Z",
    "name": "Poison Ivy",
    "description": "Poison Ivy",
    "labels": [
        "remote-access-trojan"
    ]
}
```

```
In [ ]: from stix2 import indicator

        # add STIX object to TAXIICollectionStore
        ind = Indicator(description="Smokey Bear implant",
                        labels=["malicious-activity"],
                        pattern="[file:hashes.'SHA-256' = '09c7e05a39a59428743635242e4a867c932140a909

        tc_store.add(ind)
```

2.11 Technical Specification Support

2.11.1 How imports will work

Imports can be used in different ways depending on the use case and support levels.

People who want to support the latest version of STIX 2.X without having to make changes, can implicitly use the latest version:

```
In [ ]: import stix2

        stix2.Indicator()
```

or,

```
In [ ]: from stix2 import Indicator

        Indicator()
```

People who want to use an explicit version:

```
In [ ]: import stix2.v20

        stix2.v20.Indicator()
```

or,

```
In [ ]: from stix2.v20 import Indicator

        Indicator()
```

or even,

```
In [ ]: import stix2.v20 as stix2

        stix2.Indicator()
```

The last option makes it easy to update to a new version in one place per file, once you've made the deliberate action to do this.

People who want to use multiple versions in a single file:

```
In [ ]: import stix2

        stix2.v20.Indicator()
        stix2.v21.Indicator()
```

or,

```
In [ ]: from stix2 import v20, v21

        v20.Indicator()
        v21.Indicator()
```

or (less preferred):

```
In [ ]: from stix2.v20 import Indicator as Indicator_v20
        from stix2.v21 import Indicator as Indicator_v21

        Indicator_v20()
        Indicator_v21()
```

2.11.2 How parsing will work

If the `version` positional argument is not provided. The data will be parsed using the latest version of STIX 2.X supported by the `stix2` library.

You can lock your *parse()* method to a specific STIX version by:

```
In [3]: from stix2 import parse

        indicator = parse("""{
            "type": "indicator",
            "id": "indicator--dbcbd659-c927-4f9a-994f-0a2632274394",
            "created": "2017-09-26T23:33:39.829Z",
            "modified": "2017-09-26T23:33:39.829Z",
            "labels": [
                "malicious-activity"
            ],
            "name": "File hash for malware variant",
            "pattern": "[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']",
            "valid_from": "2017-09-26T23:33:39.829952Z"
```

```
        }""", version="2.0")
        print(indicator)
```

```
Out[3]: <IPython.core.display.HTML object>
```

Keep in mind that if a 2.1 or higher object is parsed, the operation will fail.

2.11.3 How will custom content work

CustomObject, *CustomObservable*, *CustomMarking* and *CustomExtension* must be registered explicitly by STIX version. This is a design decision since properties or requirements may change as the STIX Technical Specification advances.

You can perform this by:

```
In [ ]: import stix2

        # Make my custom observable available in STIX 2.0
        @stix2.v20.CustomObservable('x-new-object-type',
                                    (("prop", stix2.properties.BooleanProperty())))
        class NewObject2(object):
            pass


        # Make my custom observable available in STIX 2.1
        @stix2.v21.CustomObservable('x-new-object-type',
                                    (("prop", stix2.properties.BooleanProperty())))
        class NewObject2(object):
            pass
```

2.12 Versioning

To create a new version of an existing object, specify the property(ies) you want to change and their new values:

```
In [3]: from stix2 import Indicator

        indicator = Indicator(created="2016-01-01T08:00:00.000Z",
                              name="File hash for suspicious file",
                              labels=["anomalous-activity"],
                              pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']")

        indicator2 = indicator.new_version(name="File hash for Foobar malware",
                                           labels=["malicious-activity"])
        print(indicator2)
```

```
Out[3]: <IPython.core.display.HTML object>
```

The modified time will be updated to the current time unless you provide a specific value as a keyword argument. Note that you can't change the `type`, `id`, or `created` properties.

```
In [4]: indicator.new_version(id="indicator--cc42e358-8b9b-493c-9646-6ecd73b41c21")
```

```
UnmodifiablePropertyError: These properties cannot be changed when making a new version: id.
```

To revoke an object:

```
In [5]: indicator2 = indicator2.revoke()
        print(indicator2)
```

```
Out[5]: <IPython.core.display.HTML object>
```

CHAPTER 3

API Reference

This section of documentation contains information on all of the classes and functions in the `stix2` API, as given by the package's docstrings.

Note: All the classes and functions detailed in the pages below are importable directly from *stix2*. See also: *How imports will work*.

Python APIs for STIX 2.

core	STIX 2.0 Objects that are neither SDOs nor SROs.
datastore	Python STIX 2.0 DataStore API
environment	
exceptions	STIX 2 error classes.
markings	Functions for working with STIX 2 Data Markings.
patterns	Classes to aid in working with the STIX 2 patterning language.
properties	Classes for representing properties of STIX Objects and Cyber Observables.
utils	Utility functions and classes for the stix2 library.
v20.common	STIX 2 Common Data Types and Properties.
v20.observables	STIX 2.0 Cyber Observable Objects.
v20.sdo	STIX 2.0 Domain Objects
v20.sro	STIX 2.0 Relationship Objects.

3.1 core

STIX 2.0 Objects that are neither SDOs nor SROs.

class Bundle(**args*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **spec_version** (**)
- **objects** (*List of STIX Objects*)

class STIXObjectProperty(*allow_custom=False*)

clean(*value*)

parse(*data*, *allow_custom=False*, *version=None*)
Deserialize a string or file-like object into a STIX object.

Parameters

- **data** (*str, dict, file-like object*) – The STIX 2 content to be parsed.
- **allow_custom** (*bool*) – Whether to allow custom properties or not. Default: False.
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

Returns An instantiated Python STIX object.

3.2 datastore

Python STIX 2.0 DataStore API

filesystem	Python STIX 2.0 FileSystem Source/Sink
filters	Filters for Python STIX 2.0 DataSources, DataSinks, DataStores
memory	Python STIX 2.0 Memory Source/Sink
taxii	Python STIX 2.x TAXIICollectionStore

3.2.1 filesystem

Python STIX 2.0 FileSystem Source/Sink

class FileSystemSink(*stix_dir*, *allow_custom=False*, *bundlify=False*)
Interface for adding/pushing STIX objects to file directory of STIX objects.

Can be paired with a FileSystemSource, together as the two components of a FileSystemStore.

Parameters

- **stix_dir** (*str*) – path to directory of STIX objects.
- **allow_custom** (*bool*) – Whether to allow custom STIX content to be added to the FileSystemSource. Default: False
- **bundlify** (*bool*) – Whether to wrap objects in bundles when saving them. Default: False.

add(*stix_data=None*, *version=None*)
Add STIX objects to file directory.

Parameters

- **stix_data** (*STIX object OR dict OR str OR list*) – valid STIX 2.0 content in a STIX object (or list of), dict (or list of), or a STIX 2.0 json encoded string.
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

Note: `stix_data` can be a Bundle object, but each object in it will be saved separately; you will be able to retrieve any of the objects the Bundle contained, but not the Bundle itself.

---

stix_dir

class FileSystemSource(*stix_dir*, *allow_custom=True*)

Interface for searching/retrieving STIX objects from a STIX object file directory.

Can be paired with a FileSystemSink, together as the two components of a FileSystemStore.

Parameters

- **stix_dir** (*str*) – path to directory of STIX objects
- **allow_custom** (*bool*) – Whether to allow custom STIX content to be added to the FileSystemSink. Default: True

all_versions(*stix_id*, *version=None*, *_composite_filters=None*)

Retrieve STIX object from file directory via STIX ID, all versions.

Note: Since FileSystem sources/sinks don't handle multiple versions of a STIX object, this operation is unnecessary. Pass call to get().

Parameters

- **stix_id** (*str*) – The STIX ID of the STIX objects to be retrieved.
- **_composite_filters** (*set*) – set of filters passed from the parent CompositeDataSource, not user supplied
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

Returns

(list) –

of STIX objects that has the supplied STIX ID. The STIX objects are loaded from their json files, parsed into a python STIX objects and then returned

get(*stix_id*, *version=None*, *_composite_filters=None*)

Retrieve STIX object from file directory via STIX ID.

Parameters

- **stix_id** (*str*) – The STIX ID of the STIX object to be retrieved.
- **_composite_filters** (*set*) – set of filters passed from the parent CompositeDataSource, not user supplied
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

Returns

(STIX object) –

STIX object that has the supplied STIX ID. The STIX object is loaded from its json file, parsed into a python STIX object and then returned

query(*query=None*, *version=None*, *_composite_filters=None*)
Search and retrieve STIX objects based on the complete query.

A "complete query" includes the filters from the query, the filters attached to this FileSystemSource, and any filters passed from a CompositeDataSource (i.e. _composite_filters).

Parameters

- **query** (*list*) – list of filters to search on
- **_composite_filters** (*set*) – set of filters passed from the CompositeDataSource, not user supplied
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

Returns

(list) –

list of STIX objects that matches the supplied query. The STIX objects are loaded from their json files, parsed into a python STIX objects and then returned.

stix_dir

class FileSystemStore(*stix_dir*, *allow_custom=None*, *bundlify=False*)
Interface to a file directory of STIX objects.

FileSystemStore is a wrapper around a paired FileSystemSink and FileSystemSource.

Parameters

- **stix_dir** (*str*) – path to directory of STIX objects
- **allow_custom** (*bool*) – whether to allow custom STIX content to be pushed/retrieved. Defaults to True for FileSystemSource side(retrieving data) and False for FileSystemSink side(pushing data). However, when parameter is supplied, it will be applied to both FileSystemSource and FileSystemSink.
- **bundlify** (*bool*) – whether to wrap objects in bundles when saving them. Default: False.

source
FileSystemSource – FileSystemSource

sink
FileSystemSink – FileSystemSink

3.2.2 filters

Filters for Python STIX 2.0 DataSources, DataSinks, DataStores

class Filter
STIX 2 filters that support the querying functionality of STIX 2 DataStores and DataSources.

Initialized like a Python tuple.

Parameters

- **property** (*str*) – filter property name, corresponds to STIX 2 object property
- **op** (*str*) – operator of the filter

- **value** (*str*) – filter property value

Example

Filter("id", "=", "malware–0f862b01-99da-47cc-9bdb-db4a86a95bb1")

apply_common_filters(*stix_objs*, *query*)
Evaluate filters against a set of STIX 2.0 objects.

Supports only STIX 2.0 common property properties.

Parameters

- **stix_objs** (*list*) – list of STIX objects to apply the query to
- **query** (*set*) – set of filters (combined form complete query)

Yields STIX objects that successfully evaluate against the query.

FILTER_OPS = ['=', '!=', 'in', '>', '<', '>=', '<=']
Supported filter value types

3.2.3 memory

Python STIX 2.0 Memory Source/Sink

Note: Not worrying about STIX versioning. The in memory STIX data at anytime will only hold one version of a STIX object. As such, when save() is called, the single versions of all the STIX objects are what is written to file.

---

class MemorySink(*stix_data=None*, *allow_custom=True*, *version=None*, *_store=False*)
Interface for adding/pushing STIX objects to an in-memory dictionary.

Designed to be paired with a MemorySource, together as the two components of a MemoryStore.

Parameters

- **stix_data** (*dict OR list*) – valid STIX 2.0 content in bundle or a list.
- **_store** (*bool*) – whether the MemorySink is a part of a MemoryStore, in which case "stix_data" is a direct reference to shared memory with DataSource. Not user supplied
- **allow_custom** (*bool*) – whether to allow custom objects/properties when exporting STIX content to file. Default: True.

_data
dict – the in-memory dict that holds STIX objects. If part of a MemoryStore, the dict is shared with a MemorySource

add(*stix_data*, *version=None*)
Add STIX objects to MemoryStore/Sink.

Adds STIX objects to an in-memory dictionary for fast lookup. Recursive function, breaks down STIX Bundles and lists.

Parameters

- **stix_data** (*list OR dict OR STIX object*) – STIX objects to be added
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

save_to_file(*file_path*)
Write SITX objects from in-memory dictionary to JSON file, as a STIX Bundle.

Parameters **file_path** (*str*) – file path to write STIX data to

class MemorySource(*stix_data=None*, *allow_custom=True*, *version=None*, *_store=False*)
Interface for searching/retrieving STIX objects from an in-memory dictionary.

Designed to be paired with a MemorySink, together as the two components of a MemoryStore.

Parameters

- **stix_data** (*dict OR list OR STIX object*) – valid STIX 2.0 content in bundle or list.
- **_store** (*bool*) – if the MemorySource is a part of a MemoryStore, in which case "stix_data" is a direct reference to shared memory with DataSink. Not user supplied
- **allow_custom** (*bool*) – whether to allow custom objects/properties when importing STIX content from file. Default: True.

_data
dict – the in-memory dict that holds STIX objects. If part of a MemoryStore, the dict is shared with a MemorySink

all_versions(*stix_id*, *_composite_filters=None*)
Retrieve STIX objects from in-memory dict via STIX ID, all versions of it

Note: Since Memory sources/sinks don't handle multiple versions of a STIX object, this operation is unnecessary. Translate call to get().

Parameters

- **stix_id** (*str*) – The STIX ID of the STIX 2 object to retrieve.
- **_composite_filters** (*set*) – set of filters passed from the parent CompositeDataSource, not user supplied

Returns

(list) –

list of STIX objects that has the supplied ID. As the MemoryStore(i.e. MemorySink) adds STIX objects to memory as they are supplied (either as python dictionary or STIX object), it is returned in the same form as it as added

get(*stix_id*, *_composite_filters=None*)
Retrieve STIX object from in-memory dict via STIX ID.

Parameters

- **stix_id** (*str*) – The STIX ID of the STIX object to be retrieved.
- **_composite_filters** (*set*) – set of filters passed from the parent CompositeDataSource, not user supplied

Returns

(dict OR STIX object) –

STIX object that has the supplied ID. As the MemoryStore(i.e. MemorySink) adds STIX objects to memory as they are supplied (either as python dictionary or STIX object), it is returned in the same form as it as added

load_from_file(*file_path*, *version=None*)
Load STIX data from JSON file.

File format is expected to be a single JSON STIX object or JSON STIX bundle.

Parameters

- **file_path** (*str*) – file path to load STIX data from
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

query(*query=None*, *_composite_filters=None*)
Search and retrieve STIX objects based on the complete query.

A "complete query" includes the filters from the query, the filters attached to this MemorySource, and any filters passed from a CompositeDataSource (i.e. _composite_filters).

Parameters

- **query** (*list*) – list of filters to search on
- **_composite_filters** (*set*) – set of filters passed from the CompositeDataSource, not user supplied

Returns

(list) –

list of STIX objects that matches the supplied query. As the MemoryStore(i.e. MemorySink) adds STIX objects to memory as they are supplied (either as python dictionary or STIX object), it is returned in the same form as it as added.

class MemoryStore(*stix_data=None*, *allow_custom=True*, *version=None*)
Interface to an in-memory dictionary of STIX objects.

MemoryStore is a wrapper around a paired MemorySink and MemorySource.

Note: It doesn't make sense to create a MemoryStore by passing in existing MemorySource and MemorySink because there could be data concurrency issues. As well, just as easy to create new MemoryStore.

Parameters

- **stix_data** (*list OR dict OR STIX object*) – STIX content to be added
- **allow_custom** (*bool*) – whether to allow custom STIX content. Only applied when export/input functions called, i.e. load_from_file() and save_to_file(). Defaults to True.
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

_data
dict – the in-memory dict that holds STIX objects

source
MemorySource – MemorySource

sink
MemorySink – MemorySink

load_from_file(**args*, ***kwargs*)
Load STIX data from JSON file.

File format is expected to be a single JSON STIX object or JSON STIX bundle.

Parameters

- **file_path** (*str*) – file path to load STIX data from
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

save_to_file(**args*, ***kwargs*)
Write SITX objects from in-memory dictionary to JSON file, as a STIX Bundle.

Parameters **file_path** (*str*) – file path to write STIX data to

3.2.4 taxii

Python STIX 2.x TAXIICollectionStore

class TAXIICollectionSink(*collection*, *allow_custom=False*)
Provides an interface for pushing STIX objects to a local/remote TAXII Collection endpoint.

Parameters

- **collection** (*taxii2.Collection*) – TAXII2 Collection instance
- **allow_custom** (*bool*) – Whether to allow custom STIX content to be added to the TAXIICollectionSink. Default: False

add(*stix_data*, *version=None*)
Add/push STIX content to TAXII Collection endpoint

Parameters

- **stix_data** (*STIX object OR dict OR str OR list*) – valid STIX 2.0 content in a STIX object (or Bundle), STIX onject dict (or Bundle dict), or a STIX 2.0 json encoded string, or list of any of the following
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

class TAXIICollectionSource(*collection*, *allow_custom=True*)
Provides an interface for searching/retrieving STIX objects from a local/remote TAXII Collection endpoint.

Parameters

- **collection** (*taxii2.Collection*) – TAXII Collection instance
- **allow_custom** (*bool*) – Whether to allow custom STIX content to be added to the FileSystemSink. Default: True

all_versions(*stix_id*, *version=None*, *_composite_filters=None*)
Retrieve STIX object from local/remote TAXII Collection endpoint, all versions of it

Parameters

- **stix_id** (*str*) – The STIX ID of the STIX objects to be retrieved.
- **_composite_filters** (*set*) – set of filters passed from the parent CompositeDataSource, not user supplied
