pyTenable Documentation

Release 26.6.1

Steven McGrath

Jun 10, 2026

CONTENTS

class TenableIO(*access_key: str | None = None*, *secret_key: str | None = None*, ***kwargs*)

The Tenable Vulnerability Management object is the primary interaction point for users to interface with Tenable Vulnerability Management via the pyTenable library. All the API endpoint classes that have been written will be grafted onto this class.

Parameters

- **access_key** (*str, optional*) – The user's API access key for Tenable Vulnerability Management. If an access key isn't specified, then the library will attempt to read the environment variable `TIO_ACCESS_KEY` to acquire the key.
- **secret_key** (*str, optional*) – The user's API secret key for Tenable Vulnerability Management. If a secret key isn't specified, then the library will attempt to read the environment variable `TIO_SECRET_KEY` to acquire the key.
- **url** (*str, optional*) – The base URL that the paths will be appended onto. The default is `https://cloud.tenable.com`
- **retries** (*int, optional*) – The number of retries to make before failing a request. The default is 5.
- **backoff** (*float, optional*) – If a 429 response is returned, how much do we want to backoff if the response didn't send a Retry-After header. The default backoff is 1 second.
- **vendor** (*str, optional*) – The vendor name for the User-Agent string.
- **product** (*str, optional*) – The product name for the User-Agent string.
- **build** (*str, optional*) – The version or build identifier for the User-Agent string.
- **timeout** (*int, optional*) – The connection timeout parameter informing the library how long to wait in seconds for a stalled response before terminating the connection. If unspecified, the default is 120 seconds.

Examples

Basic Example:

```
>>> from tenable.io import TenableIO
>>> tio = TenableIO('ACCESS_KEY', 'SECRET_KEY')
```

Example with proper identification:

```
>>> tio = TenableIO('ACCESS_KEY', 'SECRET_KEY',
>>>     vendor='Company Name',
>>>     product='My Awesome Widget',
>>>     build='1.0.0')
```

Example with proper identification leveraging environment variables for access and secret keys:

```
>>> tio = TenableIO(
>>>     vendor='Company Name', product='Widget', build='1.0.0')
```

property access_control

The interface object for the *Tenable Vulnerability Management Access Control APIs*.

property agent_config

The interface object for the *Tenable Vulnerability Management Agent Config APIs*.

property agent_exclusions

The interface object for the *Tenable Vulnerability Management Agent Exclusions APIs*.

property agent_groups

The interface object for the *Tenable Vulnerability Management Agent Groups APIs*.

property agents

The interface object for the *Tenable Vulnerability Management Agents APIs*.

property assets

The interface object for the *Tenable Vulnerability Management assets APIs*.

property audit_log

The interface object for the *Tenable Vulnerability Management Audit Log APIs*.

property credentials

The interface object for the *Tenable Vulnerability Management Credentials APIs*.

property cs

The interface object for the *Tenable Vulnerability Management Container Security APIs*.

property editor

The interface object for the *Tenable Vulnerability Management Editor APIs*.

property exclusions

The interface object for the *Tenable Vulnerability Management Exclusions APIs*.

property exports

The interface object for the *Tenable Vulnerability Management Exports APIs*.

property files

The interface object for the *Tenable Vulnerability Management Files APIs*.

property filters

The interface object for the *Tenable Vulnerability Management Filters APIs*.

property folders

The interface object for the *Tenable Vulnerability Management Folders APIs*.

property groups

The interface object for the *Tenable Vulnerability Management Groups APIs*.

property networks

The interface object for the *Tenable Vulnerability Management Networks APIs*.

property pci: *PCIASVAPI*

The interface for *TVM PCI-ACV APIs*.

property permissions

The interface object for the *Tenable Vulnerability Management Permissions APIs*.

property plugins

The interface object for the *Tenable Vulnerability Management Plugins APIs*.

property policies

The interface object for the *Tenable Vulnerability Management Policies APIs*.

property remediationscans

The interface object for the *Tenable Vulnerability Management Remediation Scans APIs*.

property scanner_groups

The interface object for the *Tenable Vulnerability Management Scanner Groups APIs*.

property scanners

The interface object for the *Tenable Vulnerability Management Scanners APIs*.

property scans

The interface object for the *Tenable Vulnerability Management Scans APIs*.

property server

The interface object for the *Tenable Vulnerability Management Server APIs*.

property session

The interface object for the *Tenable Vulnerability Management Session APIs*.

property tags

The interface object for the *Tenable Vulnerability Management Tags APIs*.

property users

The interface object for the *Tenable Vulnerability Management Users APIs*.

property was

The interface object for the *Tenable Vulnerability Management WAS APIs*.

property workbenches

The interface object for the *Tenable Vulnerability Management Workbenches APIs*.

CHAPTER

ONE

CONTAINER SECURITY

The following sub-package allows for interaction with the Tenable Vulnerability Management Container Security APIs.

class ContainerSecurity(*api: APISession*)

property images

The interface object for the *Tenable Vulnerability Management Container Security Images APIs*.

property reports

The interface object for the *Tenable Vulnerability Management Container Security Reports APIs*.

property repositories

The interface object for the *Tenable Vulnerability Management Container Security Repository APIs*.

1.1 Images

The following methods allow for interaction into the Tenable Vulnerability Management Container Security images API endpoints.

Methods available on `tio.cs.images`:

class ImagesAPI(*api: APISession*)

delete(*repository: str*, *image: str*, *tag: str*) → None

Deleted the specified image.

API Documentation

Parameters

- **repository** (*str*) – The repository name.
- **image** (*str*) – The image name.
- **tag** (*str*) – The tag name.

Examples

```
>>> tio.cs.images.delete('centos', '7', 'latest')
```

details(*repository: str*, *image: str*, *tag: str*) → Dict

Returns the details for the specified image.

API Documentation # noqa: E501

Parameters

- **repository** (*str*) – The repository name.
- **image** (*str*) – The image name.
- **tag** (*str*) – The tag name.

Examples

```
>>> tio.cs.images.details('centos', '7', 'latest')
```

list(*name: str | None = None*, *repo: str | None = None*, *tag: str | None = None*, *has_malware: bool | None = None*, *score: int | None = None*, *score_operator: Literal['EQ', 'GT', 'LT'] | None = None*, *os: str | None = None*, *offset: int = 0*, *limit: int = 1000*, *return_json: bool = False*) → Dict | CSIterator

Returns the list of images stored within Container Security.

API Documentation

Parameters

- **name** (*str, optional*) – Image name to filter on. Filter is case-sensitive and enforces an exact match.
- **repo** (*str, optional*) – Repository name to filter on. Filter is case-sensitive and enforces an exact match.
- **tag** (*str, optional*) – Tag to filter on. Filter is case-sensitive and enforces an exact match.
- **has_malware** (*bool, optional*) – Specifies whether to return only images with malware associated to them.
- **score** (*int, optional*) – The score value to filter on.
- **score_operator** (*str, optional*) – The score operator to use with the score value. Supported operations are `EQ` (equal), `GT` (greater-than), and `LT` (less-than).
- **os** (*str, optional*) – The operating system to filter on. Filter is case-sensitive and enforces an exact match.
- **offset** (*int, optional*) – The number of records to skip before starting to return data.
- **limit** (*int, optional*) – The number of records to return for each page of data.
- **return_json** (*bool, optional*) – If set, then the response will instead be a Dict object instead of an iterable.

Examples

Using the default iterable:

```
>>> for image in tio.cs.images.list():
...     print(image)
```

Getting the raw JSON response:

```
>>> resp = tio.cs.images.list(return_json=True)
>>> for item in resp['items']:
...     print(item)
```

1.2 Reports

The following methods allow for interaction into the Tenable Vulnerability Management Container Security report API endpoints.

Methods available on `tio.cs.reports`:

class ReportsAPI(*api: APISession*)

report(*repository: str*, *image: str*, *tag: str*) → Dict

Returns the report for the specified image.

API Documentation

Parameters

- **repository** (*str*) – The repository name.
- **image** (*str*) – The image name.
- **tag** (*str*) – The tag name.

Examples

```
>>> tio.cs.reports.report('centos', '7', 'latest')
```

1.3 Repositories

The following methods allow for interaction into the Tenable Vulnerability Management Container Security repositories API endpoints.

Methods available on `tio.cs.repositories`:

class RepositoriesAPI(*api: APISession*)

delete(*name: str*) → None

Deleted the specified repository.

API Documentation # noqa: E501

Parameters

name (*str*) – The repository name.

Examples

```
>>> tio.cs.repositories.delete('centos')
```

details(*name: str*) → Dict

Returns the details for the specified repository.

API Documentation # noqa: E501

Parameters

name (*str*) – The repository name.

Examples

```
>>> tio.cs.repositories.details('centos')
```

list(*name: str | None = None*, *contains: str | None = None*, *offset: int = 0*, *limit: int = 1000*, *return_json: bool = False*) → Dict | CSIterator

Returns the list of images stored within Container Security.

