libtaxii Documentation

Release 1.1.115

The MITRE Corporation

Nov 12, 2019

Contents

Version: 1.1.115

libtaxii is a Python library that provides:

1. An object representation of TAXII Messages
2. Serialization/deserialization to and from XML and Python dictionaries
3. An HTTP/HTTPS TAXII Client

Version: 1.1.115

CHAPTER 1

Installation

1.1 Recommended Installation

Use pip:

```
$ pip install libtaxii
```

You might also want to consider using a virtualenv.

1.2 Dependencies

The libtaxii library is developed on Python 2.7 and tested against both Python 2.6 and 2.7. Besides the Python Standard Library, libtaxii relies on the following Python libraries:

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

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

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

Note: In libtaxii 1.0.101 and earlier, the M2Crypto library was also required. This dependency was removed as of libtaxii 1.0.102.

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

1.3 Manual Installation

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

1. Download and install the *dependencies* above. Although setuptools will generally install dependencies automatically, installing the dependencies manually beforehand helps distinguish errors in dependency installation from errors in libtaxii installation. Make sure you check to ensure the versions you install are compatible with the version of libtaxii you plan to install.
2. Download the desired version of libtaxii from PyPI or the GitHub releases page. The steps below assume you are using the 1.1.115 release.
3. Extract the downloaded file. This will leave you with a directory named libtaxii-1.1.115.

```
$ tar -zxf libtaxii-1.1.115.tar.gz
$ ls
libtaxii-1.1.115 libtaxii-1.1.115.tar.gz
```

OR

```
$ unzip libtaxii-1.1.115.zip
$ ls
libtaxii-1.1.115 libtaxii-1.1.115.zip
```

4. Run the installation script.

```
$ cd libtaxii-1.1.115
$ python setup.py install
```

5. Test the installation.

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

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

1.4 Further Information

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

Version: 1.1.115

CHAPTER 2

Getting Started

This page gives an introduction to **libtaxii** and how to use it. Please note that this page is being actively worked on and feedback is welcome.

2.1 Modules

The libtaxii library contains the following modules:

- **libtaxii** - Contains version info and some methods for getting TAXII Messages from HTTP responses. (Implemented in `libtaxii/__init__.py`)
- **libtaxii.clients.** - TAXII HTTP and HTTPS clients. (Implemented in `libtaxii/clients.py`)
- **libtaxii.common** - Contains functions and classes useful for all versions of TAXII
- **libtaxii.constants** - Contains constants for TAXII
- **libtaxii.messages_10** - Creating, handling, and parsing TAXII 1.0 messages. (Implemented in `libtaxii/messages_10.py`)
- **libtaxii.messages_11** - Creating, handling, and parsing TAXII 1.1 messages. (Implemented in `libtaxii/messages_11.py`)
- **libtaxii.taxii_default_query** - Creating, handling and parsing TAXII Default Queries. (Implemented in `libtaxii/taxii_default_query.py`) *New in libtaxii 1.1.100*.
- **libtaxii.validation** - Common data validation functions used across libtaxii. (Implemented in `libtaxii/validation.py`)

2.2 TAXII Messages Module Structure

In the TAXII message modules (*libtaxii.messages_10* and *libtaxii.messages_11*), there is a class corresponding to each type of TAXII message. For example, there is a `DiscoveryRequest` class for the Discovery Request message:

```
import libtaxii.messages_11 as tm11
discovery_request = tm11.DiscoveryRequest( ... )
```

For types that can been used across multiple messages (e.g., a Content Block can exist in both Poll Response and Inbox Message), the corresponding class (`ContentBlock`) is (and always has always been) defined at the module level.

```
content_block = tm11.ContentBlock( ... )
```

Other types that are used exclusively within a particular TAXII message type were previously defined as nested classes on the corresponding message class; however, they are now defined at the top level of the module. For example, a Service Instance is only used in a Discovery Response message, so the class representing a Service Instance, now just `ServiceInstance`, was previously `DiscoveryResponse.ServiceInstance`. The latter name still works for backward compatibilty reasons, but is deprecated and may be removed in the future.

```
service_instance = tm11.ServiceInstance( ... )
service_instance = tm11.DiscoveryRequest.ServiceInstance( ... )
```

See the *API Documentation* for proper constructor arguments for each type above.

2.3 TAXII Message Serialization and Deserialization

Each class in the message modules has serialization and deserialization methods for XML Strings, Python dictionaries, and LXML ElementTrees. All serialization methods (`to_*()`) are instance methods called on specific objects (e.g., `discovery_request.to_xml()`). Deserialization methods (`from_*()`) are class methods and should be called on the class itself (e.g., `tm11.DiscoveryRequest.from_xml(xml_string)`).

Each class in messages.py defines the following:

- `from_xml(xml_string)` - Creates an instance of the class from an XML String.
- `to_xml()` - Creates the XML representation of an instance of a class.
- `from_dict(dictionary)` - Creates an instance of the class from a Python dictionary.
- `to_dict()` - Creates the Python dictionary representation of an instance of a class.
- `from_etree(lxml_etree)` - Creates an instance of the class from an LXML Etree.
- `to_etree()` - Creates the LXML Etree representation of an instance of a class.

To create a TAXII Message from XML:

```
xml_string = '<taxii:Discovery_Response ... />'  # Note: Invalid XML
discovery_response = tm11.DiscoveryResponse.from_xml(xml_string)
```

To create an XML string from a TAXII Message:

```
new_xml_string = discovery_response.to_xml()
```

The same approach can be used for Python dictionaries:

```
msg_dict = { ... }  # Note: Invalid dictionary syntax
discovery_response = tm11.DiscoveryResponse.from_dict(msg_dict)
new_dict = discovery_response.to_dict()
```

and for LXML ElementTrees:

```
msg_etree = etree.Element( ... )  # Note: Invalid Element constructor
discovery_response = tm11.DiscoveryResponse.from_etree(msg_etree)
new_etree = discovery_response.to_etree()
```

2.4 Schema Validating TAXII Messages

You can use libtaxii to Schema Validate XML, etree, and file representations of TAXII Messages. XML Schema validation cannot be performed on a TAXII Message Python object, since XML Schema validation can only be performed on XML.

A full code example of XML Schema validation can be found in *API Documentation*

2.5 TAXII Clients

The **libtaxii.clients** module defines a single class `HttpClient` capable of invoking TAXII services over both HTTP and HTTPS. The client is a fairly straighforward wrapper around Python's builtin `httplib` and supports the use of of both HTTP Basic and TLS Certificate authentication.

Example usage of clients:

```
import libtaxii as t
import libtaxii.clients as tc
import libtaxii.messages_11 as tm11
from libtaxii.constants import *

client = tc.HttpClient()
client.set_auth_type(tc.HttpClient.AUTH_BASIC)
client.set_use_https(True)
client.set_auth_credentials({'username': 'MyUsername', 'password': 'MyPassword'})

discovery_request = tm11.DiscoveryRequest(tm11.generate_message_id())
discovery_xml = discovery_request.to_xml()

http_resp = client.call_taxii_service2('example.com', '/pollservice/', VID_TAXII_XML_
↪11, discovery_xml)
taxii_message = t.get_message_from_http_response(http_resp, discovery_request.message_
↪id)
print taxii_message.to_xml()
```

Version: 1.1.115

CHAPTER 3

Scripts

This page provides documentation on the scripts that are included with libtaxii. All clients are configured to use the Hail A TAXII server (http://hailataxii.com/) by default; provide command line options for specifying most aspects of the script (e.g., host, port, client certs, username/password, HTTP or HTTPS, etc); and support TAXII 1.1 unless otherwise noted.

Note that the scripts *should* be callable from anywhere on the command line as long as you have the python scripts directory on your path.

3.1 Script Listing

- **discovery_client** - Issues a Discovery Request to a Discovery Service
- **fulfillment_client** - Issues a Poll Fulfillment Request to a Poll Service and writes the resulting content to file
- **inbox_client** - Issues an Inbox Message with one Content Block to an Inbox Service
- **poll_client** - Issues a Poll Request to a Poll Service and writes the resulting content to file
- **poll_client_10** - Issues a Poll Request to a TAXII 1.0 Poll Service and writes the resulting content to file
- **query_client** - Issues a Query for an IP Address or Hash to a Poll Service and writes the resulting content to file

3.2 Common Command Line Arguments

All scripts use these command line arguments:

- `-h, --help` - Shows help text
- `-u, --url` - Specified the URL to connect to.
- `--cert` - Specifies the file location of the certificate to use for authentication. If provided, `--key` must also be provided.
- `--key` - Specifies the file location of the key to use for authentication.

- `--username` - Specifies the username to use for authentication. If provided, `--pass` must also be provided.
- `--pass` - Specifies the password to use for authentication.
- `--proxy` - Specifies proxy settings (e.g. `http://proxy.example.com:80/`, or `noproxy` to not use any proxy). If omitted, the system's proxy settings will be used.
- `--xml-output` - Specifies that the XML messages should be printed instead of the default textual representation

Note: As of libtaxii 1.1.106, the following arguments are now deprecated in favor of `--url`

- `--host` - Specifies the host to connect to (e.g., `hailataxii.com`)
- `--port` - Specifies the port to connect on (e.g., `80`)
- `--path` - Specifies the path portion of the URL to connect to (e.g., `/services/discovery`)
- `--https` - Specifies whether to use HTTPS or not (e.g., True or False)

For example, to call the discovery_client using all these arguments, you would do: `discovery_client --url http://hailataxii.com/taxii-discovery-service --cert MyCert.crt --key MyKey.key --username foo --pass bar --proxy http://myproxy.example.com:80 --xml-output`

3.3 Additional Discovery Client Command Line Arguments

The Discovery Client does not use any additional command line arguments.

3.4 Additional Poll Fulfillment Client Command Line Arguments

In addition to the command line arguments listed above, the Poll Fulfillment Client uses these:

- `--collection` - The collection being requested
- `--result_id` - The result id being requested (required)
- `--result_part_number` - The result part number being requested (defaults to 1)

Example: `fulfillment_client --collection MyCollection --result_id someId --result_part_number 1`

3.5 Additional Inbox Client Command Line Arguments

In addition to the command line arguments listed above, the Inbox Client uses these:

- `--content-binding` - The Content Binding ID to use for the Content Block (Defaults to STIX XML 1.1)
- `--subtype` - The Content Binding ID subtype to use for the Content Block (Optional; Defaults to None)
- `--content-file` - The file location (e.g., /tmp/mydata) containing the data to send in the Content Block. Defaults to a built-in STIX 1.1 XML document.
- `--dcn` - The Destination Collection Name that is specified in the Inbox Message, requesting that the recipient make the sent content available on the specified Destination Collection Name. TAXII supports multiple DCNs, but this script only supports one.

Example: `inbox_client --content-binding urn:stix.mitre.org:xml:1.1 --content-file stix_file.xml`

3.6 Additional Poll Client Command Line Arguments

In addition to the command line arguments listed above, the Poll Client uses these:

- `--collection` - The Collection Name to Poll. Defaults to 'default'
- `--begin_timestamp` - The Begin Timestamp Label to used bound the Poll Request. Defaults to None.
- `--end_timestamp` - The End Timestamp Label to used bound the Poll Request. Defaults to None.
- `--subscription-id` - The Subscription ID for this Poll Request
- `--dest-dir` - The directory to save Content Blocks to. Defaults to the current directory.

Example: `poll_client --collection MyCollection`

3.7 Additional Poll Client 1.0 Command Line Arguments

In addition to the command line arguments listed above, the Poll Client 1.0 uses these:

- `--feed` - The Data Feed to Poll. Defaults to 'default'
- `--begin_timestamp` - The Begin Timestamp Label to used bound the Poll Request. Defaults to None.
- `--end_timestamp` - The End Timestamp Label to used bound the Poll Request. Defaults to None.
- `--subscription-id` - The Subscription ID to use when polling
- `--dest-dir` - The directory to save Content Blocks to. Defaults to the current directory.

Example: `poll_client_10 --feed MyFeedName --subscription-id SomeSubscriptionId`

3.8 Additional Query Client Command Line Arguments

In addition to the command line arguments listed above, the Query Client uses these:

- `--collection` - The collection to Poll (recall that a query is part of a Poll Request). Defaults to 'default_queryable'.
- `--allow_asynch` - Whether asynchronous Polling is supported. Defaults to True (Use the Poll Fulfillment client to request asynchronous results!)
- `--ip` - The IP to query on. One of –ip or –hash must be specified.
- `--hash` - The file hash to query on. One of –ip or –hash must be specified.
- `--dest-dir` - The directory to save Content Blocks to. Defaults to the current directory.