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

Returns (see query() as all_versions() is just a wrapper)

get(*stix_id*, *version=None*, *_composite_filters=None*)
Retrieve STIX object from local/remote STIX Collection endpoint.

Parameters

- **stix_id** (*str*) – The STIX ID of the STIX object to be retrieved.
- **_composite_filters** (*set*) – set of filters passed from the parent CompositeDataSource, not user supplied
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

Returns

(STIX object) –

STIX object that has the supplied STIX ID. The STIX object is received from TAXII has dict, parsed into a python STIX object and then returned

query(*query=None*, *version=None*, *_composite_filters=None*)

Search and retreive STIX objects based on the complete query

A "complete query" includes the filters from the query, the filters attached to MemorySource, and any filters passed from a CompositeDataSource (i.e. _composite_filters)

Parameters

- **query** (*list*) – list of filters to search on
- **_composite_filters** (*set*) – set of filters passed from the CompositeDataSource, not user supplied
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

Returns

(list) –

list of STIX objects that matches the supplied query. The STIX objects are received from TAXII as dicts, parsed into python STIX objects and then returned.

class TAXIICollectionStore(*collection*, *allow_custom=None*)

Provides an interface to a local/remote TAXII Collection of STIX data. TAXIICollectionStore is a wrapper around a paired TAXIICollectionSink and TAXIICollectionSource.

Parameters

- **collection** (*taxii2.Collection*) – TAXII Collection instance
- **allow_custom** (*bool*) – whether to allow custom STIX content to be pushed/retrieved. Defaults to True for TAXIICollectionSource side(retrieving data) and False for TAXIICollectionSink side(pushing data). However, when parameter is supplied, it will be applied to both TAXIICollectionSource/Sink.

class CompositeDataSource

Controller for all the attached DataSources.

A user can have a single CompositeDataSource as an interface the a set of DataSources. When an API call is made to the CompositeDataSource, it is delegated to each of the (real) DataSources that are attached to it.

DataSources can be attached to CompositeDataSource for a variety of reasons, e.g. common filters, organization, less API calls.

data_sources
list – A dictionary of DataSource objects; to be controlled and used by the Data Source Controller object.

add_data_source(*data_source*)
Attach a DataSource to CompositeDataSource instance

Parameters **data_source** (DataSource) – a stix2.DataSource to attach to the CompositeDataSource

add_data_sources(*data_sources*)
Attach list of DataSources to CompositeDataSource instance

Parameters **data_sources** (*list*) – stix2.DataSources to attach to CompositeDataSource

all_versions(*stix_id*, *_composite_filters=None*)
Retrieve all versions of a STIX object by STIX ID.

Federated all_versions retrieve method - iterates through all DataSources defined in "data_sources".

A composite data source will pass its attached filters to each configured data source, pushing filtering to them to handle.

Parameters

- **stix_id** (*str*) – id of the STIX objects to retrieve.
- **_composite_filters** (*list*) – a list of filters passed from a CompositeDataSource (i.e. if this CompositeDataSource is attached to a parent CompositeDataSource), not user supplied.

Returns *all_data (list)* – list of STIX objects that have the specified id

get(*stix_id*, *_composite_filters=None*)
Retrieve STIX object by STIX ID

Federated retrieve method, iterates through all DataSources defined in the "data_sources" parameter. Each data source has a specific API retrieve-like function and associated parameters. This function does a federated retrieval and consolidation of the data returned from all the STIX data sources.

A composite data source will pass its attached filters to each configured data source, pushing filtering to them to handle.

Parameters

- **stix_id** (*str*) – the id of the STIX object to retrieve.
- **_composite_filters** (*list*) – a list of filters passed from a CompositeDataSource (i.e. if this CompositeDataSource is attached to another parent CompositeDataSource), not user supplied.

Returns *stix_obj* – the STIX object to be returned.

get_all_data_sources()

has_data_sources()

query(*query=None*, *_composite_filters=None*)
Retrieve STIX objects that match a query.

Federate the query to all DataSources attached to the Composite Data Source.

Parameters

- **query** (*list*) – list of filters to search on.

- **_composite_filters** (*list*) – a list of filters passed from a CompositeDataSource (i.e. if this CompositeDataSource is attached to a parent CompositeDataSource), not user supplied.

Returns *all_data (list)* – list of STIX objects to be returned

related_to(**args*, ***kwargs*)

Retrieve STIX Objects that have a Relationship involving the given STIX object.

Only one of *source_only* and *target_only* may be *True*.

Federated related objects method - iterates through all DataSources defined in "data_sources".

Parameters

- **obj** (*STIX object OR dict OR str*) – The STIX object (or its ID) whose related objects will be looked up.
- **relationship_type** (*str*) – Only retrieve objects related by this Relationships type. If None, all related objects will be returned, regardless of type.
- **source_only** (*bool*) – Only examine Relationships for which this object is the source_ref. Default: False.
- **target_only** (*bool*) – Only examine Relationships for which this object is the target_ref. Default: False.

Returns *(list)* – List of STIX objects related to the given STIX object.

relationships(**args*, ***kwargs*)

Retrieve Relationships involving the given STIX object.

Only one of *source_only* and *target_only* may be *True*.

Federated relationships retrieve method - iterates through all DataSources defined in "data_sources".

Parameters

- **obj** (*STIX object OR dict OR str*) – The STIX object (or its ID) whose relationships will be looked up.
- **relationship_type** (*str*) – Only retrieve Relationships of this type. If None, all relationships will be returned, regardless of type.
- **source_only** (*bool*) – Only retrieve Relationships for which this object is the source_ref. Default: False.
- **target_only** (*bool*) – Only retrieve Relationships for which this object is the target_ref. Default: False.

Returns *(list)* – List of Relationship objects involving the given STIX object.

remove_data_source(*data_source_id*)

Remove DataSource from the CompositeDataSource instance

Parameters **data_source_id** (*str*) – DataSource IDs.

remove_data_sources(*data_source_ids*)

Remove DataSources from the CompositeDataSource instance

Parameters **data_source_ids** (*list*) – DataSource IDs

class DataSink

An implementer will create a concrete subclass from this class for the specific DataSink.

id
str – A unique UUIDv4 to identify this DataSink.

add(*stix_objs*)
Method for storing STIX objects.

Implement: Specific data sink API calls, processing, functionality required for adding data to the sink

Parameters **stix_objs** (*list*) – a list of STIX objects (where each object is a STIX object)

class DataSource
An implementer will create a concrete subclass from this class for the specific DataSource.

id
str – A unique UUIDv4 to identify this DataSource.

filters
set – A collection of filters attached to this DataSource.

all_versions(*stix_id*)
Implement: Similar to get() except returns list of all object versions of the specified "id". In addition, implement the specific data source API calls, processing, functionality required for retrieving data from the data source.

Parameters **stix_id** (*str*) – The id of the STIX 2.0 object to retrieve. Should return a list of objects, all the versions of the object specified by the "id".

Returns *stix_objs (list)* – a list of STIX objects

creator_of(*obj*)
Retrieve the Identity refered to by the object's *created_by_ref*.

Parameters **obj** – The STIX object whose *created_by_ref* property will be looked up.

Returns The STIX object's creator, or None, if the object contains no *created_by_ref* property or the object's creator cannot be found.

get(*stix_id*)
Implement: Specific data source API calls, processing, functionality required for retrieving data from the data source

Parameters **stix_id** (*str*) – the id of the STIX 2.0 object to retrieve. Should return a single object, the most recent version of the object specified by the "id".

Returns *stix_obj* – the STIX object

query(*query=None*)
Implement: The specific data source API calls, processing, functionality required for retrieving query from the data source

Parameters **query** (*list*) – a list of filters (which collectively are the query) to conduct search on.

Returns *stix_objs (list)* – a list of STIX objects

related_to(*obj*, *relationship_type=None*, *source_only=False*, *target_only=False*)
Retrieve STIX Objects that have a Relationship involving the given STIX object.

Only one of *source_only* and *target_only* may be *True*.

Parameters

- **obj** (*STIX object OR dict OR str*) – The STIX object (or its ID) whose related objects will be looked up.

- **relationship_type** (*str*) – Only retrieve objects related by this Relationships type. If None, all related objects will be returned, regardless of type.
- **source_only** (*bool*) – Only examine Relationships for which this object is the source_ref. Default: False.
- **target_only** (*bool*) – Only examine Relationships for which this object is the target_ref. Default: False.

Returns *(list)* – List of STIX objects related to the given STIX object.

relationships(*obj*, *relationship_type=None*, *source_only=False*, *target_only=False*)

Retrieve Relationships involving the given STIX object.

Only one of *source_only* and *target_only* may be *True*.

Parameters

- **obj** (*STIX object OR dict OR str*) – The STIX object (or its ID) whose relationships will be looked up.
- **relationship_type** (*str*) – Only retrieve Relationships of this type. If None, all relationships will be returned, regardless of type.
- **source_only** (*bool*) – Only retrieve Relationships for which this object is the source_ref. Default: False.
- **target_only** (*bool*) – Only retrieve Relationships for which this object is the target_ref. Default: False.

Returns *(list)* – List of Relationship objects involving the given STIX object.

class DataStoreMixin(*source=None*, *sink=None*)

Provides mechanisms for storing and retrieving STIX data. The specific behavior can be customized by subclasses.

Parameters

- **source** (DataSource) – An existing DataSource to use as this DataStore's DataSource component
- **sink** (DataSink) – An existing DataSink to use as this DataStore's DataSink component

id

str – A unique UUIDv4 to identify this DataStore.

source

DataSource – An object that implements DataSource class.

sink

DataSink – An object that implements DataSink class.

add(**args*, ***kwargs*)

Method for storing STIX objects.

Define custom behavior before storing STIX objects using the associated DataSink. Translates add() to the appropriate DataSink call.

Parameters **stix_objs** (*list*) – a list of STIX objects

all_versions(**args*, ***kwargs*)

Retrieve all versions of a single STIX object by ID.

Translate all_versions() call to the appropriate DataSource call.

Parameters **stix_id** (*str*) – the id of the STIX object to retrieve.

Returns *stix_objs (list)* – a list of STIX objects

creator_of(**args*, ***kwargs*)
Retrieve the Identity refered to by the object's *created_by_ref*.

Translate creator_of() call to the appropriate DataSource call.

Parameters **obj** – The STIX object whose *created_by_ref* property will be looked up.

Returns The STIX object's creator, or None, if the object contains no *created_by_ref* property or the object's creator cannot be found.

get(**args*, ***kwargs*)
Retrieve the most recent version of a single STIX object by ID.

Translate get() call to the appropriate DataSource call.

Parameters **stix_id** (*str*) – the id of the STIX object to retrieve.

Returns

stix_obj –

the single most recent version of the STIX object specified by the "id".

query(**args*, ***kwargs*)
Retrieve STIX objects matching a set of filters.

Translate query() call to the appropriate DataSource call.

Parameters **query** (*list*) – a list of filters (which collectively are the query) to conduct search on.

Returns *stix_objs (list)* – a list of STIX objects

related_to(**args*, ***kwargs*)
Retrieve STIX Objects that have a Relationship involving the given STIX object.

Translate related_to() call to the appropriate DataSource call.

Only one of *source_only* and *target_only* may be *True*.

Parameters

- **obj** (*STIX object OR dict OR str*) – The STIX object (or its ID) whose related objects will be looked up.
- **relationship_type** (*str*) – Only retrieve objects related by this Relationships type. If None, all related objects will be returned, regardless of type.
- **source_only** (*bool*) – Only examine Relationships for which this object is the source_ref. Default: False.
- **target_only** (*bool*) – Only examine Relationships for which this object is the target_ref. Default: False.

Returns *(list)* – List of STIX objects related to the given STIX object.

relationships(**args*, ***kwargs*)
Retrieve Relationships involving the given STIX object.

Translate relationships() call to the appropriate DataSource call.

Only one of *source_only* and *target_only* may be *True*.

Parameters

- **obj** (*STIX object OR dict OR str*) – The STIX object (or its ID) whose relationships will be looked up.
- **relationship_type** (*str*) – Only retrieve Relationships of this type. If None, all relationships will be returned, regardless of type.
- **source_only** (*bool*) – Only retrieve Relationships for which this object is the source_ref. Default: False.
- **target_only** (*bool*) – Only retrieve Relationships for which this object is the target_ref. Default: False.

Returns *(list)* – List of Relationship objects involving the given STIX object.

make_id()

3.3 environment

class Environment(*factory=<stix2.environment.ObjectFactory object>*, *store=None*, *source=None*, *sink=None*)

