PyIntelOwl

Release 4.4.3

Matteo Lodi

Apr 18, 2023

USAGE

Robust Python **SDK** and **Command Line Client** for interacting with IntelOwl API.

CHAPTER

ONE

INSTALLATION

```
$ pip install pyintelowl
```

CHAPTER

TWO

USAGE AS CLI

On successful installation, The `pyintelowl` entryscript should be directly invokable. For example,

```
$ pyintelowl
Usage: pyintelowl [OPTIONS] COMMAND [ARGS]...

Options:
-d, --debug  Set log level to DEBUG
--version    Show the version and exit.
-h, --help   Show this message and exit.

Commands:
analyse                Send new analysis request
analyzer-healthcheck   Send healthcheck request for an analyzer...
config                 Set or view config variables
connector-healthcheck  Send healthcheck request for a connector
get-analyzer-config    Get current state of `analyzer_config.json` from...
get-connector-config   Get current state of `connector_config.json` from...
get-playbook-config    Get current state of `playbook_config.json` from...
jobs                   Manage Jobs
tags                   Manage tags
```

Configuration:

You can use `set` to set the config variables and `get` to view them.

Listing 1: View on asciinema

```
$ pyintelowl config set -k 4bf03f20add626e7138f4023e4cf52b8 -u "http://localhost:80"
$ pyintelowl config get
```

Hint: The CLI would is well-documented which will help you navigate various commands easily. Invoke `pyintelowl -h` or `pyintelowl <command> -h` to get help.

CHAPTER

THREE

USAGE AS SDK/LIBRARY

```python
from pyintelowl import IntelOwl, IntelOwlClientException
obj = IntelOwl(
    "4bf03f20add626e7138f4023e4cf52b8",
    "http://localhost:80",
    None,
)
"""
obj = IntelOwl(
    "<your_api_key>",
    "<your_intelowl_instance_url>",
    "optional<path_to_pem_file>"
    "optional<proxies>"
)
"""

try:
    ans = obj.get_analyzer_configs()
    print(ans)
except IntelOwlClientException as e:
    print("Oh no! Error: ", e)
```

Tip: We very much **recommend** going through the *pyintelowl.pyintelowl.IntelOwl* docs.

CHAPTER

FOUR

INDEX

4.1 Modules

4.1.1 IntelOwl class

class pyintelowl.pyintelowl.**IntelOwl**(*token: str*, *instance_url: str*, *certificate: Optional[str] = None*, *proxies: Optional[dict] = None*, *logger: Optional[Logger] = None*, *cli: bool = False*)

Bases: `object`

analyzer_healthcheck(*analyzer_name: str*) → Optional[bool]

Send analyzer(docker-based) health check request.

Method: GET Endpoint: `/api/analyzer/{analyzer_name}/healthcheck`

Parameters
analyzer_name (*str*) – name of analyzer

Raises
IntelOwlClientException – on client/HTTP error

Returns
success or not

Return type
Bool

ask_analysis_availability(*md5: str*, *analyzers: Optional[List[str]] = None*, *check_reported_analysis_too: bool = False*, *minutes_ago: Optional[int] = None*) → Dict

Search for already available analysis.

Endpoint: `/api/ask_analysis_availability`

Parameters

- **md5** (*str*) – md5sum of the observable or file
- **analyzers** (*List[str], optional*) –
- **trigger.** (*list of analyzers to*) –
- **analyzers.** (*Defaults to None meaning automatically select all configured*) –
- **check_reported_analysis_too** (*bool, optional*) –

- **False.** (*Check against all existing jobs. Defaults to*) –
- **minutes_ago** (*int, optional*) –
- **analysis.** (*number of minutes to check back for*) –
- **limits.** (*Default is None so the check does not have any time*) –

Raises
IntelOwlClientException – on client/HTTP error

Returns
JSON body

Return type
Dict

connector_healthcheck(*connector_name: str*) → Optional[bool]

Send connector health check request.

Method: GET Endpoint: `/api/connector/{connector_name}/healthcheck`

Parameters
connector_name (*str*) – name of connector

Raises
IntelOwlClientException – on client/HTTP error

Returns
success or not

Return type
Bool

create_tag(*label: str*, *color: str*)

Creates new tag by sending a POST Request Endpoint: `/api/tags`

Parameters

- **label** (*[str]*) – [Label of the tag to be created]
- **color** (*[str]*) – [Color of the tag to be created]

delete_job_by_id(*job_id: int*) → bool

Send delete job request.