Example: `query_client --collection MyQueryCollection --ip 10.0.0.0`

Version: 1.1.115

CHAPTER 4

Release Notes

4.1 1.1.115 (2019-11-12)

(diff)

- #239 Convert the HTTP response body to a string type (PY3 this will be bytes) (@sddj)

4.2 1.1.114 (2019-07-26)

(diff)

- #237 Support converting dicts to content bindings (@danielsamuels)
- #238 Provide XMLParser copies instead of reusing the cached instance. Prevents future messages to lose namespace

4.3 1.1.113 (2019-04-11)

(diff)

- #234 Add ability to load a configuration file when executing a script
- #232 Fix TLS handshake failure when a server requires SNI (@marcelslotema)

4.4 1.1.112 (2018-11-27)

(diff)

- #227 Fixes to poll_client script (Python3 compatibility)
- #226 Clean-up documentation warnings

- #228 Fix 'HTTPMessage' has no attribute 'getheader' (Python3 compatibility)
- #225 Fix checks that involve xpath (lxml) to prevent FutureWarning message
- #230 Fix parsing status message round-trip (@danielsamuels)

4.5 1.1.111 (2017-06-07)

(diff)

- Fix #222, #224 - Update clients.py to work with Python 2.6, 3.3, 3.5, and 3.6.
- Fix #221 - Add Python 3.6 support.
- Fix #219 - Handle Unicode- and byte-strings consistently.
- Fix #214 - Add timeout parameter to call_taxii_service2 (@mbekavac)
- Fix #192 - Add support for STIX 1.2.
- Add user_agent parameter to call_taxii_service2 (@kralka)

4.6 1.1.110 (2016-09-08)

(diff)

- Fix #210 - Use hailataxii.com in examples instead of taxiitest.mitre.org (@clenk)
- Fix #183 - Update incorrect comment (@daybarr)
- Fix SMIME Content Binding ID typo (@brlogan)

4.7 1.1.109 (2015-11-16)

(diff)

- Fix #203 - Fix SSL context on older Python 2.7 versions (@traut)

4.8 1.1.108 (2015-10-29)

(diff)

- Support password-protected SSL keys (@traut)
- Fix #200 - Bad encodings no longer generate Exceptions (@MarkDavidson)

4.9 1.1.107 (2015-08-05)

(diff)

- Fix #184 - Use proxy for both HTTP and HTTPS (@nadavc)
- Fix #187 - Handle numeric values in taxii_default_query (@stkyle)

- Update Example Query documentation (@stkyle)
- Fix #189 - Update how constants are used and referenced (@stkyle)
- Show HTTP error code in StatusMessage.message (@ahippo)
- Python 3 compatibility (@rjprins)

4.10 1.1.106

(diff)

- Thank you to the multiple contributors for this release: @traut, @gtback, @wbolster, and @MarkDavidson, and thank you to those who filed issues that were fixed in this release.
- Timestamp labels can now be provided as a string or as a python datetime object. Previously, only datetime objects were permitted.
- Some big changes to TAXII Client command line args. Deprecated URL components (e.g., –host, –port) in favor of specifying a single url (–url)
- Added a TAXII Inbox 1.0 client
- Decreased the likelihood of future message ID collisions
- A variety of improvements in the following areas: data validation, typos, documentation, DRY-ness, overall repo quality (thanks @gtback)
- Multiple code cleanup changes (Thanks in part to @traut of IntelWorks)

4.11 1.1.105

(diff)

- Fixed multiple XML parsing related vulnerabilities (Thanks @guidovranken of IntelWorks for the vulnerability research!)

4.12 1.1.104

(diff)

- Fixed a bug where libtaxii did not properly handle XML values for Extended Headers
- Added checking for required status detail keys in Status Messages
- Improved data validation in various places, fixed various bugs, and improved documentation
- Improved filename generation in scripts (Thanks @guidovranken!)

4.13 1.1.103

(diff)

In terms of code organization, there are a few big changes beginning to take place in this version of libtaxii. Constants and commonly used classes/functions are being moved to common locations (libtaxii.constants and libtaxii.common,

respectively). Also, nested classes (e.g., messages_11.DiscoveryResponse.ServiceInstance) have been de-nested (e.g., is now messages_11.ServiceInstance). All of these changes are intended to make using libtaxii easier. For the time being, backward compatibility has been maintained, but many of these changes may result in a backward compatibility breaking change in a future, major release of libtaxii.

Major changes:

- libtaxii.constants, a new source file, was created. The definition for all constants in libtaxii have been moved to libtaxii.constants. Aliases to the previous definition locations have been retained for backward compatibility and may be removed in a future major release.
- libtaxii.common, a new source file for containing classes and methods common to TAXII, was created. Some common classes and functions have been moved into libtaxii.common, and more will be moved over time. Aliases to the previous classes and functions have been retained for backward compatibility and may be removed in a future major release. (Thanks, @gtback!)
- Promoted nested classes to module-level classes in messages_10, messages_11, and taxii_default_query. Aliases to the previous, nested, classes have been retained for backward compatibility and may be removed in a future major release. (Thanks, @gtback!)
- A 'to_text()' method has been added to all TAXII Message classes. 'to_text()' provides a "nicely formatted" human readable representation of a TAXII Message and its components. The 'to_text()' method was added to support libtaxii's scripts. There is no 'from_text()' method as this is not intended to be a serialization/deserialization feature, but a readability feature.
- Lowered the required version of lxml to 2.2.3, the latest available on RHEL 6. (Thanks to @mblayman for raising this)
- Lowered the required version of python-dateutil to 1.4.1, the latest available on RHEL 6. (Thanks to @mblayman for raising this)
- TAXII 1.1 StatusMessages now raise a ValueError when required a Status Detail is not set.
- TAXII XML Validation has a new methodology: See validation.SchemaValidator (http://libtaxii.readthedocs.org/en/latest/api/validation.html#libtaxii.validation.SchemaValidator)
- Related: validate_xml(...) has been deprecated and may be removed in a future major release.

Minor changes:

- Tons of PEP8 improvements (Thanks, @gtback!)
- TAXII Scripts have been entirely reorganized to be more DRY.
- Added two constants for Proxy Settings (SYSTEM_PROXY and NO_PROXY). These supersede the need to use either None or 'noproxy', which were not as clear to developers.
- Improved documentation, Tox usage, and Travis-CI usage. (Thanks, @gtback!)
- SMIME Content Binding added (application/x-pks7-mime)
- For Python 2.6, argparse is now a requirement
- Added constants for TAXII Default Query Parameters and Relationships

Bug fixes:

- In messages_11.PollResponse, the result_part_number parameter is now set by the constructor.

4.14 1.1.102

(diff)

The biggest change was the addition of scripts to libtaxii. Now when you install libtaxii, you get a number of scripts that are by default configured to hit the TAXII Test server (taxiitest.mitre.org). You can specify a number of parameters on the command line to change where/how the scripts connect. The scripts are:

- discovery_client - Calls a TAXII 1.1 Discovery Service
- fulfillment_client - Calls a TAXII 1.1 Poll Service for Poll Fulfillment
- inbox_client - Calls a TAXII 1.1 Inbox Service. Comes packaged with a STIX document to use by default.
- poll_client - Calls a TAXII 1.1 Poll Service
- poll_client_10 - Calls a TAXII 1.0 Poll Service (Note: Does not work with taxiitest.mitre.org, as taxiitest.mitre.org is TAXII 1.1 only)
- query_client - Calls a TAXII 1.1 Poll Service with a query on IP or File Hash (Note: As of 6/11/2014; Works with the master branch of YETI, and will work with YETI after the next release of YETI)

We also had a number of bug fixes and impprovements for this version of libtaxii:

- Unicode strings work round trip (Hat tip to Ben Yates for reporting the issue)
- Added TONS of documentation (http://libtaxii.readthedocs.org/en/latest/index.html). Big thanks to @gtback and @slsnow!
- Fixed some issues in ContentBlock.content where certain data was not serializing/deserializing properly
- Streamlined serialization of XML documents to avoid a double-parse in certain cases
- Added a Content Binding ID for STIX XML 1.1.1
- Added an optional pretty_print argument to all to_xml() functions. e.g., to_xml(pretty_print=True)
- Added the three TAXII Default Query Status Type to libtaxii.taxii_default_query
- Fixed a bug where custom Status Types were prohibited
- Added Travis CI

4.15 1.1.101

(diff)

Lots of changes in this release, including some important bug fixes.

- The equals method for all TAXII Messages was fixed (previous it would incorrectly return True in many cases).
- Fixed various serialization/deserialization issues uncovered by the now correctly implemented equals methods.
- Added a defined Content-Type for TAXII XML 1.1.
- Corrected the value of ST_UNSUPPORTED_PROTOCOL.
- Fixed a bug when parsing non-TAXII responses.
- Fixed a bug where the Subscription ID was not allowed to be none in ManageFeedSubscriptionRequest (The Subscription ID must be None for subscription requests with an action of SUBSCRIBE).
- Fixed a bug where DeliveryParameters were not permitted to be None in a ManageFeedSubscriptionRequest.
- Added code to permit the setting of certain HTTP Headers (Accept, X-TAXII-Accept).
- Improved libtaxii's handling of non-XML content that looks like XML
- Added Constants for TAXII Headers (and updated the code to use them).

- Improved handling of non-registered Query formats (now an exception is raised; previously None was returned).
- libtaxii now provides an X-TAXII-Services header.

4.16 1.1.100

(diff)

This version contains known bugs. Use a more recent version of libtaxii when possible.

- First release that supports TAXII 1.1.
- No changes to TAXII 1.0 code.
- Added documentation for Messages 1.1 API and TAXII Default Query.

4.17 1.0.107

(diff)

- Fixed an issue that was causing invalid TAXII XML to be generated (Thanks [@JamesNK](https://github.com/JamesNK)).
- Fixed an issue in the messages test suite that caused the invalid XML to not be caught.

4.18 1.0.106

(diff)

- Added validation to messages.py. This should not cause any backwards compatibility issues, but there may be things we didn't catch. Please report any instances of this via the issue tracker.
- Modified the internals of `from_dict()` and `from_xml()` in many cases to support how validation now works.
- Added constructor arguments to HttpClient. Default behavior is still the same.
- Added the ability to specify whether or not an HTTP Server's SSL Certificate should be verified.
- Prettified some of the documentation.
- Added documentation in certain places where there was none previously.

4.19 1.0.105

(diff)

- Added support for JSON (Thanks to [@ics](https://github.com/ics), Alex Ciobanu of CERT EU).
- callTaxiiService2 now supports user-specified content_types (Thanks to Alex Ciobanu of CERT EU).
- Fixed Issue #18, libtaxii.messages now permits users to specify any lxml parser for parsing XML. A default parser is used when one is not specified, which is unchanged from previous usage.

4.20 1.0.104

(diff)

- Many of the comments were aligned with PEP8 guidelines (thanks [@gtback](https://github.com/gtback)!)
- Added a new authentication mechanism (AUTH_CERT_BASIC) to clients.py. This authentication mechanism supports Certificate Authentication plus HTTP Basic authentication.
- Added clients.HttpClient.callTaxiiService2, which supersedes callTaxiiService. The previous version of callTaxiiService couldn't handle proxies well, which now have better support.
- Added better proxy support to client.HttpClient via the setProxy() function.

4.21 1.0.103

(diff)

This version fixes a schema validation bug. Schema validation did not work prior to this version.

4.22 1.0.102

(diff)

This version adds better proxy support to libtaxii in libtaxii.clients. A function to set a proxy (setProxy) was added as well as a new callTaxiiService2 function that can properly use proxies. The original callTaxiiService function did not support proxies well. The APIs have the full documentation for callTaxiiService, callTaxiiService2, and setProxy (Client API).

4.23 1.0.101

(diff)

This version added missing source files for distribution on PyPI. No functionality changes were made.

4.24 1.0.100

(diff)

Version 1.0.100 represents the first TAXII 1.0 compliant version of libtaxii. This version removes all code not compliant with TAXII 1.0.

4.25 1.0.090

(diff)

This version of libtaxii has components that are TAXII 1.0 conformant and experimental functionality that conforms to a draft version of TAXII. This version should only be used to transition from 1.0.000draft to 1.0.100.

4.26 1.0.000draft

This version of libtaxii represents experimental functionality that conforms to a draft version of TAXII. This code should no longer be used. For those using this code, you should upgrade to 1.0.090 and migrate your code to use the TAXII 1.0 components, then transition to 1.0.100.

CHAPTER 5

API Reference

Version: 1.1.115

5.1 API Documentation

libtaxii Module

Version: 1.1.115

5.1.1 Functions

libtaxii.**get_message_from_http_response**(*http_response*, *in_response_to*)

Create a TAXII message from an HTTPResponse object.

This function parses the `httplib.HTTPResponse` by reading the X-TAXII-Content-Type HTTP header to determine if the message binding is supported. If the X-TAXII-Content-Type header is present and the value indicates a supported Message Binding, this function will attempt to parse the HTTP Response body.

If the X-TAXII-Content-Type header is not present, this function will attempt to build a Failure Status Message per the HTTP Binding 1.0 specification.

If the X-TAXII-Content-Type header is present and indicates an unsupported Message Binding, this function will raise a ValueError.

Parameters

- **http_response** (*httplib.HTTPResponse*) – the HTTP response to parse
- **in_response_to** (*str*) – the default value for in_response_to

libtaxii.common Module

Version: 1.1.115

Common utility classes and functions used throughout libtaxii.

5.1.2 Functions

libtaxii.common.**get_xml_parser**()
Return the XML parser currently in use.

If one has not already been set (via *set_xml_parser()*), a new etree.XMLParser is constructed with no_network=True and huge_tree=False.

libtaxii.common.**set_xml_parser**(*xml_parser=None*)
Set the libtaxii.messages XML parser.

Parameters **xml_parser** (*etree.XMLParser*) – The parser to use to parse TAXII XML.

5.1.3 Classes

class libtaxii.common.**TAXIIBase**
Base class for all TAXII Messages and Message component types.

libtaxii users should not need to use this class directly.

classmethod **from_dict**(*d*)
Create an instance of this class from a dictionary.

Subclasses must implement this method.

classmethod **from_etree**(*src_etree*)
Create an instance of this class from an etree.

Subclasses must implement this method.

classmethod **from_xml**(*xml*)
Create an instance of this class from XML.

Subclasses should not need to implement this method.

sort_key
This property allows list of TAXII objects to be compared efficiently. The __eq__ method uses this property to sort the lists before comparisons are made.

Subclasses must implement this property.

to_dict()
Create a dictionary representation of this class.

Subclasses must implement this method.

to_etree()
Create an etree representation of this class.

Subclasses must implement this method.

to_json()
Create a JSON object of this class.

Assumes any binary content will be UTF-8 encoded.

to_text(*line_prepend=''*)
Create a nice looking (this is a subjective term!) textual representation of this class. Subclasses should implement this method.