API Documentation # noqa: E501

Parameters

- **name** (*str, optional*) – Image name to filter on. Filter is case-sensitive and enforces an exact match.
- **contains** (*str, optional*) – Partial name to filter on. Filter is case-sensitive.
- **offset** (*int, optional*) – The number of records to skip before starting to return data.
- **limit** (*int, optional*) – The number of records to return for each page of data.
- **return_json** (*bool, optional*) – If set, then the response will instead be a Dict object instead of an iterable.

Examples

Using the default iterable:

```
>>> for repo in tio.cs.repositories.list():
...     print(repo)
```

Getting the raw JSON response:

```
>>> resp = tio.cs.repositories.list(return_json=True)
>>> for item in resp['items']:
...     print(item)
```

CHAPTER

TWO

PCI-ASV

The following methods allow for interaction into the TVM PCI-ASV API endpoints.

Methods available on `tio.pci`:

class PCIASVAPI(*api: APISession*)

property attestations: *AttestationsAPI*

The interface for *TVM PCI-ASV Attestations APIs*.

property scans: *ScansAPI*

The interface for *TVM PCI-ASV Scans APIs*.

2.1 Attestations

The following methods allow for interaction into the TVM PCI Attestation API API endpoints.

Methods available on `tio.pci.attestations`:

class AttestationsAPI(*api: APISession*)

assets(*id: ~uuid.UUID | str*, ***, *sort: ~typing.List[tuple[str, ~typing.Literal['asc', 'desc']]] | None = None*, *limit: int = 1000*, *offset: int = 0*, *iterator: ~typing.Type[~tenable.io.pci.iterators.PCIAttestationAssetsIterator] | None = <class 'tenable.io.pci.iterators.PCIAttestationAssetsIterator'>*) → PCIAttestationAssetsIterator | dict[str, Any]

Retrieves the list of assets associated to the attestation ID provided.

Parameters

- **sort** – Sort the results based on the field and sort order.
- **limit** – The number of results to return for each page
- **offset** – Where within the page data to start the page.
- **iterator** – Should an iterator be returned or a page of data? If set to *None*, the page will be returned instead of the iterable.

details(*id: UUID | str*) → dict[str, Any]

Retrieves the details of a specific attestation.

Parameters

id – The attestation UUID to retrieve.

disputes(*id: ~uuid.UUID | str*, ***, *sort: ~typing.List[tuple[str, ~typing.Literal['asc', 'desc']]] | None = None*, *limit: int = 1000*, *offset: int = 0*, *iterator: ~typing.Type[~tenable.io.pci.iterators.PCIDisputesIterator] | None = <class 'tenable.io.pci.iterators.PCIDisputesIterator'>*) → PCIDisputesIterator | dict[str, Any]

Retrieves the list of disputes for the attestation ID provided.

Parameters

- **sort** – Sort the results based on the field and sort order.
- **limit** – The number of results to return for each page
- **offset** – Where within the page data to start the page.
- **iterator** – Should an iterator be returned or a page of data? If set to *None*, the page will be returned instead of the iterable.

failures(*id: ~uuid.UUID | str*, ***, *sort: ~typing.List[tuple[str, ~typing.Literal['asc', 'desc']]] | None = None*, *limit: int = 1000*, *offset: int = 0*, *iterator: ~typing.Type[~tenable.io.pci.iterators.PCIUndisputedFailuresIterator] | None = <class 'tenable.io.pci.iterators.PCIUndisputedFailuresIterator'>*) → PCIUndisputedFailuresIterator | dict[str, Any]

Retrieves the list of undisputed failures for the attestation ID provided.

Parameters

- **sort** – Sort the results based on the field and sort order.
- **limit** – The number of results to return for each page
- **offset** – Where within the page data to start the page.
- **iterator** – Should an iterator be returned or a page of data? If set to *None*, the page will be returned instead of the iterable.

list(***, *status_type: list[~typing.Literal['IN_PROGRESS', 'NEEDS_WORK', 'IN_REVIEW', 'INFO_REQUESTED', 'INFO_PROVIDED', 'FAILED', 'PASSED', 'CLOSED']] | None = None*, *sort: list[tuple[str, ~typing.Literal['asc', 'desc']]] | None = None*, *limit: int = 1000*, *offset: int = 0*, *iterator: ~typing.Type[~tenable.io.pci.iterators.PCIAttestationsIterator] | None = <class 'tenable.io.pci.iterators.PCIAttestationsIterator'>*) → PCIAttestationsIterator | dict[str, Any]

Returns a list of PCI attestations.

Parameters

- **status_type** – Filters the attestations based on the status types defined.
- **sort** – Sort the results based on the field and sort order.
- **limit** – The number of results to return for each page
- **offset** – Where within the page data to start the page.
- **iterator** – Should an iterator be returned or a page of data? If set to *None*, the page will be returned instead of the iterable.

2.2 Scans

The following methods allow for interaction into the TVM PCI Scans API API endpoints.

Methods available on `tio.pci.scans`:

class ScansAPI(*api: APISession*)

list(*, *sort: list[tuple[str, ~typing.Literal['asc', 'desc']]] | None = None*, *limit: int = 1000*, *offset: int = 0*, *iterator: ~typing.Type[~tenable.io.pci.iterators.PCIScansIterator] | None = <class 'tenable.io.pci.iterators.PCIScansIterator'>*) → PCIScansIterator | dict[str, Any]

Returns a list of PCI scans.

Parameters

- **sort** – Sort the results based on the field and sort order.
- **limit** – The number of results to return for each page
- **offset** – Where within the page data to start the page.
- **iterator** – Should an iterator be returned or a page of data? If set to *None*, the page will be returned instead of the iterable.

CHAPTER

THREE

ACCESS-CONTROL

The following methods allow for interaction into the Tenable Vulnerability Management API endpoints.

Methods available on `tio.v3.access_control`:

class AccessControlAPI(*api: APISession*)

create(*permission: Dict*) → Dict

Creates a new permission

access-control: create

Parameters
permission (*dict*) – the permission details object that needs to be created,

Returns
The resource record for the new permission.

Return type
dict

Example

```
>>> permission = {
...     "actions": ["CanView","CanEdit"],
...     "objects": [
...         {
...             "type": "Tag",
...             "uuid": "10bd7477-2961-402c-92fb-d7f6a8dc9399",
...             "name": "TGG,DE"
...         }
...     ],
...     "subjects": [
...         {
...             "name": "dummy_user@tenable.com",
...             "type": "User",
...             "uuid": "4f948c212-ae2c-4d0b-bab4-0fc1088a85bd"
...         }
...     ],
...     "name": "permission name"
... }
...
...  tio.access_control.create(permission)
```

delete(*permission_uuid: UUID*) → Dict

Delete the specified permission

access-control: delete

Parameters
permission_uuid (*str*) – the uuid of the permission to remove

Return type
dict

Examples

```
>>> tio.access_control.delete(
...     '4f948c22-ae2c-4d0b-bab4-0fc1088a85bd'
... )
```

details(*uuid: UUID*) → Dict

Retrieves the details of the specified permission.

access-control: details :param uuid: the uuid of the permission to retrieve :type uuid: str

Returns
The resource record for the specified permission

Return type
dict

Examples

Get specific permission details:

```
>>> tio.access_control.details(
...     '4f948c22-ae2c-4d0b-bab4-0fc1088a85bd'
... )
```

get_current_user_permission() → Dict

Retrieves current user permission details

access-control : current user permission

Returns
The resource record for the current user permission.

Return type
dict

Examples

Get specific user-group permission details:

```
>>> tio.access_control.get_current_user_permission(
...     '4f948c22-ae2c-4d0b-bab4-0fc1088a85bd'
... )
```

get_user_group_permission(*user_group_uuid: UUID*) → Dict

Retrieves user group permission details

access-control : user group permission

Parameters
user_group_uuid (*str*) – the uuid of the user-group to retrieve

Returns
The resource record for the user-group permission

Return type
dict

Examples

Get specific user-group permission details:

```
>>> tio.access_control.get_user_group_permission(
...     '4f948c22-ae2c-4d0b-bab4-0fc1088a85bd'
... )
```

get_user_permission(*user_uuid: UUID*) → Dict

Retrieves user permission details

access-control: user permission

Parameters
user_uuid (*str*) – the uuid of the user to retrieve

Returns
The resource record for the user permissions

Return type
dict

Examples

Get specific user permission details:

```
>>> tio.access_control.get_user_permission(
...     '4f948c22-ae2c-4d0b-bab4-0fc1088a85bd'
... )
```

list() → List

Returns a list of permissions in your container.

Returns
List of permissions.

Return type
list

Examples

```
>>> for permission in tio.access_control.list():
...     pprint(permission)
```

update(*permission_uuid: UUID*, *permission: Dict*) → Dict

update permission

access-control : update

Parameters

- **permission_uuid** (*str*) – permission uuid to be updated
- **permission** (*dict*) – the permission details object that needs to be updated, permission details object example :

Returns
Update successfully requested.