Method: DELETE Endpoint: `/api/jobs/{job_id}`

Parameters
job_id (*int*) – id of job to kill

Raises
IntelOwlClientException – on client/HTTP error

Returns
deleted or not

Return type
Bool

delete_tag_by_id(*tag_id: int*) → bool

Send delete tag request.

Method: DELETE Endpoint: `/api/tags/{tag_id}`

Parameters
tag_id (*int*) – id of tag to delete

Raises
IntelOwlClientException – on client/HTTP error

Returns
deleted or not

Return type
Bool

download_sample(*job_id: int*) → bytes

Download file sample from job.

Method: GET Endpoint: `/api/jobs/{job_id}/download_sample`

Parameters
job_id (*int*) – id of job to download sample from

Raises
IntelOwlClientException – on client/HTTP error

Returns
Raw file data.

Return type
Bytes

edit_tag(*tag_id: Union[int, str]*, *label: str*, *color: str*)

Edits existing tag by sending PUT request Endpoint: `api/tags`

Parameters

- **id** (*[int]*) – [Id of the existing tag]
- **label** (*[str]*) – [Label of the tag to be created]
- **color** (*[str]*) – [Color of the tag to be created]

get_all_jobs() → List[Dict[str, Any]]

Fetch list of all jobs.

Endpoint: `/api/jobs`

Raises
IntelOwlClientException – on client/HTTP error

Returns
Dict with 3 keys: “count”, “total_pages”, “results”

Return type
Dict

get_all_tags() → List[Dict[str, str]]

Fetch list of all tags.

Endpoint: `/api/tags`

Raises
IntelOwlClientException – on client/HTTP error

Returns
List of tags

Return type
List[Dict[str, str]]

`get_analyzer_configs()`

Get current state of *analyzer_config.json* from the IntelOwl instance.

Endpoint: `/api/get_analyzer_configs`

`get_connector_configs()`

Get current state of *connector_config.json* from the IntelOwl instance.

Endpoint: `/api/get_connector_configs`

`get_job_by_id`(*job_id: Union[int, str]*) → Dict[str, Any]

Fetch job info by ID. Endpoint: `/api/jobs/{job_id}`

Parameters
job_id (*Union[int, str]*) – Job ID

Raises
IntelOwlClientException – on client/HTTP error

Returns
JSON body.

Return type
Dict[str, Any]

`static get_md5`(*to_hash: AnyStr*, *type_='observable'*) → str

Returns md5sum of given observable or file object.

Parameters

- **to_hash** (*AnyStr*) – either an observable string, file contents as bytes or path to a file
- **type** (*Union["observable", "binary", "file"], optional*) – *observable*, *binary*, *file*. Defaults to "observable".

Raises
IntelOwlClientException – on client/HTTP error

Returns
md5sum

Return type
str

`get_playbook_configs()`

Get current state of *playbook_config.json* from the IntelOwl instance.

Endpoint: `/api/get_playbook_configs`

`get_tag_by_id`(*tag_id: Union[int, str]*) → Dict[str, str]

Fetch tag info by ID.

Endpoint: `/api/tag/{tag_id}`

Parameters
tag_id (*Union[int, str]*) – Tag ID

Raises
IntelOwlClientException – on client/HTTP error

Returns
Dict with 3 keys: *id*, *label* and *color*.

Return type
Dict[str, str]

kill_analyzer(*job_id: int*, *analyzer_name: str*) → bool

Send kill running/pending analyzer request.

Method: PATCH Endpoint: `/api/jobs/{job_id}/analyzer/{analyzer_name}/kill`

Parameters

- **job_id** (*int*) – id of job
- **analyzer_name** (*str*) – name of analyzer to kill

Raises
IntelOwlClientException – on client/HTTP error

Returns
killed or not

Return type
Bool

kill_connector(*job_id: int*, *connector_name: str*) → bool

Send kill running/pending connector request.