Note that this is just a convenience method for making TAXII Messages nice to read for humans and may change drastically in future versions of libtaxii.

to_xml(*pretty_print=False*)
Create an XML representation of this class.

Subclasses should not need to implement this method.

libtaxii.constants Module

Version: 1.1.115

5.1.4 Constants

Version IDs

The following constants can be used as TAXII Version IDs

libtaxii.constants.**VID_TAXII_SERVICES_10 = 'urn:taxii.mitre.org:services:1.0'**
Version ID for the TAXII Services Specification 1.0

libtaxii.constants.**VID_TAXII_SERVICES_11 = 'urn:taxii.mitre.org:services:1.1'**
Version ID for the TAXII Services Specification 1.1

libtaxii.constants.**VID_TAXII_XML_10 = 'urn:taxii.mitre.org:message:xml:1.0'**
Version ID for the TAXII XML Message Binding Specification 1.0

libtaxii.constants.**VID_TAXII_XML_11 = 'urn:taxii.mitre.org:message:xml:1.1'**
Version ID for the TAXII XML Message Binding Specification 1.1

libtaxii.constants.**VID_TAXII_HTTP_10 = 'urn:taxii.mitre.org:protocol:http:1.0'**
Version ID for the TAXII HTTP Protocol Binding Specification 1.0

libtaxii.constants.**VID_TAXII_HTTPS_10 = 'urn:taxii.mitre.org:protocol:https:1.0'**
Version ID for the TAXII HTTPS Protocol Binding Specification 1.0

The following are third-party Version IDs included in libtaxii for convenience.

libtaxii.constants.**VID_CERT_EU_JSON_10 = 'urn:cert.europa.eu:message:json:1.0'**
Version ID for the CERT EU JSON Message Binding

Content Binding IDs

The following constants should be used as the Content Binding ID for STIX XML.

libtaxii.constants.**CB_STIX_XML_10 = 'urn:stix.mitre.org:xml:1.0'**
Content Binding ID for STIX XML 1.0

libtaxii.constants.**CB_STIX_XML_101 = 'urn:stix.mitre.org:xml:1.0.1'**
Content Binding ID for STIX XML 1.0.1

libtaxii.constants.**CB_STIX_XML_11 = 'urn:stix.mitre.org:xml:1.1'**
Content Binding ID for STIX XML 1.1

libtaxii.constants.**CB_STIX_XML_111 = 'urn:stix.mitre.org:xml:1.1.1'**
Content Binding ID for STIX XML 1.1.1

libtaxii.constants.**CB_STIX_XML_12 = 'urn:stix.mitre.org:xml:1.2'**
Content Binding ID for STIX XML 1.2

These other Content Binding IDs are included for convenience as well.

libtaxii.constants.**CB_CAP_11 = 'urn:oasis:names:tc:emergency:cap:1.1'**
Content Binding ID for CAP 1.1

libtaxii.constants.**CB_XENC_122002 = 'http://www.w3.org/2001/04/xmlenc#'**
Content Binding ID for XML Encryption

libtaxii.constants.**CB_SMIME = 'application/x-pkcs7-mime'**
Content Binding ID for SMIME

Namespace Map

This constant contains commonly namespaces and aliases in TAXII.

libtaxii.constants.**NS_MAP = {'taxii': 'http://taxii.mitre.org/messages/taxii_xml_binding-1', 'taxii_11': 'http://taxii.mitr**
Namespace map of namespaces libtaxii knows about

Message Types

libtaxii.constants.**MSG_STATUS_MESSAGE = 'Status_Message'**
Constant identifying a Status Message

libtaxii.constants.**MSG_DISCOVERY_REQUEST = 'Discovery_Request'**
Constant identifying a Discovery Request Message

libtaxii.constants.**MSG_DISCOVERY_RESPONSE = 'Discovery_Response'**
Constant identifying a Discovery Response Message

libtaxii.constants.**MSG_FEED_INFORMATION_REQUEST = 'Feed_Information_Request'**
Constant identifying a Feed Information Request Message

libtaxii.constants.**MSG_FEED_INFORMATION_RESPONSE = 'Feed_Information_Response'**
Constant identifying a Feed Information Response Message

libtaxii.constants.**MSG_MANAGE_FEED_SUBSCRIPTION_REQUEST = 'Subscription_Management_Request'**
Constant identifying a Subscription Management Request Message

libtaxii.constants.**MSG_MANAGE_FEED_SUBSCRIPTION_RESPONSE = 'Subscription_Management_Response'**
Constant identifying a Subscription Management Response Message

libtaxii.constants.**MSG_POLL_REQUEST = 'Poll_Request'**
Constant identifying a Poll Request Message

libtaxii.constants.**MSG_POLL_RESPONSE = 'Poll_Response'**
Constant identifying a Poll Response Message

libtaxii.constants.**MSG_INBOX_MESSAGE = 'Inbox_Message'**
Constant identifying a Inbox Message

libtaxii.constants.**MSG_TYPES_10 = ('Status_Message', 'Discovery_Request', 'Discovery_Response', 'Feed_Informati**
TAXII 1.0 Message Types

libtaxii.constants.**MSG_POLL_FULFILLMENT_REQUEST = 'Poll_Fulfillment'**
Constant identifying a Status Message

libtaxii.constants.**MSG_COLLECTION_INFORMATION_REQUEST = 'Collection_Information_Request'**
Constant identifying a Collection Information Request

libtaxii.constants.**MSG_COLLECTION_INFORMATION_RESPONSE = 'Collection_Information_Response'**
Constant identifying a Collection Information Response

libtaxii.constants.**MSG_MANAGE_COLLECTION_SUBSCRIPTION_REQUEST = 'Subscription_Management_Request'**
Constant identifying a Subscription Request

libtaxii.constants.**MSG_MANAGE_COLLECTION_SUBSCRIPTION_RESPONSE = 'Subscription_Management_Respons**
Constant identifying a Subscription Response

libtaxii.constants.**MSG_TYPES_11 = ('Status_Message', 'Discovery_Request', 'Discovery_Response', 'Collection_Infor**
Tuple of all TAXII 1.1 Message Types

Status Types

These constants are used in `StatusMessage`.

libtaxii.constants.**ST_BAD_MESSAGE = 'BAD_MESSAGE'**
Constant identifying a Status Type of Bad Message

libtaxii.constants.**ST_DENIED = 'DENIED'**
Constant identifying a Status Type of Denied

libtaxii.constants.**ST_FAILURE = 'FAILURE'**
Constant identifying a Status Type of Failure

libtaxii.constants.**ST_NOT_FOUND = 'NOT_FOUND'**
Constant identifying a Status Type of Not Found

libtaxii.constants.**ST_POLLING_UNSUPPORTED = 'POLLING_UNSUPPORTED'**
Constant identifying a Status Type of Polling Unsupported

libtaxii.constants.**ST_RETRY = 'RETRY'**
Constant identifying a Status Type of Retry

libtaxii.constants.**ST_SUCCESS = 'SUCCESS'**
Constant identifying a Status Type of Success

libtaxii.constants.**ST_UNAUTHORIZED = 'UNAUTHORIZED'**
Constant identifying a Status Type of Unauthorized

libtaxii.constants.**ST_UNSUPPORTED_MESSAGE_BINDING = 'UNSUPPORTED_MESSAGE'**
Constant identifying a Status Type of Unsupported Message Binding

libtaxii.constants.**ST_UNSUPPORTED_CONTENT_BINDING = 'UNSUPPORTED_CONTENT'**
Constant identifying a Status Type of Unsupported Content Binding

libtaxii.constants.**ST_UNSUPPORTED_PROTOCOL = 'UNSUPPORTED_PROTOCOL'**
Constant identifying a Status Type of Unsupported Protocol Binding

libtaxii.constants.**ST_TYPES_10 = ('BAD_MESSAGE', 'DENIED', 'FAILURE', 'NOT_FOUND', 'POLLING_UNSU**
Tuple of all TAXII 1.0 Status Types

libtaxii.constants.**ST_ASYNCHRONOUS_POLL_ERROR = 'ASYNCHRONOUS_POLL_ERROR'**
Constant identifying a Status Type of Asynchronous Poll Error

libtaxii.constants.**ST_DESTINATION_COLLECTION_ERROR = 'DESTINATION_COLLECTION_ERROR'**
Constant identifying a Status Type of Destination Collection Error

libtaxii.constants.**ST_INVALID_RESPONSE_PART = 'INVALID_RESPONSE_PART'**
Constant identifying a Status Type of Invalid Response Part

libtaxii.constants.**ST_NETWORK_ERROR = 'NETWORK_ERROR'**
Constant identifying a Status Type of Network Error

libtaxii.constants.**ST_PENDING = 'PENDING'**
Constant identifying a Status Type of Pending

libtaxii.constants.**ST_UNSUPPORTED_QUERY = 'UNSUPPORTED_QUERY'**
Constant identifying a Status Type of Unsupported Query Format

libtaxii.constants.**ST_TYPES_11 = ('ASYNCHRONOUS_POLL_ERROR', 'BAD_MESSAGE', 'DENIED', 'DESTINA**
Tuple of all TAXII 1.1 Status types

Subscription Actions

These constants are used in `ManageFeedSubscriptionRequest`

libtaxii.constants.**ACT_SUBSCRIBE = 'SUBSCRIBE'**
Constant identifying an Action of Subscribe

libtaxii.constants.**ACT_UNSUBSCRIBE = 'UNSUBSCRIBE'**
Constant identifying an Action of Unsubscribe

libtaxii.constants.**ACT_STATUS = 'STATUS'**
Constant identifying an Action of Status

libtaxii.constants.**ACT_TYPES_10 = ('SUBSCRIBE', 'UNSUBSCRIBE', 'STATUS')**
Tuple of all TAXII 1.0 Action Types

libtaxii.constants.**ACT_PAUSE = 'PAUSE'**
Constant identifying an Action of Pause

libtaxii.constants.**ACT_RESUME = 'RESUME'**
Constant identifying an Action of Resume

libtaxii.constants.**ACT_TYPES_11 = ('SUBSCRIBE', 'PAUSE', 'RESUME', 'UNSUBSCRIBE', 'STATUS')**
Tuple of all TAXII 1.1 Action types

Service Types

These constants are used to indicate the type of service.

libtaxii.constants.**SVC_INBOX = 'INBOX'**
Constant identifying a Service Type of Inbox

libtaxii.constants.**SVC_POLL = 'POLL'**
Constant identifying a Service Type of Poll

libtaxii.constants.**SVC_FEED_MANAGEMENT = 'FEED_MANAGEMENT'**
Constant identifying a Service Type of Feed Management

libtaxii.constants.**SVC_DISCOVERY = 'DISCOVERY'**
Constant identifying a Service Type of Discovery

libtaxii.constants.**SVC_TYPES_10 = ('INBOX', 'POLL', 'FEED_MANAGEMENT', 'DISCOVERY')**
Tuple of all TAXII 1.0 Service Types

libtaxii.constants.**SVC_COLLECTION_MANAGEMENT = 'COLLECTION_MANAGEMENT'**
Constant identifying a Service Type of Collection Management. "Feed Management" was renamed to "Collection Management" in TAXII 1.1.

libtaxii.constants.**SVC_TYPES_11 = ('INBOX', 'POLL', 'COLLECTION_MANAGEMENT', 'DISCOVERY')**
Tuple of all TAXII 1.1 Service Types

Subscription Statuses

These constants are used in `ManageCollectionSubscriptionResponse`

`libtaxii.constants.`**SS_ACTIVE = 'ACTIVE'**
Subscription Status of Active

`libtaxii.constants.`**SS_PAUSED = 'PAUSED'**
Subscription Status of Paused

`libtaxii.constants.`**SS_UNSUBSCRIBED = 'UNSUBSCRIBED'**
Subscription Status of Unsubscribed

`libtaxii.constants.`**SS_TYPES_11 = ('ACTIVE', 'PAUSED', 'UNSUBSCRIBED')**
Tuple of all TAXII 1.1 Subscription Statues

Response Types

These constants are used to indicate the type of response returned.

`libtaxii.constants.`**RT_FULL = 'FULL'**
Constant identifying a response type of Full

`libtaxii.constants.`**RT_COUNT_ONLY = 'COUNT_ONLY'**
Constant identifying a response type of Count only

`libtaxii.constants.`**RT_TYPES_11 = ('FULL', 'COUNT_ONLY')**
Tuple of all TAXII 1.1 Response Types

Collection Types

These constants are used to indicate the type of collection.

`libtaxii.constants.`**CT_DATA_FEED = 'DATA_FEED'**
Constant identifying a collection type of Data Feed

`libtaxii.constants.`**CT_DATA_SET = 'DATA_SET'**
Constant identifying a collection type of Data Set

`libtaxii.constants.`**CT_TYPES_11 = ('DATA_FEED', 'DATA_SET')**
Tuple of all TAXII 1.1 Collection Types

Status Details

These constants are used in `StatusMessage`.

