stix2 Documentation

Release 1.4.0

OASIS

Apr 03, 2020

Contents:

Welcome to the STIX 2 Python API's documentation. This library is designed to help you work with STIX 2 content. For more information about STIX 2, see the website of the OASIS Cyber Threat Intelligence Technical Committee.

Get started with an *overview* of the library, then take a look at the *guides and tutorials* to see how to use it. For information about a specific class or function, see the *API reference*.

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 the `stix2` library:

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 `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 spec.

1.3 Architecture

The `stix2` library is 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, the **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.

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 Sources` 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). For more information, check out *this Environment tutorial*.

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. For more information, check out *this Workbench tutorial*.

CHAPTER 2

User's Guide

This section of documentation contains guides and tutorials on how to use the `stix2` library.

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:

[3]:
```
from stix2 import Indicator

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

[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:

[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:

```
[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:

```
[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:

```
[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:

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

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

Or access properties using the standard Python attribute syntax:

```
[9]: indicator.name
```

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

Warning

Note that there are several attributes on these objects used for method names. Accessing those will return a bound method, not the attribute value.

Attempting to modify any attributes will raise an error:

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

```
TypeError: 'Indicator' object does not support item assignment
```

```
[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:

[12]:
```
from stix2 import Malware

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

[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.

[13]:
```
from stix2 import Relationship

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

[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:

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

[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:

[15]:
```
from stix2 import Bundle

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

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

2.1.4 Creating Cyber Observable References

Cyber Observable Objects have properties that can reference other Cyber Observable Objects. In order to create those references, use the `_valid_refs` property as shown in the following examples. It should be noted that `_valid_refs` is necessary when creating references to Cyber Observable Objects since some embedded references can only point to certain types, and `_valid_refs` helps ensure consistency.

There are two cases.

Case 1: Specifying the type of the Cyber Observable Objects being referenced

In the following example, the IPv4Address object has its `resolves_to_refs` property specified. As per the spec, this property's value must be a list of reference(s) to MACAddress objects. In this case, those references are strings that state the type of the Cyber Observable Object being referenced, and are provided in `_valid_refs`.

[16]:
```
from stix2 import IPv4Address

ip4 = IPv4Address(
    _valid_refs={"1": "mac-addr", "2": "mac-addr"},
    value="177.60.40.7",
    resolves_to_refs=["1", "2"]
)

print(ip4)
```

[16]:
```
<IPython.core.display.HTML object>
```

Case 2: Specifying the name of the Cyber Observable Objects being referenced

The following example is just like the one provided in Case 1 above, with one key difference: instead of using strings to specify the type of the Cyber Observable Objects being referenced in `_valid_refs`, the referenced Cyber Observable Objects are created beforehand and then their names are provided in `_valid_refs`.

[17]:
```
from stix2 import MACAddress

mac_addr_a = MACAddress(value="a1:b2:c3:d4:e5:f6")
mac_addr_b = MACAddress(value="a7:b8:c9:d0:e1:f2")

ip4_valid_refs = IPv4Address(
    _valid_refs={"1": mac_addr_a, "2": mac_addr_b},
    value="177.60.40.7",
    resolves_to_refs=["1", "2"]
)

print(ip4_valid_refs)
```

[17]:
```
<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:

[3]:
```
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`:

[4]:
```
identity = Identity(name="John Smith",
                    identity_class="individual",
                    custom_properties={
                        "x_foo": "bar"
                    })
print(identity)
```

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

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

[5]:
```
identity2 = Identity(name="John Smith",
                     identity_class="individual",
                     x_foo="bar",
                     allow_custom=True)
print(identity2)
```

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

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

[6]:
```
from stix2 import parse

input_string = """{
    "type": "identity",
    "id": "identity--311b2d2d-f010-4473-83ec-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)
```

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

To remove a custom properties, use `new_version()` and set it to `None`.

[7]:
```
identity4 = identity3.new_version(x_foo=None)
print(identity4)
```

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

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__`.

[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.

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

[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:

[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.

[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)
```

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

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

[12]:
```
input_string3 = """{
    "type": "x-foobar",
```

(continues on next page)

(continued from previous page)

```
    "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 decorator.
```

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.

[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)
```

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

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

[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)
```

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

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

2.2.4 ID-Contributing Properties for Custom Cyber Observables

STIX 2.1 Cyber Observables (SCOs) have deterministic IDs, meaning that the ID of a SCO is based on the values of some of its properties. Thus, if multiple cyber observables of the same type have the same values for their ID-contributing properties, then these SCOs will have the same ID. UUIDv5 is used for the deterministic IDs, using the namespace `"00abedb4-aa42-466c-9c01-fed23315a9b7"`. A SCO's ID-contributing properties may consist of a combination of required properties and optional properties.

If a SCO type does not have any ID contributing properties defined, or all of the ID-contributing properties are not present on the object, then the SCO uses a randomly-generated UUIDv4. Thus, you can optionally define which of your custom SCO's properties should be ID-contributing properties. Similar to standard SCOs, your custom SCO's ID-contributing properties can be any combination of the SCO's required and optional properties.

You define the ID-contributing properties when defining your custom SCO with the `CustomObservable` decorator. After the list of properties, you can optionally define the list of id-contributing properties. If you do not want to specify any id-contributing properties for your custom SCO, then you do not need to do anything additional.

See the example below:

[15]:
```
from stix2.v21 import CustomObservable # IDs and Deterministic IDs are NOT part of␣
↪STIX 2.0 Custom Observables

@CustomObservable('x-new-observable-2', [
    ('a_property', properties.StringProperty(required=True)),
    ('property_2', properties.IntegerProperty()),
], [
    'a_property'
])
class NewObservable2():
    pass

new_observable_a = NewObservable2(a_property="A property", property_2=2000)
print(new_observable_a)

new_observable_b = NewObservable2(a_property="A property", property_2=3000)
print(new_observable_b)

new_observable_c = NewObservable2(a_property="A different property", property_2=3000)
print(new_observable_c)
```

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

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

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

In this example, `a_property` is the only id-contributing property. Notice that the ID for `new_observable_a` and `new_observable_b` is the same since they have the same value for the id-contributing `a_property` property.

2.2.5 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:

[16]:
```
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)
```

[16]:
```
<IPython.core.display.HTML object>
```

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

[17]:
```
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)
```

[17]:
```
<IPython.core.display.HTML object>
```

[17]:
```
<IPython.core.display.HTML object>
```

2.3 DataStore API

The `stix2` library features an interface for pulling and pushing STIX 2 content. This interface consists of *DataStore*, *DataSource* and *DataSink* constructs: a *DataSource* for pulling STIX 2 content, a *DataSink* for pushing STIX 2

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. The `stix2` library 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

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

# create FileSystemStore
fs = FileSystemSource("/tmp/stix2_source")

# create TAXIICollectionSource
colxn = Collection('http://127.0.0.1:5000/trustgroup1/collections/91a7b528-80eb-42ed-
↪a74d-c6fbd5a26116/')
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)
```

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

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

2.3.2 Filters

The `stix2` DataStore suites - *FileSystem*, *Memory*, and *TAXII* - all use the *Filters* module to allow for the querying of STIX content. Filters can be used to 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 2)
- only return content that can be shared with external parties (e.g. only content that has TLP:GREEN markings)

Filters can be created and supplied with every call 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 STIX 2 objects can be filtered on. In addition, TAXII 2 Filtering parameters for fields can also be used in filters.

TAXII2 filter fields:

- added_after
- id
- type
- version

Supported operators:

- =
- !=
- in
- >
- <
- >=
- <=
- contains

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

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

Filter Examples

[3]:
```
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.

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

fs = FileSystemStore("/tmp/stix2_store")
fs_source = FileSystemSource("/tmp/stix2_source")

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

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

# can also attach filters to a Source
# attach multiple filters to FileSystemSource
fs_source.filters.add([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.add([f1,f2])
```

Note: The ‘‘defanged‘‘ property is now always included (implicitly) for STIX 2.1 Cyber Observable Objects (SCOs)

This is important to remember if you are writing a filter that involves checking the `objects` property of a STIX 2.1 `ObservedData` object. If any of the objects associated with the `objects` property are STIX 2.1 SCOs, then your filter must include the `defanged` property. For an example, refer to `filters[14]` & `filters[15]` in stix2/test/v21/test_datastore_filters.py

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.

```
[10]: 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 =
      ↪'d41d8cd98f00b204e9800998ecf8427e']")
      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.

```
[11]: print(mem.creator_of(mal))
```

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

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

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

```
[12]: 3
```

You can limit it to only specific relationship types:

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

```
[13]: [Relationship(type='relationship', id='relationship--3b9cb248-5c2c-425d-85d0-
      ↪680bfef6e69d', created='2018-04-05T20:43:54.134Z', modified='2018-04-05T20:43:54.
      ↪134Z', relationship_type='indicates', source_ref='indicator--61deb2a5-305a-490e-
      ↪83b3-9839a9677368', target_ref='malware--9fe343d8-edf7-4f4a-bb6c-a221fb75142d')]
```

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

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

```
[14]: [Relationship(type='relationship', id='relationship--8d322508-423b-4d51-be85-
      ↪a95ad083f8af', created='2018-04-05T20:43:54.134Z', modified='2018-04-05T20:43:54.
      ↪134Z', relationship_type='targets', source_ref='malware--9fe343d8-edf7-4f4a-bb6c-
      ↪a221fb75142d', target_ref='identity--b67cf8d4-cc1a-4bb7-9402-fffcff17c9a9')]
```

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

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

```
[15]: [Relationship(type='relationship', id='relationship--3b9cb248-5c2c-425d-85d0-
      ↪680bfef6e69d', created='2018-04-05T20:43:54.134Z', modified='2018-04-05T20:43:54.
      ↪134Z', relationship_type='indicates', source_ref='indicator--61deb2a5-305a-490e-
      ↪83b3-9839a9677368', target_ref='malware--9fe343d8-edf7-4f4a-bb6c-a221fb75142d'),
       Relationship(type='relationship', id='relationship--93e5afe0-d1fb-4315-8d08-
      ↪10951f7a99b6', created='2018-04-05T20:43:54.134Z', modified='2018-04-05T20:43:54.
      ↪134Z', relationship_type='uses', source_ref='campaign--edfd885c-bc31-4051-9bc2-
      ↪08e057542d56', target_ref='malware--9fe343d8-edf7-4f4a-bb6c-a221fb75142d')]
```

(continues on next page)

(continued from previous page)

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.

[16]:
```
mem.related_to(mal, target_only=True, relationship_type='uses')
```

[16]:
```
[Campaign(type='campaign', id='campaign--edfd885c-bc31-4051-9bc2-08e057542d56',␣
↪created='2018-04-05T20:43:54.117Z', modified='2018-04-05T20:43:54.117Z', name=
↪'Charge', description='Attack!')]
```

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.

[1]:
```
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.

[6]:
```
from stix2 import CompositeDataSource, FileSystemSink, FileSystemSource, MemorySource

src = CompositeDataSource()
src.add_data_sources([MemorySource(), FileSystemSource("/tmp/stix2_source")])
env2 = Environment(source=src,
                   sink=FileSystemSink("/tmp/stix2_sink"))
```

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

[7]:
```
from stix2 import Indicator

indicator = Indicator(id="indicator--a740531e-63ff-4e49-a9e1-a0a3eed0e3e7",
                      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*.

[8]:
```
print(env.get("indicator--a740531e-63ff-4e49-a9e1-a0a3eed0e3e7"))
```

[8]:
```
<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.

[13]:
```
from stix2 import Indicator, ObjectFactory

factory = ObjectFactory(created_by_ref="identity--311b2d2d-f010-4473-83ec-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.

[14]:
```
ind = factory.create(Indicator,
                     labels=["malicious-activity"],
                     pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']")
print(ind)
```

[14]:
```
<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.

[15]:
```
factory2 = ObjectFactory(created_by_ref="identity--311b2d2d-f010-4473-83ec-
↪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)
```

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

[16]:
```
ind3 = env2.create(Indicator,
                       created_by_ref="identity--962cabe5-f7f3-438a-9169-585a8c971d12
↪",
                       labels=["malicious-activity"],
                       pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']
↪")
print(ind3)
```

[16]:
```
<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*:

[17]:
```
environ = Environment(ObjectFactory(created_by_ref="identity--311b2d2d-f010-4473-83ec-
↪1edf84858f4c"),
                      MemoryStore())

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

[17]:
```
<IPython.core.display.HTML object>
```

2.5 Checking Semantic Equivalence

The *Environment* has a function for checking if two STIX Objects are semantically equivalent. For each supported object type, the algorithm checks if the values for a specific set of properties match. Then each matching property is weighted since every property doesn't represent the same level of importance for semantic equivalence. The result will be the sum of these weighted values, in the range of 0 to 100. A result of 0 means that the the two objects are not equivalent, and a result of 100 means that they are equivalent.

TODO: Add a link to the committee note when it is released.

There are a number of use cases for which calculating semantic equivalence may be helpful. It can be used for echo detection, in which a STIX producer who consumes content from other producers wants to make sure they are not creating content they have already seen or consuming content they have already created.

Another use case for this functionality is to identify identical or near-identical content, such as a vulnerability shared under three different nicknames by three different STIX producers. A third use case involves a feed that aggregates data from multiple other sources. It will want to make sure that it is not publishing duplicate data.

Below we will show examples of the semantic equivalence results of various objects. Unless otherwise specified, the ID of each object will be generated by the library, so the two objects will not have the same ID. This demonstrates that the semantic equivalence algorithm only looks at specific properties for each object type.

Please note that you will need to install a few extra dependencies in order to use the semantic equivalence functions. You can do this using:

```
pip install stix2[semantic]
```

2.5.1 Attack Pattern Example

For Attack Patterns, the only properties that contribute to semantic equivalence are `name` and `external_references`, with weights of 30 and 70, respectively. In this example, both attack patterns have the same external reference but the second has a slightly different yet still similar name.

[3]:
```
import stix2
from stix2 import Environment, MemoryStore
from stix2.v21 import AttackPattern

env = Environment(store=MemoryStore())

ap1 = AttackPattern(
    name="Phishing",
    external_references=[
        {
```

(continues on next page)

(continued from previous page)

```
            "url": "https://example2",
            "source_name": "some-source2",
        },
    ],
)
ap2 = AttackPattern(
    name="Spear phishing",
    external_references=[
        {
            "url": "https://example2",
            "source_name": "some-source2",
        },
    ],
)
print(env.semantically_equivalent(ap1, ap2))
```

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

2.5.2 Campaign Example

For Campaigns, the only properties that contribute to semantic equivalence are `name` and `aliases`, with weights of 60 and 40, respectively. In this example, the two campaigns have completely different names, but slightly similar descriptions. The result may be higher than expected because the Jaro-Winkler algorithm used to compare string properties looks at the edit distance of the two strings rather than just the words in them.

[4]:
```
from stix2.v21 import Campaign

c1 = Campaign(
    name="Someone Attacks Somebody",)

c2 = Campaign(
    name="Another Campaign",)
print(env.semantically_equivalent(c1, c2))
```

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

2.5.3 Identity Example

For Identities, the only properties that contribute to semantic equivalence are `name`, `identity_class`, and `sectors`, with weights of 60, 20, and 20, respectively. In this example, the two identities are identical, but are missing one of the contributing properties. The algorithm only compares properties that are actually present on the objects. Also note that they have completely different description properties, but because description is not one of the properties considered for semantic equivalence, this difference has no effect on the result.

[5]:
```
from stix2.v21 import Identity

id1 = Identity(
    name="John Smith",
    identity_class="individual",
    description="Just some guy",
)
id2 = Identity(
    name="John Smith",
```

(continues on next page)

(continued from previous page)

```
    identity_class="individual",
    description="A person",
)
print(env.semantically_equivalent(id1, id2))
```

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

2.5.4 Indicator Example

For Indicators, the only properties that contribute to semantic equivalence are `indicator_types`, `pattern`, and `valid_from`, with weights of 15, 80, and 5, respectively. In this example, the two indicators have patterns with different hashes but the same indicator_type and valid_from. For patterns, the algorithm currently only checks if they are identical.