Method: PATCH Endpoint: `/api/jobs/{job_id}/connector/{connector_name}/kill`

Parameters

- **job_id** (*int*) – id of job
- **connector_name** (*str*) – name of connector to kill

Raises
IntelOwlClientException – on client/HTTP error

Returns
killed or not

Return type
Bool

kill_running_job(*job_id: int*) → bool

Send kill_running_job request.

Method: PATCH Endpoint: `/api/jobs/{job_id}/kill`

Parameters
job_id (*int*) – id of job to kill

Raises
IntelOwlClientException – on client/HTTP error

Returns
killed or not

Return type
Bool

logger: Logger

retry_analyzer(*job_id: int*, *analyzer_name: str*) → bool

Send retry failed/killed analyzer request.

Method: PATCH Endpoint: `/api/jobs/{job_id}/analyzer/{analyzer_name}/retry`

Parameters

- **job_id** (*int*) – id of job
- **analyzer_name** (*str*) – name of analyzer to retry

Raises
IntelOwlClientException – on client/HTTP error

Returns
success or not

Return type
Bool

retry_connector(*job_id: int*, *connector_name: str*) → bool

Send retry failed/killed connector request.

Method: PATCH Endpoint: `/api/jobs/{job_id}/connector/{connector_name}/retry`

Parameters

- **job_id** (*int*) – id of job
- **connector_name** (*str*) – name of connector to retry

Raises
IntelOwlClientException – on client/HTTP error

Returns
success or not

Return type
Bool

send_analysis_batch(*rows: List[Dict]*)

Send multiple analysis requests. Can be mix of observable or file analysis requests.

Used by the pyintelowl CLI.

Parameters
rows (*List[Dict]*) – Each row should be a dictionary with keys, *value*, *type*, *check*, *tlp*, *analyzers_list*, *connectors_list*, *runtime_config tags_list*.

send_file_analysis_playbook_request(*filename: str*, *binary: bytes*, *tlp: Optional[typing_extensions.Literal[WHITE, GREEN, AMBER, RED, CLEAR]] = None*, *playbooks_requested: Optional[List[str]] = None*, *runtime_configuration: Optional[Dict] = None*, *tags_labels: Optional[List[str]] = None*) → Dict

Send playbook analysis request for a file.

Endpoint: `/api/playbook/analyze_multiple_files`

Parameters

- **filename** (*str*) – Filename
- **binary** (*bytes*) – File contents as bytes

- **playbooks_requested** (*List[str], optional*) – List of specific playbooks to invoke. Defaults to `[]` i.e. all playbooks.
- **tlp** (*str, optional*) – TLP for the analysis. (options: `WHITE, GREEN, AMBER, RED`). Defaults to `WHITE`.
- **runtime_configuration** (*Dict, optional*) – Overwrite configuration for analyzers. Defaults to `{}`.
- **tags_labels** (*List[str], optional*) – List of tag labels to assign (creates non-existing tags)

Raises
IntelOwlClientException – on client/HTTP error

Returns
JSON body

Return type
Dict

send_file_analysis_request(*filename: str*, *binary: bytes*, *tlp: Optional[typing_extensions.Literal[WHITE, GREEN, AMBER, RED, CLEAR]] = None*, *analyzers_requested: Optional[List[str]] = None*, *connectors_requested: Optional[List[str]] = None*, *runtime_configuration: Optional[Dict] = None*, *tags_labels: Optional[List[str]] = None*) → Dict

Send analysis request for a file.

Endpoint: `/api/analyze_file`

Parameters

- **filename** (*str*) – Filename
- **binary** (*bytes*) – File contents as bytes
- **analyzers_requested** (*List[str], optional*) – List of analyzers to invoke Defaults to `[]` i.e. all analyzers.
- **connectors_requested** (*List[str], optional*) – List of specific connectors to invoke. Defaults to `[]` i.e. all connectors.
- **tlp** (*str, optional*) – TLP for the analysis. (options: `WHITE, GREEN, AMBER, RED`). Defaults to `WHITE`.
- **runtime_configuration** (*Dict, optional*) – Overwrite configuration for analyzers. Defaults to `{}`.
- **tags_labels** (*List[str], optional*) – List of tag labels to assign (creates non-existing tags)

Raises
IntelOwlClientException – on client/HTTP error

Returns
JSON body

Return type
Dict

send_observable_analysis_playbook_request(*observable_name: str*, *tlp: Optional[typing_extensions.Literal[WHITE, GREEN, AMBER, RED, CLEAR]] = None*, *playbooks_requested: Optional[List[str]] = None*, *runtime_configuration: Optional[Dict] = None*, *tags_labels: Optional[List[str]] = None*, *observable_classification: Optional[str] = None*) → Dict

Send playbook analysis request for an observable.