`libtaxii.constants.`**SD_ACCEPTABLE_DESTINATION = 'ACCEPTABLE_DESTINATION'**
Constant Identifying the Acceptable Destination Status Detail

`libtaxii.constants.`**SD_MAX_PART_NUMBER = 'MAX_PART_NUMBER'**
Constant Identifying the Max Part Number Status Detail

`libtaxii.constants.`**SD_ITEM = 'ITEM'**
Constant Identifying the Item Status Detail

`libtaxii.constants.`**SD_ESTIMATED_WAIT = 'ESTIMATED_WAIT'**
Constant Identifying the Estimated Wait Status Detail

libtaxii.constants.**SD_RESULT_ID = 'RESULT_ID'**
Constant Identifying the Result ID Status Detail

libtaxii.constants.**SD_WILL_PUSH = 'WILL_PUSH'**
Constant Identifying the Will Push Status Detail

libtaxii.constants.**SD_SUPPORTED_BINDING = 'SUPPORTED_BINDING'**
Constant Identifying the Supported Binding Status Detail

libtaxii.constants.**SD_SUPPORTED_CONTENT = 'SUPPORTED_CONTENT'**
Constant Identifying the Supported Content Status Detail

libtaxii.constants.**SD_SUPPORTED_PROTOCOL = 'SUPPORTED_PROTOCOL'**
Constant Identifying the Supported Protocol Status Detail

libtaxii.constants.**SD_SUPPORTED_QUERY = 'SUPPORTED_QUERY'**
Constant Identifying the Supported Query Status Detail

libtaxii.constants.**SD_TYPES_11 = ('ACCEPTABLE_DESTINATION', 'MAX_PART_NUMBER', 'ITEM', 'ESTIMA**
Tuple of all TAXII 1.1 Status Detail Keys

libtaxii.constants.**SD_CAPABILITY_MODULE = 'CAPABILITY_MODULE'**
(For TAXII Default Query) Constant identifying supported Capability Modules

libtaxii.constants.**SD_PREFERRED_SCOPE = 'PREFERRED_SCOPE'**
(For TAXII Default Query) Constant identifying Preferred Scopes

libtaxii.constants.**SD_ALLOWED_SCOPE = 'ALLOWED_SCOPE'**
(For TAXII Default Query) Constant identifying Allowed Scopes

libtaxii.constants.**SD_TARGETING_EXPRESSION_ID = 'TARGETING_EXPRESSION_ID'**
(For TAXII Default Query) Constant identifying supported Targeting Expression IDs

Query Formats

These constants are used to indicate query format.

..autodata:: FID_TAXII_DEFAULT_QUERY_10

Query Capability Modules

These constants are used to indicate TAXII Default Query Capability Modules

libtaxii.constants.**CM_CORE = 'urn:taxii.mitre.org:query:capability:core-1'**
Capability Module ID for Core

libtaxii.constants.**CM_REGEX = 'urn:taxii.mitre.org:query:capability:regex-1'**
Capability Module ID for Regex

libtaxii.constants.**CM_TIMESTAMP = 'urn:taxii.mitre.org:query:capability:timestamp-1'**
Capability Module ID for Timestamp

libtaxii.constants.**CM_IDS = ('urn:taxii.mitre.org:query:capability:core-1', 'urn:taxii.mitre.org:query:capability:regex**
Tuple of all capability modules defined in TAXII Default Query 1.0

Query Operators

These constants are used to identify the operator in :py:class'Criteria'

libtaxii.constants.**OP_OR = 'OR'**
Operator OR

libtaxii.constants.**OP_AND = 'AND'**
Operator AND

libtaxii.constants.**OP_TYPES = ('OR', 'AND')**
Tuple of all operators

Query Status Types

TAXII Default Query 1.0 identifies three additional Status Types:

libtaxii.constants.**ST_UNSUPPORTED_CAPABILITY_MODULE = 'UNSUPPORTED_CAPABILITY_MODULE'**
Status Type indicating an unsupported capability module

libtaxii.constants.**ST_UNSUPPORTED_TARGETING_EXPRESSION = 'UNSUPPORTED_TARGETING_EXPRESSION**
Status Type indicating an unsupported targeting expression

libtaxii.constants.**ST_UNSUPPORTED_TARGETING_EXPRESSION_ID = 'UNSUPPORTED_TARGETING_EXPRESS**
Status Type indicating an unsupported targeting expression id

Query Parameters

These constants are used to identify parameters.

libtaxii.constants.**P_VALUE = 'value'**
Parameter name – value

libtaxii.constants.**P_MATCH_TYPE = 'match_type'**
Parameter name – match_type

libtaxii.constants.**P_CASE_SENSITIVE = 'case_sensitive'**
Parameter name – case_sensitive

libtaxii.constants.**P_NAMES = ('value', 'match_type', 'case_sensitive')**
Tuple of all parameter names

Query Relationships

These constants are used to identify relationships

libtaxii.constants.**R_EQUALS = 'equals'**
Relationship name – equals

libtaxii.constants.**R_NOT_EQUALS = 'not_equals'**
Relationship name – not_requals

libtaxii.constants.**R_GREATER_THAN = 'greater_than'**
Relationship name – greater_than

libtaxii.constants.**R_GREATER_THAN_OR_EQUAL = 'greater_than_or_equal'**
Relationship name – greater_than_or_equal

libtaxii.constants.**R_LESS_THAN = 'less_than'**
Relationship name – less_than

libtaxii.constants.**R_LESS_THAN_OR_EQUAL = 'less_than_or_equal'**
Relationship name – less_than_or_equal

libtaxii.constants.**R_DOES_NOT_EXIST = 'does_not_exist'**
Relationship name – does_not_exist

libtaxii.constants.**R_EXISTS = 'exists'**
Relationship name – exists

libtaxii.constants.**R_BEGINS_WITH = 'begins_with'**
Relationship name – begins_with

libtaxii.constants.**R_ENDS_WITH = 'ends_with'**
Relationship name – ends_with

libtaxii.constants.**R_CONTAINS = 'contains'**
Relationship name – contains

libtaxii.constants.**R_MATCHES = 'matches'**
Relationship name – matches

libtaxii.constants.**R_NAMES = ('equals', 'not_equals', 'greater_than', 'greater_than_or_equal', 'less_than', 'less_than_**
Tuple of all relationship names

libtaxii.clients Module

Version: 1.1.115

5.1.5 Classes

class libtaxii.clients.**HttpClient**(*auth_type=0*, *auth_credentials=None*, *use_https=False*)

call_taxii_service2(*host*, *path*, *message_binding*, *post_data*, *port=None*, *get_params_dict=None*, *content_type=None*, *headers=None*, *user_agent=None*, *timeout=None*)
Call a TAXII service.

Note: this uses urllib2 instead of httplib, and therefore returns a different kind of object than `call_taxii_service()`.

Returns `urllib2.Response`

set_auth_credentials(*auth_credentials_dict*)
Set the authentication credentials used later when making a request.

Note that it is possible to pass in one dict containing credentials for different authentication types and swap between them later.

Parameters **dict** (*auth_credentials_dict*) – The dictionary containing authentication credentials. e.g.: - {'key_file': '/path/to/key.key', 'cert_file': '/path/to/cert.crt'} - {'username': 'abc', 'password': 'xyz'} - Or both, if both username/password and certificate based auth are used

set_auth_type(*auth_type*)
Set the authentication type for this client.

Parameters **auth_type** (*string*) – Must be one of `AUTH_NONE`, `AUTH_BASIC`, or `AUTH_CERT`

set_proxy(*proxy_string=None*)
Set the proxy settings to use when making a connection.

Parameters **proxy_string** (*string*) – Proxy address formatted like http://proxy.example.com:80. Set to SYSTEM_PROXY to use the system proxy; set to NO_PROXY to use no proxy.

set_use_https(*bool_*)
Indicate whether the HttpClient should use HTTP or HTTPs. The default is HTTP.

Parameters **bool** (*bool*) – The new use_https value.

set_verify_server(*verify_server=False*, *ca_file=None*)
Tell libtaxii whether to verify the server's ssl certificate using the provided ca_file.

Parameters **verify_server** (*bool*) – Flag indicating whether or not libtaxii should verify the server.

5.1.6 Examples

TAXII clients have three types of authentication credentials: None, HTTP Basic, and TLS Certificate. This section demonstrates usage of all three auth types.

All examples assume the following imports:

```
import libtaxii as t
import libtaxii.messages_11 as tm11
import libtaxii.clients as tc
from libtaxii.common import generate_message_id
from libtaxii.constants import *
from dateutil.tz import tzutc
```

Using No Credentials

```
client = tc.HttpClient()
client.set_auth_type(tc.HttpClient.AUTH_NONE)
client.set_use_https(False)

discovery_request = tm11.DiscoveryRequest(generate_message_id())
discovery_xml = discovery_request.to_xml(pretty_print=True)

http_resp = client.call_taxii_service2('hailataxii.com', '/taxii-discovery-service',␣
↪VID_TAXII_XML_11, discovery_xml)
taxii_message = t.get_message_from_http_response(http_resp, discovery_request.message_
↪id)
print taxii_message.to_xml(pretty_print=True)
```

Using Basic HTTP Auth

```
client = tc.HttpClient()
client.set_auth_type(tc.HttpClient.AUTH_BASIC)
client.set_auth_credentials({'username': 'guest', 'password': 'guest'})

discovery_request = tm11.DiscoveryRequest(generate_message_id())
discovery_xml = discovery_request.to_xml(pretty_print=True)

http_resp = client.call_taxii_service2('hailataxii.com', '/taxii-discovery-service',␣
↪VID_TAXII_XML_11, discovery_xml)
```

```
taxii_message = t.get_message_from_http_response(http_resp, discovery_request.message_
↪id)
print taxii_message.to_xml(pretty_print=True)
```

Using TLS Certificate Auth

Note: The following code is provided as an example of this authentication method, but will not work as-is, because Hail A Taxii does not support TLS.

```
client = tc.HttpClient()
client.set_use_https(True)
client.set_auth_type(tc.HttpClient.AUTH_CERT)
client.set_auth_credentials({'key_file': '../PATH_TO_KEY_FILE.key', 'cert_file': '../
↪PATH_TO_CERT_FILE.crt'})

discovery_request = tm11.DiscoveryRequest(generate_message_id())
discovery_xml = discovery_request.to_xml(pretty_print=True)

http_resp = client.call_taxii_service2('hailataxii.com', '/taxii-discovery-service/',␣
↪VID_TAXII_XML_11, discovery_xml)
taxii_message = t.get_message_from_http_response(http_resp, discovery_request.message_
↪id)
print taxii_message.to_xml(pretty_print=True)
```

libtaxii.messages_10 Module

Version: 1.1.115

Creating, handling, and parsing TAXII 1.0 messages.

Note: The examples on this page assume that you have run the equivalent of

```
import datetime
from dateutil.tz import tzutc
import libtaxii as t
import libtaxii.messages_10 as tm10
from libtaxii.constants import *
```

5.1.7 Status Message

class libtaxii.messages_10.**StatusMessage**(*message_id*, *in_response_to*, *extended_headers=None*, *status_type=None*, *status_detail=None*, *message=None*)

A TAXII Status message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **in_response_to** (*str*) – Contains the Message ID of the message to which this is a response. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**

- **status_type** (*str*) – One of the defined Status Types or a third-party- defined Status Type. **Required**
- **status_detail** (*str*) – A field for additional information about this status in a machine-readable format. **Optional or Prohibited** depending on `status_type`. See TAXII Specification for details.
- **message** (*str*) – Additional information for the status. There is no expectation that this field be interpretable by a machine; it is instead targeted to a human operator. **Optional**

Example:

```
status_message1 = tm10.StatusMessage(
        message_id=tm10.generate_message_id(),
        in_response_to="12345",
        status_type=ST_SUCCESS,
        status_detail='Machine-processable info here!',
        message='This is a message.')
```

5.1.8 Discovery Request

class libtaxii.messages_10.**DiscoveryRequest**(*message_id*, *in_response_to=None*, *extended_headers=None*)

A TAXII Discovery Request message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**

Example:

```
ext_headers = {'name1': 'val1', 'name2': 'val2'}
discovery_request = tm10.DiscoveryRequest(
        message_id=tm10.generate_message_id(),
        extended_headers=ext_headers)
```

5.1.9 Discovery Response

class libtaxii.messages_10.**DiscoveryResponse**(*message_id*, *in_response_to*, *extended_headers=None*, *service_instances=None*)

A TAXII Discovery Response message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **in_response_to** (*str*) – Contains the Message ID of the message to which this is a response. **Optional**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **service_instances** (list of *ServiceInstance*) – a list of service instances that this response contains. **Optional**

class libtaxii.messages_10.**ServiceInstance**(*service_type*, *services_version*, *protocol_binding*, *service_address*, *message_bindings*, *inbox_service_accepted_content=None*, *available=None*, *message=None*)

The Service Instance component of a TAXII Discovery Response Message.

Parameters

- **service_type** (*string*) – identifies the Service Type of this Service Instance. **Required**
- **services_version** (*string*) – identifies the TAXII Services Specification to which this Service conforms. **Required**
- **protocol_binding** (*string*) – identifies the protocol binding supported by this Service. **Required**
- **service_address** (*string*) – identifies the network address of the TAXII Daemon that hosts this Service. **Required**
- **message_bindings** (*list of strings*) – identifies the message bindings supported by this Service instance. **Required**
- **inbox_service_accepted_content** (*list of strings*) – identifies content bindings that this Inbox Service is willing to accept. **Optional**
- **available** (*boolean*) – indicates whether the identity of the requester (authenticated or otherwise) is allowed to access this TAXII Service. **Optional**
- **message** (*string*) – contains a message regarding this Service instance. **Optional**

The message_bindings list must contain at least one value.