[6]:
```
from stix2.v21 import Indicator

ind1 = Indicator(
    indicator_types=['malicious-activity'],
    pattern_type="stix",
    pattern="[file:hashes.MD5 = 'd41d8cd98f00b204e9800998ecf8427e']",
    valid_from="2017-01-01T12:34:56Z",
)
ind2 = Indicator(
    indicator_types=['malicious-activity'],
    pattern_type="stix",
    pattern="[file:hashes.MD5 = '79054025255fb1a26e4bc422aef54eb4']",
    valid_from="2017-01-01T12:34:56Z",
)
print(env.semantically_equivalent(ind1, ind2))
```

```
Indicator pattern equivalence is not fully defined; will default to zero if not␣
↪completely identical
```

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

If the patterns were identical the result would have been 100.

2.5.5 Location Example

For Locations, the only properties that contribute to semantic equivalence are `longitude`/`latitude`, `region`, and `country`, with weights of 34, 33, and 33, respectively. In this example, the two locations are Washington, D.C. and New York City. The algorithm computes the distance between two locations using the haversine formula and uses that to influence equivalence.

[7]:
```
from stix2.v21 import Location

loc1 = Location(
    latitude=38.889,
    longitude=-77.023,
)
loc2 = Location(
    latitude=40.713,
    longitude=-74.006,
)
print(env.semantically_equivalent(loc1, loc2))
```

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

2.5.6 Malware Example

For Malware, the only properties that contribute to semantic equivalence are `malware_types` and `name`, with weights of 20 and 80, respectively. In this example, the two malware objects only differ in the strings in their malware_types lists. For lists, the algorithm bases its calculations on the intersection of the two lists. An empty intersection will result in a 0, and a complete intersection will result in a 1 for that property.

[8]:
```
from stix2.v21 import Malware

MALWARE_ID = "malware--9c4638ec-f1de-4ddb-abf4-1b760417654e"

mal1 = Malware(id=MALWARE_ID,
    malware_types=['ransomware'],
    name="Cryptolocker",
    is_family=False,
    )
mal2 = Malware(id=MALWARE_ID,
    malware_types=['ransomware', 'dropper'],
    name="Cryptolocker",
    is_family=False,
    )
print(env.semantically_equivalent(mal1, mal2))
```

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

2.5.7 Threat Actor Example

For Threat Actors, the only properties that contribute to semantic equivalence are `threat_actor_types`, `name`, and `aliases`, with weights of 20, 60, and 20, respectively. In this example, the two threat actors have the same id properties but everything else is different. Since the id property does not factor into semantic equivalence, the result is not very high. The result is not zero because of the "Token Sort Ratio" algorithm used to compare the `name` property.

[9]:
```
from stix2.v21 import ThreatActor

THREAT_ACTOR_ID = "threat-actor--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f"

ta1 = ThreatActor(id=THREAT_ACTOR_ID,
    threat_actor_types=["crime-syndicate"],
    name="Evil Org",
    aliases=["super-evil"],
)
ta2 = ThreatActor(id=THREAT_ACTOR_ID,
    threat_actor_types=["spy"],
    name="James Bond",
    aliases=["007"],
)
print(env.semantically_equivalent(ta1, ta2))
```

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

2.5.8 Tool Example

For Tools, the only properties that contribute to semantic equivalence are `tool_types` and `name`, with weights of 20 and 80, respectively. In this example, the two tools have the same values for properties that contribute to semantic equivalence but one has an additional, non-contributing property.

[10]:
```
from stix2.v21 import Tool

t1 = Tool(
    tool_types=["remote-access"],
    name="VNC",
)
t2 = Tool(
    tool_types=["remote-access"],
    name="VNC",
    description="This is a tool"
)
print(env.semantically_equivalent(t1, t2))
```

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

2.5.9 Vulnerability Example

For Vulnerabilities, the only properties that contribute to semantic equivalence are `name` and `external_references`, with weights of 30 and 70, respectively. In this example, the two vulnerabilities have the same name but one also has an external reference. The algorithm doesn't take into account any semantic equivalence contributing properties that are not present on both objects.

[11]:
```
from stix2.v21 import Vulnerability

vuln1 = Vulnerability(
    name="Heartbleed",
    external_references=[
        {
            "url": "https://example",
            "source_name": "some-source",
        },
    ],
)
vuln2 = Vulnerability(
    name="Heartbleed",
)
print(env.semantically_equivalent(vuln1, vuln2))
```

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

2.5.10 Other Examples

Comparing objects of different types will result in a `ValueError`.

[12]:
```
print(env.semantically_equivalent(ind1, vuln1))
```

```
ValueError: The objects to compare must be of the same type!
```

Some object types do not have a defined method for calculating semantic equivalence and by default will give a warning and a result of zero.

[13]:
```
from stix2.v21 import Report

r1 = Report(
    report_types=["campaign"],
    name="Bad Cybercrime",
    published="2016-04-06T20:03:00.000Z",
    object_refs=["indicator--a740531e-63ff-4e49-a9e1-a0a3eed0e3e7"],
)
r2 = Report(
    report_types=["campaign"],
    name="Bad Cybercrime",
    published="2016-04-06T20:03:00.000Z",
    object_refs=["indicator--a740531e-63ff-4e49-a9e1-a0a3eed0e3e7"],
)
print(env.semantically_equivalent(r1, r2))
```

```
'report' type has no 'weights' dict specified & thus no semantic equivalence method␣
↪to call!
```

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

By default, comparing objects of different spec versions will result in a `ValueError`.

[14]:
```
from stix2.v20 import Identity as Identity20

id20 = Identity20(
    name="John Smith",
    identity_class="individual",
)
print(env.semantically_equivalent(id2, id20))
```

```
ValueError: The objects to compare must be of the same spec version!
```

You can optionally allow comparing across spec versions by providing a configuration dictionary using `ignore_spec_version` like in the next example:

[15]:
```
from stix2.v20 import Identity as Identity20

id20 = Identity20(
    name="John Smith",
    identity_class="individual",
)
print(env.semantically_equivalent(id2, id20, **{"_internal": {"ignore_spec_version":␣
↪True}}))
```

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

2.5.11 Detailed Results

If your logging level is set to `DEBUG` or higher, the function will log more detailed results. These show the semantic equivalence and weighting for each property that is checked, to show how the final result was arrived at.

[16]:
```
import logging
logging.basicConfig(format='%(message)s')
```

(continues on next page)

(continued from previous page)

```
logger = logging.getLogger()
logger.setLevel(logging.DEBUG)

ta3 = ThreatActor(
    threat_actor_types=["crime-syndicate"],
    name="Evil Org",
    aliases=["super-evil"],
)
ta4 = ThreatActor(
    threat_actor_types=["spy"],
    name="James Bond",
    aliases=["007"],
)
print(env.semantically_equivalent(ta3, ta4))

logger.setLevel(logging.ERROR)
```

```
Starting semantic equivalence process between:␣
↪'threat-actor-664624c7-394e-49ad-ae2a-12f7a48a54a3' and␣
↪'threat-actor-1d67719e-6be6-4194-9226-1685986514f5'
-              partial_string_based 'Evil Org' 'James Bond'  result: '11'
'name' check - weight: 60, contributing score: 6.6
-              partial_list_based '['crime-syndicate']' '['spy']'    result: '0.0'
'threat_actor_types' check - weight: 20, contributing score: 0.0
-              partial_list_based '['super-evil']' '['007']' result: '0.0'
'aliases' check - weight: 20, contributing score: 0.0
Matching Score: 6.6, Sum of Weights: 100.0
```

[16]:
```
<IPython.core.display.HTML object>
```

You can also retrieve the detailed results in a dictionary so the detailed results information can be accessed and used more programatically. The *semantically_equivalent()* function takes an optional third argument, called `prop_scores`. This argument should be a dictionary into which the detailed debugging information will be stored.

Using `prop_scores` is simple: simply pass in a dictionary to `semantically_equivalent()`, and after the function is done executing, the dictionary will have the various scores in it. Specifically, it will have the overall `matching_score` and `sum_weights`, along with the weight and contributing score for each of the semantic equivalence contributing properties.

For example:

[18]:
```
ta5 = ThreatActor(
    threat_actor_types=["crime-syndicate", "spy"],
    name="Evil Org",
    aliases=["super-evil"],
)
ta6 = ThreatActor(
    threat_actor_types=["spy"],
    name="James Bond",
    aliases=["007"],
)

prop_scores = {}
print("Semantic equivalence score using standard weights: %s" % (env.semantically_
↪equivalent(ta5, ta6, prop_scores)))
print(prop_scores)
for prop in prop_scores:
    if prop not in ["matching_score", "sum_weights"]:
```

(continues on next page)

(continued from previous page)

```
        print ("Prop: %s | weight: %s | contributing_score: %s" % (prop, prop_
↪scores[prop]['weight'], prop_scores[prop]['contributing_score']))
    else:
        print ("%s: %s" % (prop, prop_scores[prop]))
```

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

2.5.12 Custom Comparisons

If you wish, you can customize semantic equivalence comparisons. Specifically, you can do any of three things: - Provide custom weights for each semantic equivalence contributing property - Provide custom comparison functions for individual semantic equivalence contributing properties - Provide a custom semantic equivalence function for a specific object type

The `weights` dictionary

In order to do any of the aforementioned (*optional*) custom comparisons, you will need to provide a `weights` dictionary as the last parameter to the *semantically_equivalent()* method call.

The weights dictionary should contain both the weight and the comparison function for each property. You may use the default weights and functions, or provide your own.

Existing comparison functions

For reference, here is a list of the comparison functions already built in the codebase (found in *stix2/environment.py*):

- *custom_pattern_based*
- *exact_match*
- *partial_external_reference_based*
- *partial_list_based*
- *partial_location_distance*
- *partial_string_based*
- *partial_timestamp_based*

For instance, if we wanted to compare two of the `ThreatActors` from before, but use our own weights, then we could do the following:

[19]:
```
weights = {
    "threat-actor": {                                                    # You must␣
↪specify the object type
        "name": (30, stix2.environment.partial_string_based),           # Each␣
↪property's value must be a tuple
        "threat_actor_types": (50, stix2.environment.partial_list_based), # The 1st␣
↪component must be the weight
        "aliases": (20, stix2.environment.partial_list_based)            # The 2nd␣
↪component must be the comparison function
    }
}

print("Using standard weights: %s" % (env.semantically_equivalent(ta5, ta6)))
print("Using custom weights: %s" % (env.semantically_equivalent(ta5, ta6, **weights)))
```

[19]:
```
<IPython.core.display.HTML object>
```

[19]:
```
<IPython.core.display.HTML object>
```

Notice how there is a difference in the semantic equivalence scores, simply due to the fact that custom weights were used.

Custom Weights With prop_scores

If we want to use both `prop_scores` and `weights`, then they would be the third and fourth arguments, respectively, to `sematically_equivalent()`:

[20]:
```
prop_scores = {}
weights = {
    "threat-actor": {
        "name": (45, stix2.environment.partial_string_based),
        "threat_actor_types": (10, stix2.environment.partial_list_based),
        "aliases": (45, stix2.environment.partial_list_based),
    },
}
env.semantically_equivalent(ta5, ta6, prop_scores, **weights)
print(prop_scores)
```

[20]:
```
9.95
```

[20]:
```
<IPython.core.display.HTML object>
```

Custom Semantic Equivalence Functions

You can also write and use your own semantic equivalence functions. In the examples above, you could replace the built-in comparison functions for any or all properties. For example, here we use a custom string comparison function just for the `'name'` property:

[21]:
```
def my_string_compare(p1, p2):
    if p1 == p2:
        return 1
    else:
        return 0

weights = {
    "threat-actor": {
```

(continues on next page)

(continued from previous page)

```
        "name": (45, my_string_compare),
        "threat_actor_types": (10, stix2.environment.partial_list_based),
        "aliases": (45, stix2.environment.partial_list_based),
    },
}
print("Using custom string comparison: %s" % (env.semantically_equivalent(ta5, ta6,␣
↪**weights)))
```

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

You can also customize the comparison of an entire object type instead of just how each property is compared. To do this, provide a `weights` dictionary to `semantically_equivalent()` and in this dictionary include a key of `"method"` whose value is your custom semantic equivalence function for that object type.

If you provide your own custom semantic equivalence method, you **must also provide the weights for each of the properties** (unless, for some reason, your custom method is weights-agnostic). However, since you are writing the custom method, your weights need not necessarily follow the tuple format specified in the above code box.

Note also that if you want detailed results with `prop_scores` you will need to implement that in your custom function, but you are not required to do so.

In this next example we use our own custom semantic equivalence function to compare two `ThreatActors`, and do not support `prop_scores`.

[22]:
```
def custom_semantic_equivalence_method(obj1, obj2, **weights):
    sum_weights = 0
    matching_score = 0
    # Compare name
    w = weights['name']
    sum_weights += w
    contributing_score = w * stix2.environment.partial_string_based(obj1['name'],␣
↪obj2['name'])
    matching_score += contributing_score
    # Compare aliases only for spies
    if 'spy' in obj1['threat_actor_types'] + obj2['threat_actor_types']:
        w = weights['aliases']
        sum_weights += w
        contributing_score = w * stix2.environment.partial_list_based(obj1['aliases'],
↪ obj2['aliases'])
        matching_score += contributing_score

    return matching_score, sum_weights

weights = {
    "threat-actor": {
        "name": 60,
        "aliases": 40,
        "method": custom_semantic_equivalence_method
    }
}

print("Using standard weights: %s" % (env.semantically_equivalent(ta5, ta6)))
print("Using a custom method: %s" % (env.semantically_equivalent(ta5, ta6,␣
↪**weights)))
```

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

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

You can also write custom functions for comparing objects of your own custom types. Like in the previous example, you can use the built-in functions listed above to help with this, or write your own. In the following example we define semantic equivalence for our new `x-foobar` object type. Notice that this time we have included support for detailed results with `prop_scores`.

[23]:
```
def _x_foobar_checks(obj1, obj2, prop_scores, **weights):
    matching_score = 0.0
    sum_weights = 0.0
    if stix2.environment.check_property_present("name", obj1, obj2):
        w = weights["name"]
        sum_weights += w
        contributing_score = w * stix2.environment.partial_string_based(obj1["name"],␣
↪obj2["name"])
        matching_score += contributing_score
        prop_scores["name"] = (w, contributing_score)
    if stix2.environment.check_property_present("color", obj1, obj2):
        w = weights["color"]
        sum_weights += w
        contributing_score = w * stix2.environment.partial_string_based(obj1["color"],
↪ obj2["color"])
        matching_score += contributing_score
        prop_scores["color"] = (w, contributing_score)

    prop_scores["matching_score"] = matching_score
    prop_scores["sum_weights"] = sum_weights
    return matching_score, sum_weights

prop_scores = {}
weights = {
    "x-foobar": {
        "name": 60,
        "color": 40,
        "method": _x_foobar_checks,
    },
    "_internal": {
        "ignore_spec_version": False,
    },
}
foo1 = {
    "type":"x-foobar",
    "id":"x-foobar--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
    "name": "Zot",
    "color": "red",
}
foo2 = {
    "type":"x-foobar",
    "id":"x-foobar--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
    "name": "Zot",
    "color": "blue",
}
print(env.semantically_equivalent(foo1, foo2, prop_scores, **weights))
print(prop_scores)
```

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

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

2.6 FileSystem

The FileSystem suite contains *FileSystemStore*, *FileSystemSource* and *FileSystemSink*. Under the hood, all FileSystem objects point to a file directory (on disk) that contains STIX 2 content.