Endpoint: `/api/playbook/analyze_multiple_observables`

Parameters

- **observable_name** (*str*) – Observable value
- **playbooks_requested** (*List[str], optional*) – List of specific playbooks to invoke. Defaults to `[]` i.e. all playbooks.
- **tlp** (*str, optional*) – TLP for the analysis. (options: `WHITE, GREEN, AMBER, RED`). Defaults to `WHITE`.
- **runtime_configuration** (*Dict, optional*) – Overwrite configuration for analyzers. Defaults to `{}`.
- **tags_labels** (*List[str], optional*) – List of tag labels to assign (creates non-existing tags)
- **observable_classification** (*str*) – Observable classification, Default to None. By default launch analysis with an automatic classification. (options: `url, domain, hash, ip, generic`)

Raises

- *IntelOwlClientException* – on client/HTTP error
- *IntelOwlClientException* – on wrong observable_classification

Returns
JSON body

Return type
Dict

send_observable_analysis_request(*observable_name: str*, *tlp: Optional[typing_extensions.Literal[WHITE, GREEN, AMBER, RED, CLEAR]] = None*, *analyzers_requested: Optional[List[str]] = None*, *connectors_requested: Optional[List[str]] = None*, *runtime_configuration: Optional[Dict] = None*, *tags_labels: Optional[List[str]] = None*, *observable_classification: Optional[str] = None*) → Dict

Send analysis request for an observable.

Endpoint: `/api/analyze_observable`

Parameters

- **observable_name** (*str*) – Observable value
- **analyzers_requested** (*List[str], optional*) – List of analyzers to invoke Defaults to `[]` i.e. all analyzers.
- **connectors_requested** (*List[str], optional*) – List of specific connectors to invoke. Defaults to `[]` i.e. all connectors.

- **tlp** (*str, optional*) – TLP for the analysis. (options: `WHITE, GREEN, AMBER, RED`). Defaults to `WHITE`.
- **runtime_configuration** (*Dict, optional*) – Overwrite configuration for analyzers. Defaults to `{}`.
- **tags_labels** (*List[str], optional*) – List of tag labels to assign (creates non-existing tags)
- **observable_classification** (*str*) – Observable classification, Default to None. By default launch analysis with an automatic classification. (options: `url, domain, hash, ip, generic`)

Raises

- *IntelOwlClientException* – on client/HTTP error
- *IntelOwlClientException* – on wrong observable_classification

Returns
JSON body

Return type
Dict

property session: Session

Internal use only.

4.1.2 IntelOwlClientException class

class pyintelowl.exceptions.**IntelOwlClientException**(**args*, ***kwargs*)

Bases: `RequestException`

property error_detail: Union[Dict, AnyStr]

4.2 Tests

4.2.1 Configuration

Some tests require file samples, which can be found in the encrypted folder `tests/test_files.zip` (password: "infected"). Unzip the archive in `tests/test_files` folder before running the tests.

Please remember that these are dangerous malware! They come encrypted and locked for a reason! Do NOT run them unless you are absolutely sure of what you are doing! They are to be used only for launching specific tests that require them (`__send_analysis_request`)

- With the following constants in `__init__.py`, you can customize your tests:
  - **MOCKING_CONNECTIONS:** Mock connections to external API to test functions without a real connection or a valid API Key.
- If you prefer to use custom inputs for tests, you can change the following constants:
  - **TEST_JOB_ID**
  - **TEST_HASH**
  - **TEST_URL**

- **TEST_IP**
- **TEST_DOMAIN**
- **TEST_GENERIC**
- **TEST_FILE**
- **TEST_FILE_HASH**

4.2.2 Launch Tests

- The test requirements are specified in the `test-requirements.txt` file. Install them using,

```
$ pip3 install -r test-requirements.txt
```

- Launch the tests using `tox`:

```
$ tox
```

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex

INDEX