Return type
None

Example

```
>>> payload = {
...     "actions": ["CanView","CanEdit"],
...     "objects": [
...          {
...               "type": "Tag",
...               "uuid": "10bd7477-2961-402c-92fb-d7f6a8dc9399",
...               "name": "TGG,DE"
...          }
...     ],
...     "subjects": [
...           {
...               "name": "dummy_user@tenable.com",
...               "type": "User",
...               "uuid": "4f948c212-ae2c-4d0b-bab4-0fc1088a85bd"
...           }
...     ],
...     "name": "permission name"
... }
>>>  permission_uuid_val = "212-ae2c-4d0b-bab4-0fc1088a85bd"
>>>  tio.v3.access_control.update(permission_uuid_val, payload)
```

CHAPTER

FOUR

AGENT CONFIG

The following methods allow for interaction into the Tenable Vulnerability Management agent config API endpoints.

Methods available on `tio.agent_config`:

class AgentConfigAPI(*api: APISession*)

This will contain all methods related to agent config

details(*scanner_id=1*)

Returns the current agent configuration.

agent-config: details

Parameters
scanner_id (*int, optional*) – The scanner ID.

Returns
Dictionary of the current settings.

Return type
dict

Examples

```
>>> details = tio.agent_config.details()
>>> pprint(details)
```

edit(*scanner_id=1*, *software_update=None*, *auto_unlink=None*)

Edits the agent configuration.

agent-config: edit

Parameters

- **scanner_id** (*int, optional*) – The scanner ID.
- **software_update** (*bool, optional*) – If True, software updates are enabled for agents (exclusions may override this). If false, software updates for all agents are disabled.
- **auto_unlink** (*int, optional*) – If true, agent auto-unlinking is enabled, allowing agents to automatically unlink themselves after a given period of time. If the value is 0 or false, auto-unlinking is disabled. True values are between 1 and 365.

Returns
Dictionary of the applied settings is returned if successfully applied.

Return type
dict

Examples

Enabling auto-unlinking for agents after 30 days:

```
>>> tio.agent_config.edit(auto_unlink=30)
```

Disabling auto-unlinking for agents:

```
>>> tio.agent_config.edit(auto_unlink=False)
```

Enabling software updates for agents:

```
>>> tio.agent_config.edit(software_update=True)
```

CHAPTER

FIVE

AGENT EXCLUSIONS

The following methods allow for interaction into the Tenable Vulnerability Management agent exclusions API endpoints.

Methods available on `tio.agent_exclusions`:

class AgentExclusionsAPI(*api: APISession*)

create(*name*, *scanner_id=1*, *start_time=None*, *end_time=None*, *timezone=None*, *description=None*, *frequency=None*, *interval=None*, *weekdays=None*, *day_of_month=None*, *enabled=True*)

Creates a new agent exclusion.

agent-exclusions: create

Parameters

- **name** (*str*) – The name of the exclusion to create.
- **scanner_id** (*int, optional*) – The scanner id.
- **description** (*str, optional*) – Some further detail about the exclusion.
- **start_time** (*datetime*) – When the exclusion should start.
- **end_time** (*datetime*) – When the exclusion should end.
- **timezone** (*str, optional*) – The timezone to use for the exclusion. The default if none is specified is to use UTC. For the list of usable timezones, please refer to: https://cloud.tenable.com/api#/resources/scans/timezones
- **frequency** (*str, optional*) – The frequency of the rule. The string inputted will be up-cased. Valid values are: `ONETIME`, `DAILY`, `WEEKLY`, `MONTHLY`, `YEARLY`. Default value is `ONETIME`.
- **interval** (*int, optional*) – The interval of the rule. The default interval is 1
- **weekdays** (*list, optional*) – List of 2-character representations of the days of the week to repeat the frequency rule on. Valid values are: *SU, MO, TU, WE, TH, FR, SA* Default values: `['SU', 'MO', 'TU', 'WE', 'TH', 'FR', 'SA']`
- **day_of_month** (*int, optional*) – The day of the month to repeat a **MONTHLY** frequency rule on. The default is today.
- **enabled** (*bool, optional*) – enable/disable exclusion. The default is `True`

Returns
Dictionary of the newly minted exclusion.

Return type
dict

Examples

Creating a one-time exclusion:

```
>>> from datetime import datetime, timedelta
>>> exclusion = tio.agent_exclusions.create(
...     'Example One-Time Agent Exclusion',
...     ['127.0.0.1'],
...     start_time=datetime.utcnow(),
...     end_time=datetime.utcnow() + timedelta(hours=1))
```

Creating a daily exclusion:

```
>>> exclusion = tio.agent_exclusions.create(
...     'Example Daily Agent Exclusion',
...     ['127.0.0.1'],
...     frequency='daily',
...     start_time=datetime.utcnow(),
...     end_time=datetime.utcnow() + timedelta(hours=1))
```

Creating a weekly exclusion:

```
>>> exclusion = tio.agent_exclusions.create(
...     'Example Weekly Exclusion',
...     ['127.0.0.1'],
...     frequency='weekly',
...     weekdays=['mo', 'we', 'fr'],
...     start_time=datetime.utcnow(),
...     end_time=datetime.utcnow() + timedelta(hours=1))
```

Creating a monthly esxclusion:

```
>>> exclusion = tio.agent_exclusions.create(
...     'Example Monthly Agent Exclusion',
...     ['127.0.0.1'],
...     frequency='monthly',
...     day_of_month=1,
...     start_time=datetime.utcnow(),
...     end_time=datetime.utcnow() + timedelta(hours=1))
```

Creating a yearly exclusion:

```
>>> exclusion = tio.agent_exclusions.create(
...     'Example Yearly Agent Exclusion',
...     ['127.0.0.1'],
...     frequency='yearly',
...     start_time=datetime.utcnow(),
...     end_time=datetime.utcnow() + timedelta(hours=1))
```

delete(*exclusion_id*, *scanner_id=1*)

Delete an agent exclusion.

agent-exclusions: delete

Parameters

- **exclusion_id** (*int*) – The id of the exclusion object in Tenable Vulnerability Management
- **scanner_id** (*int, optional*) – The id of the scanner

Returns
The Exclusion was successfully deleted

Return type
None

Examples

```
>>> tio.agent_exclusions.delete(1)
```

details(*exclusion_id*, *scanner_id=1*)

Retrieve the details for a specific agent exclusion.

agent-exclusion: details

Parameters

- **exclusion_id** (*int*) – The id of the exclusion object in Tenable Vulnerability Management
- **scanner_id** (*int, optional*) – The id of the scanner

Returns
The exclusion resource dictionary.

Return type
dict

Examples

```
>>> exclusion = tio.agent_exclusions.details(1)
```

edit(*exclusion_id*, *scanner_id=1*, *name=None*, *start_time=None*, *end_time=None*, *timezone=None*, *description=None*, *frequency=None*, *interval=None*, *weekdays=None*, *day_of_month=None*, *enabled=None*)

Edit an existing agent exclusion.

agent-exclusions: edit

The edit function will first gather the details of the exclusion that will be edited and will overlay the changes on top. The result will then be pushed back to the API to modify the exclusion.

Parameters

- **exclusion_id** (*int*) – The id of the exclusion object in Tenable Vulnerability Management
- **scanner_id** (*int, optional*) – The scanner id.
- **name** (*str, optional*) – The name of the exclusion to create.
- **description** (*str, optional*) – Some further detail about the exclusion.
- **start_time** (*datetime, optional*) – When the exclusion should start.

- **end_time** (*datetime, optional*) – When the exclusion should end.
- **timezone** (*str, optional*) – The timezone to use for the exclusion. The default if none is specified is to use UTC.
- **frequency** (*str, optional*) – The frequency of the rule. The string inputted will be up-cased. Valid values are: *ONETIME, DAILY, WEEKLY, MONTHLY, YEARLY*.
- **interval** (*int, optional*) – The interval of the rule.
- **weekdays** (*list, optional*) – List of 2-character representations of the days of the week to repeat the frequency rule on. Valid values are: *SU, MO, TU, WE, TH, FR, SA* Default values: `['SU', 'MO', 'TU', 'WE', 'TH', 'FR', 'SA']`
- **day_of_month** (*int, optional*) – The day of the month to repeat a **MONTHLY** frequency rule on.
- **enabled** (*bool, optional*) – enable/disable exclusion.

Returns
Dictionary of the newly minted exclusion.

Return type
dict

Examples

```
>>> exclusion = tio.agent_exclusions.edit(1, name='New Name')
```

list(*scanner_id=1*)

Lists all of the currently configured agent exclusions.

agent-exclusions: list

Parameters
scanner_id (*int, optional*) – The scanner identifier to be used.

Returns
List of agent exclusions.