Abstract away some of the nasty details of working with STIX content.

Parameters

- **factory** (ObjectFactory, *optional*) – Factory for creating objects with common defaults for certain properties.
- **store** (*DataStore, optional*) – Data store providing the source and sink for the environment.
- **source** (DataSource, *optional*) – Source for retrieving STIX objects.
- **sink** (DataSink, *optional*) – Destination for saving STIX objects. Invalid if *store* is also provided.

get(**args*, ***kwargs*)

Retrieve the most recent version of a single STIX object by ID.

Translate get() call to the appropriate DataSource call.

Parameters **stix_id** (*str*) – the id of the STIX object to retrieve.

Returns

stix_obj –

the single most recent version of the STIX object specified by the "id".

all_versions(**args*, ***kwargs*)

Retrieve all versions of a single STIX object by ID.

Translate all_versions() call to the appropriate DataSource call.

Parameters **stix_id** (*str*) – the id of the STIX object to retrieve.

Returns *stix_objs (list)* – a list of STIX objects

query(**args*, ***kwargs*)

Retrieve STIX objects matching a set of filters.

Translate query() call to the appropriate DataSource call.

Parameters **query** (*list*) – a list of filters (which collectively are the query) to conduct search on.

Returns *stix_objs (list)* – a list of STIX objects

creator_of(**args*, ***kwargs*)
Retrieve the Identity refered to by the object's *created_by_ref*.

Translate creator_of() call to the appropriate DataSource call.

Parameters **obj** – The STIX object whose *created_by_ref* property will be looked up.

Returns The STIX object's creator, or None, if the object contains no *created_by_ref* property or the object's creator cannot be found.

relationships(**args*, ***kwargs*)
Retrieve Relationships involving the given STIX object.

Translate relationships() call to the appropriate DataSource call.

Only one of *source_only* and *target_only* may be *True*.

Parameters

- **obj** (*STIX object OR dict OR str*) – The STIX object (or its ID) whose relationships will be looked up.
- **relationship_type** (*str*) – Only retrieve Relationships of this type. If None, all relationships will be returned, regardless of type.
- **source_only** (*bool*) – Only retrieve Relationships for which this object is the source_ref. Default: False.
- **target_only** (*bool*) – Only retrieve Relationships for which this object is the target_ref. Default: False.

Returns *(list)* – List of Relationship objects involving the given STIX object.

related_to(**args*, ***kwargs*)
Retrieve STIX Objects that have a Relationship involving the given STIX object.

Translate related_to() call to the appropriate DataSource call.

Only one of *source_only* and *target_only* may be *True*.

Parameters

- **obj** (*STIX object OR dict OR str*) – The STIX object (or its ID) whose related objects will be looked up.
- **relationship_type** (*str*) – Only retrieve objects related by this Relationships type. If None, all related objects will be returned, regardless of type.
- **source_only** (*bool*) – Only examine Relationships for which this object is the source_ref. Default: False.
- **target_only** (*bool*) – Only examine Relationships for which this object is the target_ref. Default: False.

Returns *(list)* – List of STIX objects related to the given STIX object.

add(**args*, ***kwargs*)
Method for storing STIX objects.

Define custom behavior before storing STIX objects using the associated DataSink. Translates add() to the appropriate DataSink call.

Parameters **stix_objs** (*list*) – a list of STIX objects

add_filter(**args*, ***kwargs*)

add_filters(**args*, ***kwargs*)

create(**args*, ***kwargs*)
Create a STIX object using object factory defaults.

Parameters

- **cls** – the python-stix2 class of the object to be created (eg. Indicator)
- ****kwargs** – The property/value pairs of the STIX object to be created

parse(**args*, ***kwargs*)
Deserialize a string or file-like object into a STIX object.

Parameters

- **data** (*str, dict, file-like object*) – The STIX 2 content to be parsed.
- **allow_custom** (*bool*) – Whether to allow custom properties or not. Default: False.
- **version** (*str*) – Which STIX2 version to use. (e.g. "2.0", "2.1"). If None, use latest version.

Returns An instantiated Python STIX object.

class ObjectFactory(*created_by_ref=None*, *created=None*, *external_references=None*, *object_marking_refs=None*, *list_append=True*)
Easily create STIX objects with default values for certain properties.

Parameters

- **created_by_ref** (*optional*) – Default created_by_ref value to apply to all objects created by this factory.
- **created** (*optional*) – Default created value to apply to all objects created by this factory.
- **external_references** (*optional*) – Default *external_references* value to apply to all objects created by this factory.
- **object_marking_refs** (*optional*) – Default *object_marking_refs* value to apply to all objects created by this factory.
- **list_append** (*bool, optional*) – When a default is set for a list property like *external_references* or *object_marking_refs* and a value for that property is passed into *create()*, if this is set to True, that value will be added to the list alongside the default. If this is set to False, the passed in value will replace the default. Defaults to True.

create(*cls*, ***kwargs*)
Create a STIX object using object factory defaults.

Parameters

- **cls** – the python-stix2 class of the object to be created (eg. Indicator)
- ****kwargs** – The property/value pairs of the STIX object to be created

3.4 exceptions

STIX 2 error classes.

exception AtLeastOnePropertyError(*cls*, *properties*)
Violating a constraint of a STIX object type that at least one of the given properties must be populated.

exception DependentPropertiesError(*cls*, *dependencies*)
Violating interproperty dependency constraint of a STIX object type.

exception DictionaryKeyError(*key*, *reason*)
Dictionary key does not conform to the correct format.

exception ExtraPropertiesError(*cls*, *properties*)
One or more extra properties were provided when constructing STIX object.

exception ImmutableError(*cls*, *key*)
Attempted to modify an object after creation.

exception InvalidObjRefError(*cls*, *prop_name*, *reason*)
A STIX Cyber Observable Object contains an invalid object reference.

exception InvalidSelectorError(*cls*, *key*)
Granular Marking selector violation. The selector must resolve into an existing STIX object property.

exception InvalidValueError(*cls*, *prop_name*, *reason*)
An invalid value was provided to a STIX object's `__init__`.

exception MarkingNotFoundError(*cls*, *key*)
Marking violation. The marking reference must be present in SDO or SRO.

exception MissingPropertiesError(*cls*, *properties*)
Missing one or more required properties when constructing STIX object.

exception MutuallyExclusivePropertiesError(*cls*, *properties*)
Violating interproperty mutually exclusive constraint of a STIX object type.

exception ParseError(*msg*)
Could not parse object.

exception RevokeError(*called_by*)
Attempted to an operation on a revoked object.

exception STIXError
Base class for errors generated in the stix2 library.

exception UnmodifiablePropertyError(*unchangable_properties*)
Attempted to modify an unmodifiable property of object when creating a new version.

3.5 markings

Functions for working with STIX 2 Data Markings.

These high level functions will operate on both object-level markings and granular markings unless otherwise noted in each of the functions.

Note: These functions are also available as methods on SDOs, SROs, and Marking Definitions. The corresponding methods on those classes are identical to these functions except that the *obj* parameter is omitted.

granular_markings	Functions for working with STIX 2.0 granular markings.
object_markings	Functions for working with STIX 2.0 object markings.
utils	Utility functions for STIX 2.0 data markings.

3.5.1 granular_markings

Functions for working with STIX 2.0 granular markings.

add_markings(*obj*, *marking*, *selectors*)
Append a granular marking to the granular_markings collection.

Parameters

- **obj** – An SDO or SRO object.
- **marking** – identifier or list of marking identifiers that apply to the properties selected by *selectors*.
- **selectors** – list of type string, selectors must be relative to the TLO in which the properties appear.

Raises `InvalidSelectorError` – If *selectors* fail validation.

Returns A new version of the given SDO or SRO with specified markings added.

clear_markings(*obj*, *selectors*)
Remove all granular markings associated with the selectors.

Parameters

- **obj** – An SDO or SRO object.
- **selectors** – string or list of selectors strings relative to the SDO or SRO in which the properties appear.

Raises

- `InvalidSelectorError` – If *selectors* fail validation.
- `MarkingNotFoundError` – If markings to remove are not found on the provided SDO or SRO.

Returns A new version of the given SDO or SRO with specified markings cleared.

get_markings(*obj*, *selectors*, *inherited=False*, *descendants=False*)
Get all granular markings associated to with the properties.

Parameters

- **obj** – An SDO or SRO object.
- **selectors** – string or list of selector strings relative to the SDO or SRO in which the properties appear.
- **inherited** – If True, include markings inherited relative to the properties.
- **descendants** – If True, include granular markings applied to any children relative to the properties.

Raises `InvalidSelectorError` – If *selectors* fail validation.

Returns *list* – Marking identifiers that matched the selectors expression.

is_marked(*obj*, *marking=None*, *selectors=None*, *inherited=False*, *descendants=False*)
Check if field is marked by any marking or by specific marking(s).

Parameters

- **obj** – An SDO or SRO object.

- **marking** – identifier or list of marking identifiers that apply to the properties selected by *selectors*.
- **selectors** – string or list of selectors strings relative to the SDO or SRO in which the properties appear.
- **inherited** – If True, return markings inherited from the given selector.
- **descendants** – If True, return granular markings applied to any children of the given selector.

Raises `InvalidSelectorError` – If *selectors* fail validation.

Returns

bool –

True if selectors is found on internal SDO or SRO collection. False otherwise.

Note: When a list of marking identifiers is provided, if ANY of the provided marking identifiers match, True is returned.

---

remove_markings(*obj*, *marking*, *selectors*)

Remove a granular marking from the granular_markings collection.

Parameters

- **obj** – An SDO or SRO object.
- **marking** – identifier or list of marking identifiers that apply to the properties selected by *selectors*.
- **selectors** – string or list of selectors strings relative to the SDO or SRO in which the properties appear.

Raises

- `InvalidSelectorError` – If *selectors* fail validation.
- `MarkingNotFoundError` – If markings to remove are not found on the provided SDO or SRO.

Returns A new version of the given SDO or SRO with specified markings removed.

set_markings(*obj*, *marking*, *selectors*)

Remove all granular markings associated with selectors and append a new granular marking. Refer to *clear_markings* and *add_markings* for details.

Parameters

- **obj** – An SDO or SRO object.
- **selectors** – string or list of selector strings relative to the SDO or SRO in which the properties appear.
- **marking** – identifier or list of marking identifiers that apply to the properties selected by *selectors*.

Returns A new version of the given SDO or SRO with specified markings removed and new ones added.

3.5.2 object_markings

Functions for working with STIX 2.0 object markings.

add_markings(*obj*, *marking*)
Append an object level marking to the object_marking_refs collection.

Parameters

- **obj** – A SDO or SRO object.
- **marking** – identifier or list of identifiers to apply SDO or SRO object.

Returns A new version of the given SDO or SRO with specified markings added.

clear_markings(*obj*)
Remove all object level markings from the object_marking_refs collection.

Parameters **obj** – A SDO or SRO object.

Returns A new version of the given SDO or SRO with object_marking_refs cleared.

get_markings(*obj*)
Get all object level markings from the given SDO or SRO object.

Parameters **obj** – A SDO or SRO object.

Returns

list –

Marking identifiers contained in the SDO or SRO. Empty list if no markings are present in *object_marking_refs*.

is_marked(*obj*, *marking=None*)
Check if SDO or SRO is marked by any marking or by specific marking(s).

Parameters

- **obj** – A SDO or SRO object.
- **marking** – identifier or list of marking identifiers that apply to the SDO or SRO object.

Returns *bool* – True if SDO or SRO has object level markings. False otherwise.

Note: When an identifier or list of identifiers is provided, if ANY of the provided marking refs match, True is returned.

---

remove_markings(*obj*, *marking*)
Remove an object level marking from the object_marking_refs collection.

Parameters

- **obj** – A SDO or SRO object.
- **marking** – identifier or list of identifiers that apply to the SDO or SRO object.

Raises `MarkingNotFoundError` – If markings to remove are not found on the provided SDO or SRO.

Returns A new version of the given SDO or SRO with specified markings removed.

set_markings(*obj*, *marking*)
Remove all object level markings and append new object level markings to the collection. Refer to *clear_markings* and *add_markings* for details.

Parameters

- **obj** – A SDO or SRO object.
- **marking** – identifier or list of identifiers to apply in the SDO or SRO object.

Returns A new version of the given SDO or SRO with specified markings removed and new ones added.

3.5.3 utils

Utility functions for STIX 2.0 data markings.

build_granular_marking(*granular_marking*)
Return a dictionary with the required structure for a granular marking.

compress_markings(*granular_markings*)
Compress granular markings list.

If there is more than one marking identifier matches. It will collapse into a single granular marking.

Example