Example:

```
discovery_response = tm10.DiscoveryResponse(
        message_id=tm10.generate_message_id(),
        in_response_to=discovery_request.message_id)

service_instance = tm10.ServiceInstance(
        service_type=SVC_INBOX,
        services_version=VID_TAXII_SERVICES_10,
        protocol_binding=VID_TAXII_HTTPS_10,
        service_address='https://example.com/inbox/',
        message_bindings=[VID_TAXII_XML_10],
        inbox_service_accepted_content=[CB_STIX_XML_10],
        available=True,
        message='This is a sample inbox service instance')

discovery_response.service_instances.append(service_instance)

# Alternatively, you could define the service instance(s) first and use the
# following:

service_instance_list = [service_instance]
discovery_response = tm10.DiscoveryResponse(
        message_id=tm10.generate_message_id(),
        in_response_to=discovery_request.message_id,
        service_instances=service_instance_list)
```

5.1.10 Feed Information Request

class libtaxii.messages_10.**FeedInformationRequest**(*message_id*, *in_response_to=None*, *extended_headers=None*)

A TAXII Feed Information Request message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**

Example:

```
ext_headers = {'name1': 'val1', 'name2': 'val2'}
feed_information_request= tm10.FeedInformationRequest(
        message_id=tm10.generate_message_id(),
        extended_headers=ext_headers)
```

5.1.11 Feed Information Response

class libtaxii.messages_10.**FeedInformationResponse**(*message_id*, *in_response_to*, *extended_headers=None*, *feed_informations=None*)

A TAXII Feed Information Response message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **in_response_to** (*str*) – Contains the Message ID of the message to which this is a response. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **feed_informations** (*list of FeedInformation*) – A list of FeedInformation objects to be contained in this response. **Optional**

class libtaxii.messages_10.**FeedInformation**(*feed_name*, *feed_description*, *supported_contents*, *available=None*, *push_methods=None*, *polling_service_instances=None*, *subscription_methods=None*)

The Feed Information component of a TAXII Feed Information Response Message.

Parameters

- **feed_name** (*str*) – the name by which this TAXII Data Feed is identified. **Required**
- **feed_description** (*str*) – a prose description of this TAXII Data Feed. **Required**
- **supported_contents** (*list of str*) – Content Binding IDs indicating which types of content are currently expressed in this TAXII Data Feed. **Required**
- **available** (*boolean*) – whether the identity of the requester (authenticated or otherwise) is allowed to access this TAXII Service. **Optional** Default: None, indicating "unknown"
- **push_methods** (*list of PushMethod objects*) – the protocols that can be used to push content via a subscription. **Optional**

- **polling_service_instances** (*list of PollingServiceInstance objects*) – the bindings and address a Consumer can use to interact with a Poll Service instance that supports this TAXII Data Feed. **Optional**
- **subscription_methods** (*list of SubscriptionMethod objects*) – the protocol and address of the TAXII Daemon hosting the Feed Management Service that can process subscriptions for this TAXII Data Feed. **Optional**

The absense of `push_methods` indicates no push methods. The absense of `polling_service_instances` indicates no polling services. At least one of `push_methods` and `polling_service_instances` must not be empty. The absense of `subscription_methods` indicates no subscription services.

class `libtaxii.messages_10.`**PushMethod**(*push_protocol*, *push_message_bindings*)
The Push Method component of a TAXII Feed Information component.

Parameters

- **push_protocol** (*str*) – a protocol binding that can be used to push content to an Inbox Service instance. **Required**
- **push_message_bindings** (*list of str*) – the message bindings that can be used to push content to an Inbox Service instance using the protocol identified in the Push Protocol field. **Required**

class `libtaxii.messages_10.`**PollingServiceInstance**(*poll_protocol*, *poll_address*, *poll_message_bindings*)
The Polling Service Instance component of a TAXII Feed Information component.

Parameters

- **poll_protocol** (*str*) – the protocol binding supported by this Poll Service instance. **Required**
- **poll_address** (*str*) – the address of the TAXII Daemon hosting this Poll Service instance. **Required**
- **poll_message_bindings** (*list of str*) – the message bindings supported by this Poll Service instance. **Required**

class `libtaxii.messages_10.`**SubscriptionMethod**(*subscription_protocol*, *subscription_address*, *subscription_message_bindings*)
The Subscription Method component of a TAXII Feed Information component.

Parameters

- **subscription_protocol** (*str*) – the protocol binding supported by this Feed Management Service instance. **Required**
- **subscription_address** (*str*) – the address of the TAXII Daemon hosting this Feed Management Service instance. **Required**.
- **subscription_message_bindings** (*list of str*) – the message bindings supported by this Feed Management Service Instance. **Required**

Example:

```
push_method1 = tm10.PushMethod(
        push_protocol=VID_TAXII_HTTP_10,
        push_message_bindings=[VID_TAXII_XML_10])

polling_service1 = tm10.PollingServiceInstance(
        poll_protocol=VID_TAXII_HTTP_10,
```

```
        poll_address='http://example.com/PollService/',
        poll_message_bindings=[VID_TAXII_XML_10])

subscription_service1 = tm10.SubscriptionMethod(
        subscription_protocol=VID_TAXII_HTTP_10,
        subscription_address='http://example.com/SubsService/',
        subscription_message_bindings=[VID_TAXII_XML_10])

feed1 = tm10.FeedInformation(
        feed_name='Feed1',
        feed_description='Description of a feed',
        supported_contents=[CB_STIX_XML_10],
        available=True,
        push_methods=[push_method1],
        polling_service_instances=[polling_service1],
        subscription_methods=[subscription_service1])

feed_information_response1 = tm10.FeedInformationResponse(
        message_id=tm10.generate_message_id(),
        in_response_to=tm10.generate_message_id(),
        feed_informations=[feed1])
```

5.1.12 Manage Feed Subscription Request

class libtaxii.messages_10.**ManageFeedSubscriptionRequest**(*message_id*, *extended_headers=None*, *feed_name=None*, *action=None*, *subscription_id=None*, *delivery_parameters=None*)

A TAXII Manage Feed Subscription Request message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **feed_name** (*str*) – the name of the TAXII Data Feed to which the action applies. **Required**
- **action** (*str*) – the requested action to take. **Required**
- **subscription_id** (*str*) – the ID of a previously created subscription. **Required** if `action==ACT_UNSUBSCRIBE`, else **Prohibited**.
- **delivery_parameters** (*list of DeliveryParameters*) – the delivery parameters for this request. **Optional** Absence means delivery is not requested.

Example:

```
delivery_parameters1 = tm10.DeliveryParameters(
        inbox_protocol=VID_TAXII_HTTP_10,
        inbox_address='http://example.com/inbox',
        delivery_message_binding=VID_TAXII_XML_10,
        content_bindings=[CB_STIX_XML_10])
```

```
manage_feed_subscription_request1 = tm10.ManageFeedSubscriptionRequest(
        message_id=tm10.generate_message_id(),
        feed_name='SomeFeedName',
        action=ACT_UNSUBSCRIBE,
        subscription_id='SubsId056',
        delivery_parameters=delivery_parameters1)
```

5.1.13 Manage Feed Subscription Response

class libtaxii.messages_10.**ManageFeedSubscriptionResponse**(*message_id*, *in_response_to*, *extended_headers=None*, *feed_name=None*, *message=None*, *subscription_instances=None*)

A TAXII Manage Feed Subscription Response message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **in_response_to** (*str*) – Contains the Message ID of the message to which this is a response. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **feed_name** (*str*) – the name of the TAXII Data Feed to which the action applies. **Required**
- **message** (*str*) – additional information for the message recipient. **Optional**
- **subscription_instances** (*list of SubscriptionInstance*) – **Optional**

class libtaxii.messages_10.**SubscriptionInstance**(*subscription_id*, *delivery_parameters=None*, *poll_instances=None*)

The Subscription Instance component of the Manage Feed Subscription Response message.

Parameters

- **subscription_id** (*str*) – the id of the subscription. **Required**
- **delivery_parameters** (libtaxii.messages_10.DeliveryParameters) – the parameters for this subscription. **Required** if responding to message with `action==ACT_STATUS`, otherwise **Prohibited**
- **poll_instances** (*list of PollInstance*) – Each Poll Instance represents an instance of a Poll Service that can be contacted to retrieve content associated with the new Subscription. **Optional**

class libtaxii.messages_10.**PollInstance**(*poll_protocol*, *poll_address*, *poll_message_bindings=None*)

The Poll Instance component of the Manage Feed Subscription Response message.

Parameters

- **poll_protocol** (*str*) – The protocol binding supported by this instance of a Polling Service. **Required**

- **poll_address** (*str*) – the address of the TAXII Daemon hosting this Poll Service. **Required**
- **poll_message_bindings** (*list of str*) – one or more message bindings that can be used when interacting with this Poll Service instance. **Required**

Example:

```
poll_instance1 = tm10.PollInstance(
        poll_protocol=VID_TAXII_HTTP_10,
        poll_address='http://example.com/poll',
        poll_message_bindings=[VID_TAXII_XML_10])

subscription_instance1 = tm10.SubscriptionInstance(
        subscription_id='SubsId234',
        delivery_parameters=[delivery_parameters1],
        poll_instances=[poll_instance1])

manage_feed_subscription_response1 = tm10.ManageFeedSubscriptionResponse(
        message_id=tm10.generate_message_id(),
        in_response_to="12345",
        feed_name='Feed001',
        message='This is a message',
        subscription_instances=[subscription_instance1])
```

5.1.14 Poll Request

class libtaxii.messages_10.**PollRequest**(*message_id*, *extended_headers=None*, *feed_name=None*, *exclusive_begin_timestamp_label=None*, *inclusive_end_timestamp_label=None*, *subscription_id=None*, *content_bindings=None*)

A TAXII Poll Request message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **feed_name** (*str*) – the name of the TAXII Data Feed that is being polled. **Required**
- **exclusive_begin_timestamp_label** (*datetime*) – a Timestamp Label indicating the beginning of the range of TAXII Data Feed content the requester wishes to receive. **Optional**
- **inclusive_end_timestamp_label** (*datetime*) – a Timestamp Label indicating the end of the range of TAXII Data Feed content the requester wishes to receive. **Optional**
- **subscription_id** (*str*) – the existing subscription the Consumer wishes to poll. **Optional**
- **content_bindings** (*list of str*) – the type of content that is requested in the response to this poll. **Optional**, defaults to accepting all content bindings.

Example:

```
poll_request1 = tm10.PollRequest(
        message_id=tm10.generate_message_id(),
```

```
        feed_name='TheFeedToPoll',
        exclusive_begin_timestamp_label=datetime.datetime.now(tzutc()),
        inclusive_end_timestamp_label=datetime.datetime.now(tzutc()),
        subscription_id='SubsId002',
        content_bindings=[CB_STIX_XML_10])
```

5.1.15 Poll Response

class libtaxii.messages_10.**PollResponse**(*message_id*, *in_response_to*, *extended_headers=None*, *feed_name=None*, *inclusive_begin_timestamp_label=None*, *inclusive_end_timestamp_label=None*, *subscription_id=None*, *message=None*, *content_blocks=None*)

A TAXII Poll Response message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **in_response_to** (*str*) – Contains the Message ID of the message to which this is a response. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **feed_name** (*str*) – the name of the TAXII Data Feed that was polled. **Required**
- **inclusive_begin_timestamp_label** (*datetime*) – a Timestamp Label indicating the beginning of the range this response covers. **Optional**
- **inclusive_end_timestamp_label** (*datetime*) – a Timestamp Label indicating the end of the range this response covers. **Required**
- **subscription_id** (*str*) – the Subscription ID for which this content is being provided. **Optional**
- **message** (*str*) – additional information for the message recipient. **Optional**
- **content_blocks** (*list of ContentBlock*) – piece of content and additional information related to the content. **Optional**

Example:

```
poll_response1 = tm10.PollResponse(
        message_id=tm10.generate_message_id(),
        in_response_to="12345",
        feed_name='FeedName',
        inclusive_begin_timestamp_label=datetime.datetime.now(tzutc()),
        inclusive_end_timestamp_label=datetime.datetime.now(tzutc()),
        subscription_id='SubsId001',
        message='This is a message.',
        content_blocks=[])
```

5.1.16 Inbox Message

class libtaxii.messages_10.**InboxMessage**(*message_id*, *in_response_to=None*, *extended_headers=None*, *message=None*, *subscription_information=None*, *content_blocks=None*)

A TAXII Inbox message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **message** (*str*) – prose information for the message recipient. **Optional**
- **subscription_information** (libtaxii.messages_10.SubscriptionInformation) – This field is only present if this message is being sent to provide content in accordance with an existing TAXII Data Feed subscription. **Optional**
- **content_blocks** (*list of ContentBlock*) – Inbox content. **Optional**

class libtaxii.messages_10.**SubscriptionInformation**(*feed_name*, *subscription_id*, *inclusive_begin_timestamp_label*, *inclusive_end_timestamp_label*)

The Subscription Information component of a TAXII Inbox message.

Parameters

- **feed_name** (*str*) – the name of the TAXII Data Feed from which this content is being provided. **Required**
- **subscription_id** (*str*) – the Subscription ID for which this content is being provided. **Required**
- **inclusive_begin_timestamp_label** (*datetime*) – a Timestamp Label indicating the beginning of the time range this Inbox Message covers. **Optional**
- **inclusive_end_timestamp_label** (*datetime*) – a Timestamp Label indicating the end of the time range this Inbox Message covers. **Optional**

Example:

```
cb1 = tm10.ContentBlock(CB_STIX_XML_11, "")

subscription_information1 = tm10.SubscriptionInformation(
        feed_name='SomeFeedName',
        subscription_id='SubsId021',
        inclusive_begin_timestamp_label=datetime.datetime.now(tzutc()),
        inclusive_end_timestamp_label=datetime.datetime.now(tzutc()))

inbox_message1 = tm10.InboxMessage(
        message_id=tm10.generate_message_id(),
        message='This is a message.',
        subscription_information=subscription_information1,
        content_blocks=[cb1])
```

5.1.17 Other Classes

class libtaxii.messages_10.**TAXIIMessage**(*message_id*, *in_response_to=None*, *extended_headers=None*)

Encapsulate properties common to all TAXII Messages (such as headers).

This class is extended by each Message Type (e.g., DiscoveryRequest), with each subclass containing subclass-specific information

class libtaxii.messages_10.**ContentBlock**(*content_binding*, *content*, *timestamp_label=None*, *padding=None*)

A TAXII Content Block.