Return type
list

Examples

```
>>> for exclusion in tio.agent_exclusions.list():
...     pprint(exclusion)
```

CHAPTER

SIX

AGENT GROUPS

The following methods allow for interaction into the Tenable Vulnerability Management agent groups API endpoints.

Methods available on `tio.agent_groups`:

class **AgentGroupsAPI**(*api: APISession*)

add_agent(*group_id*, **agent_ids*, ***kw*)

Adds an agent or multiple agents to the agent group specified.

agent-groups: add-agent

Parameters

- **group_id** (*int*) – The id of the group
- ***agent_ids** (*int*) – The id or uuid of the agent. If passing multiple, they must all be either numeric id or uuid.
- **scanner_id** (*int, optional*) – The id of the scanner

Returns

If adding a singular agent, a `None` response will be returned. If adding multiple agents, a `dict` response will be returned with a task record.

Return type

`dict` or `None`

Examples

Adding a singular agent:

```
>>> tio.agent_groups.add_agent(1, 1)
```

Adding multiple agents:

```
>>> tio.agent_groups.add_agent(1, 1, 2, 3)
```

Adding multiple agents by uuid:

```
>>> tio.agent_groups.add_agent(1, 'uuid-1', 'uuid-2', 'uuid-3')
```

configure(*group_id*, *name*, *scanner_id=1*)

Renames an existing agent group.

agent-groups: configure

Parameters

- **group_id** (*int*) – The id of the group
- **name** (*str*) – The new name for the agent group
- **scanner_id** (*int, optional*) – The id of the scanner

Returns
None

Examples

```
>>> tio.agent_groups.configure(1, 'New Name')
```

create(*name*, *scanner_id=1*)

Creates a new agent group.

agent-groups: create

Parameters

- **name** (*str*) – The name of the agent group
- **scanner_id** (*int, optional*) – The id of the scanner to add the agent group to

Returns
The dictionary object representing the newly minted agent group

Return type
dict

Examples

```
>>> group = tio.agent_groups.create('New Agent Group')
```

delete(*group_id*, *scanner_id=1*)

Delete an agent group.

agent-groups: delete

Parameters

- **group_id** (*int*) – The id of the agent group to delete
- **scanner_id** (*int, optional*) – The id of the scanner

Returns
None

Examples

```
>>> tio.agent_groups.delete(1)
```

delete_agent(*group_id*, **agent_ids*, ***kw*)

Delete one or many agents from an agent group.

agent-groups: delete-agent

Parameters

- **group_id** (*int*) – The id of the agent group to remove the agent from
- ***agent_ids** (*int*) – The id of the agent to be removed
- **scanner_id** (*int, optional*) – The id of the scanner

Returns

If deleting a singular agent, a `None` response will be returned. If deleting multiple agents, a `dict` response will be returned with a Job resource record.

Return type

`dict` or `None`

Examples

Delete a singular agent from an agent group:

```
>>> tio.agent_groups.delete_agent(1, 1)
```

Delete multiple agents from an agent group:

```
>>> tio.agent_groups.delete_agent(1, 1, 2, 3)
```

details(*group_id*, **filters*, ***kw*)

Retrieve the details about the specified agent group.

agent-groups: details

Parameters

- **group_id** (*int*) – The id of the agent group.
- ***filters** (*tuple, optional*) – Filters are tuples in the form of ('NAME', 'OPERATOR', 'VALUE'). Multiple filters can be used and will filter down the data being returned from the API.

  ### Examples

  - `('distro', 'match', 'win')`
  - `('name', 'nmatch', 'home')`

  As the filters may change and sortable fields may change over time, it's highly recommended that you look at the output of the filters:agents-filters endpoint to get more details.

- **filter_type** (*str, optional*) – The filter_type operator determines how the filters are combined together. `and` will inform the API that all the filter conditions must be met for an agent to be returned, whereas `or` would mean that if any of the conditions are met, the agent record will be returned.
- **limit** (*int, optional*) – The number of records to retrieve. Default is 50
- **offset** (*int, optional*) – The starting record to retrieve. Default is 0.
- **scanner_id** (*int, optional*) – The identifier the scanner that the agent communicates to.
- **sort** (*tuple, optional*) – A tuple of tuples identifying the field and sort order of the field.
- **wildcard** (*str, optional*) – A string to pattern match against all available fields returned.
- **wildcard_fields** (*list, optional*) – A list of fields to optionally restrict the wildcard matching to.

Returns
The dictionary object representing the requested agent group

Return type
dict

Examples

```
>>> group = tio.agent_groups.details(1)
>>> pprint(group)
```

list(*scanner_id=1*)

Retrieves the list of agent groups configured

agent-groups: list

Parameters
scanner_id (*int, optional*) – The id of the scanner

Returns
Listing of agent group resource records

Return type
list

Examples

>>>> for agent_group in tio.agent_groups.list(): ... pprint(agent_group)

task_status(*group_id*, *task_uuid*, *scanner_id=1*)

Retrieves the current status of a bulk task.

bulk-operations: bulk-agent-group-status

Parameters

- **group_id** (*int*) – The id of the group
- **task_uuid** (*str*) – The id of the task

- **scanner_id** (*int, optional*) – The id of the scanner

Returns
Task resource

Return type
dict

Examples

```
>>> item = tio.agent_groups.add_agent(1, 21, 22, 23)
>>> task = tio.agent_groups.task_status(item['task_uuid'])
>>> pprint(task)
```

CHAPTER

SEVEN

AGENTS

The following methods allow for interaction into the Tenable Vulnerability Management agents API endpoints.

Methods available on `tio.agents`:

class AgentsAPI(*api: APISession*)

details(*agent_id*, *scanner_id=1*)

Retrieves the details of an agent.

agents: get

Parameters

- **agent_id** (*int*) – The identifier of the agent.
- **scanner_id** (*int, optional*) – The identifier of the scanner. Default is 1.

Returns
The agent dictionary record.

Return type
dict

Examples

```
>>> agent = tio.agents.details(1)
>>> pprint(agent)
```

list(**filters*, ***kw*)

Get the listing of configured agents from Tenable Vulnerability Management.

agents: list

Parameters

- ***filters** (*tuple, optional*) – Filters are tuples in the form of ('NAME', 'OPERATOR', 'VALUE'). Multiple filters can be used and will filter down the data being returned from the API.

Examples

- `('distro', 'match', 'win')`
- `('name', 'nmatch', 'home')`

As the filters may change and sortable fields may change over time, it's highly recommended that you look at the output of the filters:agents-filters endpoint to get more details.

- **filter_type** (*str, optional*) – The filter_type operator determines how the filters are combined together. `and` will inform the API that all the filter conditions must be met for an agent to be returned, whereas `or` would mean that if any of the conditions are met, the agent record will be returned.
- **limit** (*int, optional*) – The number of records to retrieve. Default is 50
- **offset** (*int, optional*) – The starting record to retrieve. Default is 0.
- **scanner_id** (*int, optional*) – The identifier the scanner that the agent communicates to.
- **sort** (*tuple, optional*) – A tuple of tuples identifying the field and sort order of the field.
- **wildcard** (*str, optional*) – A string to pattern match against all available fields returned.
- **wildcard_fields** (*list, optional*) – A list of fields to optionally restrict the wildcard matching to.

Returns
An iterator that handles the page management of the requested records.

Return type
`AgentsIterator`

Examples

Getting the listing of all agents:

```
>>> for agent in tio.agents.list():
...     pprint(agent)
```

Retrieving all of the windows agents:

```
>>> for agent in tio.agents.list(('distro', 'match', 'win')):
...     pprint(agent)
```

task_status(*task_uuid*, *scanner_id=1*)

Retrieves the current status of the task requested.

bulk-operations: bulk-agent-status

Parameters

- **task_uuid** (*str*) – The id of the agent
- **scanner_id** (*int, optional*) – The id of the scanner

Returns
Task resource

Return type
dict

Examples

```
>>> item = tio.agents.unlink(21, 22, 23)
>>> task = tio.agent.task_status(item['task_uuid'])
>>> pprint(task)
```

unlink(**agent_ids*, ***kw*)

Unlink one or multiple agents from the Tenable Vulnerability Management instance.

agents: delete

Parameters

- ***agent_ids** (*int*) – The ID of the agent to delete
- **scanner_id** (*int, optional*) – The identifier the scanner that the agent communicates to.

Returns
If unlinking a singular agent, a None response will be returned. If unlinking multiple agents, a dict response will be returned with a task record.

Return type
dict or None

Examples

Unlink a singular agent:

```
>>> tio.agents.unlink(1)
```

Unlink many agents:

```
>>> tio.agents.unlink(1, 2, 3)
```

CHAPTER

EIGHT

ASSETS

The following methods allow for interaction into the Tenable Vulnerability Management assets API endpoints.