```
>>> compress_markings([
...     {
...         "selectors": [
...             "description"
...         ],
...         "marking_ref": "marking-definition--613f2e26-407d-48c7-9eca-
↪b8e91df99dc9"
...     },
...     {
...         "selectors": [
...             "name"
...         ],
...         "marking_ref": "marking-definition--613f2e26-407d-48c7-9eca-
↪b8e91df99dc9"
...     }
... ])
[
    {
        "selectors": [
            "description",
            "name"
        ],
        "marking_ref": "marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9"
    }
]
```

Parameters **granular_markings** – The granular markings list property present in a SDO or SRO.

Returns *list* – A list with all markings collapsed.

convert_to_list(*data*)
Convert input into a list for further processing.

convert_to_marking_list(*data*)
Convert input into a list of marking identifiers.

expand_markings(*granular_markings*)
Expand granular markings list.

If there is more than one selector per granular marking. It will be expanded using the same marking_ref.

Example

```
>>> expand_markings([
...     {
...         "selectors": [
...             "description",
...             "name"
...         ],
...         "marking_ref": "marking-definition--613f2e26-407d-48c7-9eca-
↪b8e91df99dc9"
...     }
... ])
[
    {
        "selectors": [
            "description"
        ],
        "marking_ref": "marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9"
    },
    {
        "selectors": [
            "name"
        ],
        "marking_ref": "marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9"
    }
]
```

Parameters **granular_markings** – The granular markings list property present in a SDO or SRO.

Returns *list* – A list with all markings expanded.

iterpath(*obj*, *path=None*)
Generator which walks the input `obj` model.

Each iteration yields a tuple containing a list of ancestors and the property value.

Parameters

- **obj** – An SDO or SRO object.
- **path** – None, used recursively to store ancestors.

Example

```
>>> for item in iterpath(obj):
>>>     print(item)
(['type'], 'campaign')
```