Parameters

- **content_binding** (*str*) – a Content Binding ID or nesting expression indicating the type of content contained in the Content field of this Content Block. **Required**
- **content** (*string or etree*) – a piece of content of the type specified by the Content Binding. **Required**
- **timestamp_label** (*datetime*) – the Timestamp Label associated with this Content Block. **Optional**
- **padding** (*string*) – an arbitrary amount of padding for this Content Block. **Optional**

Example:

```
cb1 = tm10.ContentBlock(
        content_binding=CB_STIX_XML_10,
        content='<stix:STIX_Package xmlns:stix="http://stix.mitre.org/stix-1"/>')
```

class libtaxii.messages_10.**DeliveryParameters**(*inbox_protocol=None*, *inbox_address=None*, *delivery_message_binding=None*, *content_bindings=None*)

Delivery Parameters.

Parameters

- **inbox_protocol** (*str*) – identifies the protocol to be used when pushing TAXII Data Feed content to a Consumer's TAXII Inbox Service implementation. **Required**
- **inbox_address** (*str*) – identifies the address of the TAXII Daemon hosting the Inbox Service to which the Consumer requests content for this TAXII Data Feed to be delivered. **Required**
- **delivery_message_binding** (*str*) – identifies the message binding to be used to send pushed content for this subscription. **Required**
- **content_bindings** (*list of str*) – contains Content Binding IDs indicating which types of contents the Consumer requests to receive for this TAXII Data Feed. **Optional**

5.1.18 Functions

libtaxii.messages_10.**generate_message_id**(*maxlen=5*, *version='urn:taxii.mitre.org:services:1.0'*)

Generate a TAXII Message ID.

Parameters **maxlen** (*int*) – maximum length of the ID, in characters

Example

```
msg_id = tm11.generate_message_id()
message = tm11.DiscoveryRequest(msg_id)
# Or...
message = tm11.DiscoveryRequest(tm11.generate_message_id())
```

libtaxii.messages_10.**validate_xml**(*xml_string*)

Note that this function has been deprecated. Please see libtaxii.validators.SchemaValidator.

Validate XML with the TAXII XML Schema 1.0.

Parameters **xml_string** (*str*) – The XML to validate.

Example

```
is_valid = tm10.validate_xml(message.to_xml())
```

libtaxii.messages_10.**get_message_from_xml**(*xml_string*, *encoding='utf_8'*)

Create a TAXIIMessage object from an XML string.

This function automatically detects which type of Message should be created based on the XML.

Parameters **xml_string** (*str*) – The XML to parse into a TAXII message.

Example

```
message_xml = message.to_xml()
new_message = tm10.get_message_from_xml(message_xml)
```

libtaxii.messages_10.**get_message_from_dict**(*d*)

Create a TAXIIMessage object from a dictonary.

This function automatically detects which type of Message should be created based on the 'message_type' key in the dictionary.

Parameters **d** (*dict*) – The dictionary to build the TAXII message from.

Example

```
message_dict = message.to_dict()
new_message = tm10.get_message_from_dict(message_dict)
```

libtaxii.messages_10.**get_message_from_json**(*json_string*, *encoding='utf_8'*)

Create a TAXIIMessage object from a JSON string.

This function automatically detects which type of Message should be created based on the JSON.

Parameters **json_string** (*str*) – The JSON to parse into a TAXII message.

libtaxii.messages_11 Module

Version: 1.1.115

Creating, handling, and parsing TAXII 1.1 messages.

Note: The examples on this page assume that you have run the equivalent of

```
import datetime
from dateutil.tz import tzutc
import libtaxii as t
import libtaxii.messages_11 as tm11
from libtaxii.constants import *
```

5.1.19 Status Message

class libtaxii.messages_11.**StatusMessage**(*message_id*, *in_response_to*, *extended_headers=None*, *status_type=None*, *status_detail=None*, *message=None*)

A TAXII Status message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **in_response_to** (*str*) – Contains the Message ID of the message to which this is a response. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **status_type** (*str*) – One of the defined Status Types or a third-party- defined Status Type. **Required**
- **status_detail** (*dict*) – A field for additional information about this status in a machine-readable format. **Required or Optional** depending on status_type. See TAXII Specification for details.
- **message** (*str*) – Additional information for the status. There is no expectation that this field be interpretable by a machine; it is instead targeted to a human operator. **Optional**

Example:

```
sm03 = tm11.StatusMessage(
        message_id='SM03',
        in_response_to=tm11.generate_message_id(),
        status_type=ST_DESTINATION_COLLECTION_ERROR,
        status_detail={'ACCEPTABLE_DESTINATION': ['Collection1','Collection2']})
```

5.1.20 Discovery Request

class libtaxii.messages_11.**DiscoveryRequest**(*message_id*, *in_response_to=None*, *extended_headers=None*)

A TAXII Discovery Request message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**

Example:

```
headers={'ext_header1': 'value1', 'ext_header2': 'value2'}
discovery_request = tm11.DiscoveryRequest(
        message_id=tm11.generate_message_id(),
        extended_headers=headers)
```

5.1.21 Discovery Response

class libtaxii.messages_11.**DiscoveryResponse**(*message_id*, *in_response_to*, *extended_headers=None*, *service_instances=None*)

A TAXII Discovery Response message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **in_response_to** (*str*) – Contains the Message ID of the message to which this is a response. **Optional**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **service_instances** (list of *ServiceInstance*) – a list of service instances that this response contains. **Optional**

class libtaxii.messages_11.**ServiceInstance**(*service_type*, *services_version*, *protocol_binding*, *service_address*, *message_bindings*, *inbox_service_accepted_content=None*, *available=None*, *message=None*, *supported_query=None*)

The Service Instance component of a TAXII Discovery Response Message.

Parameters

- **service_type** (*string*) – identifies the Service Type of this Service Instance. **Required**
- **services_version** (*string*) – identifies the TAXII Services Specification to which this Service conforms. **Required**
- **protocol_binding** (*string*) – identifies the protocol binding supported by this Service. **Required**
- **service_address** (*string*) – identifies the network address of the TAXII Daemon that hosts this Service. **Required**
- **message_bindings** (*list of strings*) – identifies the message bindings supported by this Service instance. **Required**
- **inbox_service_accepted_content** (*list of ContentBinding objects*) – identifies content bindings that this Inbox Service is willing to accept. **Optional**
- **available** (*boolean*) – indicates whether the identity of the requester (authenticated or otherwise) is allowed to access this TAXII Service. **Optional**
- **message** (*string*) – contains a message regarding this Service instance. **Optional**
- **supported_query** (*SupportedQuery*) – contains a structure indicating a supported query. **Optional**

The `message_bindings` list must contain at least one value. The `supported_query` parameter is optional when `service_type` is `SVC_POLL`.

Example:

```
discovery_response = tm11.DiscoveryResponse(
        message_id=tm11.generate_message_id(),
        in_response_to=discovery_request.message_id)

service_instance = tm11.ServiceInstance(
        service_type=SVC_POLL,
        services_version=VID_TAXII_SERVICES_11,
        protocol_binding=VID_TAXII_HTTP_10,
        service_address='http://example.com/poll/',
        message_bindings=[VID_TAXII_XML_11],
        available=True,
        message='This is a message.',
        #supported_query=[tdq1],
        )

discovery_response.service_instances.append(service_instance)

# Alternatively, you could define the service instance(s) first and use the
# following:

service_instance_list = [service_instance]
discovery_response = tm11.DiscoveryResponse(
        message_id=tm11.generate_message_id(),
        in_response_to=discovery_request.message_id,
        service_instances=service_instance_list)
```

5.1.22 Collection Information Request

class libtaxii.messages_11.**CollectionInformationRequest**(*message_id*, *in_response_to=None*, *extended_headers=None*)

A TAXII Collection Information Request message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**

Example:

```
ext_headers = {'name1': 'val1', 'name2': 'val2'}
collection_information_request = tm11.CollectionInformationRequest(
        message_id='CIReq01',
        extended_headers=ext_headers)
```

5.1.23 Collection Information Response

class libtaxii.messages_11.**CollectionInformationResponse**(*message_id*, *in_response_to*, *extended_headers=None*, *collection_informations=None*)

A TAXII Collection Information Response message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **in_response_to** (*str*) – Contains the Message ID of the message to which this is a response. **Optional**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **collection_informations** (*list of CollectionInformation objects*) – A list of CollectionInformation objects to be contained in this response. **Optional**

class libtaxii.messages_11.**CollectionInformation**(*collection_name*, *collection_description*, *supported_contents=None*, *available=None*, *push_methods=None*, *polling_service_instances=None*, *subscription_methods=None*, *collection_volume=None*, *collection_type='DATA_FEED'*, *receiving_inbox_services=None*)

The Collection Information component of a TAXII Collection Information Response Message.

Parameters

- **collection_name** (*str*) – the name by which this TAXII Data Collection is identified. **Required**
- **collection_description** (*str*) – a prose description of this TAXII Data Collection. **Required**
- **supported_contents** (*list of str*) – Content Binding IDs indicating which types of content are currently expressed in this TAXII Data Collection. **Optional**
- **available** (*boolean*) – whether the identity of the requester (authenticated or otherwise) is allowed to access this TAXII Service. **Optional** Default: `None`, indicating "unknown"
- **push_methods** (*list of PushMethod objects*) – the protocols that can be used to push content via a subscription. **Optional**
- **polling_service_instances** (*list of PollingServiceInstance objects*) – the bindings and address a Consumer can use to interact with a Poll Service instance that supports this TAXII Data Collection. **Optional**
- **subscription_methods** (*list of SubscriptionMethod objects*) – the protocol and address of the TAXII Daemon hosting the Collection Management Service that can process subscriptions for this TAXII Data Collection. **Optional**
- **collection_volume** (*int*) – the typical number of messages per day. **Optional**
- **collection_type** (*str*) – the type ofo this collection. **Optional**, defaults to `CT_DATA_FEED`.

- **receiving_inbox_services** (*list of ReceivingInboxService objects*) – TODO: FILL THIS IN. **Optional**

If `supported_contents` is omitted, then the collection supports all content bindings. The absense of `push_methods` indicates no push methods. The absense of `polling_service_instances` indicates no polling services. The absense of `subscription_methods` indicates no subscription services. The absense of `receiving_inbox_services` indicates no receiving inbox services.

class `libtaxii.messages_11.`**PushMethod**(*push_protocol*, *push_message_bindings*)
The Push Method component of a TAXII Collection Information component.

Parameters

- **push_protocol** (*str*) – a protocol binding that can be used to push content to an Inbox Service instance. **Required**
- **push_message_bindings** (*list of str*) – the message bindings that can be used to push content to an Inbox Service instance using the protocol identified in the Push Protocol field. **Required**

class `libtaxii.messages_11.`**PollingServiceInstance**(*poll_protocol*, *poll_address*, *poll_message_bindings*)
The Polling Service Instance component of a TAXII Collection Information component.

Parameters

- **poll_protocol** (*str*) – the protocol binding supported by this Poll Service instance. **Required**
- **poll_address** (*str*) – the address of the TAXII Daemon hosting this Poll Service instance. **Required**
- **poll_message_bindings** (*list of str*) – the message bindings supported by this Poll Service instance. **Required**

class `libtaxii.messages_11.`**SubscriptionMethod**(*subscription_protocol*, *subscription_address*, *subscription_message_bindings*)
The Subscription Method component of a TAXII Collection Information component.

Parameters

- **subscription_protocol** (*str*) – the protocol binding supported by this Collection Management Service instance. **Required**
- **subscription_address** (*str*) – the address of the TAXII Daemon hosting this Collection Management Service instance. **Required**.
- **subscription_message_bindings** (*list of str*) – the message bindings supported by this Collection Management Service Instance. **Required**

class `libtaxii.messages_11.`**ReceivingInboxService**(*inbox_protocol*, *inbox_address*, *inbox_message_bindings*, *supported_contents=None*)
The Receiving Inbox Service component of a TAXII Collection Information component.

Parameters

- **inbox_protocol** (*str*) – Indicates the protocol this Inbox Service uses. **Required**
- **address** (*inbox*) – Indicates the address of this Inbox Service. **Required**
- **inbox_message_bindings** (*list of str*) – Each string indicates a message binding that this inbox service uses. **Required**

- **supported_contents** (*list of ContentBinding objects*) – Each object indicates a Content Binding this inbox service can receive. **Optional**. Setting to `None` means that all Content Bindings are supported.

Example:

```
push_method1 = tm11.PushMethod(
        push_protocol=VID_TAXII_HTTP_10,
        push_message_bindings=[VID_TAXII_XML_11])

poll_service1 = tm11.PollingServiceInstance(
        poll_protocol=VID_TAXII_HTTPS_10,
        poll_address='https://example.com/PollService1',
        poll_message_bindings=[VID_TAXII_XML_11])

poll_service2 = tm11.PollingServiceInstance(
        poll_protocol=VID_TAXII_HTTPS_10,
        poll_address='https://example.com/PollService2',
        poll_message_bindings=[VID_TAXII_XML_11])

subs_method1 = tm11.SubscriptionMethod(
        subscription_protocol=VID_TAXII_HTTPS_10,
        subscription_address='https://example.com/SubscriptionService',
        subscription_message_bindings=[VID_TAXII_XML_11])

inbox_service1 = tm11.ReceivingInboxService(
        inbox_protocol=VID_TAXII_HTTPS_10,
        inbox_address='https://example.com/InboxService',
        inbox_message_bindings=[VID_TAXII_XML_11],
        supported_contents=None)

collection1 = tm11.CollectionInformation(
        collection_name='collection1',
        collection_description='This is a collection',
        supported_contents=[tm11.ContentBinding(CB_STIX_XML_101)],
        available=False,
        push_methods=[push_method1],
        polling_service_instances=[poll_service1, poll_service2],
        subscription_methods=[subs_method1],
        collection_volume=4,
        collection_type=CT_DATA_FEED,
        receiving_inbox_services=[inbox_service1])

collection_response1 = tm11.CollectionInformationResponse(
        message_id='CIR01',
        in_response_to='0',
        collection_informations=[collection1])
```

5.1.24 Manage Collection Subscription Request

class libtaxii.messages_11.**ManageCollectionSubscriptionRequest**(*message_id*, *extended_headers=None*, *collection_name=None*, *action=None*, *subscription_id=None*, *subscription_parameters=None*, *push_parameters=None*)

A TAXII Manage Collection Subscription Request message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **collection_name** (*str*) – the name of the TAXII Data Collection to which the action applies. **Required**
- **action** (*str*) – the requested action to take. **Required**
- **subscription_id** (*str*) – the ID of a previously created subscription. **Prohibited** if action==ACT_SUBSCRIBE, else **Required**
- **subscription_parameters** (SubscriptionParameters) – The parameters for this subscription. **Optional**
- **push_parameters** (*list of PushParameter*) – the push parameters for this request. **Optional** Absence means push is not requested.