Methods available on `tio.assets`:

class AssetsAPI(*api: APISession*)

This will contain all methods related to Assets

asset_import(*source*, **assets*)

Imports asset information into Tenable Vulnerability Management from an external source.

assets: import

Imports a list of asset definition dictionaries. Each asset record must contain at least one of the following attributes: `fqdn`, `ipv4`, `netbios_name`, `mac_address`. Each record may also contain additional properties.

Parameters

- ***assets** (*dict*) – One or more asset definition dictionaries
- **source** (*str*) – An identifier to be used to upload the assets.

Returns

The job UUID.

Return type

str

Examples

import single asset:

```
>>> tio.assets.asset_import('example_source', {
...     'fqdn': ['example.py.test'],
...     'ipv4': ['192.168.254.1'],
...     'netbios_name': 'example',
...     'mac_address': ['00:00:00:00:00:00']
... })
```

import multiple asset:

```
>>> tio.assets.asset_import('multiple_asset_example_source',
...     {
...         'fqdn': ['example_one.py.test'],
```

(continues on next page)

(continued from previous page)

```
...         'ipv4': ['192.168.1.1'],
...         'netbios_name': 'example_one',
...         'mac_address': ['00:00:00:00:00:00']
...     },{
...         'fqdn': ['example_two.py.test'],
...         'ipv4': ['192.168.255.1'],
...         'netbios_name': 'example_two',
...         'mac_address': ['00:00:00:00:00:00']
...     })
```

assign_tags(*action*, *assets*, *tags*)

Add/remove tags for asset(s).

tags: assign-asset-tags

Parameters

- **action** (*str*) – Specifies whether to add or remove tags. Valid values: add, remove.
- **assets** (*List[str]*) – An array of asset UUIDs.
- **tags** (*List[str]*) – An array of tag value UUIDs.

Returns
The job Resource record.

Return type
dict

Examples

```
>>> asset = tio.assets.assign_tags(
...     'add', ['00000000-0000-0000-0000-000000000000'],
...     ['00000000-0000-0000-0000-000000000000'])
```

bulk_delete(**filters*, *hard_delete=None*, *filter_type=None*)

Deletes the specified assets.

assets: bulk_delete

Parameters
***filters** – A defined filter tuple consisting of the name, operator, and value. Example: `('host.hostname', 'match', 'asset.com')`.

Returns
Returns the number of deleted assets.

Return type
dict

Examples

```
>>> asset = tio.assets.bulk_delete(
...     ('host.hostname', 'match', 'asset.com'), filter_type='or')
>>> pprint(asset)
```

delete(**uuid*)

Deletes the asset.

NOTE: This method is simply a link to the bulk delete API with the appropriate
filter pre-populated. This is here for backwards-compatability reasons as the older workbench API has been removed. It's highly recommended to move to the bulk delete endpoint instead.

Parameters
asset_uuid (*str*) – The unique identifier for the asset.

Return type
None

Examples

```
>>> asset_id = '00000000-0000-0000-0000-000000000000'
>>> tio.asset.delete(asset_id)
```

details(*uuid*)

Retrieves the details about a specific asset.

assets: asset-info

Parameters
uuid (*str*) – The UUID (unique identifier) for the asset.

Returns
Asset resource definition.

Return type
dict

Examples

```
>>> asset = tio.assets.details(
...     '00000000-0000-0000-0000-000000000000')
```

import_job_details(*uuid*)

Returns the details about a specific asset import job.

assets: import-job-info

uuid (str):
The UUID (unique identifier) for the job.

Returns
The job Resource record.

Return type
dict

Examples

```
>>> job = tio.assets.import_job_details(
...     '00000000-0000-0000-0000-000000000000')
>>> pprint(job)
```

list()

Returns a list of assets.

assets: list-assets

Returns
List of asset records.

Return type
list

Examples

```
>>> for asset in tio.assets.list():
...     pprint(asset)
```

list_import_jobs()

Returns a list of asset import jobs.

assets: list-import-jobs

Returns
List of job records.

Return type
list

Examples

```
>>> for job in tio.assets.list_import_jobs():
...     pprint(job)
```

move_assets(*source*, *destination*, *targets*)

Moves assets from the specified network to another network.

assets: move-assets

source (str):
The UUID of the network currently associated with the assets.

destination (str):
The UUID of the network to associate with the specified assets.

targets (list):
The IPv4 addresses of the assets to move.

Returns
Returns the number of moved assets.

Return type
int

Examples

```
>>> asset = tio.assets.move_assets('00000000-0000-0000-0000-000000000000',
...         '10000000-0000-0000-0000-000000000001', ["127.0.0.1"])
>>> pprint(asset)
```

tags(*uuid*)

Retrieves the details about a specific asset.

tags: asset-tags

Parameters
uuid (*str*) – The UUID (unique identifier) for the asset.

Returns
Asset resource definition.

Return type
dict

Examples

```
>>> asset = tio.assets.tags(
...     '00000000-0000-0000-0000-000000000000')
```

update_acr(*assets_uuid_list*, *reason*, *value*, *note=''*)

Updates ACR for the provided asset UUID's with reason(s).

Parameters

- **assets_uuid_list** (*list*) – Asset UUID's which are being updated.
- **reason** (*list*) – List of reason(s).
- **value** (*str*) – New ACR value for assets, ranging from 1 - 10.
- **note** (*str*) – Additional note if any.

Returns
Status code for the request.

Return type
int

Examples

```
>>> tio.assets.update_acr(
... ['00000000-0000-0000-0000-000000000000', '00000000-0000-0000-0000-
↪000000000001'],
... ['Other'], 1, 'some notes')
```

CHAPTER

NINE

AUDIT LOG

The following methods allow for interaction into the Tenable Vulnerability Management audit log API endpoints.

Methods available on `io.audit_log`:

class **AuditLogAPI**(*api: APISession*)

events(**filters: Tuple[str, str, str], limit: int = 1000, filter_type: Literal['and', 'or'] = 'and', sort: str | None = None, token: str | None = '0', return_json: bool = False*)

Retrieve audit logs from Tenable Vulnerability Management.

audit-log: events

Parameters

- ***filters** (*tuple, optional*) – Filters to allow the user to get to a specific subset of data within the audit log. For a more detailed listing of what filters are available, please refer to the API documentation linked above, however some examples are as such:
  - `('date', 'gt', '2017-07-05')`
  - `('date', 'lt', '2017-07-07')`
  - `('actor_id', 'match', '6000a811-8422-4096-83d3-e4d44f7d')`
  - `('target_id', 'match', '6000a811-8422-4096-83d3-e4d447d')`
- **limit** (*int, optional*) – The limit of how many events to return. The API will default to 50 unless otherwise specified.
- **filter_type** (*str, optional*) – if multiple filters are present, how should we combine the filters? Supported values are *and* or *or*. If left unspecified, the default is *and*.
- **sort** (*str, optional*) – Should any soerting be performed on the resulting data? The format is *FIELD_NAME:DIRECTION*. For example, supplying *received:desc* would sort the results by the received field in descencing order.
- **token** (*str, optional*) – The *next* token to request the next page.
- **return_json** (*bool, optional*) – Should we return the JSON response instead of iterable?

Returns
List of event records

Return type
`AuditLogIterator`

Examples

```
>>> events = tio.audit_log.events(('date', 'gt', '2018-01-01'))
>>> for e in events:
...     pprint(e)
```

CHAPTER

TEN

CREDENTIALS

The following methods allow for interaction into the Tenable Vulnerability Management credentials API endpoints.

Methods available on `tio.credentials`:

class CredentialsAPI(*api: APISession*)

create(*cred_name*, *cred_type*, *description=None*, *permissions=None*, ***settings*)

Creates a new managed credential.

credentials: create

Parameters

- **cred_name** (*str*) – The name of the credential.
- **cred_type** (*str*) – The type of credential to create. For a list of values refer to the output of the *types()* method.
- **description** (*str, optional*) – A description for the credential.
- **permissions** (*list, optional*) – A list of permissions (in either tuple or native dict format) detailing whom is allowed to use or edit this credential set. For the dictionary format, refer to the API docs. The tuple format uses the customary `(type, perm, uuid)` format.

  ### Examples

  - `('user', 32, user_uuid)`
  - `('group', 32, group_uuid)`
  - `('user', 'use', user_uuid)`

- ****settings** (*dict, optional*) – Additional keywords passed will be added to the settings dict within the API call. As this dataset can be highly variable, it will not be validated and simply passed as-is.

Returns

The UUID of the newly created credential.

Return type

str

Examples

```
>>> group_id = '00000000-0000-0000-0000-000000000000'
>>> tio.credentials.create('SSH Account', 'SSH',
...     permissions=[('group', 'use', group_id)],
...     username='user1',
...     password='sekretsquirrel',
...     escalation_account='root',
...     escalation_password='sudopassword',
...     elevate_privileges_with='sudo',
...     bin_directory='/usr/bin',
...     custom_password_prompt='')
```

delete(*id*)

Deletes the specified credential.

credentials: delete

Parameters
id (*str*) – The UUID of the credential to retrieve.