```
...
(['cybox', 'objects', '[0]', 'hashes', 'sha1'],
↪'cac35ec206d868b7d7cb0b55f31d9425b075082b')
```

Returns

tuple –

Containing two items: a list of ancestors and the property value.

validate(*obj*, *selectors*)
Given an SDO or SRO, check that each selector is valid.

add_markings(*obj*, *marking*, *selectors=None*)
Append a marking to this object.

Parameters

- **obj** – An SDO or SRO object.
- **marking** – identifier or list of marking identifiers that apply to the properties selected by *selectors*.
- **selectors** – string or list of selectors strings relative to the SDO or SRO in which the properties appear.

Raises `InvalidSelectorError` – If *selectors* fail validation.

Returns A new version of the given SDO or SRO with specified markings added.

Note: If `selectors` is None, operations will be performed on object level markings. Otherwise on granular markings.

---

clear_markings(*obj*, *selectors=None*)
Remove all markings associated with the selectors.

Parameters

- **obj** – An SDO or SRO object.
- **selectors** – string or list of selectors strings relative to the SDO or SRO in which the field(s) appear(s).

Raises

- `InvalidSelectorError` – If *selectors* fail validation.
- `MarkingNotFoundError` – If markings to remove are not found on the provided SDO or SRO.

Returns A new version of the given SDO or SRO with specified markings cleared.

Note: If `selectors` is None, operations will be performed on object level markings. Otherwise on granular markings.

---

get_markings(*obj*, *selectors=None*, *inherited=False*, *descendants=False*)
Get all markings associated to the field(s) specified by selectors.

Parameters

- **obj** – An SDO or SRO object.
- **selectors** – string or list of selectors strings relative to the SDO or SRO in which the properties appear.
- **inherited** – If True, include object level markings and granular markings inherited relative to the properties.
- **descendants** – If True, include granular markings applied to any children relative to the properties.

Returns *list* – Marking identifiers that matched the selectors expression.

Note: If `selectors` is None, operation will be performed only on object level markings.

---

is_marked(*obj*, *marking=None*, *selectors=None*, *inherited=False*, *descendants=False*)
Check if field(s) is marked by any marking or by specific marking(s).

Parameters

- **obj** – An SDO or SRO object.
- **marking** – identifier or list of marking identifiers that apply to the properties selected by *selectors*.
- **selectors** – string or list of selectors strings relative to the SDO or SRO in which the field(s) appear(s).
- **inherited** – If True, include object level markings and granular markings inherited to determine if the properties is/are marked.
- **descendants** – If True, include granular markings applied to any children of the given selector to determine if the properties is/are marked.

Returns

bool –

True if selectors is found on internal SDO or SRO collection. False otherwise.

Note: When a list of marking identifiers is provided, if ANY of the provided marking identifiers match, True is returned.

If `selectors` is None, operation will be performed only on object level markings.

---

remove_markings(*obj*, *marking*, *selectors=None*)
Remove a marking from this object.

Parameters

- **obj** – An SDO or SRO object.
- **marking** – identifier or list of marking identifiers that apply to the properties selected by *selectors*.
- **selectors** – string or list of selectors strings relative to the SDO or SRO in which the properties appear.

Raises

- `InvalidSelectorError` – If *selectors* fail validation.
- `MarkingNotFoundError` – If markings to remove are not found on the provided SDO or SRO.

Returns A new version of the given SDO or SRO with specified markings removed.

Note: If `selectors` is None, operations will be performed on object level markings. Otherwise on granular markings.

---

set_markings(*obj*, *marking*, *selectors=None*)

Remove all markings associated with selectors and appends a new granular marking. Refer to *clear_markings* and *add_markings* for details.

Parameters

- **obj** – An SDO or SRO object.
- **marking** – identifier or list of marking identifiers that apply to the properties selected by *selectors*.
- **selectors** – string or list of selectors strings relative to the SDO or SRO in which the properties appear.

Returns A new version of the given SDO or SRO with specified markings removed and new ones added.

Note: If `selectors` is None, operations will be performed on object level markings. Otherwise on granular markings.

3.6 patterns

Classes to aid in working with the STIX 2 patterning language.

class AndBooleanExpression(*operands*)

class AndObservationExpression(*operands*)

class BasicObjectPathComponent(*property_name*, *is_key=False*)

class BinaryConstant(*value*)

class BooleanConstant(*value*)

class EqualityComparisonExpression(*lhs*, *rhs*, *negated=False*)

class FloatConstant(*value*)

class FollowedByObservationExpression(*operands*)

class GreaterThanComparisonExpression(*lhs*, *rhs*, *negated=False*)

class GreaterThanEqualComparisonExpression(*lhs*, *rhs*, *negated=False*)

class HashConstant(*value*, *type*)

class HexConstant(*value*)

class InComparisonExpression(*lhs*, *rhs*, *negated=False*)

class IntegerConstant(*value*)

class IsSubsetComparisonExpression(*lhs*, *rhs*, *negated=False*)

class IsSupersetComparisonExpression(*lhs*, *rhs*, *negated=False*)

class LessThanComparisonExpression(*lhs*, *rhs*, *negated=False*)

class LessThanEqualComparisonExpression(*lhs*, *rhs*, *negated=False*)

class LikeComparisonExpression(*lhs*, *rhs*, *negated=False*)

class ListConstant(*values*)

class ListObjectPathComponent(*property_name*, *index*)

class MatchesComparisonExpression(*lhs*, *rhs*, *negated=False*)

class ObjectPath(*object_type_name*, *property_path*)

    static make_object_path(*lhs*)

    merge(*other*)

class ObservationExpression(*operand*)

class OrBooleanExpression(*operands*)

class OrObservationExpression(*operands*)

class ParentheticalExpression(*exp*)

class QualifiedObservationExpression(*observation_expression*, *qualifier*)

class ReferenceObjectPathComponent(*reference_property_name*)

class RepeatQualifier(*times_to_repeat*)

class StartStopQualifier(*start_time*, *stop_time*)

class StringConstant(*value*)

class TimestampConstant(*value*)

class WithinQualifier(*number_of_seconds*)

escape_quotes_and_backslashes(*s*)

make_constant(*value*)

3.7 properties

Classes for representing properties of STIX Objects and Cyber Observables.

class BinaryProperty(*required=False*, *fixed=None*, *default=None*, *type=None*)

    clean(*value*)

class BooleanProperty(*required=False*, *fixed=None*, *default=None*, *type=None*)

    clean(*value*)

class DictionaryProperty(*required=False*, *fixed=None*, *default=None*, *type=None*)

clean(*value*)

class EmbeddedObjectProperty(*type*, *required=False*)

clean(*value*)

class EnumProperty(*allowed*, ***kwargs*)

clean(*value*)

class FloatProperty(*required=False*, *fixed=None*, *default=None*, *type=None*)

clean(*value*)

class HashesProperty(*required=False*, *fixed=None*, *default=None*, *type=None*)

clean(*value*)

class HexProperty(*required=False*, *fixed=None*, *default=None*, *type=None*)

clean(*value*)

class IDProperty(*type*)

clean(*value*)

default()

class IntegerProperty(*required=False*, *fixed=None*, *default=None*, *type=None*)

clean(*value*)

class ListProperty(*contained*, ***kwargs*)

clean(*value*)

class ObjectReferenceProperty(*valid_types=None*, ***kwargs*)

class PatternProperty(***kwargs*)

clean(*value*)

class Property(*required=False*, *fixed=None*, *default=None*, *type=None*)

Represent a property of STIX data type.

Subclasses can define the following attributes as keyword arguments to `__init__()`.

Parameters

- **required** (*bool*) – If `True`, the property must be provided when creating an object with that property. No default value exists for these properties. (Default: `False`)
- **fixed** – This provides a constant default value. Users are free to provide this value explicity when constructing an object (which allows you to copy **all** values from an existing object

to a new object), but if the user provides a value other than the `fixed` value, it will raise an error. This is semantically equivalent to defining both:

- a `clean()` function that checks if the value matches the fixed value, and
- a `default()` function that returns the fixed value.

Subclasses can also define the following functions:

- **`def clean(self, value) -> any:`**
  - Return a value that is valid for this property. If `value` is not valid for this property, this will attempt to transform it first. If `value` is not valid and no such transformation is possible, it should raise a ValueError.
- **`def default(self):`**
  - provide a default value for this property.
  - **`default()` can return the special value `NOW` to use the current** time. This is useful when several timestamps in the same object need to use the same default value, so calling now() for each property– likely several microseconds apart– does not work.

Subclasses can instead provide a lambda function for `default` as a keyword argument. `clean` should not be provided as a lambda since lambdas cannot raise their own exceptions.

When instantiating Properties, `required` and `default` should not be used together. `default` implies that the property is required in the specification so this function will be used to supply a value if none is provided. `required` means that the user must provide this; it is required in the specification and we can't or don't want to create a default value.

clean(*value*)

class ReferenceProperty(*required=False*, *type=None*)

clean(*value*)

class SelectorProperty(*type=None*)

clean(*value*)

class StringProperty(***kwargs*)

clean(*value*)

class TimestampProperty(*precision=None*, ***kwargs*)

clean(*value*)

class TypeProperty(*type*)

3.8 utils

Utility functions and classes for the stix2 library.

class STIXdatetime

deduplicate(*stix_obj_list*)
Deduplicate a list of STIX objects to a unique set.

Reduces a set of STIX objects to unique set by looking at 'id' and 'modified' fields - as a unique object version is determined by the combination of those fields

Note: Be aware, as can be seen in the implementation of deduplicate(),that if the "stix_obj_list" argument has multiple STIX objects of the same version, the last object version found in the list will be the one that is returned.

Parameters **stix_obj_list** (*list*) – list of STIX objects (dicts)

Returns A list with a unique set of the passed list of STIX objects.

find_property_index(*obj*, *properties*, *tuple_to_find*)
Recursively find the property in the object model, return the index according to the _properties OrderedDict. If it's a list look for individual objects.

format_datetime(*dttm*)
Convert a datetime object into a valid STIX timestamp string.

1. Convert to timezone-aware
2. Convert to UTC
3. Format in ISO format
4. Ensure correct precision a. Add subsecond value if non-zero and precision not defined
5. Add "Z"

get_class_hierarchy_names(*obj*)
Given an object, return the names of the class hierarchy.

get_dict(*data*)
Return data as a dictionary.

Input can be a dictionary, string, or file-like object.

get_timestamp()
Return a STIX timestamp of the current date and time.

get_type_from_id(*stix_id*)

new_version(*data*, ***kwargs*)
Create a new version of a STIX object, by modifying properties and updating the `modified` property.

parse_into_datetime(*value*, *precision=None*)
Parse a value into a valid STIX timestamp object.

remove_custom_stix(*stix_obj*)
remove any custom STIX objects or properties

Warning: This function is a best effort utility, in that it will remove custom objects and properties based on the type names; i.e. if "x-" prefixes object types, and "**x_**" prefixes property types. According to the STIX2 spec, those naming conventions are a SHOULDs not MUSTs, meaning that valid custom STIX content may ignore those conventions and in effect render this utility function invalid when used on that STIX content.

Parameters **stix_obj** (*dict OR python-stix obj*) – a single python-stix object or dict of a STIX object

Returns A new version of the object with any custom content removed

revoke(*data*)
Revoke a STIX object.

Returns A new version of the object with `revoked` set to `True`.

3.9 common

STIX 2 Common Data Types and Properties.

class ExternalReference(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **source_name** (*String, required*)
- **description** (*String*)
- **url** (*String*)
- **hashes** (*Hashes*)
- **external_id** (*String*)

class GranularMarking(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **marking_ref** (*Reference, required*)
- **selectors** (*List of Selectors, required*)

class KillChainPhase(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **kill_chain_name** (*String, required*)
- **phase_name** (*String, required*)

class MarkingDefinition(***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **definition_type** (*String, required*)
- **definition** (*Marking, required*)

class MarkingProperty(*required=False*, *fixed=None*, *default=None*, *type=None*)
Represent the marking objects in the `definition` property of marking-definition objects.

clean(*value*)

class StatementMarking(*statement=None*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **statement** (*String, required*)

class TLPMarking(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **tlp** (*String, required*)

CustomMarking(*type='x-custom-marking'*, *properties=None*)
Custom STIX Marking decorator.

Example