The directory and file structure of the intended STIX 2 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 STIX 2 content directory contains subdirectories, each of which aligns to a STIX 2 domain object type (i.e. "attack-pattern", "campaign", "malware", etc.). Within each STIX 2 domain object subdirectory are JSON files that are STIX 2 domain objects of the specified type. The name of the json files correspond to the ID of the STIX 2 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 STIX 2 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 STIX 2 content will only be retrieved or pushed, then a *FileSystemSource* and *FileSystemSink* can be used individually. They can also be used individually when STIX 2 content will be retrieved from one distinct file directory and pushed to another.

2.6.1 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 STIX 2 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.

2.6.2 FileSystem Examples

FileSystemStore

Use the FileSystemStore when you want to both retrieve STIX content from the file system and push STIX content to it, too.

[4]:
```
from stix2 import FileSystemStore

# create FileSystemStore
fs = FileSystemStore("/tmp/stix2_store")

# 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)
```

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

[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' =
↪'54b7e05e39a59428743635242e4a867c932140a999f52a1e54fa7ee6a440c73b']")

ind1 = Indicator(description="Crusades C2 implant 2",
                 labels=["malicious-activity"],
                 pattern="[file:hashes.'SHA-256' =
↪'64c7e05e40a59511743635242e4a867c932140a999f52a1e54fa7ee6a440c73b']")
```

(continues on next page)

(continued from previous page)

```
# 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

Use the FileSystemSource when you only want to retrieve STIX content from the file system.

[6]:
```
from stix2 import FileSystemSource

# create FileSystemSource
fs_source = FileSystemSource("/tmp/stix2_source")

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

# for visual purposes
print(ap)
```

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

[7]:
```
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.id)
```

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

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

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

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

[8]:
```
# 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.id)
```

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

FileSystemSink

Use the FileSystemSink when you only want to push STIX content to the file system.

[10]:
```
from stix2 import FileSystemSink, Campaign, Indicator

# create FileSystemSink
fs_sink = FileSystemSink("/tmp/stix2_sink")

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

ind = Indicator(description="Crusades C2 implant",
                labels=["malicious-activity"],
                pattern="[file:hashes.'SHA-256' =
↪'54b7e05e39a59428743635242e4a867c932140a999f52a1e54fa7ee6a440c73b']")

ind1 = Indicator(description="Crusades C2 implant",
                 labels=["malicious-activity"],
                 pattern="[file:hashes.'SHA-256' =
↪'54b7e05e39a59428743635242e4a867c932140a999f52a1e54fa7ee6a440c73b']")

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

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

2.7 Data Markings

2.7.1 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.

[7]:
```
from stix2 import Indicator, TLP_AMBER

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

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

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

[8]:
```
from stix2 import MarkingDefinition, StatementMarking

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

[8]:
```
<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).

[9]:
```
indicator2 = Indicator(labels=["malicious-activity"],
                       pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']
↪",
                       object_marking_refs=marking_definition)
print(indicator2)
```

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

[10]:
```
indicator3 = Indicator(labels=["malicious-activity"],
                       pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']
↪",
                       object_marking_refs="marking-definition--f88d31f6-486f-44da-
↪b317-01333bde0b82")
print(indicator3)
```

[10]:
```
<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).

[11]:
```
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)
```

[11]:
```
<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:

[12]:
```
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!
```

2.7.2 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).

[13]:
```
indicator4 = indicator.add_markings(marking_definition)
print(indicator4)
```

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

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

[14]:
```
indicator5 = indicator4.remove_markings(marking_definition)
print(indicator5)
```

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

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

[15]:
```
from stix2 import TLP_GREEN

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

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

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

[16]:
```
indicator7 = indicator5.clear_markings()
print(indicator7)
```

[16]:
```
<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.

2.7.3 Evaluating Data Markings

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

[17]:
```
indicator6.get_markings()
```

[17]:
```
['marking-definition--13680b12-3d19-4b42-abe6-0d31effe5368',
 'marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da']
```

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

[18]:
```
from stix2 import get_markings

get_markings(malware, 'name')
```

[18]:
```
['marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9']
```

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

[19]:
```
malware.get_markings('name')
```

```
[19]: ['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.

```
[20]: indicator.is_marked(TLP_AMBER.id)
[20]: True
```

```
[21]: malware.is_marked(TLP_WHITE.id, 'name')
[21]: True
```

```
[22]: malware.is_marked(TLP_WHITE.id, 'description')
[22]: False
```

Extracting Lang Data Markings or marking-definition Data Markings

If you need a specific kind of marking, you can also filter them using the API. By default the library will get both types of markings by default. You can choose between `lang=True/False` or `marking_ref=True/False` depending on your use-case.

```
[16]: from stix2 import v21

v21_indicator = v21.Indicator(
    description="Una descripcion sobre este indicador",
    pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']",
    object_marking_refs=['marking-definition--f88d31f6-486f-44da-b317-01333bde0b82'],
    indicator_types=['malware'],
    granular_markings=[
        {
            'selectors': ['description'],
            'lang': 'es'
        },
        {
            'selectors': ['description'],
            'marking_ref': 'marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da'
        }
    ]
)
print(v21_indicator)

# Gets both lang and marking_ref markings for 'description'
print(v21_indicator.get_markings('description'))

# Exclude lang markings from results
print(v21_indicator.get_markings('description', lang=False))

# Exclude marking-definition markings from results
print(v21_indicator.get_markings('description', marking_ref=False))
```

```
{
    "type": "indicator",
    "spec_version": "2.1",
    "id": "indicator-634ef462-d6b5-48bc-9d9f-b46a6919227c",
```

(continues on next page)

(continued from previous page)

```
    "created": "2019-05-03T18:36:44.354Z",
    "modified": "2019-05-03T18:36:44.354Z",
    "description": "Una descripcion sobre este indicador",
    "indicator_types": [
        "malware"
    ],
    "pattern": "[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']",
    "valid_from": "2019-05-03T18:36:44.354443Z",
    "object_marking_refs": [
        "marking-definition-f88d31f6-486f-44da-b317-01333bde0b82"
    ],
    "granular_markings": [
        {
            "lang": "es",
            "selectors": [
                "description"
            ]
        },
        {
            "marking_ref": "marking-definition-34098fce-860f-48ae-8e50-ebd3cc5e41da",
            "selectors": [
                "description"
            ]
        }
    ]
}
['es', 'marking-definition-34098fce-860f-48ae-8e50-ebd3cc5e41da']
['marking-definition-34098fce-860f-48ae-8e50-ebd3cc5e41da']
['es']
```

In this same manner, calls to `clear_markings` and `set_markings` also have the ability to operate in for one or both types of markings.

[5]:
```
print(v21_indicator.clear_markings("description"))  # By default, both types of␣
↪markings will be removed
```

```
{
    "type": "indicator",
    "spec_version": "2.1",
    "id": "indicator-a612665a-2df4-4fd2-851c-7fbb8c92339a",
    "created": "2019-05-03T19:13:59.010Z",
    "modified": "2019-05-03T19:15:41.173Z",
    "description": "Una descripcion sobre este indicador",
    "indicator_types": [
        "malware"
    ],
    "pattern": "[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']",
    "valid_from": "2019-05-03T19:13:59.010624Z",
    "object_marking_refs": [
        "marking-definition-f88d31f6-486f-44da-b317-01333bde0b82"
    ]
}
```

[13]:
```
# If lang is False, no lang markings will be removed
print(v21_indicator.clear_markings("description", lang=False))
```

```
{
```

(continues on next page)

(continued from previous page)

```
    "type": "indicator",
    "spec_version": "2.1",
    "id": "indicator-982aeb4d-4dd3-4b04-aa50-a1d00c31986c",
    "created": "2019-05-03T19:19:26.542Z",
    "modified": "2019-05-03T19:20:51.818Z",
    "description": "Una descripcion sobre este indicador",
    "indicator_types": [
        "malware"
    ],
    "pattern": "[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']",
    "valid_from": "2019-05-03T19:19:26.542267Z",
    "object_marking_refs": [
        "marking-definition-f88d31f6-486f-44da-b317-01333bde0b82"
    ],
    "granular_markings": [
        {
            "lang": "es",
            "selectors": [
                "description"
            ]
        }
    ]
}
```

[2]:
```
# If marking_ref is False, no marking-definition markings will be removed
print(v21_indicator.clear_markings("description", marking_ref=False))
```

```
{
    "type": "indicator",
    "spec_version": "2.1",
    "id": "indicator-de0316d6-38e1-43c2-af4f-649305251864",
    "created": "2019-05-03T19:40:21.459Z",
    "modified": "2019-05-03T19:40:26.431Z",
    "description": "Una descripcion sobre este indicador",
    "indicator_types": [
        "malware"
    ],
    "pattern": "[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']",
    "valid_from": "2019-05-03T19:40:21.459582Z",
    "object_marking_refs": [
        "marking-definition-f88d31f6-486f-44da-b317-01333bde0b82"
    ],
    "granular_markings": [
        {
            "marking_ref": "marking-definition-34098fce-860f-48ae-8e50-ebd3cc5e41da",
            "selectors": [
                "description"
            ]
        }
    ]
}
```

2.8 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.

2.8.1 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 and retrieves STIX content as python-stix2 objects.

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.

2.8.2 Memory Examples

MemoryStore

[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' =
↪'54b7e05e39a59428743635242e4a867c932140a999f52a1e54fa7ee6a440c73b']")

mem.add(ind)

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

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

[4]:
```
from stix2 import Malware

# add multiple STIX objects into memory
ind2 = Indicator(description="Crusades stage 2 implant",
```

(continues on next page)

(continued from previous page)

```
                labels=["malicious-activity"],
                pattern="[file:hashes.'SHA-256' =
↪'70fa62fb218dd9d936ee570dbe531dfa4e7c128ff37e6af7a6a6b2485487e50a']")
ind3 = Indicator(description="Crusades stage 2 implant variant",
                labels=["malicious-activity"],
                pattern="[file:hashes.'SHA-256' =
↪'31a45e777e4d58b97f4c43e38006f8cd6580ddabc4037905b2fad734712b582c']")
mal = Malware(labels=["rootkit"], name= "Alexios")

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

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

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

[5]:
```
from stix2 import Filter

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

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

2.8.3 load_from_file() and save_to_file()

[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("malware--9e9b87ce-2b2b-455a-8d5b-26384ccc8d52")

# for visual purposes
print(report)
```

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

2.9 Parsing STIX Content

Parsing STIX content is as easy as calling the parse() function on a JSON string, dictionary, or file-like object. 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.

Parsing a string

[3]:
```
from stix2 import parse

input_string = """{
    "type": "observed-data",
    "id": "observed-data--b67d30ff-02ac-498a-92f9-32f845f448cf",
```

(continues on next page)

(continued from previous page)

```
    "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(type(obj))
print(obj)
```

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

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

Parsing a dictionary

```
[4]: input_dict = {
    "type": "identity",
    "id": "identity--311b2d2d-f010-4473-83ec-1edf84858f4c",
    "created": "2015-12-21T19:59:11Z",
    "modified": "2015-12-21T19:59:11Z",
    "name": "Cole Powers",
    "identity_class": "individual"
}

obj = parse(input_dict)
print(type(obj))
print(obj)
```

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

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

Parsing a file-like object

```
[5]: file_handle = open("/tmp/stix2_store/course-of-action/course-of-action--d9727aee-48b8-
↪4fdb-89e2-4c49746ba4dd.json")

obj = parse(file_handle)
print(type(obj))
print(obj)
```

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

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

2.9.1 Parsing Custom STIX Content

Parsing custom STIX objects and/or STIX objects with custom properties is also completed easily with parse(). Just supply the keyword argument `allow_custom=True`. When `allow_custom` is specified, parse() will attempt to convert the supplied STIX content to known STIX 2 domain objects and/or previously defined *custom STIX 2 objects*. If the conversion cannot be completed (and `allow_custom` is specified), parse() will treat the supplied STIX 2 content as valid STIX 2 objects and return them. **Warning: Specifying allow_custom may lead to critical errors if further processing (searching, filtering, modifying etc...) of the custom content occurs where the custom content supplied is not valid STIX 2**. This is an axiomatic possibility as the `stix2` library cannot guarantee proper processing of unknown custom STIX 2 objects that were explicitly flagged to be allowed, and thus may not be valid.

For examples of parsing STIX 2 objects with custom STIX properties, see *Custom STIX Content: Custom Properties*

For examples of parsing defined custom STIX 2 objects, see *Custom STIX Content: Custom STIX Object Types*

For retrieving STIX 2 content from a source (e.g. file system, TAXII) that may possibly have custom STIX 2 content unknown to the user, the user can create a STIX 2 DataStore/Source with the flag `allow_custom=True`. As mentioned, this will configure the DataStore/Source to allow for unknown STIX 2 content to be returned (albeit not converted to full STIX 2 domain objects and properties); the `stix2` library may preclude processing the unknown content, if the content is not valid or actual STIX 2 domain objects and properties.

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

# to allow for the retrieval of unknown custom STIX2 content,
# just create *Stores/*Sources with the 'allow_custom' flag

# create FileSystemStore
fs = FileSystemSource("/path/to/stix2_data/", allow_custom=True)

# create TAXIICollectionSource
colxn = Collection('http://taxii_url')
ts = TAXIICollectionSource(colxn, allow_custom=True)
```

2.10 STIX2 Patterns

The Python `stix2` library supports STIX 2 patterning insofar that patterns may be used for the pattern property of Indicators, identical to the STIX 2 specification. `stix2` does not evaluate patterns against STIX 2 content; for that functionality see cti-pattern-matcher.

Patterns in the `stix2` library are built compositely from the bottom up, creating subcomponent expressions first before those at higher levels.

2.10.1 API Tips

ObservationExpression

Within the STIX 2 Patterning specification, Observation Expressions denote a complete expression to be evaluated against a discrete observation. In other words, an Observation Expression must be created to apply to a single Observation instance. This is further made clear by the visual brackets(`[]`) that encapsulate an Observation Expression.

Thus, whatever sub expressions that are within the Observation Expression are meant to be matched against the same Observable instance.

This requirement manifests itself within the `stix2` library via `ObservationExpression`. When creating STIX 2 observation expressions, whenever the current expression is complete, wrap it with `ObservationExpression()`. This allows the complete pattern expression - no matter its complexity - to be rendered as a proper specification-adhering string. ***Note: When pattern expressions are added to Indicator objects, the expression objects are implicitly converted to string representations***. While the extra step may seem tedious in the construction of simple pattern expressions, this explicit marking of observation expressions becomes vital when converting the pattern expressions to strings.

In all the examples, you can observe how in the process of building pattern expressions, when an Observation Expression is completed, it is wrapped with `ObservationExpression()`.

ParentheticalExpression

Do not be confused by the `ParentheticalExpression` object. It is not a distinct expression type but is also used to properly craft pattern expressions by denoting order priority and grouping of expression components. Use it in a similar manner as `ObservationExpression`, wrapping completed subcomponent expressions with `ParentheticalExpression()` if explicit ordering is required. For usage examples with `ParentheticalExpression`'s, see *here*.

BooleanExpressions vs CompoundObservationExpressions

Be careful to note the difference between these two very similar pattern components.

BooleanExpressions

- *AndBooleanExpression*
- *OrbooleanExpression*

Usage: When the boolean sub-expressions refer to the *same* root object

Example: `[domain-name:value = "www.5z8.info" AND domain-name:resolvess_to_refs[*].value = "'198.51.100.1/32'"]`

Rendering: when pattern is rendered, brackets or parenthesis will encapsulate boolean expression

CompoundObservationExpressions

- *AndObservationExpression*
- *OrObservationExpression*

Usage: When the boolean sub-expressions refer to *different* root objects

Example: `[file:name="foo.dll"] AND [process:name = "procfoo"]`

Rendering: when pattern is rendered, brackets will encapsulate each boolean sub-expression

2.10.2 Examples

Comparison Expressions

[3]:
```
from stix2 import DomainName, File, IPv4Address
from stix2 import (ObjectPath, EqualityComparisonExpression, ObservationExpression,
                   GreaterThanComparisonExpression, IsSubsetComparisonExpression,
                   FloatConstant, StringConstant)
```

Equality Comparison expressions

[7]:
```
lhs = ObjectPath("domain-name", ["value"])
ece_1 = ObservationExpression(EqualityComparisonExpression(lhs, "site.of.interest.zaz
↪"))
print("\t{}\n".format(ece_1))

lhs = ObjectPath("file", ["parent_directory_ref","path"])
ece_2 = ObservationExpression(EqualityComparisonExpression(lhs, "C:\\Windows\\System32
↪"))
print("\t{}\n".format(ece_2))
```