Example:

```
subscription_parameters1 = tm11.SubscriptionParameters()
push_parameters1 = tm11.PushParameters("", "", "")

subs_req1 = tm11.ManageCollectionSubscriptionRequest(
        message_id='SubsReq01',
        action=ACT_SUBSCRIBE,
        collection_name='collection1',
        subscription_parameters=subscription_parameters1,
        push_parameters=push_parameters1)
```

5.1.25 Manage Collection Subscription Response

class libtaxii.messages_11.**ManageCollectionSubscriptionResponse**(*message_id*, *in_response_to*, *extended_headers=None*, *collection_name=None*, *message=None*, *subscription_instances=None*)

A TAXII Manage Collection Subscription Response message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **in_response_to** (*str*) – Contains the Message ID of the message to which this is a response. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **collection_name** (*str*) – the name of the TAXII Data Collection to which the action applies. **Required**
- **message** (*str*) – additional information for the message recipient. **Optional**
- **subscription_instances** (*list of SubscriptionInstance*) – **Optional**

class libtaxii.messages_11.**SubscriptionInstance**(*subscription_id*, *status='ACTIVE'*, *subscription_parameters=None*, *push_parameters=None*, *poll_instances=None*)

The Subscription Instance component of the Manage Collection Subscription Response message.

Parameters

- **subscription_id** (*str*) – the id of the subscription. **Required**
- **status** (*str*) – One of SS_ACTIVE, SS_PAUSED, or SS_UNSUBSCRIBED. **Optional**, defaults to "ACTIVE"
- **subscription_parameters** (SubscriptionParameters) – the parameters for this subscription. **Optional** If provided, should match the request.
- **push_parameters** (PushParameters) – the push parameters for this subscription. **Optional** If provided, should match the request.
- **poll_instances** (*list of PollInstance*) – The Poll Services that can be polled to fulfill this subscription. **Optional**

class libtaxii.messages_11.**PollInstance**(*poll_protocol*, *poll_address*, *poll_message_bindings=None*)

The Poll Instance component of the Manage Collection Subscription Response message.

Parameters

- **poll_protocol** (*str*) – The protocol binding supported by this instance of a Polling Service. **Required**
- **poll_address** (*str*) – the address of the TAXII Daemon hosting this Poll Service. **Required**
- **poll_message_bindings** (*list of str*) – one or more message bindings that can be used when interacting with this Poll Service instance. **Required**

Example:

```
subscription_parameters1 = tm11.SubscriptionParameters()
push_parameters1 = tm11.PushParameters("", "", "")


poll_instance1 = tm11.PollInstance(
        poll_protocol=VID_TAXII_HTTPS_10,
        poll_address='https://example.com/poll1/',
        poll_message_bindings=[VID_TAXII_XML_11])
```

```
subs1 = tm11.SubscriptionInstance(
        subscription_id='Subs001',
        status=SS_ACTIVE,
        subscription_parameters=subscription_parameters1,
        push_parameters=push_parameters1,
        poll_instances=[poll_instance1])

subs_resp1 = tm11.ManageCollectionSubscriptionResponse(
        message_id='SubsResp01',
        in_response_to='xyz',
        collection_name='abc123',
        message='Hullo!',
        subscription_instances=[subs1])
```

5.1.26 Poll Request

class libtaxii.messages_11.**PollRequest**(*message_id*, *extended_headers=None*, *collection_name=None*, *exclusive_begin_timestamp_label=None*, *inclusive_end_timestamp_label=None*, *subscription_id=None*, *poll_parameters=None*)

A TAXII Poll Request message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **collection_name** (*str*) – the name of the TAXII Data Collection that is being polled. **Required**
- **exclusive_begin_timestamp_label** (*datetime*) – a Timestamp Label indicating the beginning of the range of TAXII Data Feed content the requester wishes to receive. **Optional for a Data Feed, Prohibited for a Data Set**
- **inclusive_end_timestamp_label** (*datetime*) – a Timestamp Label indicating the end of the range of TAXII Data Feed content the requester wishes to receive. **Optional for a Data Feed, Probited for a Data Set**
- **subscription_id** (*str*) – the existing subscription the Consumer wishes to poll. **Optional**
- **poll_parameters** (*list of PollParameters objects*) – the poll parameters for this request. **Optional**

Exactly one of `subscription_id` and `poll_parameters` is **Required**.

class libtaxii.messages_11.**PollParameters**(*response_type='FULL'*, *content_bindings=None*, *query=None*, *allow_asynch=False*, *delivery_parameters=None*)

The Poll Parameters component of a TAXII Poll Request message.

Parameters

- **response_type** (*str*) – The requested response type. Must be either `RT_FULL` or `RT_COUNT_ONLY`. **Optional**, defaults to `RT_FULL`

- **content_bindings** (*list of ContentBinding objects*) – A list of Content Bindings acceptable in response. **Optional**
- **query** (*Query*) – The query for this poll parameters. **Optional**
- **allow_asynch** (*bool*) – Indicates whether the client supports asynchronous polling. **Optional**, defaults to `False`
- **delivery_parameters** (`libtaxii.messages_11.DeliveryParameters`) – The requested delivery parameters for this object. **Optional**

If `content_bindings` in not provided, this indicates that all bindings are accepted as a response.

Example:

```
delivery_parameters1 = tm11.DeliveryParameters(
        inbox_protocol=VID_TAXII_HTTPS_10,
        inbox_address='https://example.com/inboxAddress/',
        delivery_message_binding=VID_TAXII_XML_11)

poll_params1 = tm11.PollParameters(
        allow_asynch=False,
        response_type=RT_COUNT_ONLY,
        content_bindings=[tm11.ContentBinding(binding_id=CB_STIX_XML_11)],
        #query=query1,
        delivery_parameters=delivery_parameters1)

poll_req3 = tm11.PollRequest(
        message_id='PollReq03',
        collection_name='collection100',
        exclusive_begin_timestamp_label=datetime.datetime.now(tzutc()),
        inclusive_end_timestamp_label=datetime.datetime.now(tzutc()),
        poll_parameters=poll_params1)
```

5.1.27 Poll Response

class `libtaxii.messages_11.`**PollResponse**(*message_id*, *in_response_to*, *extended_headers=None*, *collection_name=None*, *exclusive_begin_timestamp_label=None*, *inclusive_end_timestamp_label=None*, *subscription_id=None*, *message=None*, *content_blocks=None*, *more=False*, *result_id=None*, *result_part_number=1*, *record_count=None*)

A TAXII Poll Response message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **in_response_to** (*str*) – Contains the Message ID of the message to which this is a response. **Optional**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **collection_name** (*str*) – the name of the TAXII Data Collection that was polled. **Required**

- **exclusive_begin_timestamp_label** (*datetime*) – a Timestamp Label indicating the beginning of the range this response covers. **Optional for a Data Feed, Prohibited for a Data Set**
- **inclusive_end_timestamp_label** (*datetime*) – a Timestamp Label indicating the end of the range this response covers. **Optional for a Data Feed, Prohibited for a Data Set**
- **subscription_id** (*str*) – the Subscription ID for which this content is being provided. **Optional**
- **message** (*str*) – additional information for the message recipient. **Optional**
- **content_blocks** (*list of ContentBlock*) – piece of content and additional information related to the content. **Optional**
- **more** (*bool*) – Whether there are more result parts. **Optional**, defaults to `False`
- **result_id** (*str*) – The ID of this result. **Optional**
- **result_part_number** (*int*) – The result part number of this response. **Optional**
- **record_count** (RecordCount) – The number of records and whether the count is a lower bound. **Optional**

Example:

```
cb1 = tm11.ContentBlock(CB_STIX_XML_11, "")
cb2 = tm11.ContentBlock(CB_STIX_XML_11, "")

count = tm11.RecordCount(record_count=22, partial_count=False)

poll_resp1 = tm11.PollResponse(
        message_id='PollResp1',
        in_response_to='tmp',
        collection_name='blah',
        exclusive_begin_timestamp_label=datetime.datetime.now(tzutc()),
        inclusive_end_timestamp_label=datetime.datetime.now(tzutc()),
        subscription_id='24',
        message='This is a test message',
        content_blocks=[cb1, cb2],
        more=True,
        result_id='123',
        result_part_number=1,
        record_count=count)
```

5.1.28 Inbox Message

class libtaxii.messages_11.**InboxMessage**(*message_id*, *in_response_to=None*, *extended_headers=None*, *message=None*, *result_id=None*, *destination_collection_names=None*, *subscription_information=None*, *record_count=None*, *content_blocks=None*)

A TAXII Inbox message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**

- **message** (*str*) – prose information for the message recipient. **Optional**
- **result_id** (*str*) – the result id. **Optional**
- **destination_collection_names** (*list of str*) – Each string indicates a destination collection name. **Optional**
- **subscription_information** (libtaxii.messages_11.SubscriptionInformation) – This field is only present if this message is being sent to provide content in accordance with an existing TAXII Data Collection subscription. **Optional**
- **record_count** (RecordCount) – The number of records and whether the count is a lower bound. **Optional**
- **content_blocks** (*list of ContentBlock*) – Inbox content. **Optional**

class libtaxii.messages_11.**SubscriptionInformation**(*collection_name*, *subscription_id*, *exclusive_begin_timestamp_label=None*, *inclusive_end_timestamp_label=None*)

The Subscription Information component of a TAXII Inbox message.

Parameters

- **collection_name** (*str*) – the name of the TAXII Data Collection from which this content is being provided. **Required**
- **subscription_id** (*str*) – the Subscription ID for which this content is being provided. **Required**
- **exclusive_begin_timestamp_label** (*datetime*) – a Timestamp Label indicating the beginning of the time range this Inbox Message covers. **Optional for a Data Feed, Prohibited for a Data Set**
- **inclusive_end_timestamp_label** (*datetime*) – a Timestamp Label indicating the end of the time range this Inbox Message covers. **Optional for a Data Feed, Prohibited for a Data Set**

Example:

```
cb1 = tm11.ContentBlock(CB_STIX_XML_11, "")
cb2 = tm11.ContentBlock(CB_STIX_XML_11, "")

subs_info1 = tm11.SubscriptionInformation(
        collection_name='SomeCollectionName',
        subscription_id='SubsId021',
        exclusive_begin_timestamp_label=datetime.datetime.now(tzutc()),
        inclusive_end_timestamp_label=datetime.datetime.now(tzutc()))

inbox1 = tm11.InboxMessage(
        message_id='Inbox1',
        result_id='123',
        destination_collection_names=['collection1','collection2'],
        message='Hello!',
        subscription_information=subs_info1,
        record_count=tm11.RecordCount(22, partial_count=True),
        content_blocks=[cb1, cb2])
```

5.1.29 Poll Fulfillment Request

class libtaxii.messages_11.**PollFulfillmentRequest**(*message_id*, *extended_headers=None*, *collection_name=None*, *result_id=None*, *result_part_number=None*)

A TAXII Poll Fulfillment Request message.

Parameters

- **message_id** (*str*) – A value identifying this message. **Required**
- **extended_headers** (*dict*) – A dictionary of name/value pairs for use as Extended Headers. **Optional**
- **collection_name** (*str*) – the name of the TAXII Data Collection to which the action applies. **Required**
- **result_id** (*str*) – The result id of the requested result. **Required**
- **result_part_number** (*int*) – The part number being requested. **Required**

Example:

```
pf1 = tm11.PollFulfillmentRequest(
        message_id='pf1',
        collection_name='1-800-collection',
        result_id='123',
        result_part_number=1)
```

5.1.30 Other Classes

class libtaxii.messages_11.**TAXIIMessage**(*message_id*, *in_response_to=None*, *extended_headers=None*)

Encapsulate properties common to all TAXII Messages (such as headers).

This class is extended by each Message Type (e.g., DiscoveryRequest), with each subclass containing subclass-specific information

class libtaxii.messages_11.**ContentBinding**(*binding_id*, *subtype_ids=None*)

TAXII Content Binding component

Parameters

- **binding_id** (*str*) – The content binding ID. **Required**
- **subtype_ids** (*list of str*) – the subtype IDs. **Required**

class libtaxii.messages_11.**ContentBlock**(*content_binding*, *content*, *timestamp_label=None*, *padding=None*, *message=None*)

A TAXII Content Block.