```
>>> @CustomMarking('x-custom-marking', [
...     ('property1', StringProperty(required=True)),
...     ('property2', IntegerProperty()),
... ])
... class MyNewMarkingObjectType():
...     pass
```

3.10 observables

STIX 2.0 Cyber Observable Objects.

Embedded observable object types, such as Email MIME Component, which is embedded in Email Message objects, inherit from `_STIXBase` instead of Observable and do not have a `_type` attribute.

class AlternateDataStream(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **name** (*String, required*)
- **hashes** (*Hashes*)
- **size** (*Integer*)

class ArchiveExt(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **contains_refs** (*List of Object References, required*)
- **version** (*String*)
- **comment** (*String*)

class Artifact(***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **mime_type** (*String*)

- **payload_bin** (*Binary*)
- **url** (*String*)
- **hashes** (*Hashes*)
- **extensions** (*Extensions*)

class AutonomousSystem(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **number** (*Integer, required*)
- **name** (*String*)
- **rir** (*String*)
- **extensions** (*Extensions*)

class Directory(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **path** (*String, required*)
- **path_enc** (*String*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)
- **accessed** (*Timestamp*)
- **contains_refs** (*List of Object References*)
- **extensions** (*Extensions*)

class DomainName(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **value** (*String, required*)
- **resolves_to_refs** (*List of Object References*)
- **extensions** (*Extensions*)

class EmailAddress(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **value** (*String, required*)
- **display_name** (*String*)
- **belongs_to_ref** (*Object Reference*)

- **extensions** (*Extensions*)

class EmailMIMEComponent(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **body** (*String*)
- **body_raw_ref** (*Object Reference*)
- **content_type** (*String*)
- **content_disposition** (*String*)

class EmailMessage(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **is_multipart** (*Boolean, required*)
- **date** (*Timestamp*)
- **content_type** (*String*)
- **from_ref** (*Object Reference*)
- **sender_ref** (*Object Reference*)
- **to_refs** (*List of Object References*)
- **cc_refs** (*List of Object References*)
- **bcc_refs** (*List of Object References*)
- **subject** (*String*)
- **received_lines** (*List of Strings*)
- **additional_header_fields** (*Dictionary*)
- **body** (*String*)
- **body_multipart** (*List of Embedded Objects*)
- **raw_email_ref** (*Object Reference*)
- **extensions** (*Extensions*)

class ExtensionsProperty(*enclosing_type=None*, *required=False*)

Property for representing extensions on Observable objects.

clean(*value*)

class File(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **hashes** (*Hashes*)
- **size** (*Integer*)
- **name** (*String*)

- **name_enc** (*String*)
- **magic_number_hex** (*Hex*)
- **mime_type** (*String*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)
- **accessed** (*Timestamp*)
- **parent_directory_ref** (*Object Reference*)
- **is_encrypted** (*Boolean*)
- **encryption_algorithm** (*String*)
- **decryption_key** (*String*)
- **contains_refs** (*List of Object References*)
- **content_ref** (*Object Reference*)
- **extensions** (*Extensions*)

class HTTPRequestExt(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **request_method** (*String, required*)
- **request_value** (*String, required*)
- **request_version** (*String*)
- **request_header** (*Dictionary*)
- **message_body_length** (*Integer*)
- **message_body_data_ref** (*Object Reference*)

class ICMPExt(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **icmp_type_hex** (*Hex, required*)
- **icmp_code_hex** (*Hex, required*)

class IPv4Address(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **value** (*String, required*)
- **resolves_to_refs** (*List of Object References*)
- **belongs_to_refs** (*List of Object References*)
- **extensions** (*Extensions*)

class IPv6Address(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **value** (*String, required*)
- **resolves_to_refs** (*List of Object References*)
- **belongs_to_refs** (*List of Object References*)
- **extensions** (*Extensions*)

class MACAddress (***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **value** (*String, required*)
- **extensions** (*Extensions*)

class Mutex (***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **name** (*String, required*)
- **extensions** (*Extensions*)

class NTFSExt (*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **sid** (*String*)
- **alternate_data_streams** (*List of Embedded Objects*)

class NetworkTraffic (***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **start** (*Timestamp*)
- **end** (*Timestamp*)
- **is_active** (*Boolean*)
- **src_ref** (*Object Reference*)
- **dst_ref** (*Object Reference*)
- **src_port** (*Integer*)
- **dst_port** (*Integer*)
- **protocols** (*List of Strings, required*)
- **src_byte_count** (*Integer*)
- **dst_byte_count** (*Integer*)

- **src_packets** (*Integer*)
- **dst_packets** (*Integer*)
- **ipfix** (*Dictionary*)
- **src_payload_ref** (*Object Reference*)
- **dst_payload_ref** (*Object Reference*)
- **encapsulates_refs** (*List of Object References*)
- **encapsulates_by_ref** (*Object Reference*)
- **extensions** (*Extensions*)

class ObservableProperty(*required=False*, *fixed=None*, *default=None*, *type=None*)
Property for holding Cyber Observable Objects.

clean(*value*)

class PDFExt(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **version** (*String*)
- **is_optimized** (*Boolean*)
- **document_info_dict** (*Dictionary*)
- **pdfid0** (*String*)
- **pdfid1** (*String*)

class Process(***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **is_hidden** (*Boolean*)
- **pid** (*Integer*)
- **name** (*String*)
- **created** (*Timestamp*)
- **cwd** (*String*)
- **arguments** (*List of Strings*)
- **command_line** (*String*)
- **environment_variables** (*Dictionary*)
- **opened_connection_refs** (*List of Object References*)
- **creator_user_ref** (*Object Reference*)
- **binary_ref** (*Object Reference*)
- **parent_ref** (*Object Reference*)
- **child_refs** (*List of Object References*)
- **extensions** (*Extensions*)

class RasterImageExt (*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **image_height** (*Integer*)
- **image_weight** (*Integer*)
- **bits_per_pixel** (*Integer*)
- **image_compression_algorithm** (*String*)
- **exif_tags** (*Dictionary*)

class SocketExt (*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **address_family** (*Enum, required*)
- **is_blocking** (*Boolean*)
- **is_listening** (*Boolean*)
- **protocol_family** (*Enum*)
- **options** (*Dictionary*)
- **socket_type** (*Enum*)
- **socket_descriptor** (*Integer*)
- **socket_handle** (*Integer*)

class Software (***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **name** (*String, required*)
- **cpe** (*String*)
- **languages** (*List of Strings*)
- **vendor** (*String*)
- **version** (*String*)
- **extensions** (*Extensions*)

class TCPExt (*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **src_flags_hex** (*Hex*)
- **dst_flags_hex** (*Hex*)

class UNIXAccountExt (*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **gid** (*Integer*)

- **groups** (*List of Strings*)
- **home_dir** (*String*)
- **shell** (*String*)

class URL(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **value** (*String, required*)
- **extensions** (*Extensions*)

class UserAccount(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **user_id** (*String, required*)
- **account_login** (*String*)
- **account_type** (*String*)
- **display_name** (*String*)
- **is_service_account** (*Boolean*)
- **is_privileged** (*Boolean*)
- **can_escalate_privs** (*Boolean*)
- **is_disabled** (*Boolean*)
- **account_created** (*Timestamp*)
- **account_expires** (*Timestamp*)
- **password_last_changed** (*Timestamp*)
- **account_first_login** (*Timestamp*)
- **account_last_login** (*Timestamp*)
- **extensions** (*Extensions*)

class WindowsPEBinaryExt(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **pe_type** (*String, required*)
- **imphash** (*String*)
- **machine_hex** (*Hex*)
- **number_of_sections** (*Integer*)
- **time_date_stamp** (*Timestamp*)
- **pointer_to_symbol_table_hex** (*Hex*)
- **number_of_symbols** (*Integer*)

- **size_of_optional_header** (*Integer*)
- **characteristics_hex** (*Hex*)
- **file_header_hashes** (*Hashes*)
- **optional_header** (*Embedded Object*)
- **sections** (*List of Embedded Objects*)

class WindowsPEOptionalHeaderType(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **magic_hex** (*Hex*)
- **major_linker_version** (*Integer*)
- **minor_linker_version** (*Integer*)
- **size_of_code** (*Integer*)
- **size_of_initialized_data** (*Integer*)
- **size_of_uninitialized_data** (*Integer*)
- **address_of_entry_point** (*Integer*)
- **base_of_code** (*Integer*)
- **base_of_data** (*Integer*)
- **image_base** (*Integer*)
- **section_alignment** (*Integer*)
- **file_alignment** (*Integer*)
- **major_os_version** (*Integer*)
- **minor_os_version** (*Integer*)
- **major_image_version** (*Integer*)
- **minor_image_version** (*Integer*)
- **major_subsystem_version** (*Integer*)
- **minor_subsystem_version** (*Integer*)
- **win32_version_value_hex** (*Hex*)
- **size_of_image** (*Integer*)
- **size_of_headers** (*Integer*)
- **checksum_hex** (*Hex*)
- **subsystem_hex** (*Hex*)
- **dll_characteristics_hex** (*Hex*)
- **size_of_stack_reserve** (*Integer*)
- **size_of_stack_commit** (*Integer*)
- **size_of_heap_reserve** (*Integer*)
- **size_of_heap_commit** (*Integer*)

- **loader_flags_hex** (*Hex*)
- **number_of_rva_and_sizes** (*Integer*)
- **hashes** (*Hashes*)

class WindowsPESection(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **name** (*String, required*)
- **size** (*Integer*)
- **entropy** (*Float*)
- **hashes** (*Hashes*)

class WindowsProcessExt(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **aslr_enabled** (*Boolean*)
- **dep_enabled** (*Boolean*)
- **priority** (*String*)
- **owner_sid** (*String*)
- **window_title** (*String*)
- **startup_info** (*Dictionary*)

class WindowsRegistryKey(***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **key** (*String, required*)
- **values** (*List of Embedded Objects*)
- **modified** (*Timestamp*)
- **creator_user_ref** (*Object Reference*)
- **number_of_subkeys** (*Integer*)
- **extensions** (*Extensions*)

values

class WindowsRegistryValueType(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **name** (*String, required*)
- **data** (*String*)
- **data_type** (*Enum*)

class WindowsServiceExt(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **service_name** (*String, required*)
- **descriptions** (*List of Strings*)
- **display_name** (*String*)
- **group_name** (*String*)
- **start_type** (*Enum*)
- **service_dll_refs** (*List of Object References*)
- **service_type** (*Enum*)
- **service_status** (*Enum*)

class X509Certificate(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **is_self_signed** (*Boolean*)
- **hashes** (*Hashes*)
- **version** (*String*)
- **serial_number** (*String*)
- **signature_algorithm** (*String*)
- **issuer** (*String*)
- **validity_not_before** (*Timestamp*)
- **validity_not_after** (*Timestamp*)
- **subject** (*String*)
- **subject_public_key_algorithm** (*String*)
- **subject_public_key_modulus** (*String*)
- **subject_public_key_exponent** (*Integer*)
- **x509_v3_extensions** (*Embedded Object*)
- **extensions** (*Extensions*)

class X509V3ExtenstionsType(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **basic_constraints** (*String*)
- **name_constraints** (*String*)
- **policy_constraints** (*String*)
- **key_usage** (*String*)
- **extended_key_usage** (*String*)
- **subject_key_identifier** (*String*)
- **authority_key_identifier** (*String*)

- **subject_alternative_name** (*String*)
- **issuer_alternative_name** (*String*)
- **subject_directory_attributes** (*String*)
- **crl_distribution_points** (*String*)
- **inhibit_any_policy** (*String*)
- **private_key_usage_period_not_before** (*Timestamp*)
- **private_key_usage_period_not_after** (*Timestamp*)
- **certificate_policies** (*String*)
- **policy_mappings** (*String*)

CustomExtension(*observable=None*, *type='x-custom-observable'*, *properties=None*)
Decorator for custom extensions to STIX Cyber Observables.

CustomObservable(*type='x-custom-observable'*, *properties=None*)
Custom STIX Cyber Observable Object type decorator.

Example