```
        [domain-name:value = 'site.of.interest.zaz']

        [file:parent_directory_ref.path = 'C:\\Windows\\System32']
```

Greater-than Comparison expressions

[5]:
```
lhs = ObjectPath("file", ["extensions", "windows-pebinary-ext", "sections[*]",
↪"entropy"])
gte = ObservationExpression(GreaterThanComparisonExpression(lhs, FloatConstant("7.0
↪")))
print("\t{}\n".format(gte))
```

```
        [file:extensions.windows-pebinary-ext.sections[*].entropy > 7.0]
```

IsSubset Comparison expressions

[6]:
```
lhs = ObjectPath("network-traffic", ["dst_ref", "value"])
iss = ObservationExpression(IsSubsetComparisonExpression(lhs, StringConstant(
↪"2001:0db8:dead:beef:0000:0000:0000:0000/64")))
print("\t{}\n".format(iss))
```

```
        [network-traffic:dst_ref.value ISSUBSET␣
↪'2001:0db8:dead:beef:0000:0000:0000:0000/64']
```

Compound Observation Expressions

[1]:
```
from stix2 import (IntegerConstant, HashConstant, ObjectPath,
                   EqualityComparisonExpression, AndBooleanExpression,
                   OrBooleanExpression, ParentheticalExpression,
                   AndObservationExpression, OrObservationExpression,
                   FollowedByObservationExpression, ObservationExpression)
```

AND boolean

[3]:
```
ece3 = EqualityComparisonExpression(ObjectPath("email-message", ["sender_ref", "value
↪"]), "stark@example.com")
ece4 = EqualityComparisonExpression(ObjectPath("email-message", ["subject"]),
↪"Conference Info")
abe = ObservationExpression(AndBooleanExpression([ece3, ece4]))
print("(AND)\n{}\n".format(abe))
```

```
(AND)
[email-message:sender_ref.value = 'stark@example.com' AND email-message:subject =␣
↪'Conference Info']
```

OR boolean

[4]:
```
ece5 = EqualityComparisonExpression(ObjectPath("url", ["value"]), "http://example.com/
↪foo")
ece6 = EqualityComparisonExpression(ObjectPath("url", ["value"]), "http://example.com/
↪bar")
obe = ObservationExpression(OrBooleanExpression([ece5, ece6]))
print("(OR)\n{}\n".format(obe))
```

```
(OR)
[url:value = 'http://example.com/foo' OR url:value = 'http://example.com/bar']
```

(OR) AND boolean

[5]:
```
ece7 = EqualityComparisonExpression(ObjectPath("file", ["name"]), "pdf.exe")
ece8 = EqualityComparisonExpression(ObjectPath("file", ["size"]), IntegerConstant(
↪"371712"))
ece9 = EqualityComparisonExpression(ObjectPath("file", ["created"]), "2014-01-
↪13T07:03:17Z")
obe1 = OrBooleanExpression([ece7, ece8])
pobe = ParentheticalExpression(obe1)
abe1 = ObservationExpression(AndBooleanExpression([pobe, ece9]))
print("(OR,AND)\n{}\n".format(abe1))
```

```
(OR,AND)
[(file:name = 'pdf.exe' OR file:size = 371712) AND file:created = 2014-01-13␣
↪07:03:17+00:00]
```

(AND) OR (OR) observation

[6]:
```
ece20 = ObservationExpression(EqualityComparisonExpression(ObjectPath("file", ["name
↪"]), "foo.dll"))
ece21 = ObservationExpression(EqualityComparisonExpression(ObjectPath("win-registry-
↪key", ["key"]), "HKEY_LOCAL_MACHINE\\foo\\bar"))
ece22 = EqualityComparisonExpression(ObjectPath("process", ["name"]), "fooproc")
ece23 = EqualityComparisonExpression(ObjectPath("process", ["name"]), "procfoo")
```

(continues on next page)

(continued from previous page)

```
# NOTE: we need to use AND/OR observation expression instead of just boolean
# expressions as the operands are not on the same object-type
aoe = ParentheticalExpression(AndObservationExpression([ece20, ece21]))
obe2 = ObservationExpression(OrBooleanExpression([ece22, ece23]))
ooe = OrObservationExpression([aoe, obe2])
print("(AND,OR,OR)\n{}\n".format(ooe))
```

```
(AND,OR,OR)
([file:name = 'foo.dll'] AND [win-registry-key:key = 'HKEY_LOCAL_MACHINE\\foo\\bar'])␣
↪OR [process:name = 'fooproc' OR process:name = 'procfoo']
```

FOLLOWED-BY

[7]:
```
ece10 = ObservationExpression(EqualityComparisonExpression(ObjectPath("file", ["hashes
↪", "MD5"]), HashConstant("79054025255fb1a26e4bc422aef54eb4", "MD5")))
ece11 = ObservationExpression(EqualityComparisonExpression(ObjectPath("win-registry-
↪key", ["key"]), "HKEY_LOCAL_MACHINE\\foo\\bar"))
fbe = FollowedByObservationExpression([ece10, ece11])
print("(FollowedBy)\n{}\n".format(fbe))
```

```
(FollowedBy)
[file:hashes.MD5 = '79054025255fb1a26e4bc422aef54eb4'] FOLLOWEDBY␣
↪[win-registry-key:key = 'HKEY_LOCAL_MACHINE\\foo\\bar']
```

Qualified Observation Expressions

[8]:
```
from stix2 import (TimestampConstant, HashConstant, ObjectPath,␣
↪EqualityComparisonExpression,
                   AndBooleanExpression, WithinQualifier, RepeatQualifier,␣
↪StartStopQualifier,
                   QualifiedObservationExpression, FollowedByObservationExpression,
                   ParentheticalExpression, ObservationExpression)
```

WITHIN

[9]:
```
ece10 = ObservationExpression(EqualityComparisonExpression(ObjectPath("file", ["hashes
↪", "MD5"]), HashConstant("79054025255fb1a26e4bc422aef54eb4", "MD5")))
ece11 = ObservationExpression(EqualityComparisonExpression(ObjectPath("win-registry-
↪key", ["key"]), "HKEY_LOCAL_MACHINE\\foo\\bar"))
fbe = FollowedByObservationExpression([ece10, ece11])
par = ParentheticalExpression(fbe)
qoe = QualifiedObservationExpression(par, WithinQualifier(300))
print("(WITHIN)\n{}\n".format(qoe))
```

```
(WITHIN)
([file:hashes.MD5 = '79054025255fb1a26e4bc422aef54eb4'] FOLLOWEDBY␣
↪[win-registry-key:key = 'HKEY_LOCAL_MACHINE\\foo\\bar']) WITHIN 300 SECONDS
```

REPEATS, WITHIN

[10]:
```
ece12 = EqualityComparisonExpression(ObjectPath("network-traffic", ["dst_ref", "type
↪"]), "domain-name")
ece13 = EqualityComparisonExpression(ObjectPath("network-traffic", ["dst_ref", "value
↪"]), "example.com")
abe2 = ObservationExpression(AndBooleanExpression([ece12, ece13]))
qoe1 = QualifiedObservationExpression(QualifiedObservationExpression(abe2,␣
↪RepeatQualifier(5)), WithinQualifier(180))
print("(REPEAT, WITHIN)\n{}\n".format(qoe1))
```

```
(REPEAT, WITHIN)
[network-traffic:dst_ref.type = 'domain-name' AND network-traffic:dst_ref.value =␣
↪'example.com'] REPEATS 5 TIMES WITHIN 180 SECONDS
```

START, STOP

[11]:
```
ece14 = ObservationExpression(EqualityComparisonExpression(ObjectPath("file", ["name
↪"]), "foo.dll"))
ssq = StartStopQualifier(TimestampConstant('2016-06-01T00:00:00Z'), TimestampConstant(
↪'2016-07-01T00:00:00Z'))
qoe2 = QualifiedObservationExpression(ece14, ssq)
print("(START-STOP)\n{}\n".format(qoe2))
```

```
(START-STOP)
[file:name = 'foo.dll'] START t'2016-06-01T00:00:00Z' STOP t'2016-07-01T00:00:00Z'
```

2.10.3 Attaching patterns to STIX2 Domain objects

Example

[10]:
```
from stix2 import Indicator, EqualityComparisonExpression, ObservationExpression

ece14 = ObservationExpression(EqualityComparisonExpression(ObjectPath("file", ["name
↪"]), "$$t00rzch$$.elf"))
ind = Indicator(name="Cryptotorch", labels=["malware", "ransomware"], pattern=ece14)
print(ind)
```

```
{
    "type": "indicator",
    "id": "indicator-219bc5fc-fdbf-4b54-a2fc-921be7ab3acb",
    "created": "2018-08-29T23:58:00.548Z",
    "modified": "2018-08-29T23:58:00.548Z",
    "name": "Cryptotorch",
    "pattern": "[file:name = '$$t00rzch$$.elf']",
    "valid_from": "2018-08-29T23:58:00.548391Z",
    "labels": [
        "malware",
        "ransomware"
    ]
}
```

2.11 Serializing STIX Objects

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

[3]:
```
from stix2 import Indicator

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

print(str(indicator))
```

[3]:
```
<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:

[4]:
```
print(indicator.serialize())
```

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

If you need performance but also need human-readable output, you can pass the `indent` keyword argument to `serialize()`:

[5]:
```
print(indicator.serialize(indent=4))
```

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

The only difference between this and the string representation from using `str()` is that this will not sort the keys. This works because the keyword arguments are passed to `json.dumps()` internally.

2.12 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.

2.12.1 TAXIICollection API

2.12.2 TAXIICollection Examples

TAXIICollectionSource

[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-a74d-c6fbd5a26116/", user="admin", password="Password0")
# supply the TAXII2 collection to TAXIICollection
tc_source = TAXIICollectionSource(collection)

#retrieve STIX objects by id
stix_obj = tc_source.get("malware--fdd60b30-b67c-41e3-b0b9-f01faf20d111")
stix_obj_versions = tc_source.all_versions("indicator--a932fcc6-e032-476c-826f-
↪cb970a5a1ade")

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

```
{
    "type": "malware",
    "id": "malware-fdd60b30-b67c-41e3-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-e032-476c-826f-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' =␣
↪'ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c']",
    "valid_from": "2014-05-08T09:00:00Z",
    "labels": [
        "file-hash-watchlist"
    ]
}
```

[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
```

(continues on next page)

(continued from previous page)

```
for indicator in indicators:
    print(indicator)
```

```
{
    "type": "indicator",
    "id": "indicator-a932fcc6-e032-476c-826f-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' =␣
↪'ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c']",
    "valid_from": "2014-05-08T09:00:00Z",
    "labels": [
        "file-hash-watchlist"
    ]
}
```

TAXIICollectionSink

[ ]:
```
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

[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-41e3-b0b9-f01faf20d111")

print(stix_obj2)
```

```
{
    "type": "malware",
    "id": "malware-fdd60b30-b67c-41e3-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"
    ]
}
```

[ ]:
```
from stix2 import indicator

# add STIX object to TAXIICollectionStore
ind = Indicator(description="Smokey Bear implant",
                labels=["malicious-activity"],
                pattern="[file:hashes.'SHA-256' =
↪'09c7e05a39a59428743635242e4a867c932140a909f12a1e54fa7ee6a440c73b']")

tc_store.add(ind)
```

2.12.3 Bug and Workaround

You may get an error similar to the following when adding STIX objects to a TAXIICollectionStore or TAXIICollectionSink:

```
TypeError: Object of type ThreatActor is not JSON serializable
```

This is a known bug and we are working to fix it. For more information, see this GitHub issue In the meantime, try this workaround:

[ ]:
```
tc_sink.add(json.loads(Bundle(ta).serialize()))
```

Or bypass the TAXIICollection altogether and interact with the collection itself:

[ ]:
```
collection.add_objects(json.loads(Bundle(ta).serialize()))
```

2.13 Technical Specification Support

2.13.1 How imports 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:

[ ]:
```
import stix2

stix2.Indicator()
```

or,

```
[ ]: from stix2 import Indicator

     Indicator()
```

People who want to use an explicit version:

```
[ ]: import stix2.v20

     stix2.v20.Indicator()
```

or,

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

     Indicator()
```

or even,

```
[ ]: 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:

```
[ ]: import stix2

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

or,

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

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

or (less preferred):

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

     Indicator_v20()
     Indicator_v21()
```

2.13.2 How parsing works

If the `version` positional argument is not provided. The library will make the best attempt using the "spec_version" property found on a Bundle, SDOs, and SROs.

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

[2]:
```
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)
```

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

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

2.13.3 How custom content works

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:

[ ]:
```
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
```

2.14 Versioning

To create a new version of an existing object, specify the property(ies) you want to change and their new values. For example, here we change the label from "anomalous-activity" to "malicious-activity":

[3]:
```
from stix2 import Indicator

indicator = Indicator(created="2016-01-01T08:00:00.000Z",
                      name="File hash for suspicious file",
                      description="A file indicator",
                      labels=["anomalous-activity"],
```

(continues on next page)

(continued from previous page)

```
                        pattern="[file:hashes.md5 = 'd41d8cd98f00b204e9800998ecf8427e']
↪")

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

```
[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.

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

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

You can remove optional or custom properties by setting them to `None` when you call `new_version()`.

```
[5]: indicator3 = indicator.new_version(description=None)
print(indicator3)
```

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

To revoke an object:

```
[6]: indicator4 = indicator3.revoke()
print(indicator4)
```

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

2.15 Using The Workbench

The *Workbench API* hides most of the complexity of the rest of the library to make it easy to interact with STIX data. To use it, just import everything from `stix2.workbench`:

```
[3]: from stix2.workbench import *
```

2.15.1 Retrieving STIX Data

To get some STIX data to work with, let's set up a DataSource and add it to our workbench.

```
[4]: from taxii2client import Collection

collection = Collection("http://127.0.0.1:5000/trustgroup1/collections/91a7b528-80eb-
↪42ed-a74d-c6fbd5a26116/", user="admin", password="Password0")
tc_source = TAXIICollectionSource(collection)
add_data_source(tc_source)
```

Now we can get all of the indicators from the data source.