Returns
The status of the action.

Return type
bool

Examples

```
>>> cred_uuid = '00000000-0000-0000-0000-000000000000'
>>> cred = tio.credentials.delete(cred_uuid)
```

details(*id*)

Retrieves the details of the specified credential.

credentials: details

Parameters
id (*str*) – The UUID of the credential to retrieve.

Returns
The resource record for the credential.

Return type
dict

Examples

```
>>> cred_uuid = '00000000-0000-0000-0000-000000000000'
>>> cred = tio.credentials.details(cred_uuid)
```

edit(*cred_uuid*, *cred_name=None*, *description=None*, *permissions=None*, *ad_hoc=None*, ***settings*)

Creates a new managed credential.

credentials: create

Parameters

- **ad_hoc** (*bool, optional*) – Determines whether the credential is managed (`False`) or an embedded credential in a scan or policy (`True`).
- **cred_name** (*str, optional*) – The name of the credential.
- **description** (*str, optional*) – A description for the credential.
- **permissions** (*list, optional*) – A list of permissions (in either tuple or native dict format) detailing whom is allowed to use or edit this credential set. For the dictionary format, refer to the API docs. The tuple format uses the customary `(type, perm, uuid)` format.

  #### Examples

  - `('user', 32, user_uuid)`
  - `('group', 32, group_uuid)`
  - `('user', 'use', user_uuid)`

- ****settings** (*dict, optional*) – Additional keywords passed will be added to the settings dict within the API call. As this dataset can be highly variable, it will not be validated and simply passed as-is.

Returns
The status of the update process.

Return type
`bool`

Examples

```
>>> cred_uuid = '00000000-0000-0000-0000-000000000000'
>>> tio.credentials.edit(cred_uuid,
...     password='sekretsquirrel',
...     escalation_password='sudopassword')
```

list(**filters*, ***kw*)

Get the listing of configured credentials from Tenable Vulnerability Management.

credentials: list

Parameters

- ***filters** (*tuple, optional*) – Filters are tuples in the form of ('NAME', 'OPERATOR', 'VALUE'). Multiple filters can be used and will filter down the data being returned from the API.

  #### Examples

  – `('name', 'eq', 'example')`

  As the filters may change and sortable fields may change over time, it's highly recommended that you look at the output of the `tio.filters.networks_filters()` endpoint to get more details.

- **filter_type** (*str, optional*) – The filter_type operator determines how the filters are combined together. `and` will inform the API that all of the filter conditions must be met for an access group to be returned, whereas `or` would mean that if any of the conditions are met, the access group record will be returned.
- **limit** (*int, optional*) – The number of records to retrieve. Default is 50
- **offset** (*int, optional*) – The starting record to retrieve. Default is 0.
- **owner_uuid** (*str, optional*) – The UUID of the scan owner. If specified it will limit the responses to credentials assigned to scans owned by the specified user UUID.
- **sort** (*tuple, optional*) – A tuple of tuples identifying the the field and sort order of the field.
- **wildcard** (*str, optional*) – A string to pattern match against all available fields returned.
- **wildcard_fields** (*list, optional*) – A list of fields to optionally restrict the wildcard matching to.

Returns
An iterator that handles the page management of the requested records.

Return type
`CredentialsIterator`

Examples

```
>>> for cred in tio.credentials.list():
...     pprint(cred)
```

types()

Lists all of the available credential types.

credentials: list-types

Returns
A list of the available credential types and definitions.

Return type
list

Examples

```
>>> cred_types = tio.credentials.types()
```

upload(*fobj: BinaryIO*, *file_type: str*)

Uploads a file for use with a managed credential.

credentials: upload

Parameters

- **fobj** (*FileObject*) – The file object intended to be uploaded into Tenable Vulnerability Management.
- **file_type** (*string*) – File type of the credential being uploaded.

Returns

The fileuploaded attribute

Return type

str

Examples

```
>>>     with open("hello.pem", "rb") as file:
...             response = tio.credentials.upload(file, "pem")
...
...     print(response)
```

CHAPTER

ELEVEN

EDITOR

The following methods allow for interaction into the Tenable Vulnerability Management editor API endpoints. While these endpoints are pythonized for completeness within pyTenable, the Editor API endpoints should generally be avoided unless absolutely necessary. These endpoints are used to drive the Tenable Vulnerability Management UI, and not designed to be used programmatically.

Methods available on `io.editor`:

class EditorAPI(*api: APISession*)

audits(*etype: Literal['scan', 'policy']*, *object_id: int | str*, *file_id: int | str*, *fobj=None*)

Retrieves an audit file from Tenable Vulnerability Management

editor: audits

Parameters

- **etype** (*str*) – The type of template to retrieve. Must be either `scan` or `policy`.
- **object_od** (*int*) – The unique identifier of the object.
- **file_id** (*int*) – The unique identifier of the file to export.
- **fobj** (*FileObject*) – An optional File-like object to write the file to. If none is provided a BytesIO object will be returned.

Returns
A File-like object of of the audit file.

Return type
`file`

details(*etype*, *id*)

Constructs a valid scan document from the specified item.

> **Important**
>
> Please note that the details method is reverse-engineered from the responses from the editor API, and while we are reasonably sure that the response should align almost exactly to what the API expects to be pushed to it, this method by very nature of what it's doing isn't guaranteed to always work.

Parameters

- **etype** (*str*) – The type of object to request.
- **scan_id** (*int*) – The unique identifier for the scan.

Returns
The constructed scan configuration resource.

Return type
dict

Examples

```
>>> policy = tio.editor.details('scan', 1)
>>> pprint(scan)
```

obj_details(*etype*, *id*)

Retrieves details about a specific object.

editor: template-details

Parameters

- **etype** (*str*) – The type of object to retrieve. Must be either `scan` or `policy`.
- **id** (*int*) – The unique identifier of the object.

Returns
Details of the requested object

Return type
dict

plugin_description(*policy_id*, *family_id*, *plugin_id*)

Retrieves the plugin description for the specified plugin.

editor: plugin-description

Parameters

- **policy_id** (*int*) – The identifier of the policy.
- **family_id** (*int*) – The identifier of the plugin family.
- **plugin_id** (*int*) – The identifier of the plugin within the family.

Returns
Details of the plugin requested.

Return type
dict

template_details(*etype*, *uuid*)

Retrieves details about a specific template.

editor: template-details

Parameters

- **etype** (*str*) – The type of template to retrieve. Must be either `scan` or `policy`.
- **uuid** (*str*) – The UUID (unique identifier) for the template.

Returns
Details on the requested template

Return type
dict

template_list(*etype*)

List template objects.

editor: list

Parameters
etype (*str*) – The type of object to retrieve. Must be either `scan` or `policy`.

Returns
Listing of template records.

Return type
`list`

CHAPTER

TWELVE

EXCLUSIONS

The following methods allow for interaction into the Tenable Vulnerability Management exclusions API endpoints.

Methods available on `tio.exclusions`:

class ExclusionsAPI(*api: APISession*)

This will contain all methods related to exclusions

create(*name*, *members*, *start_time=None*, *end_time=None*, *timezone=None*, *description=None*, *frequency=None*, *interval=None*, *weekdays=None*, *day_of_month=None*, *enabled=True*, *network_id=None*)

Create a scan target exclusion.

exclusions: create

Parameters

- **name** (*str*) – The name of the exclusion to create.
- **members** (*list*) – The exclusions members. Each member should be a string with either a FQDN, IP Address, IP Range, or CIDR.
- **description** (*str, optional*) – Some further detail about the exclusion.
- **start_time** (*datetime*) – When the exclusion should start.
- **end_time** (*datetime*) – When the exclusion should end.
- **timezone** (*str, optional*) – The timezone to use for the exclusion. The default if none is specified is to use UTC. For the list of usable timezones, please refer to scans: timezones
- **frequency** (*str, optional*) – The frequency of the rule. The string inputted will be up-cased. Valid values are: `ONETIME`, `DAILY`, `WEEKLY`, `MONTHLY`, `YEARLY`. Default value is `ONETIME`.
- **interval** (*int, optional*) – The interval of the rule. The default interval is 1
- **weekdays** (*list, optional*) – List of 2-character representations of the days of the week to repeat the frequency rule on. Valid values are: *SU, MO, TU, WE, TH, FR, SA* Default values: `['SU', 'MO', 'TU', 'WE', 'TH', 'FR', 'SA']`
- **day_of_month** (*int, optional*) – The day of the month to repeat a **MONTHLY** frequency rule on. The default is today.
- **enabled** (*bool, optional*) – If enabled is true, the exclusion schedule is active. If enabled is false, the exclusion is "Always Active". The default value is `True`
- **network_id** (*uuid, optional*) – The ID of the network object associated with scanners where Tenable Vulnerability Management applies the exclusion.

Returns
Dictionary of the newly minted exclusion.