```
>>> @CustomObservable('x-custom-observable', [
...     ('property1', StringProperty(required=True)),
...     ('property2', IntegerProperty()),
... ])
... class MyNewObservableType():
...     pass
```

parse_observable(*data*, *_valid_refs=None*, *allow_custom=False*)
Deserialize a string or file-like object into a STIX Cyber Observable object.

Parameters

- **data** – The STIX 2 string to be parsed.
- **_valid_refs** – A list of object references valid for the scope of the object being parsed. Use empty list if no valid refs are present.
- **allow_custom** – Whether to allow custom properties or not. Default: False.

Returns An instantiated Python STIX Cyber Observable object.

3.11 sdo

STIX 2.0 Domain Objects

class AttackPattern(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)

- **created** (*Timestamp*)
- **modified** (*Timestamp*)
- **name** (*String, required*)
- **description** (*String*)
- **kill_chain_phases** (*List of Kill Chain Phases*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class Campaign(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)
- **name** (*String, required*)
- **description** (*String*)
- **aliases** (*List of Strings*)
- **first_seen** (*Timestamp*)
- **last_seen** (*Timestamp*)
- **objective** (*String*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class CourseOfAction(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)

- **name** (*String, required*)
- **description** (*String*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class Identity(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)
- **name** (*String, required*)
- **description** (*String*)
- **identity_class** (*String, required*)
- **sectors** (*List of Strings*)
- **contact_information** (*String*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class Indicator(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)
- **name** (*String*)
- **description** (*String*)
- **pattern** (*Pattern, required*)
- **valid_from** (*Timestamp*)

- **valid_until** (*Timestamp*)
- **kill_chain_phases** (*List of Kill Chain Phases*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings, required*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class IntrusionSet(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)
- **name** (*String, required*)
- **description** (*String*)
- **aliases** (*List of Strings*)
- **first_seen** (*Timestamp*)
- **last_seen** (*Timestamp*)
- **goals** (*List of Strings*)
- **resource_level** (*String*)
- **primary_motivation** (*String*)
- **secondary_motivations** (*List of Strings*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class Malware(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)

- **name** (*String, required*)
- **description** (*String*)
- **kill_chain_phases** (*List of Kill Chain Phases*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings, required*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class ObservedData(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)
- **first_observed** (*Timestamp, required*)
- **last_observed** (*Timestamp, required*)
- **number_observed** (*Integer, required*)
- **objects** (*Observable, required*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class Report(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)
- **name** (*String, required*)
- **description** (*String*)
- **published** (*Timestamp, required*)
- **object_refs** (*List of References, required*)

- **revoked** (*Boolean*)
- **labels** (*List of Strings, required*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class STIXDomainObject(*allow_custom=False*, ***kwargs*)

class ThreatActor(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)
- **name** (*String, required*)
- **description** (*String*)
- **aliases** (*List of Strings*)
- **roles** (*List of Strings*)
- **goals** (*List of Strings*)
- **sophistication** (*String*)
- **resource_level** (*String*)
- **primary_motivation** (*String*)
- **secondary_motivations** (*List of Strings*)
- **personal_motivations** (*List of Strings*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings, required*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class Tool(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)

- **name** (*String, required*)
- **description** (*String*)
- **kill_chain_phases** (*List of Kill Chain Phases*)
- **tool_version** (*String*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings, required*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class Vulnerability(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)
- **name** (*String, required*)
- **description** (*String*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

CustomObject(*type='x-custom-type'*, *properties=None*)

Custom STIX Object type decorator.

Example

```
>>> @CustomObject('x-type-name', [
...     ('property1', StringProperty(required=True)),
...     ('property2', IntegerProperty()),
... ])
... class MyNewObjectType():
...     pass
```

Supply an `__init__()` function to add any special validations to the custom type. Don't call `super().__init__()` though - doing so will cause an error.

Example

```
>>> @CustomObject('x-type-name', [
...     ('property1', StringProperty(required=True)),
...     ('property2', IntegerProperty()),
... ])
... class MyNewObjectType():
...     def __init__(self, property2=None, **kwargs):
...         if property2 and property2 < 10:
...             raise ValueError("'property2' is too small.")
```

3.12 sro

STIX 2.0 Relationship Objects.

class Relationship (*source_ref=None*, *relationship_type=None*, *target_ref=None*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)
- **relationship_type** (*String, required*)
- **description** (*String*)
- **source_ref** (*Reference, required*)
- **target_ref** (*Reference, required*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class STIXRelationshipObject (*allow_custom=False*, ***kwargs*)

class Sighting (*sighting_of_ref=None*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.0 specification.

Properties

- **type** (*Type*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp*)
- **modified** (*Timestamp*)

- **first_seen** (*Timestamp*)
- **last_seen** (*Timestamp*)
- **count** (*Integer*)
- **sighting_of_ref** (*Reference, required*)
- **observed_data_refs** (*List of References*)
- **where_sighted_refs** (*List of References*)
- **summary** (*Boolean*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

CHAPTER 4

DataStore API

Warning: The DataStore API is still in the planning stages and may be subject to major changes. We encourage anyone with feedback to contact the maintainers to help ensure the API meets a large variety of use cases.

One prominent feature of python-stix2 will be an interface for connecting different backend data stores containing STIX content. This will allow a uniform interface for querying and saving STIX content, and allow higher level code to be written without regard to the underlying data storage format. python-stix2 will define the API and contain some default implementations of this API, but developers are encouraged to write their own implementations.

Potential functions of the API include:

- get a STIX Object by ID (returns the most recent version).
- get all versions of a STIX object by ID.
- get all relationships involving a given object, and all related objects.
- save an object.
- query for objects that match certain criteria (query syntax TBD).

For all queries, the API will include a "filter" interface that can be used to either explicitly include or exclude results with certain criteria. For example,

- only trust content from a set of object creators.
- exclude content from certain (untrusted) object creators.
- only include content with a confidence above a certain threshold (once confidence is added to STIX).
- only return content that can be shared with external parties (in other words, that has TLP:GREEN markings).

Additionally, the python-stix2 library will contain a "composite" data store, which implements the DataStore API while delegating functionality to one or more "child" data stores.

CHAPTER 5

Development Roadmap

Warning: Prior to version 1.0, all APIs are considered unstable and subject to change.

This is a list of (planned) features before version 1.0 is released.

- Serialization of all STIX and Cyber Observable objects to JSON.
- De-serialization (parsing) of all STIX and Cyber Observable objects.
- APIs for versioning (revising and revoking) STIX objects.
- APIs for marking STIX objects and interpreting markings of STIX objects.
- *DataStore API*, providing a common interface for querying sources of STIX content (such as objects in memory, on a filesystem, in a database, or via a TAXII feed).

CHAPTER 6

Contributing

We're thrilled that you're interested in contributing to python-stix2! Here are some things you should know:

- contribution-guide.org has great ideas for contributing to any open-source project (not just python-stix2).
- All contributors must sign a Contributor License Agreement. See CONTRIBUTING.md in the project repository for specifics.
- If you are planning to implement a major feature (vs. fixing a bug), please discuss with a project maintainer first to ensure you aren't duplicating the work of someone else, and that the feature is likely to be accepted.

Now, let's get started!

6.1 Setting up a development environment

We recommend using a virtualenv.

1. Clone the repository. If you're planning to make pull request, you should fork the repository on GitHub and clone your fork instead of the main repo:

```
$ git clone https://github.com/yourusername/cti-python-stix2.git
```

2. Install develoment-related dependencies:

```
$ cd cti-python-stix2
$ pip install -r requirements.txt
```

3. Install pre-commit git hooks:

```
$ pre-commit install
```

At this point you should be able to make changes to the code.

6.2 Code style

All code should follow PEP 8. We allow for line lengths up to 160 characters, but any lines over 80 characters should be the exception rather than the rule. PEP 8 conformance will be tested automatically by Tox and Travis-CI (see below).

6.3 Testing

Note: All of the tools mentioned in this section are installed when you run `pip install -r requirements.txt`.

python-stix2 uses pytest for testing. We encourage the use of test-driven development (TDD), where you write (failing) tests that demonstrate a bug or proposed new feature before writing code that fixes the bug or implements the features. Any code contributions to python-stix2 should come with new or updated tests.

To run the tests in your current Python environment, use the `pytest` command from the root project directory:

```
$ pytest
```

This should show all of the tests that ran, along with their status.

You can run a specific test file by passing it on the command line:

```
$ pytest stix2/test/test_<xxx>.py
```

To ensure that the test you wrote is running, you can deliberately add an `assert False` statement at the beginning of the test. This is another benefit of TDD, since you should be able to see the test failing (and ensure it's being run) before making it pass.

tox allows you to test a package across multiple versions of Python. Setting up multiple Python environments is beyond the scope of this guide, but feel free to ask for help setting them up. Tox should be run from the root directory of the project:

```
$ tox
```

We aim for high test coverage, using the coverage.py library. Though it's not an absolute requirement to maintain 100% coverage, all code contributions must be accompanied by tests. To run coverage and look for untested lines of code, run:

```
$ pytest --cov=stix2
$ coverage html
```

then look at the resulting report in `htmlcov/index.html`.

All commits pushed to the `master` branch or submitted as a pull request are tested with Travis-CI automatically.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X