```
[5]: response = indicators()
```

Similar functions are available for the other STIX Object types. See the full list *here*.

If you want to only retrieve *some* indicators, you can pass in one or more *Filters*. This example finds all the indicators created by a specific identity:

[6]:
```
response = indicators(filters=Filter('created_by_ref', '=', 'identity--adede3e8-bf44-
↪4e6f-b3c9-1958cbc3b188'))
```

The objects returned let you easily traverse their relationships. Get all Relationship objects involving that object with `.relationships()`, all other objects related to this object with `.related()`, and the Identity object for the creator of the object (if one exists) with `.created_by()`. For full details on these methods and their arguments, see the *Workbench API* documentation.

[7]:
```
for i in indicators():
    for rel in i.relationships():
        print(rel.source_ref)
        print(rel.relationship_type)
        print(rel.target_ref)
```

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

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

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

[8]:
```
for i in indicators():
    for obj in i.related():
        print(obj)
```

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

If there are a lot of related objects, you can narrow it down by passing in one or more *Filters* just as before. For example, if we want to get only the indicators related to a specific piece of malware (and not any entities that use it or are targeted by it):

[9]:
```
malware = get('malware--fdd60b30-b67c-41e3-b0b9-f01faf20d111')
indicator = malware.related(filters=Filter('type', '=', 'indicator'))
print(indicator[0])
```

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

2.15.2 Creating STIX Data

To create a STIX object, just use that object's class constructor. Once it's created, add it to the workbench with *save()*.

[10]:
```
identity = Identity(name="ACME Threat Intel Co.", identity_class="organization")
save(identity)
```

You can also set defaults for certain properties when creating objects. For example, let's set the default creator to be the identity object we just created:

[11]:
```
set_default_creator(identity)
```

Now when we create an indicator (or any other STIX Domain Object), it will automatically have the right `create_by_ref` value.

[12]:
```
indicator = Indicator(labels=["malicious-activity"], pattern="[file:hashes.MD5 =
↪'d41d8cd98f00b204e9800998ecf8427e']")
save(indicator)

indicator_creator = get(indicator.created_by_ref)
print(indicator_creator.name)
```

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

Defaults can also be set for the *created timestamp*, *external references* and *object marking references*.

Warning:

The workbench layer replaces STIX Object classes with special versions of them that use "wrappers" to provide extra functionality. Because of this, we recommend that you **either use the workbench layer or the rest of the library, but not both**. In other words, don't import from both `stix2.workbench` and any other submodules of `stix2`.

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 work*.

Python APIs for STIX 2.

confidence	Functions to operate with STIX2 Confidence scales.
core	
datastore	Python STIX2 DataStore API.
environment	Python STIX2 Environment API.
exceptions	STIX2 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	STIX 2.0 API Objects.
v21	STIX 2.1 API Objects.
workbench	Functions and class wrappers for interacting with STIX2 data at a high level.

3.1 confidence

Functions to operate with STIX2 Confidence scales.

scales	Functions to perform conversions between the different Confidence scales.

3.1.1 scales

Functions to perform conversions between the different Confidence scales. As specified in STIX™ Version 2.1. Part 1: STIX Core Concepts - Appendix B

admiralty_credibility_to_value(*scale_value*)

This method will transform a string value from the Admiralty Credibility scale to its confidence integer representation.

The scale for this confidence representation is the following:

Table 3: Admiralty Credibility Scale to STIX Confidence

Admiralty Credibility	STIX Confidence Value
6 - Truth cannot be judged	(Not present)
5 - Improbable	10
4 - Doubtful	30
3 - Possibly True	50
2 - Probably True	70
1 - Confirmed by other sources	90

Parameters **scale_value** (*str*) – A string value from the scale. Accepted strings are "6 - Truth cannot be judged", "5 - Improbable", "4 - Doubtful", "3 - Possibly True", "2 - Probably True" and "1 - Confirmed by other sources". Argument is case sensitive.

Returns

int –

The numerical representation corresponding to values in the Admiralty Credibility scale.

Raises `ValueError` – If *scale_value* is not within the accepted strings.

dni_to_value(*scale_value*)

This method will transform a string value from the DNI scale to its confidence integer representation.

The scale for this confidence representation is the following:

Table 4: DNI Scale to STIX Confidence

DNI Scale	STIX Confidence Value
Almost No Chance / Remote	5
Very Unlikely / Highly Improbable	15
Unlikely / Improbable	30
Roughly Even Chance / Roughly Even Odds	50
Likely / Probable	70
Very Likely / Highly Probable	85
Almost Certain / Nearly Certain	95

Parameters **scale_value** (*str*) – A string value from the scale. Accepted strings are "Almost No Chance / Remote", "Very Unlikely / Highly Improbable", "Unlikely / Improbable",

"Roughly Even Chance / Roughly Even Odds", "Likely / Probable", "Very Likely / Highly Probable" and "Almost Certain / Nearly Certain". Argument is case sensitive.

Returns

int –

The numerical representation corresponding to values in the DNI scale.

Raises `ValueError` – If *scale_value* is not within the accepted strings.

none_low_med_high_to_value(*scale_value*)

This method will transform a string value from the None / Low / Med / High scale to its confidence integer representation.

The scale for this confidence representation is the following:

Table 5: None, Low, Med, High to STIX Confidence

None/ Low/ Med/ High	STIX Confidence Value
Not Specified	Not Specified
None	0
Low	15
Med	50
High	85

Parameters **scale_value** (*str*) – A string value from the scale. Accepted strings are "None", "Low", "Med" and "High". Argument is case sensitive.

Returns

int –

The numerical representation corresponding to values in the None / Low / Med / High scale.

Raises `ValueError` – If *scale_value* is not within the accepted strings.

value_to_admiralty_credibility(*confidence_value*)

This method will transform an integer value into the Admiralty Credibility scale string representation.

The scale for this confidence representation is the following:

Table 6: STIX Confidence to Admiralty Credibility Scale

Range of Values	Admiralty Credibility
N/A	6 - Truth cannot be judged
0-19	5 - Improbable
20-39	4 - Doubtful
40-59	3 - Possibly True
60-79	2 - Probably True
80-100	1 - Confirmed by other sources

Parameters **confidence_value** (*int*) – An integer value between 0 and 100.

Returns *str* – A string corresponding to the Admiralty Credibility scale.

Raises `ValueError` – If *confidence_value* is out of bounds.

value_to_dni(*confidence_value*)

This method will transform an integer value into the DNI scale string representation.

The scale for this confidence representation is the following:

Table 7: STIX Confidence to DNI Scale

Range of Values	DNI Scale
0-9	Almost No Chance / Remote
10-19	Very Unlikely / Highly Improbable
20-39	Unlikely / Improbable
40-59	Roughly Even Chance / Roughly Even Odds
60-79	Likely / Probable
80-89	Very Likely / Highly Probable
90-100	Almost Certain / Nearly Certain

Parameters **confidence_value** (*int*) – An integer value between 0 and 100.

Returns *str* – A string corresponding to the DNI scale.

Raises `ValueError` – If *confidence_value* is out of bounds.

value_to_none_low_medium_high(*confidence_value*)

This method will transform an integer value into the None / Low / Med / High scale string representation.

The scale for this confidence representation is the following:

Table 8: STIX Confidence to None, Low, Med, High

Range of Values	None/ Low/ Med/ High
0	None
1-29	Low
30-69	Med
70-100	High

Parameters **confidence_value** (*int*) – An integer value between 0 and 100.

Returns *str* – A string corresponding to the None / Low / Med / High scale.

Raises `ValueError` – If *confidence_value* is out of bounds.

value_to_wep(*confidence_value*)

This method will transform an integer value into the WEP scale string representation.

The scale for this confidence representation is the following:

Table 9: STIX Confidence to WEP

Range of Values	WEP
0	Impossible
1-19	Highly Unlikely/Almost Certainly Not
20-39	Unlikely/Probably Not
40-59	Even Chance
60-79	Likely/Probable
80-99	Highly likely/Almost Certain
100	Certain

Parameters **confidence_value** (*int*) – An integer value between 0 and 100.

Returns *str* – A string corresponding to the WEP scale.

Raises `ValueError` – If *confidence_value* is out of bounds.

value_to_zero_ten(*confidence_value*)

This method will transform an integer value into the 0-10 scale string representation.

The scale for this confidence representation is the following:

Table 10: STIX Confidence to 0-10

Range of Values	0-10 Scale
0-4	0
5-14	1
15-24	2
25-34	3
35-44	4
45-54	5
55-64	6
65-74	7
75-84	8
95-94	9
95-100	10

Parameters **confidence_value** (*int*) – An integer value between 0 and 100.

Returns *str* – A string corresponding to the 0-10 scale.

Raises `ValueError` – If *confidence_value* is out of bounds.

wep_to_value(*scale_value*)

This method will transform a string value from the WEP scale to its confidence integer representation.

The scale for this confidence representation is the following:

Table 11: WEP to STIX Confidence

WEP	STIX Confidence Value
Impossible	0
Highly Unlikely/Almost Certainly Not	10
Unlikely/Probably Not	20
Even Chance	50
Likely/Probable	70
Highly likely/Almost Certain	90
Certain	100

Parameters **scale_value** (*str*) – A string value from the scale. Accepted strings are "Impossible", "Highly Unlikely/Almost Certainly Not", "Unlikely/Probably Not", "Even Chance", "Likely/Probable", "Highly likely/Almost Certain" and "Certain". Argument is case sensitive.

Returns

int –

The numerical representation corresponding to values in the WEP scale.

Raises `ValueError` – If *scale_value* is not within the accepted strings.

zero_ten_to_value(*scale_value*)

This method will transform a string value from the 0-10 scale to its confidence integer representation.

The scale for this confidence representation is the following:

Table 12: 0-10 to STIX Confidence

0-10 Scale	STIX Confidence Value
0	0
1	10
2	20
3	30
4	40
5	50
6	60
7	70
8	80
9	90
10	100

Parameters **scale_value** (*str*) – A string value from the scale. Accepted strings are "0" through "10" inclusive.

Returns

int –

The numerical representation corresponding to values in the 0-10 scale.

Raises ValueError – If *scale_value* is not within the accepted strings.

3.2 datastore

Python STIX2 DataStore API.

filesystem	Python STIX2 FileSystem Source/Sink
filters	Filters for Python STIX2 DataSources, DataSinks, DataStores
memory	Python STIX2 Memory Source/Sink
taxii	Python STIX2 TAXIICollection Source/Sink

3.2.1 filesystem

Python STIX2 FileSystem Source/Sink

class AuthSet(*allowed*, *prohibited*)

Represents either a whitelist or blacklist of values, where/what we must/must not search to find objects which match a query. (Maybe "AuthSet" isn't the right name, but determining authorization is a typical context in which black/white lists are used.)

The set may be empty. For a whitelist, this means you mustn't search anywhere, which means the query was

impossible to match, so you can skip searching altogether. For a blacklist, this means nothing is excluded and you must search everywhere.

BLACK = 0

WHITE = 1

auth_type
AuthSet.WHITE or AuthSet.BLACK.

Type Get the type of set

values
Get the values in this white/blacklist, as a set.

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*) – If present, it forces the parser to use the version provided. Otherwise, the library will make the best effort based on checking the "spec_version" property.

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*, *encoding='utf-8'*)
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
- **encoding** (*str*) – The encoding to use when reading a file from the filesystem.

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** (FilterSet) – collection of filters passed from the parent CompositeDataSource, not user supplied
- **version** (*str*) – If present, it forces the parser to use the version provided. Otherwise, the library will make the best effort based on checking the "spec_version" property.

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** (FilterSet) – collection of filters passed from the parent CompositeDataSource, not user supplied
- **version** (*str*) – If present, it forces the parser to use the version provided. Otherwise, the library will make the best effort based on checking the "spec_version" property.

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** (FilterSet) – collection of filters passed from the CompositeDataSource, not user supplied
- **version** (*str*) – If present, it forces the parser to use the version provided. Otherwise, the library will make the best effort based on checking the "spec_version" property.

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*, *encoding='utf-8'*)
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.
- **encoding** (*str*) – The encoding to use when reading a file from the filesystem.

source
FileSystemSource

Type *FileSystemSource*

sink
FileSystemSink

Type *FileSystemSink*

3.2.2 filters

Filters for Python STIX2 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")

class FilterSet (*filters=None*)
Internal STIX2 class to facilitate the grouping of Filters into sets. The primary motivation for this class came from the problem that Filters that had a dict as a value could not be added to a Python set as dicts are not hashable. Thus this class provides set functionality but internally stores filters in a list.

add (*filters=None*)
Add a Filter, FilterSet, or list of Filters to the FilterSet.

Operates like set, only adding unique stix2.Filters to the FilterSet

Note: method designed to be very accomodating (i.e. even accepting filters=None) as it allows for blind calls (very useful in DataStore)

Parameters **filters** – stix2.Filter OR list of stix2.Filter OR stix2.FilterSet

remove (*filters=None*)
Remove a Filter, list of Filters, or FilterSet from the FilterSet.

Note: method designed to be very accomodating (i.e. even accepting filters=None) as it allows for blind calls (very useful in DataStore)

Parameters **filters** – stix2.Filter OR list of stix2.Filter or stix2.FilterSet

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** (*iterable*) – iterable of STIX objects to apply the query to
- **query** (*non-iterator iterable*) – iterable of filters. Can't be an iterator (e.g. generator iterators won't work), since this is used in an inner loop of a nested loop. So we require the ability to traverse the filters repeatedly.

Yields STIX objects that successfully evaluate against the query.

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

3.2.3 memory

Python STIX2 Memory Source/Sink

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.
- **version** (*str*) – If present, it forces the parser to use the version provided. Otherwise, the library will make the best effort based on checking the "spec_version" property.

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

Type dict

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

- **store** – A MemoryStore, MemorySink or MemorySource object.
- **stix_data** (*list OR dict OR STIX object*) – STIX objects to be added
- **allow_custom** (*bool*) – Whether to allow custom properties as well unknown custom objects. Note that unknown custom objects cannot be parsed into STIX objects, and will be returned as is. Default: False.
- **version** (*str*) – Which STIX2 version to lock the parser to. (e.g. "2.0", "2.1"). If None, the library makes the best effort to figure out the spec representation of the object.

save_to_file(*path*, *encoding='utf-8'*)
Write SITX objects from in-memory dictionary to JSON file, as a STIX Bundle. If a directory is given, the Bundle 'id' will be used as filename. Otherwise, the provided value will be used.

Parameters

- **path** (*str*) – file path to write STIX data to.
- **encoding** (*str*) – The file encoding. Default utf-8.

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.
- **version** (*str*) – If present, it forces the parser to use the version provided. Otherwise, the library will make the best effort based on checking the "spec_version" property.

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

Type dict

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

Parameters

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

Returns *(list)* – list of STIX objects that have the supplied ID.

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** (FilterSet) – collection of filters passed from the parent CompositeDataSource, not user supplied

Returns *(STIX object)* – STIX object that has the supplied ID.

load_from_file(*file_path*, *version=None*, *encoding='utf-8'*)
Load STIX data from JSON file.

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

Parameters **path** (*str*) – file path to load STIX data from

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** (FilterSet) – collection of filters passed from the CompositeDataSource, not user supplied

Returns *(list)* – list of STIX objects that match the supplied query.

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.

_data
the in-memory dict that holds STIX objects

Type dict

source
MemorySource

Type *MemorySource*

sink
MemorySink

Type *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 **path** (*str*) – file path to load STIX data from

save_to_file(**args*, ***kwargs*)
Write SITX objects from in-memory dictionary to JSON file, as a STIX Bundle. If a directory is given, the Bundle 'id' will be used as filename. Otherwise, the provided value will be used.

Parameters

- **path** (*str*) – file path to write STIX data to.
- **encoding** (*str*) – The file encoding. Default utf-8.

3.2.4 taxii

Python STIX2 TAXIICollection Source/Sink

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 STIX2 content in a STIX object (or Bundle), STIX object dict (or Bundle dict), or a STIX2 json encoded string, or list of any of the following.
- **version** (*str*) – If present, it forces the parser to use the version provided. Otherwise, the library will make the best effort based on checking the "spec_version" property.

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.
- **version** (*str*) – If present, it forces the parser to use the version provided. Otherwise, the library will make the best effort based on checking the "spec_version" property.
- **_composite_filters** (FilterSet) – collection of filters passed from the parent CompositeDataSource, not user supplied

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.
- **version** (*str*) – If present, it forces the parser to use the version provided. Otherwise, the library will make the best effort based on checking the "spec_version" property.
- **_composite_filters** (FilterSet) – collection of filters passed from the parent CompositeDataSource, not user supplied

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
- **version** (*str*) – If present, it forces the parser to use the version provided. Otherwise, the library will make the best effort based on checking the "spec_version" property.
- **_composite_filters** (FilterSet) – collection of filters passed from the CompositeDataSource, not user supplied

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.

exception DataSourceError(*message*, *root_exception=None*)
General DataSource error instance, used primarily for wrapping lower level errors

Parameters

- **message** (*str*) – error message
- **root_exception** (*Exception*) – Exception instance of root exception in the case that DataSourceError is wrapping a lower level or other exception

class CompositeDataSource

Controller for all the attached DataSources.

A user can have a single CompositeDataSource as an interface to 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

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

Type list

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

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

Returns *list* – The 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.
- **filters** (*list*) – list of additional filters the related objects must match.

Returns *list* – The 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* – The 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
A unique UUIDv4 to identify this DataSink.

Type str

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
A unique UUIDv4 to identify this DataSource.

Type str

filters
A collection of filters attached to this DataSource.

Type *FilterSet*

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 *list* – All versions of the specified STIX object.

creator_of(*obj*)
Retrieve the Identity referred 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 *list* – The STIX objects that matched the query.

related_to(*obj*, *relationship_type=None*, *source_only=False*, *target_only=False*, *filters=None*)

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.
- **filters** (*list*) – list of additional filters the related objects must match.

Returns *list* – The 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* – The 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
A unique UUIDv4 to identify this DataStore.

Type str

source
An object that implements DataSource class.

Type *DataSource*

sink
An object that implements DataSink class.

Type *DataSink*

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

Defines custom behavior before storing STIX objects using the appropriate method call on the associated DataSink.

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 *list* – All versions of the specified STIX object.

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 *list* – The STIX objects matching the query.

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.
- **filters** (*list*) – list of additional filters the related objects must match.

Returns *list* – The 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* – The Relationship objects involving the given STIX object.

make_id()

3.3 environment

Python STIX2 Environment API.

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 *list* – All versions of the specified STIX object.

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 *list* – The STIX objects matching the query.

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

str –

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* – The 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.
- **filters** (*list*) – list of additional filters the related objects must match.

Returns *list* – The STIX objects related to the given STIX object.

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

Method for storing STIX objects.

Defines custom behavior before storing STIX objects using the appropriate method call on the associated DataSink.

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

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

str –

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

parse(**args*, ***kwargs*)
Convert a string, dict 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 as well unknown custom objects. Note that unknown custom objects cannot be parsed into STIX objects, and will be returned as is. Default: False.
- **version** (*str*) – If present, it forces the parser to use the version provided. Otherwise, the library will make the best effort based on checking the "spec_version" property. If none of the above are possible, it will use the default version specified by the library.

Returns An instantiated Python STIX object.

Warning: 'allow_custom=True' will allow for the return of any supplied STIX dict(s) that cannot be found to map to any known STIX object types (both STIX2 domain objects or defined custom STIX2 objects); NO validation is done. This is done to allow the processing of possibly unknown custom STIX objects (example scenario: I need to query a third-party TAXII endpoint that could provide custom STIX objects that I don't know about ahead of time)

static semantically_equivalent(*obj1*, *obj2*, *prop_scores={}*, ***weight_dict*)
This method is meant to verify if two objects of the same type are semantically equivalent.

Parameters

- **obj1** – A stix2 object instance
- **obj2** – A stix2 object instance
- **weight_dict** – A dictionary that can be used to override settings in the semantic equivalence process

Returns *float* – A number between 0.0 and 100.0 as a measurement of equivalence.

Warning: Course of Action, Intrusion-Set, Observed-Data, Report are not supported by this implementation. Indicator pattern check is also limited.

Note: Default weights_dict:

```
{
    "attack-pattern": {
        "name": [
            30,
            partial_string_based
        ],
        "external_references": [
            70,
            partial_external_reference_based
        ]
    },
    "campaign": {
        "name": [
            60,
```

(continues on next page)

(continued from previous page)

```
            partial_string_based
        ],
        "aliases": [
            40,
            partial_list_based
        ]
    },
    "identity": {
        "name": [
            60,
            partial_string_based
        ],
        "identity_class": [
            20,
            exact_match
        ],
        "sectors": [
            20,
            partial_list_based
        ]
    },
    "indicator": {
        "indicator_types": [
            15,
            partial_list_based
        ],
        "pattern": [
            80,
            custom_pattern_based
        ],
        "valid_from": [
            5,
            partial_timestamp_based
        ],
        "tdelta": 1
    },
    "location": {
        "longitude_latitude": [
            34,
            partial_location_distance
        ],
        "region": [
            33,
            exact_match
        ],
        "country": [
            33,
            exact_match
        ],
        "threshold": 1000.0
    },
    "malware": {
        "malware_types": [
            20,
            partial_list_based
        ],
        "name": [
```

(continues on next page)

(continued from previous page)

```
            80,
            partial_string_based
        ]
    },
    "threat-actor": {
        "name": [
            60,
            partial_string_based
        ],
        "threat_actor_types": [
            20,
            partial_list_based
        ],
        "aliases": [
            20,
            partial_list_based
        ]
    },
    "tool": {
        "tool_types": [
            20,
            partial_list_based
        ],
        "name": [
            80,
            partial_string_based
        ]
    },
    "vulnerability": {
        "name": [
            30,
            partial_string_based
        ],
        "external_references": [
            70,
            partial_external_reference_based
        ]
    },
    "_internal": {
        "ignore_spec_version": false
    }
}
```

Note: This implementation follows the Committee Note on semantic equivalence. see the Committee Note.

set_default_created(**args*, ***kwargs*)
Set default value for the *created* property.

set_default_creator(**args*, ***kwargs*)
Set default value for the *created_by_ref* property.

set_default_external_refs(**args*, ***kwargs*)
Set default external references.

set_default_object_marking_refs(**args*, ***kwargs*)
Set default object markings.

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

set_default_created(*created=None*)
Set default value for the *created* property.

set_default_creator(*creator=None*)
Set default value for the *created_by_ref* property.

set_default_external_refs(*external_references=None*)
Set default external references.

set_default_object_marking_refs(*object_marking_refs=None*)
Set default object markings.

check_property_present(*prop*, *obj1*, *obj2*)
Helper method checks if a property is present on both objects.

custom_pattern_based(*pattern1*, *pattern2*)
Performs a matching on Indicator Patterns.

Parameters

- **pattern1** – An Indicator pattern
- **pattern2** – An Indicator pattern

Returns *float* – Number between 0.0 and 1.0 depending on match criteria.

exact_match(*val1*, *val2*)
Performs an exact value match based on two values

Parameters

- **val1** – A value suitable for an equality test.
- **val2** – A value suitable for an equality test.

Returns *float* – 1.0 if the value matches exactly, 0.0 otherwise.

partial_external_reference_based(*refs1*, *refs2*)
Performs a matching on External References.

Parameters

- **refs1** – A list of external references.
- **refs2** – A list of external references.

Returns *float* – Number between 0.0 and 1.0 depending on matches.

partial_list_based(*l1*, *l2*)
Performs a partial list matching via finding the intersection between common values.

Parameters

- **l1** – A list of values.
- **l2** – A list of values.

Returns *float* – 1.0 if the value matches exactly, 0.0 otherwise.

partial_location_distance(*lat1*, *long1*, *lat2*, *long2*, *threshold*)
Given two coordinates perform a matching based on its distance using the Haversine Formula.

Parameters

- **lat1** – Latitude value for first coordinate point.
- **lat2** – Latitude value for second coordinate point.
- **long1** – Longitude value for first coordinate point.
- **long2** – Longitude value for second coordinate point.
- **threshold** (*float*) – A kilometer measurement for the threshold distance between these two points.

Returns *float* – Number between 0.0 and 1.0 depending on match.

partial_string_based(*str1*, *str2*)
Performs a partial string match using the Jaro-Winkler distance algorithm.

Parameters

- **str1** – A string value to check.
- **str2** – A string value to check.

Returns *float* – Number between 0.0 and 1.0 depending on match criteria.

partial_timestamp_based(*t1*, *t2*, *tdelta*)
Performs a timestamp-based matching via checking how close one timestamp is to another.

Parameters

- **t1** – A datetime string or STIXdatetime object.
- **t2** – A datetime string or STIXdatetime object.
- **tdelta** (*float*) – A given time delta. This number is multiplied by 86400 (1 day) to extend or shrink your time change tolerance.

Returns *float* – Number between 0.0 and 1.0 depending on match criteria.

WEIGHTS = {'_internal': {'ignore_spec_version': False}, 'attack-pattern': {'external_re
autodoc-skip:

3.4 exceptions

STIX2 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 CustomContentError(*msg*)
Custom STIX Content (SDO, Observable, Extension, etc.) detected.

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 DuplicateRegistrationError(*obj_type*, *reg_obj_type*)
A STIX object with the same type as an existing object is being registered

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 ObjectConfigurationError
Represents specification violations regarding the composition of STIX objects.

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

exception PropertyPresenceError(*message*, *cls*)
Represents an invalid combination of properties on a STIX object. This class can be used directly when the object requirements are more complicated and none of the more specific exception subclasses apply.

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

exception STIXDeprecationWarning
Represents usage of a deprecated component of a STIX specification.

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

exception TLPMarkingDefinitionError(*user_obj*, *spec_obj*)
Marking violation. The marking-definition for TLP MUST follow the mandated instances from the spec.

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 STIX2 granular markings.
object_markings	Functions for working with STIX2 object markings.
utils	Utility functions for STIX2 data markings.

3.5.1 granular_markings

Functions for working with STIX2 granular markings.

add_markings(*obj*, *marking*, *selectors*)
Append a granular marking to the granular_markings collection. The method makes a best-effort attempt to distinguish between a marking-definition or language granular marking.

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*, *marking_ref=True*, *lang=True*)
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.

- **marking_ref** (*bool*) – If False, markings that use the `marking_ref` property will not be removed.
- **lang** (*bool*) – If False, markings that use the `lang` property will not be removed.

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*, *marking_ref=True*, *lang=True*)

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** (*bool*) – If True, include markings inherited relative to the properties.
- **descendants** (*bool*) – If True, include granular markings applied to any children relative to the properties.
- **marking_ref** (*bool*) – If False, excludes markings that use `marking_ref` property.
- **lang** (*bool*) – If False, excludes markings that use `lang` property.

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** (*bool*) – string or list of selectors strings relative to the SDO or SRO in which the properties appear.
- **inherited** (*bool*) – If True, return markings inherited from the given selector.
- **descendants** (*bool*) – 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. The method makes a best-effort attempt to distinguish between a marking-definition or language granular marking.

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*, *marking_ref=True*, *lang=True*)
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*.
- **marking_ref** (*bool*) – If False, markings that use the `marking_ref` property will not be removed.
- **lang** (*bool*) – If False, markings that use the `lang` property will not be removed.

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 STIX2 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 STIX2 data markings.

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

check_tlp_marking(*marking_obj*, *spec_version*)

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"
```

(continues on next page)