Return type
dict

Examples

Creating a one-time exclusion:

```
>>> from datetime import datetime, timedelta
>>> exclusion = tio.exclusions.create(
...     'Example One-Time Exclusion',
...     ['127.0.0.1'],
...     start_time=datetime.utcnow(),
...     end_time=datetime.utcnow() + timedelta(hours=1))
```

Creating a daily exclusion:

```
>>> exclusion = tio.exclusions.create(
...     'Example Daily Exclusion',
...     ['127.0.0.1'],
...     frequency='daily',
...     start_time=datetime.utcnow(),
...     end_time=datetime.utcnow() + timedelta(hours=1))
```

Creating a weekly exclusion:

```
>>> exclusion = tio.exclusions.create(
...     'Example Weekly Exclusion',
...     ['127.0.0.1'],
...     frequency='weekly',
...     weekdays=['mo', 'we', 'fr'],
...     start_time=datetime.utcnow(),
...     end_time=datetime.utcnow() + timedelta(hours=1))
```

Creating a monthly exclusion:

```
>>> exclusion = tio.exclusions.create(
...     'Example Monthly Exclusion',
...     ['127.0.0.1'],
...     frequency='monthly',
...     day_of_month=1,
...     start_time=datetime.utcnow(),
...     end_time=datetime.utcnow() + timedelta(hours=1))
```

Creating a yearly exclusion:

```
>>> exclusion = tio.exclusions.create(
...     'Example Yearly Exclusion',
...     ['127.0.0.1'],
...     frequency='yearly',
...     start_time=datetime.utcnow(),
...     end_time=datetime.utcnow() + timedelta(hours=1))
```

delete(*exclusion_id*)

Delete a scan target exclusion.

exclusions: delete

Parameters
exclusion_id (*int*) – The exclusion identifier to delete

Returns
The exclusion was successfully deleted.

Return type
None

Examples

```
>>> tio.exclusions.delete(1)
```

details(*exclusion_id*)

Retrieve the details for a specific scan target exclusion.

exclusions: details

Parameters
exclusion_id (*int*) – The exclusion identifier.

Returns
The exclusion record requested.

Return type
dict

Examples

```
>>> exclusion = tio.exclusions.details(1)
>>> pprint(exclusion)
```

edit(*exclusion_id*, *name=None*, *members=None*, *start_time=None*, *end_time=None*, *timezone=None*, *description=None*, *frequency=None*, *interval=None*, *weekdays=None*, *day_of_month=None*, *enabled=None*, *network_id=None*)

Edit an existing scan target exclusion.

exclusions: edit

The edit function will first gather the details of the exclusion that will be edited and will overlay the changes on top. The result will then be pushed back to the API to modify the exclusion.

Parameters

- **exclusion_id** (*int*) – The id of the exclusion object in Tenable Vulnerability Management
- **scanner_id** (*int, optional*) – The scanner id.
- **name** (*str, optional*) – The name of the exclusion to create.
- **description** (*str, optional*) – Some further detail about the exclusion.
- **start_time** (*datetime, optional*) – When the exclusion should start.

- **end_time** (*datetime, optional*) – When the exclusion should end.
- **timezone** (*str, optional*) – The timezone to use for the exclusion. The default if none is specified is to use UTC.
- **frequency** (*str, optional*) – The frequency of the rule. The string inputted will be upcased. Valid values are: *ONETIME, DAILY, WEEKLY, MONTHLY, YEARLY*.
- **interval** (*int, optional*) – The interval of the rule.
- **weekdays** (*list, optional*) – List of 2-character representations of the days of the week to repeat the frequency rule on. Valid values are: *SU, MO, TU, WE, TH, FR, SA* Default values: `['SU', 'MO', 'TU', 'WE', 'TH', 'FR', 'SA']`
- **day_of_month** (*int, optional*) – The day of the month to repeat a **MONTHLY** frequency rule on.
- **enabled** (*bool, optional*) – enable/disable exclusion.
- **network_id** (*uuid, optional*) – The ID of the network object associated with scanners where Tenable Vulnerability Management applies the exclusion.

Returns
Dictionary of the newly minted exclusion.

Return type
dict

Examples

Modifying the name of an exclusion:

```
>>> exclusion = tio.exclusions.edit(1, name='New Name')
```

exclusions_import(*fobj*)

Import exclusions into Tenable Vulnerability Management.

exclusions: import

Parameters
fobj (*FileObject*) – The file object of the exclusion(s) you wish to import.

Returns
Returned if Tenable Vulnerability Management successfully imports the exclusion file.

Return type
None

Examples

```
>>> with open('import_example.csv') as exclusion:
...     tio.exclusions.exclusions_import(exclusion)
```

list()

List the currently configured scan target exclusions.

exclusions: list

Returns
List of exclusion resource records.

Return type
list

Examples

```
>>> for exclusion in tio.exclusions.list():
...     pprint(exclusion)
```

CHAPTER

THIRTEEN

EXPORTS

The following methods allow for interaction into the Tenable Vulnerability Management exports API endpoints.

Methods available on `tio.exports`:

class ExportsAPI(*api: APISession*)

assets(*, *chunk_size: int = 1000*, *include_open_ports: bool | None = None*, *include_resource_tags: bool | None = None*, *created_at: ~datetime.datetime | int | None = None*, *updated_at: ~datetime.datetime | int | None = None*, *deleted_at: ~datetime.datetime | int | None = None*, *terminated_at: ~datetime.datetime | int | None = None*, *first_scan_time: ~datetime.datetime | int | None = None*, *last_authenticated_scan_time: ~datetime.datetime | int | None = None*, *last_assessed: ~datetime.datetime | int | None = None*, *is_deleted: bool | None = None*, *is_licensed: bool | None = None*, *is_terminated: bool | None = None*, *has_plugin_results: bool | None = None*, *last_scan_id: str | None = None*, *network_id: ~uuid.UUID | str | None = None*, *sources: list[str] | None = None*, *tags: list[tuple[str, list[str] | str]] | None = None*, *servicenow_sysid: bool | None = None*, *uuid: ~uuid.UUID | str | None = None*, *timeout: int | None = None*, *when_done: bool = False*, *adopt_existing: bool = True*, *iterator: ~typing.Type[~tenable.io.exports.iterator.ExportsIterator] | None = <class 'tenable.io.exports.iterator.ExportsIterator'>*) → *ExportsIterator* | UUID

Initiate an asset export.

API Documentation

Parameters

- **last_scan_id** – Scan uuid of the scan to be exported.
- **created_at** – Assets created after this timestamp will be returned.
- **deleted_at** – Assets deleted after this timestamp will be returned.
- **first_scan_time** – Assets with a first_scan time later that this timestamp will be returned.
- **last_assessed** – Assets last scanned after this timestamp will be returned.
- **last_authenticated_scan_time** – Assets last scanned with an authenticated scan after this timestamp will be returned.
- **terminated_at** – Assets terminated after this timestamp will be returned.
- **updated_at** – Assets updated after this timestamp will be returned.
- **has_plugin_results** – Should assets only be returned if they have plugin results?
- **is_deleted** – Should we return only assets that have been deleted?
- **is_licensed** – Should we return only assets that are licensed?
- **is_terminated** – Should we return assets that have been terminated?

- **servicenow_sysid** – Should we return assets that have a ServiceNOW sysid? if `True` only assets with an id will be returned. if `False` only assets without an id will be returned.
- **include_open_ports** – Should we include open ports of assets in the exported chunks?
- **chunk_size** – How many asset objects should be returned per chunk of data?
- **network_id** – Only assets within the specified network UUID will be returned.
- **sources** – Only assets with a source matching one of these source values will be returned. Note that this value is case-sensitive.
- **tags** (*list[tuple[str, str]], optional*) – A list of tag pairs to filter the results on. The tag pairs should be presented as `('CATEGORY', 'VALUE')`.
- **uuid** – A predefined export UUID to use for generating an ExportIterator. Using this parameter will ignore all of the filter arguments.
- **when_done** – When creating the iterator, setting this flag to true will tell the iterator to wait until the export job has completed before processing the first chunk. The default behaviour is to start processing chunks of data as soon as they become available.
- **timeout** – If specified, determines a timeout in seconds to wait for the export job to sit in the queue before cancelling the job and raising a `TioExportsTimeout` error. Once a job has started to be processed, the timeout is ignored.
- **iterator** – Supports overloading the iterator class to be used to process the datachunks. If set to `None`, then the job UUID will be returned instead of an iterator.
- **adopt_existing** – Should we automatically adopt an existing Job UUID with we receive a 409 conflict?