Parameters

- **content_binding** (ContentBinding) – a Content Binding ID or nesting expression indicating the type of content contained in the Content field of this Content Block. **Required**
- **content** (*string or etree*) – a piece of content of the type specified by the Content Binding. **Required**
- **timestamp_label** (*datetime*) – the Timestamp Label associated with this Content Block. **Optional**

- **padding** (*string*) – an arbitrary amount of padding for this Content Block. **Optional**
- **message** (*string*) – a message associated with this ContentBlock. **Optional**

Example:

```
cb001 = tm11.ContentBlock(
        content_binding=tm11.ContentBinding(CB_STIX_XML_11),
        content='<stix:STIX_Package xmlns:stix="http://stix.mitre.org/stix-1"/>',
        timestamp_label=datetime.datetime.now(tzutc()),
        message='Hullo!',
        padding='The quick brown fox jumped over the lazy dogs.')
```

class libtaxii.messages_11.**DeliveryParameters**(*inbox_protocol*, *inbox_address*, *delivery_message_binding*)

Set up Delivery Parameters.

Parameters

- **inbox_protocol** (*str*) – identifies the protocol to be used when pushing TAXII Data Collection content to a Consumer's TAXII Inbox Service implementation. **Required**
- **inbox_address** (*str*) – identifies the address of the TAXII Daemon hosting the Inbox Service to which the Consumer requests content for this TAXII Data Collection to be delivered. **Required**
- **delivery_message_binding** (*str*) – identifies the message binding to be used to send pushed content for this subscription. **Required**

class libtaxii.messages_11.**PushParameters**(*inbox_protocol*, *inbox_address*, *delivery_message_binding*)

Set up Push Parameters.

Parameters

- **inbox_protocol** (*str*) – identifies the protocol to be used when pushing TAXII Data Collection content to a Consumer's TAXII Inbox Service implementation. **Required**
- **inbox_address** (*str*) – identifies the address of the TAXII Daemon hosting the Inbox Service to which the Consumer requests content for this TAXII Data Collection to be delivered. **Required**
- **delivery_message_binding** (*str*) – identifies the message binding to be used to send pushed content for this subscription. **Required**

class libtaxii.messages_11.**RecordCount**(*record_count*, *partial_count=False*)

Information summarizing the number of records.

Parameters

- **record_count** (*int*) – The number of records
- **partial_count** (*bool*) – Whether the number of records is a partial count

class libtaxii.messages_11.**SubscriptionParameters**(*response_type='FULL'*, *content_bindings=None*, *query=None*)

TAXII Subscription Parameters.

Parameters

- **response_type** (*str*) – The requested response type. Must be either `RT_FULL` or `RT_COUNT_ONLY`. **Optional**, defaults to `RT_FULL`
- **content_bindings** (*list of ContentBinding objects*) – A list of Content Bindings acceptable in response. **Optional**

- **query** (*Query*) – The query for this poll parameters. **Optional**

5.1.31 Functions

libtaxii.messages_11.**generate_message_id**(*maxlen=5*, *version='urn:taxii.mitre.org:services:1.0'*)

Generate a TAXII Message ID.

Parameters **maxlen** (*int*) – maximum length of the ID, in characters

Example

```
msg_id = tm11.generate_message_id()
message = tm11.DiscoveryRequest(msg_id)
# Or...
message = tm11.DiscoveryRequest(tm11.generate_message_id())
```

libtaxii.messages_11.**validate_xml**(*xml_string*)

Note that this function has been deprecated. Please see libtaxii.validators.SchemaValidator.

Validate XML with the TAXII XML Schema 1.1.

Parameters **xml_string** (*str*) – The XML to validate.

Example

```
is_valid = tm11.validate_xml(message.to_xml())
```

libtaxii.messages_11.**get_message_from_xml**(*xml_string*, *encoding='utf_8'*)

Create a TAXIIMessage object from an XML string.

This function automatically detects which type of Message should be created based on the XML.

Parameters

- **xml_string** (*str*) – The XML to parse into a TAXII message.
- **encoding** (*str*) – The encoding of the string; defaults to UTF-8

Example

```
message_xml = message.to_xml()
new_message = tm11.get_message_from_xml(message_xml)
```

libtaxii.messages_11.**get_message_from_dict**(*d*)

Create a TAXIIMessage object from a dictonary.

This function automatically detects which type of Message should be created based on the 'message_type' key in the dictionary.

Parameters **d** (*dict*) – The dictionary to build the TAXII message from.

Example

```
message_dict = message.to_dict()
new_message = tm11.get_message_from_dict(message_dict)
```

libtaxii.messages_11.**get_message_from_json**(*json_string*, *encoding='utf_8'*)
Create a TAXIIMessage object from a JSON string.

This function automatically detects which type of Message should be created based on the JSON.

Parameters **json_string** (*str*) – The JSON to parse into a TAXII message.

libtaxii.taxii_default_query Module

Version: 1.1.115

Creating, handling, and parsing TAXII Default Queries.

5.1.32 Classes

Default Query

class libtaxii.taxii_default_query.**DefaultQuery**(*targeting_expression_id*, *criteria*)
Bases: libtaxii.messages_11.Query

Conveys a TAXII Default Query.

Parameters

- **targeting_expression_id** (*string*) – The targeting_expression used in the query
- **criteria** (*DefaultQuery.Criteria*) – The criteria of the query

class **Criteria**(*operator*, *criteria=None*, *criterion=None*)
Bases: *libtaxii.common.TAXIIBase*

Represents criteria for a *DefaultQuery*. **Note**: At least one criterion OR criteria MUST be present

Parameters

- **operator** (*str*) – The logical operator (should be one of *OP_AND* or *OP_OR*)
- **criteria** (*DefaultQuery.Criteria*) – The criteria for the query
- **criterion** (*DefaultQuery.Criterion*) – The criterion for the query

class **Criterion**(*target*, *test*, *negate=False*)
Bases: *libtaxii.common.TAXIIBase*

Represents criterion for a *DefaultQuery.Criteria*

Parameters

- **target** (*string*) – A targeting expression identifying the target
- **test** (*DefaultQuery.Criterion.Test*) – The test to be applied to the target
- **negate** (*bool*) – Whether the result of applying the test to the target should be negated

class **Test**(*capability_id*, *relationship*, *parameters=None*)
Bases: *libtaxii.common.TAXIIBase*
Parameters

- **capability_id** (*string*) – The ID of the capability module that defines the relationship & parameters
- **relationship** (*string*) – The relationship (e.g., equals)
- **parameters** (dict of key/value pairs) – The parameters for the relationship.

Example

```
import libtaxii.taxii_default_query as tdq
from libtaxii.taxii_default_query import Test
from libtaxii.taxii_default_query import Criterion
from libtaxii.taxii_default_query import Criteria
from libtaxii.constants import *
import datetime

################################################################################
# A Taxii Default Query *Test* gives the consumer granular control over the
# Target of a Query by applying unambiguos relationship requirements specified
# using a standardized vocabulary.

# Each Relationship (e.g. equals, matches, greater_than, etc.) in a Capability
# Module defines a set of paramater fields, capable of expressing that
# relation.

# The *equals* relationship, of the Core Capability Module,  returns True if
# the target matches the value exactly. If the target merely contains the
# value (but does not match exactly) the relationship Test returns False.
test_equals = Test(capability_id=CM_CORE,
                   relationship='equals',
                   parameters={'value': 'Test value',
                               'match_type': 'case_sensitive_string'})

# The *matches* relationship, in the context of the Regular Expression
# Capability Module, returns true if the target matches the regular expression
# contained in the value.
test_matches = Test(capability_id=CM_REGEX,
                    relationship='matches',
                    parameters={'value': '[A-Z]*',
                                'case_sensitive': True})

# The *greater than* relationship, in the context of the Timestamp Capability
# Module returns True if the target's timestamp indicates a later time than
# that specified by this value. This relationship is only valid for timestamp
# comparisons.
test_timestamp = Test(capability_id=CM_TIMESTAMP,
                      relationship='greater_than',
                      parameters={'value': datetime.datetime.now()})


################################################################################
# A *Criterion* specifies how a Target is evaluated against a Test. Within a
# Criterion, the Target is used to identify a specific region of a record to
# which the Test should be applied. Slash Notation Targeting Expression syntax,
# in conjunction with a Targeting Expression Vocabulary, are used to form a
# Targeting Expression

# A Multi-field Wildcard (**). This indicates any Node or series of Nodes,
# specified by double asterisks.
criterion1 = Criterion(target='**',
                       test=test_equals)
```

```
# Indicates that *id* fields in the STIX Indicator construct are in scope
criterion2 = Criterion(target='STIX_Package/Indicators/Indicator/@id',
                       test=test_matches)

# Indicates that all STIX Description fields are in scope
criterion3 = Criterion(target='**/Description',
                       test=test_timestamp)


################################################################################
# *Criteria* consist of a logical operator (and/or) that should be applied to
# child Criteria and Criterion to determine whether content matches this query.

criteria1 = Criteria(operator=OP_AND,
                     criterion=[criterion1])

criteria2 = Criteria(operator=OP_OR,
                     criterion=[criterion1, criterion2, criterion3])

criteria3 = Criteria(operator=OP_AND,
                     criterion=[criterion1, criterion3],
                     criteria=[criteria2])


################################################################################
# query1, query2 and query3 would be able to be used in TAXII requests that
# contain queries (e.g., PollRequest Messages)
query1 = tdq.DefaultQuery(targeting_expression_id=CB_STIX_XML_111,
                          criteria=criteria1)

query2 = tdq.DefaultQuery(targeting_expression_id=CB_STIX_XML_111,
                          criteria=criteria3)

query3 = tdq.DefaultQuery(targeting_expression_id=CB_STIX_XML_111,
                          criteria=criteria2)
```

Default Query Info

class libtaxii.taxii_default_query.**DefaultQueryInfo**(*targeting_expression_infos*, *capability_modules*)

Bases: libtaxii.messages_11.SupportedQuery

Used to describe the TAXII Default Queries that are supported.

Parameters

- **targeting_expression_infos** (list of *TargetingExpressionInfo* objects) – Describe the supported targeting expressions
- **capability_modules** (list of str) – Indicate the supported capability modules

class **TargetingExpressionInfo**(*targeting_expression_id*, *preferred_scope=None*, *allowed_scope=None*)

Bases: *libtaxii.common.TAXIIBase*

This class describes supported Targeting Expressions

Parameters

- **targeting_expression_id** (*string*) – The supported targeting expression ID
- **preferred_scope** (list of string) – Indicates the preferred scope of queries
- **allowed_scope** (list of string) – Indicates the allowed scope of queries

Example

```
import libtaxii.taxii_default_query as tdq
from libtaxii.taxii_default_query import TargetingExpressionInfo
from libtaxii.constants import *

################################################################################
# *TargetingExpressionInfo* describes which expressions are available to
# a consumer when submitting a query to a taxii service. A
# `targetting_expression_id` indicates a suppoted targetting vocabulary
# TargetingExpressionInfo also contains the permissible scope of queries.

# This example has no preferred scope, and allows any scope
tei_01 = TargetingExpressionInfo(
    targeting_expression_id=CB_STIX_XML_111,
    preferred_scope=[],
    allowed_scope=['**'])

# This example prefers the Indicator scope and allows no other scope
tei_02 = TargetingExpressionInfo(
    targeting_expression_id=CB_STIX_XML_111,
    preferred_scope=['STIX_Package/Indicators/Indicator/**'],
    allowed_scope=[])


################################################################################
# *DefaultQueryInfo* describes the TAXII Default Queries that are supported
# using a list of TargetExpressionInfo objects, and a list of capability
# module identifiers.
tdqi1 = tdq.DefaultQueryInfo(
    targeting_expression_infos=[tei_01, tei_02],
    capability_modules=[CM_CORE])
```

libtaxii.validation Module

Version: 1.1.115

5.1.33 Validate TAXII Content

class libtaxii.validation.**SchemaValidator**(*schema_file*)

A helper class for TAXII Schema Validation.

Example

See validate_etree(...) for an example how to use this class

validate_etree(*etree_xml*)

Validate an LXML etree with the specified schema_file.

Parameters

- **etree_xml** (*etree*) – The XML to validate.
- **schema_file** (*str*) – The schema file to validate against

Returns A SchemaValidationResult object

Raises `lxml.etree.XMLSyntaxError` – When the XML to be validated is not well formed

Example

```
from libtaxii import messages_11
from libtaxii.validation import SchemaValidator, TAXII_11_SCHEMA
from lxml.etree import XMLSyntaxError

sv = SchemaValidator(TAXII_11_SCHEMA)

try:
   result = sv.validate_etree(some_etree)
   # Note that validate_string() and validate_file() can also be used
except XMLSyntaxError:
    # Handle this exception, which occurs when
    # some_xml_string is not valid XML (e.g., 'foo')

if not result.valid:
    for error in result.error_log:
        print error
    sys.exit(1)

# At this point, the XML is schema valid
do_something(some_xml_string)
```

validate_file(*file_location*)
A wrapper for validate_etree. Parses file_location, turns it into an etree, then calls validate_etree(...)

validate_string(*xml_string*)
A wrapper for validate_etree. Parses xml_string, turns it into an etree, then calls validate_etree(...)

`libtaxii.validation.`**TAXII_10_SCHEMA** **Use TAXII 1.0 schema for validation.**
Automatically-calculated path to the bundled TAXII 1.0 schema.

`libtaxii.validation.`**TAXII_11_SCHEMA** **Use TAXII 1.1 schema for validation.**
Automatically-calculated path to the bundled TAXII 1.1 schema.

class `libtaxii.validation.`**TAXII10Validator**
Bases: *libtaxii.validation.SchemaValidator*

A *SchemaValidator* that uses the TAXII 1.0 Schemas

class `libtaxii.validation.`**TAXII11Validator**
Bases: *libtaxii.validation.SchemaValidator*

A *SchemaValidator* that uses the TAXII 1.1 Schemas

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

CHAPTER 7

Offline Documentation

To download the latest documentation for offline use, please use one of these links:

- PDF
- HTML

Python Module Index

l

Index

H

I

L

M

N

O

P

R

S

T

V