(continued from previous page)

```
...         ],
...         "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*, *marking_ref=True*, *lang=True*)
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).
- **marking_ref** (*bool*) – If False, markings that use the `marking_ref` property will not be removed.
- **lang** (*bool*) – If False, markings that use the `lang` property will not be removed.

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*, *marking_ref=True*, *lang=True*)
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** (*bool*) – If True, include object level markings and granular markings inherited relative to the properties.
- **descendants** (*bool*) – If True, include granular markings applied to any children relative to the properties.

- **marking_ref** (*bool*) – If False, excludes markings that use `marking_ref` property.
- **lang** (*bool*) – If False, excludes markings that use `lang` property.

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** (*bool*) – If True, include object level markings and granular markings inherited to determine if the properties is/are marked.
- **descendants** (*bool*) – 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*, *marking_ref=True*, *lang=True*)
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.
- **marking_ref** (*bool*) – If False, markings that use the `marking_ref` property will not be removed.
- **lang** (*bool*) – If False, markings that use the `lang` property will not be removed.

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*)
'AND' Boolean Pattern Expression. Only use if both operands are of the same root object.

Parameters **operands** (*list*) – AND operands

class AndObservationExpression(*operands*)
'AND' Compound Observation Pattern Expression

Parameters **operands** (*str*) – compound observation operands

class BasicObjectPathComponent(*property_name*, *is_key*)
Basic object path component (for an observation or expression)

By "Basic", implies that the object path component is not a list, object reference or futher referenced property, i.e. terminal component

Parameters

- **property_name** (*str*) – object property name
- **is_key** (*bool*) – is dictionary key, default: False

class BinaryConstant(*value*, *from_parse_tree=False*)
Pattern binary constant

Parameters **value** (*str*) – base64 encoded string value

class BooleanConstant(*value*)
Pattern boolean constant

Parameters **value** (*str OR int*) – (str) 'true', 't' for True; 'false', 'f' for False (int) 1 for True; 0 for False

class EqualityComparisonExpression(*lhs*, *rhs*, *negated=False*)
Pattern Equality Comparison Expression

Parameters

- **lhs** (*ObjectPath OR str*) – object path of left-hand-side component of expression
- **rhs** (*ObjectPath OR str*) – object path of right-hand-side component of expression
- **negated** (*bool*) – comparison expression negated. Default: False

class FloatConstant(*value*)

class FollowedByObservationExpression(*operands*)
Pattern 'Followed by' Compound Observation Expression

Parameters **operands** (*str*) – compound observation operands

class GreaterThanComparisonExpression(*lhs*, *rhs*, *negated=False*)
Pattern Greater-than Comparison Expression

Parameters

- **lhs** (*ObjectPath OR str*) – object path of left-hand-side component of expression
- **rhs** (*ObjectPath OR str*) – object path of right-hand-side component of expression
- **negated** (*bool*) – comparison expression negated. Default: False

class GreaterThanEqualComparisonExpression(*lhs*, *rhs*, *negated=False*)
Pattern Greater-Than-or-Equal-to Comparison Expression

Parameters

- **lhs** (*ObjectPath OR str*) – object path of left-hand-side component of expression
- **rhs** (*ObjectPath OR str*) – object path of right-hand-side component of expression
- **negated** (*bool*) – comparison expression negated. Default: False

class HashConstant(*value*, *type*)
Pattern hash constant

Parameters

- **value** (*str*) – hash value
- **type** (*str*) – hash algorithm name. Supported hash algorithms: "MD5", "MD6", RIPEMD160", "SHA1", "SHA224", "SHA256", "SHA384", "SHA512", "SHA3224", "SHA3256", "SHA3384", "SHA3512", "SSDEEP", "WHIRLPOOL"

class HexConstant(*value*, *from_parse_tree=False*)
Pattern hexadecimal constant

Parameters **value** (*str*) – hexadecimal value

class InComparisonExpression(*lhs*, *rhs*, *negated=False*)
'in' Comparison Expression

Parameters

- **lhs** (*ObjectPath OR str*) – object path of left-hand-side component of expression

- **rhs** (*ObjectPath OR str*) – object path of right-hand-side component of expression
- **negated** (*bool*) – comparison expression negated. Default: False

class IntegerConstant(*value*)
Pattern interger constant

Parameters **value** (*int*) – integer value

class IsSubsetComparisonExpression(*lhs*, *rhs*, *negated=False*)
'is subset' Comparison Expression

Parameters

- **lhs** (*ObjectPath OR str*) – object path of left-hand-side component of expression
- **rhs** (*ObjectPath OR str*) – object path of right-hand-side component of expression
- **negated** (*bool*) – comparison expression negated. Default: False

class IsSupersetComparisonExpression(*lhs*, *rhs*, *negated=False*)
'is super set' Comparison Expression

Parameters

- **lhs** (*ObjectPath OR str*) – object path of left-hand-side component of expression
- **rhs** (*ObjectPath OR str*) – object path of right-hand-side component of expression
- **negated** (*bool*) – comparison expression negated. Default: False

class LessThanComparisonExpression(*lhs*, *rhs*, *negated=False*)
Pattern Less-than Comparison Expression

Parameters

- **lhs** (*ObjectPath OR str*) – object path of left-hand-side component of expression
- **rhs** (*ObjectPath OR str*) – object path of right-hand-side component of expression
- **negated** (*bool*) – comparison expression negated. Default: False

class LessThanEqualComparisonExpression(*lhs*, *rhs*, *negated=False*)
Pattern Less-Than-or-Equal-to Comparison Expression

Parameters

- **lhs** (*ObjectPath OR str*) – object path of left-hand-side component of expression
- **rhs** (*ObjectPath OR str*) – object path of right-hand-side component of expression
- **negated** (*bool*) – comparison expression negated. Default: False

class LikeComparisonExpression(*lhs*, *rhs*, *negated=False*)
'like' Comparison Expression

Parameters

- **lhs** (*ObjectPath OR str*) – object path of left-hand-side component of expression
- **rhs** (*ObjectPath OR str*) – object path of right-hand-side component of expression
- **negated** (*bool*) – comparison expression negated. Default: False

class ListConstant(*values*)
Pattern list constant

Parameters **value** (*list*) – list of values

class ListObjectPathComponent(*property_name*, *index*)
List object path component (for an observation or expression)

Parameters

- **property_name** (*str*) – list object property name
- **index** (*int*) – index of the list property's value that is specified

class MatchesComparisonExpression(*lhs*, *rhs*, *negated=False*)
'Matches' Comparison Expression

Parameters

- **lhs** (*ObjectPath OR str*) – object path of left-hand-side component of expression
- **rhs** (*ObjectPath OR str*) – object path of right-hand-side component of expression
- **negated** (*bool*) – comparison expression negated. Default: False

class ObjectPath(*object_type_name*, *property_path*)
Pattern operand object (property) path

Parameters

- **object_type_name** (*str*) – name of object type for corresponding object path component
- **property_path** (*_ObjectPathComponent OR str*) – object path

static make_object_path(*lhs*)
Create ObjectPath from string encoded object path

Parameters **lhs** (*str*) – object path of left-hand-side component of expression

merge(*other*)
Extend the object property with that of the supplied object property path

class ObservationExpression(*operand*)
Observation Expression

Parameters **operand** (*str*) – observation expression operand

class OrBooleanExpression(*operands*)
'OR' Boolean Pattern Expression. Only use if both operands are of the same root object

Parameters **operands** (*list*) – OR operands

class OrObservationExpression(*operands*)
Pattern 'OR' Compound Observation Expression

Parameters **operands** (*str*) – compound observation operands

class ParentheticalExpression(*exp*)
Pattern Parenthetical Observation Expression

Parameters **exp** (*str*) – observation expression

class QualifiedObservationExpression(*observation_expression*, *qualifier*)
Pattern Qualified Observation Expression

Parameters

- **observation_expression** (*PatternExpression OR _CompoundObservationExpression OR*) – pattern expression
- **qualifier** (*_ExpressionQualifier*) – pattern expression qualifier

class ReferenceObjectPathComponent(*reference_property_name*)
Reference object path component (for an observation or expression)

Parameters **reference_property_name** (*str*) – reference object property name

class RepeatQualifier(*times_to_repeat*)
Pattern Repeat Qualifier

Parameters **times_to_repeat** (*int*) – times the qualifiers is repeated

class StartStopQualifier(*start_time*, *stop_time*)
Pattern Start/Stop Qualifier

Parameters

- **start_time** (*TimestampConstant OR datetime.date*) – start timestamp for qualifier
- **stop_time** (*TimestampConstant OR datetime.date*) – stop timestamp for qualifier

class StringConstant(*value*, *from_parse_tree=False*)
Pattern string constant

Parameters **value** (*str*) – string value

class TimestampConstant(*value*)
Pattern timestamp constant

Parameters **value** (*datetime.datetime OR str*) – if string, must be a timestamp string

class WithinQualifier(*number_of_seconds*)
Pattern 'Within' Qualifier

Parameters **number_of_seconds** (*int*) – seconds value for 'within' qualifier

escape_quotes_and_backslashes(*s*)

make_constant(*value*)
Convert value to Pattern constant, best effort attempt at determining root value type and corresponding conversion

Parameters **value** – value to convert to Pattern constant

quote_if_needed(*x*)

3.7 properties

Classes for representing properties of STIX Objects and Cyber Observables.

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

clean(*value*)

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

clean(*value*)

class DictionaryProperty(*spec_version='2.0'*, ***kwargs*)

clean(*value*)

class EmbeddedObjectProperty(*type*, ***kwargs*)

clean(*value*)

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

clean(*value*)

class ExtensionsProperty(*spec_version='2.0'*, *allow_custom=False*, *enclosing_type=None*, *required=False*)

Property for representing extensions on Observable objects.

clean(*value*)

class FloatProperty(*min=None*, *max=None*, ***kwargs*)

clean(*value*)

class HashesProperty(*spec_version='2.0'*, ***kwargs*)

clean(*value*)

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

clean(*value*)

class IDProperty(*type*, *spec_version='2.0'*)

clean(*value*)

default()

class IntegerProperty(*min=None*, *max=None*, ***kwargs*)

clean(*value*)

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

clean(*value*)

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

class ObservableProperty(*spec_version='2.0'*, *allow_custom=False*, **args*, ***kwargs*)

Property for holding Cyber Observable Objects.

clean(*value*)

class PatternProperty(***kwargs*)

class Property(*required=False*, *fixed=None*, *default=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 an exception.
- **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(*valid_types=None*, *invalid_types=None*, *spec_version='2.0'*, ***kwargs*)

clean(*value*)

class STIXObjectProperty(*spec_version='2.0'*, *allow_custom=False*, **args*, ***kwargs*)

clean(*value*)

class SelectorProperty(*required=False*, *fixed=None*, *default=None*)

clean(*value*)

class StringProperty(***kwargs*)

clean(*value*)

class TimestampProperty(*precision='any'*, *precision_constraint='exact'*, ***kwargs*)

clean(*value*)

class TypeProperty(*type*, *spec_version='2.0'*)

enumerate_types(*types*, *spec_version*)

***types* is meant to be a list; it may contain specific object types and/or** the any of the words "SCO", "SDO", or "SRO"

Since "SCO", "SDO", and "SRO" are general types that encompass various specific object types, once each of those words is being processed, that word will be removed from *return_types*, so as not to mistakenly allow objects to be created of types "SCO", "SDO", or "SRO"

3.8 utils

Utility functions and classes for the STIX2 library.

class Precision
Timestamp format precisions.

ANY = 1

MILLISECOND = 3

SECOND = 2

class PrecisionConstraint
Timestamp precision constraints. These affect how the Precision values are applied when formatting a timestamp.

These constraints don't really make sense with the ANY precision, so they have no effect in that case.

EXACT = 1

MIN = 2

class STIXdatetime
Bundle a datetime with some format-related metadata, so that JSON serialization has the info it needs to produce compliant timestamps.

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*, *search_key*, *search_value*)
Search (recursively) for the given key and value in the given object. Return an index for the key, relative to whatever object it's found in.

Parameters

- **obj** – The object to search (list, dict, or stix object)
- **search_key** – A search key
- **search_value** – A search value

Returns *int* – An index; -1 if the key and value aren't found

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 warranted, according to precision settings
5. Add "Z"

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

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

get_type_from_id(*stix_id*)

is_marking(*obj_or_id*)
Determines whether the given object or object ID is/is for a marking definition.

Parameters **obj_or_id** – A STIX object or object ID as a string.

Returns True if a marking definition, False otherwise.

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=<Precision.ANY: 1>*, *precision_constraint=<PrecisionConstraint.EXACT: 1>*)
Parse a value into a valid STIX timestamp object. Also, optionally adjust precision of fractional seconds. This allows alignment with JSON serialization requirements, and helps ensure we're not using extra precision which would be lost upon JSON serialization. The precision info will be embedded in the returned object, so that JSON serialization will format it correctly.

Parameters

- **value** – A datetime.datetime or datetime.date instance, or a string
- **precision** – A precision value: either an instance of the Precision enum, or a string naming one of the enum values (case-insensitive)
- **precision_constraint** – A precision constraint value: either an instance of the PrecisionConstraint enum, or a string naming one of the enum values (case-insensitive)

Returns A STIXdatetime instance, which is a datetime but also carries the precision info necessary to properly JSON-serialize it.

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 v20

STIX 2.0 API Objects.

bundle	STIX 2.0 Bundle Representation.
common	STIX 2.0 Common Data Types and Properties.
observables	STIX 2.0 Cyber Observable Objects.
sdo	STIX 2.0 Domain Objects.
sro	STIX 2.0 Relationship Objects.

3.9.1 bundle

STIX 2.0 Bundle Representation.

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

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

Properties

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

get_obj(*obj_uuid*)

3.9.2 common

STIX 2.0 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

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **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*)

serialize(*pretty=False*, *include_optional_defaults=False*, ***kwargs*)

Serialize a STIX object.

Parameters

- **pretty** (*bool*) – If True, output properties following the STIX specs formatting. This includes indentation. Refer to notes for more details. (Default: `False`)
- **include_optional_defaults** (*bool*) – Determines whether to include optional properties set to the default value defined in the spec.
- ****kwargs** – The arguments for a json.dumps() call.

Examples

```
>>> import stix2
>>> identity = stix2.Identity(name='Example Corp.', identity_class=
↪'organization')
>>> print(identity.serialize(sort_keys=True))
{"created": "2018-06-08T19:03:54.066Z", ... "name": "Example Corp.", "type":
↪"identity"}
>>> print(identity.serialize(sort_keys=True, indent=4))
{
    "created": "2018-06-08T19:03:54.066Z",
    "id": "identity--d7f3e25a-ba1c-447a-ab71-6434b092b05e",
    "identity_class": "organization",
    "modified": "2018-06-08T19:03:54.066Z",
    "name": "Example Corp.",
    "type": "identity"
}
```

Returns *str* – The serialized JSON object.

Note: The argument `pretty=True` will output the STIX object following spec order. Using this argument greatly impacts object serialization performance. If your use case is centered across machine-to-machine operation it is recommended to set `pretty=False`.

When `pretty=True` the following key-value pairs will be added or overridden: indent=4, separators=(",", ": "), item_sort_key=sort_by.

class MarkingProperty (*required=False*, *fixed=None*, *default=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

```
>>> from stix2 import CustomMarking
>>> from stix2.properties import IntegerProperty, StringProperty
>>> @CustomMarking('x-custom-marking', [
...     ('property1', StringProperty(required=True)),
...     ('property2', IntegerProperty()),
... ])
... class MyNewMarkingObjectType():
...     pass
```

3.9.3 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 `_STIXBase20` 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

- **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

- **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

- **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

- **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

- **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

- **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 File(***kwargs*)

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

Properties

- **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

- **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

- **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

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

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

Properties

- **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

- **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 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

- **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_width** (*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

- **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

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

class UserAccount(***kwargs*)

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

Properties

- **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

- **key** (*String, required*)
- **values** (*List of Embedded Objects*)
- **modified** (*Timestamp*)
- **creator_user_ref** (*Object Reference*)
- **number_of_subkeys** (*Integer*)
- **extensions** (*Extensions*)

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

- **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-ext'*, *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

```
>>> from stix2.v20 import CustomObservable
>>> from stix2.properties import IntegerProperty, StringProperty
>>> @CustomObservable('x-custom-observable', [
...     ('property1', StringProperty(required=True)),
...     ('property2', IntegerProperty()),
... ])
```

(continues on next page)

(continued from previous page)

```
... class MyNewObservableType():
...     pass
```

3.9.4 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

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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

- **id** (*ID*)

- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **name** (*String*)
- **description** (*String*)
- **pattern** (*Pattern, required*)
- **valid_from** (*Timestamp, default: current date/time*)
- **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

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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(**args*, ***kwargs*)

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

Properties

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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 ThreatActor(*allow_custom=False*, ***kwargs*)

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

Properties

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)

- **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

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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

```
>>> from stix2.v20 import CustomObject
>>> from stix2.properties import IntegerProperty, StringProperty
>>> @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

```
>>> from stix2.v20 import CustomObject
>>> from stix2.properties import IntegerProperty, StringProperty
>>> @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.")
```

3.9.5 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

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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 Sighting(*sighting_of_ref=None*, ***kwargs*)

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

Properties

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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*)

3.10 v21

STIX 2.1 API Objects.

bundle	STIX 2.1 Bundle Representation.
common	STIX 2.1 Common Data Types and Properties.
observables	STIX 2.1 Cyber Observable Objects.
sdo	STIX 2.1 Domain Objects.
sro	STIX 2.1 Relationship Objects.

3.10.1 bundle

STIX 2.1 Bundle Representation.

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **id** (*ID*)
- **objects** (*List of STIX Objects*)

get_obj(*obj_uuid*)

3.10.2 common

STIX 2.1 Common Data Types and Properties.

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

For more detailed information on this object's properties, see the STIX 2.1 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 2.1 specification.

Properties

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

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

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

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **object_ref** (*Reference, required*)
- **object_modified** (*Timestamp*)
- **contents** (*Dictionary, required*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class MarkingDefinition(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)

- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **definition_type** (*String, required*)
- **name** (*String*)
- **definition** (*Marking, required*)

serialize(*pretty=False*, *include_optional_defaults=False*, ***kwargs*)

Serialize a STIX object.

Parameters

- **pretty** (*bool*) – If True, output properties following the STIX specs formatting. This includes indentation. Refer to notes for more details. (Default: `False`)
- **include_optional_defaults** (*bool*) – Determines whether to include optional properties set to the default value defined in the spec.
- ****kwargs** – The arguments for a json.dumps() call.

Examples

```
>>> import stix2
>>> identity = stix2.Identity(name='Example Corp.', identity_class=
↪'organization')
>>> print(identity.serialize(sort_keys=True))
{"created": "2018-06-08T19:03:54.066Z", ... "name": "Example Corp.", "type":
↪"identity"}
>>> print(identity.serialize(sort_keys=True, indent=4))
{
    "created": "2018-06-08T19:03:54.066Z",
    "id": "identity--d7f3e25a-ba1c-447a-ab71-6434b092b05e",
    "identity_class": "organization",
    "modified": "2018-06-08T19:03:54.066Z",
    "name": "Example Corp.",
    "type": "identity"
}
```

Returns *str* – The serialized JSON object.

Note: The argument `pretty=True` will output the STIX object following spec order. Using this argument greatly impacts object serialization performance. If your use case is centered across machine-to-machine operation it is recommended to set `pretty=False`.

When `pretty=True` the following key-value pairs will be added or overridden: indent=4, separators=(“,”, “: “), item_sort_key=sort_by.

---

class MarkingProperty(*required=False*, *fixed=None*, *default=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 2.1 specification.

Properties

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

class TLPMarking(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

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

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

Example

```
>>> from stix2.v21 import CustomMarking
>>> from stix2.properties import IntegerProperty, StringProperty
>>> @CustomMarking('x-custom-marking', [
...     ('property1', StringProperty(required=True)),
...     ('property2', IntegerProperty()),
... ])
... class MyNewMarkingObjectType():
...     pass
```

3.10.3 observables

STIX 2.1 Cyber Observable Objects.

Embedded observable object types, such as Email MIME Component, which is embedded in Email Message objects, inherit from `_STIXBase21` 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 2.1 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 2.1 specification.

Properties

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

class Artifact(***kwargs*)
For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **id** (*ID*)
- **mime_type** (*String*)
- **payload_bin** (*Binary*)
- **url** (*String*)
- **hashes** (*Hashes*)
- **encryption_algorithm** (*String*)
- **decryption_key** (*String*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

class AutonomousSystem(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **id** (*ID*)
- **number** (*Integer, required*)
- **name** (*String*)
- **rir** (*String*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

class Directory(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **id** (*ID*)
- **path** (*String, required*)
- **path_enc** (*String*)
- **ctime** (*Timestamp*)
- **mtime** (*Timestamp*)
- **atime** (*Timestamp*)
- **contains_refs** (*List of References*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)

- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

class DomainName(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **id** (*ID*)
- **value** (*String, required*)
- **resolves_to_refs** (*List of References*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

class EmailAddress(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **id** (*ID*)
- **value** (*String, required*)
- **display_name** (*String*)
- **belongs_to_ref** (*Reference*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

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

For more detailed information on this object's properties, see the STIX 2.1 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 2.1 specification.

Properties

- **id** (*ID*)
- **is_multipart** (*Boolean, required*)

- **date** (*Timestamp*)
- **content_type** (*String*)
- **from_ref** (*Reference*)
- **sender_ref** (*Reference*)
- **to_refs** (*List of References*)
- **cc_refs** (*List of References*)
- **bcc_refs** (*List of References*)
- **message_id** (*String*)
- **subject** (*String*)
- **received_lines** (*List of Strings*)
- **additional_header_fields** (*Dictionary*)
- **body** (*String*)
- **body_multipart** (*List of Embedded Objects*)
- **raw_email_ref** (*Reference*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

class File(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **id** (*ID*)
- **hashes** (*Hashes*)
- **size** (*Integer*)
- **name** (*String*)
- **name_enc** (*String*)
- **magic_number_hex** (*Hex*)
- **mime_type** (*String*)
- **ctime** (*Timestamp*)
- **mtime** (*Timestamp*)
- **atime** (*Timestamp*)
- **parent_directory_ref** (*Reference*)
- **contains_refs** (*List of References*)
- **content_ref** (*Reference*)
- **extensions** (*Extensions*)

- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

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

For more detailed information on this object's properties, see the STIX 2.1 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 2.1 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 2.1 specification.

Properties

- **id** (*ID*)
- **value** (*String, required*)
- **resolves_to_refs** (*List of References*)
- **belongs_to_refs** (*List of References*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

class IPv6Address(***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **id** (*ID*)
- **value** (*String, required*)
- **resolves_to_refs** (*List of References*)
- **belongs_to_refs** (*List of References*)

- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

class MACAddress (***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **id** (*ID*)
- **value** (*String, required*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

class Mutex (***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **id** (*ID*)
- **name** (*String, required*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

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

For more detailed information on this object's properties, see the STIX 2.1 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 2.1 specification.

Properties

- **id** (*ID*)
- **start** (*Timestamp*)
- **end** (*Timestamp*)
- **is_active** (*Boolean*)

- **src_ref** (*Reference*)
- **dst_ref** (*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** (*Reference*)
- **dst_payload_ref** (*Reference*)
- **encapsulates_refs** (*List of References*)
- **encapsulated_by_ref** (*Reference*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

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

For more detailed information on this object's properties, see the STIX 2.1 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 2.1 specification.

Properties

- **id** (*ID*)
- **is_hidden** (*Boolean*)
- **pid** (*Integer*)
- **created_time** (*Timestamp*)
- **cwd** (*String*)
- **command_line** (*String*)
- **environment_variables** (*Dictionary*)

- **opened_connection_refs** (*List of References*)
- **creator_user_ref** (*Reference*)
- **image_ref** (*Reference*)
- **parent_ref** (*Reference*)
- **child_refs** (*List of References*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **image_height** (*Integer*)
- **image_width** (*Integer*)
- **bits_per_pixel** (*Integer*)
- **exif_tags** (*Dictionary*)

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

For more detailed information on this object's properties, see the STIX 2.1 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 2.1 specification.

Properties

- **id** (*ID*)
- **name** (*String, required*)
- **cpe** (*String*)
- **swid** (*String*)
- **languages** (*List of Strings*)
- **vendor** (*String*)

- **version** (*String*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

class TCPExt(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.1 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 2.1 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 2.1 specification.

Properties

- **id** (*ID*)
- **value** (*String, required*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

class UserAccount(***kwargs*)
For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **id** (*ID*)
- **user_id** (*String*)
- **credential** (*String*)
- **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*)
- **credential_last_changed** (*Timestamp*)
- **account_first_login** (*Timestamp*)
- **account_last_login** (*Timestamp*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

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

For more detailed information on this object's properties, see the STIX 2.1 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 2.1 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 2.1 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 2.1 specification.

Properties

- **aslr_enabled** (*Boolean*)

- **dep_enabled** (*Boolean*)
- **priority** (*String*)
- **owner_sid** (*String*)
- **window_title** (*String*)
- **startup_info** (*Dictionary*)
- **integrity_level** (*Enum*)

class WindowsRegistryKey (***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **id** (*ID*)
- **key** (*String*)
- **values** (*List of Embedded Objects*)
- **modified_time** (*Timestamp*)
- **creator_user_ref** (*Reference*)
- **number_of_subkeys** (*Integer*)
- **extensions** (*Extensions*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **name** (*String*)
- **data** (*String*)
- **data_type** (*Enum*)

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **service_name** (*String*)
- **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 2.1 specification.

Properties

- **id** (*ID*)
- **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*)
- **spec_version** (*String*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **defanged** (*Boolean*)

class X509V3ExtenstionsType(*allow_custom=False*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 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-ext'*, *properties=None*)
Decorator for custom extensions to STIX Cyber Observables.

CustomObservable(*type='x-custom-observable'*, *properties=None*, *id_contrib_props=None*)
Custom STIX Cyber Observable Object type decorator.

Example

```
>>> from stix2.v21 import CustomObservable
>>> from stix2.properties import IntegerProperty, StringProperty
>>> @CustomObservable('x-custom-observable', [
...     ('property1', StringProperty(required=True)),
...     ('property2', IntegerProperty()),
... ])
... class MyNewObservableType():
...     pass
```

3.10.4 sdo

STIX 2.1 Domain Objects.

class AttackPattern(*allow_custom=False*, ***kwargs*)
For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **name** (*String, required*)
- **description** (*String*)
- **aliases** (*List of Strings*)
- **kill_chain_phases** (*List of Kill Chain Phases*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **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 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **name** (*String, required*)
- **description** (*String*)
- **aliases** (*List of Strings*)
- **first_seen** (*Timestamp*)
- **last_seen** (*Timestamp*)
- **objective** (*String*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **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 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **name** (*String, required*)
- **description** (*String*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)

- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **created_by_ref** (*Reference*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **name** (*String*)
- **description** (*String*)
- **context** (*String, required*)
- **object_refs** (*List of References, required*)

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **name** (*String, required*)
- **description** (*String*)
- **roles** (*List of Strings*)
- **identity_class** (*String*)
- **sectors** (*List of Strings*)
- **contact_information** (*String*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)

- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class Indicator(**args*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **name** (*String*)
- **description** (*String*)
- **indicator_types** (*List of Strings*)
- **pattern** (*Pattern, required*)
- **pattern_type** (*String, required*)
- **pattern_version** (*String*)
- **valid_from** (*Timestamp, required, default: current date/time*)
- **valid_until** (*Timestamp*)
- **kill_chain_phases** (*List of Kill Chain Phases*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)

- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **name** (*String, required*)
- **description** (*String*)
- **infrastructure_types** (*List of Strings*)
- **aliases** (*List of Strings*)
- **kill_chain_phases** (*List of Kill Chain Phases*)
- **first_seen** (*Timestamp*)
- **last_seen** (*Timestamp*)

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **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*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)

- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **name** (*String*)
- **description** (*String*)
- **latitude** (*Float*)
- **longitude** (*Float*)
- **precision** (*Float*)
- **region** (*String*)
- **country** (*String*)
- **administrative_area** (*String*)
- **city** (*String*)
- **street_address** (*String*)
- **postal_code** (*String*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

to_maps_url(*map_engine='Google Maps'*)

Return URL to this location in an online map engine.

Google Maps is the default, but Bing maps are also supported.

Parameters **map_engine** (*str*) – Which map engine to find the location in

Returns The URL of the location in the given map engine.

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)

- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **name** (*String*)
- **description** (*String*)
- **malware_types** (*List of Strings*)
- **is_family** (*Boolean, required*)
- **aliases** (*List of Strings*)
- **kill_chain_phases** (*List of Kill Chain Phases*)
- **first_seen** (*Timestamp*)
- **last_seen** (*Timestamp*)
- **operating_system_refs** (*List of References*)
- **architecture_execution_envs** (*List of Strings*)
- **implementation_languages** (*List of Strings*)
- **capabilities** (*List of Strings*)
- **sample_refs** (*List of References*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **created_by_ref** (*Reference*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)

- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)
- **product** (*String, required*)
- **version** (*String*)
- **host_vm_ref** (*Reference*)
- **operating_system_ref** (*Reference*)
- **installed_software_refs** (*List of References*)
- **configuration_version** (*String*)
- **modules** (*List of Strings*)
- **analysis_engine_version** (*String*)
- **analysis_definition_version** (*String*)
- **submitted** (*Timestamp*)
- **analysis_started** (*Timestamp*)
- **analysis_ended** (*Timestamp*)
- **result_name** (*String*)
- **result** (*String*)
- **analysis_sco_refs** (*List of References*)
- **sample_ref** (*Reference*)

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **abstract** (*String*)
- **content** (*String, required*)
- **authors** (*List of Strings*)
- **object_refs** (*List of References, required*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)

- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class ObservedData(**args*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **first_observed** (*Timestamp, required*)
- **last_observed** (*Timestamp, required*)
- **number_observed** (*Integer, required*)
- **objects** (*Observable*)
- **object_refs** (*List of References*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **explanation** (*String*)
- **authors** (*List of Strings*)
- **opinion** (*Enum, required*)
- **object_refs** (*List of References, required*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)

- **lang** (*String*)
- **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 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **name** (*String, required*)
- **description** (*String*)
- **report_types** (*List of Strings*)
- **published** (*Timestamp, required*)
- **object_refs** (*List of References, required*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

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

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **name** (*String, required*)
- **description** (*String*)
- **threat_actor_types** (*List of Strings*)
- **aliases** (*List of Strings*)
- **first_seen** (*Timestamp*)

- **last_seen** (*Timestamp*)
- **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*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **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 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **name** (*String, required*)
- **description** (*String*)
- **tool_types** (*List of Strings*)
- **aliases** (*List of Strings*)
- **kill_chain_phases** (*List of Kill Chain Phases*)
- **tool_version** (*String*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **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 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **name** (*String, required*)
- **description** (*String*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **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

```
>>> from stix2.v21 import CustomObject
>>> from stix2.properties import IntegerProperty, StringProperty
>>> @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

```
>>> from stix2.v21 import CustomObject
>>> from stix2.properties import IntegerProperty, StringProperty
>>> @CustomObject('x-type-name', [
...     ('property1', StringProperty(required=True)),
...     ('property2', IntegerProperty()),
... ])
... class MyNewObjectType():
...     def __init__(self, property2=None, **kwargs):
```

(continues on next page)

(continued from previous page)

```
...         if property2 and property2 < 10:
...             raise ValueError("'property2' is too small.")
```

3.10.5 sro

STIX 2.1 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 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **relationship_type** (*String, required*)
- **description** (*String*)
- **source_ref** (*Reference, required*)
- **target_ref** (*Reference, required*)
- **start_time** (*Timestamp*)
- **stop_time** (*Timestamp*)
- **revoked** (*Boolean*)
- **labels** (*List of Strings*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

class Sighting(*sighting_of_ref=None*, ***kwargs*)

For more detailed information on this object's properties, see the STIX 2.1 specification.

Properties

- **spec_version** (*String*)
- **id** (*ID*)
- **created_by_ref** (*Reference*)
- **created** (*Timestamp, default: current date/time*)
- **modified** (*Timestamp, default: current date/time*)
- **description** (*String*)
- **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*)
- **confidence** (*Integer*)
- **lang** (*String*)
- **external_references** (*List of External References*)
- **object_marking_refs** (*List of References*)
- **granular_markings** (*List of Granular Markings*)

3.11 workbench

Functions and class wrappers for interacting with STIX2 data at a high level.

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

set_default_creator(**args*, ***kwargs*)
Set default value for the *created_by_ref* property.

set_default_created(**args*, ***kwargs*)
Set default value for the *created* property.

set_default_external_refs(**args*, ***kwargs*)
Set default external references.

set_default_object_marking_refs(**args*, ***kwargs*)
Set default object markings.

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 *list* – All versions of the specified STIX object.

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 *list* – The STIX objects matching the query.

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

str –

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* – The 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.
- **filters** (*list*) – list of additional filters the related objects must match.

Returns *list* – The STIX objects related to the given STIX object.

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

Defines custom behavior before storing STIX objects using the appropriate method call on the associated DataSink.

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

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

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

parse(**args*, ***kwargs*)
Convert a string, dict 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 as well unknown custom objects. Note that unknown custom objects cannot be parsed into STIX objects, and will be returned as is. Default: False.
- **version** (*str*) – If present, it forces the parser to use the version provided. Otherwise, the library will make the best effort based on checking the "spec_version" property. If none of the above are possible, it will use the default version specified by the library.

Returns An instantiated Python STIX object.

Warning: 'allow_custom=True' will allow for the return of any supplied STIX dict(s) that cannot be found to map to any known STIX object types (both STIX2 domain objects or defined custom STIX2 objects); NO validation is done. This is done to allow the processing of possibly unknown custom STIX objects (example scenario: I need to query a third-party TAXII endpoint that could provide custom STIX objects that I don't know about ahead of time)

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

attack_patterns(*filters=None*)
Retrieve all Attack Pattern objects.

Parameters **filters** (*list, optional*) – A list of additional filters to apply to the query.

campaigns (*filters=None*)
Retrieve all Campaign objects.

Parameters **filters** (*list, optional*) – A list of additional filters to apply to the query.

courses_of_action (*filters=None*)
Retrieve all Course of Action objects.

Parameters **filters** (*list, optional*) – A list of additional filters to apply to the query.

identities (*filters=None*)
Retrieve all Identity objects.

Parameters **filters** (*list, optional*) – A list of additional filters to apply to the query.

indicators (*filters=None*)
Retrieve all Indicator objects.

Parameters **filters** (*list, optional*) – A list of additional filters to apply to the query.

intrusion_sets (*filters=None*)
Retrieve all Intrusion Set objects.

Parameters **filters** (*list, optional*) – A list of additional filters to apply to the query.

malware (*filters=None*)
Retrieve all Malware objects.

Parameters **filters** (*list, optional*) – A list of additional filters to apply to the query.

observed_data (*filters=None*)
Retrieve all Observed Data objects.

Parameters **filters** (*list, optional*) – A list of additional filters to apply to the query.

reports (*filters=None*)
Retrieve all Report objects.

Parameters **filters** (*list, optional*) – A list of additional filters to apply to the query.

threat_actors (*filters=None*)
Retrieve all Threat Actor objects.

Parameters **filters** (*list, optional*) – A list of additional filters to apply to the query.

tools (*filters=None*)
Retrieve all Tool objects.

Parameters **filters** (*list, optional*) – A list of additional filters to apply to the query.

vulnerabilities (*filters=None*)
Retrieve all Vulnerability objects.

Parameters **filters** (*list, optional*) – A list of additional filters to apply to the query.

CHAPTER 4

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!

4.1 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.

4.2 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).

4.3 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.

4.4 Adding a dependency

One of the pre-commit hooks we use in our develoment environment enforces a consistent ordering to imports. If you need to add a new library as a dependency please add it to the *known_third_party* section of *.isort.cfg* to make sure the import is sorted correctly.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

Symbols

A

B

C

D

E

T

U

V

W

X

Z