Examples

Iterating over the results of an asset export:

```
>>> for asset in tio.exports.assets():
...     print(asset)
```

Getting hosts that have been updated within the last 24 hours

```
>>> assets = tio.exports.assets(
...     updated_at=int(arrow.now().shift(days=-1).timestamp())
... )
```

Getting assets that have the the `Region:Chicago` tag:

```
>>> assets = tio.exports.assets(
...     tags=[('Region', 'Chicago')]
... )
```

assets_v2(*, *chunk_size: int = 1000*, *include_open_ports: bool | None = None*, *include_resource_tags: bool | None = None*, *since: ~datetime.datetime | int | None = None*, *created_at: ~datetime.datetime | int | None = None*, *updated_at: ~datetime.datetime | int | None = None*, *deleted_at: ~datetime.datetime | int | None = None*, *terminated_at: ~datetime.datetime | int | None = None*, *first_scan_time: ~datetime.datetime | int | None = None*, *last_authenticated_scan_time: ~datetime.datetime | int | None = None*, *last_assessed: ~datetime.datetime | int | None = None*, *is_deleted: bool | None = None*, *is_licensed: bool | None = None*, *is_terminated: bool | None = None*, *has_plugin_results: bool | None = None*, *servicenow_sysid: bool | None = None*, *last_scan_id: str | None = None*, *network_id: ~uuid.UUID | str | None = None*, *sources: list[str] | None = None*, *types: list[str] | None = None*, *iterator: ~typing.Type[~tenable.io.exports.iterator.ExportsIterator] | None = <class 'tenable.io.exports.iterator.ExportsIterator'>*, *timeout: int | None = None*, *adopt_existing: bool = True*, *when_done: bool = False*, *uuid: ~uuid.UUID | str | None = None*) → *ExportsIterator* | UUID

Initiate an asset v2 export.

API Documentation

Parameters

- **last_scan_id** – Scan uuid of the scan to be exported.
- **created_at** – Assets created after this timestamp will be returned.
- **deleted_at** – Assets deleted after this timestamp will be returned.
- **first_scan_time** – Return Assets with a first_scan time later that this timestamp will.
- **last_assessed** – Assets last scanned after this timestamp will be returned.
- **last_authenticated_scan_time** – Return Assets last scanned with an authenticated scan after this timestamp.
- **terminated_at** – Assets terminated after this timestamp will be returned.
- **updated_at** – Assets updated after this timestamp will be returned.
- **has_plugin_results** – Should assets only be returned if they have plugin results?
- **is_deleted** – Should we return only assets that have been deleted?
- **is_licensed** – Should we return only assets that are licensed?
- **is_terminated** – Should we return assets that have been terminated?
- **servicenow_sysid** – Should we return assets that have a ServiceNOW sysid? If `True` only assets with an id will be returned. If `False` only assets without an id will be returned.
- **include_open_ports** – Should we include open ports of assets in the exported chunks?
- **chunk_size** – How many asset objects should be returned per chunk of data?
- **network_id** – Only assets within the specified network UUID will be returned.
- **sources** – Only assets with a source matching one of these source values will be returned. Note that this value is case-sensitive.
- **types** – Only assets with specified type will be returned.
- **since** – Returns all assets that were updated, deleted, or terminated since the specified date regardless of state. The timestamp is specified in seconds since epoc (unix timestamp).
- **uuid** – A predefined export UUID to use for generating an ExportIterator. Using this parameter will ignore all of the filter arguments.

- **when_done** – When creating the iterator, setting this flag to true will tell the iterator to wait until the export job has completed before processing the first chunk. The default behaviour is to start processing chunks of data as soon as they become available.
- **timeout** – If specified, determines a timeout in seconds to wait for the export job to sit in the queue before cancelling the job and raising a `TioExportsTimeout` error. Once a job has started o be processed, the timeout is ignored.
- **iterator** – Supports overloading the iterator class to be used to process the datachunks. If set to `None`, then the job UUID will be returned instead of an iterator.
- **adopt_existing** – Should we automatically adopt an existing Job UUID with we receive a 409 conflict?

Examples

Iterating over the results of an asset export:

```
>>> for asset in tio.exports.assets_v2():
...     print(asset)
```

Getting hosts that have been updated within the last 24 hours

```
>>> assets = tio.exports.assets_v2(
...     updated_at=int(arrow.now().shift(days=-1).timestamp())
... )
```

Getting assets that have the the `host` type:

```
>>> assets = tio.exports.assets_v2(
...     types=['host']
... )
```

cancel(*export_type: Literal['vulns', 'assets', 'compliance', 'was']*, *export_uuid: UUID*, *version: str | None = None*) → str

Cancels the specified export job.

API Documentation for cancel export jobs with assets, compliance, and vulnerabilities datatypes.

Parameters

- **export_type** – The type of export job that we are to cancel.
- **export_uuid** – The export job's unique identifier.
- **version** – The export type version.

Returns

The status of the job.

Return type

str

Example

```
>>> tio.exports.cancel('vuln', '{UUID}')
'CANCELLED'
```

compliance(*, *num_findings: int = 5000*, *asset: list[~uuid.UUID | str] | None = None*, *last_seen: ~datetime.datetime | int | None = None*, *first_seen: ~datetime.datetime | int | None = None*, *last_observed: ~datetime.datetime | int | None = None*, *indexed_at: ~datetime.datetime | int | None = None*, *since: ~datetime.datetime | int | None = None*, *audit_name: str | None = None*, *audit_file_name: str | None = None*, *compliance_results: list[~typing.Literal['PASSED', 'FAILED', 'WARNING', 'SKIPPED', 'ERROR', 'UNKNOWN']] | None = None*, *ipv4_addresses: list[str | ~ipaddress.IPv4Address] | None = None*, *ipv6_addresses: list[str | ~ipaddress.IPv6Address] | None = None*, *network_id: ~uuid.UUID | str | None = None*, *plugin_id: list[int] | None = None*, *state: list[~typing.Literal['info', 'low', 'medium', 'high', 'critical']] | None = None*, *tags: list[tuple[str, list[str] | str]] | None = None*, *iterator: ~typing.Type[~tenable.io.exports.iterator.ExportsIterator] | None = <class 'tenable.io.exports.iterator.ExportsIterator'>*, *when_done: bool = False*, *uuid: ~uuid.UUID | str | None = None*, *timeout: int | None = None*, *adopt_existing: bool = True*) → *ExportsIterator* | UUID

Initiate a compliance export.

API Documentation

Parameters

- **asset** – A list of assets to return compliance results for.
- **first_seen** – Returns findings with a first seen time newer than the specified unix timestamp.
- **last_seen** – Returns findings with a last seen time newer than the specified unix timestamp.
- **ipv4_addresses** – Returns Compliance findings found for the provided list of ipv4 addresses.
- **ipv6_addresses** – Returns Compliance findings found for the provided list of ipv6 addresses.
- **plugin_name** – Returns Compliance findings for the specified list of plugin names.
- **plugin_id** – Returns Compliance findings for the specified list of plugin IDs.
- **audit_name** – Restricts compliance findings to those associated with the specified audit.
- **audit_file_name** – Restricts compliance findings to those associated with the specified audit file name.
- **compliance_results** – Restricts compliance findings to those associated with the specified list of compliance results, such as PASSED, FAILED, SKIPPED, ERROR, UNKNOWN etc.
- **last_observed** – Restricts compliance findings to those that were last observed on or after the specified unix timestamp.
- **indexed_at** – Restricts compliance findings to those that were updated or indexed into Tenable Vulnerability Management on or after the specified unix timestamp.
- **since** – Same as indexed_at. Restricts compliance findings to those that were updated or indexed into Tenable Vulnerability Management on or after the specified unix timestamp.

- **state** – Restricts compliance findings to those associated with the provided list of states, such as open, reopened and fixed.
- **tags** – A list of tag pairs to filter the results on. The tag pairs should be presented as `('CATEGORY', ['VALUE'])`.
- **network_id** – Returns Compliance findings for the specified network ID.
- **num_findings** – The number of findings to return per chunk of data.
- **uuid** – A predefined export UUID to use for generating an ExportIterator. Using this parameter will ignore all of the filter arguments.
- **when_done** – When creating the iterator, setting this flag to true will tell the iterator to wait until the export job has completed before processing the first chunk. The default behaviour is to start processing chunks of data as soon as they become available.
- **timeout** – If specified, determines a timeout in seconds to wait for the export job to sit in the queue before cancelling the job and raising a `TioExportsTimeout` error. Once a job has started to be processed, the timeout is ignored.
- **iterator** – Supports overloading the iterator class to be used to process the datachunks. If set to None, then the job ID will be returned instead of an iterator.
- **adopt_existing** – Should we automatically adopt an existing Job UUID with we receive a 409 conflict?

Examples

```
>>> for findings in tio.exports.compliance():
...     print(finding)
```

download_chunk(*export_type: Literal['vulns', 'assets', 'compliance', 'was']*, *export_uuid: UUID*, *chunk_id: int*, *version: str | None = None*, *retries: int = 3*) → list[dict[str, Any]]

Downloads an export chunk from the specified job.

API Documentation for downloading an export chunk for assets, compliance, and vulnerabilities.

Parameters

- **export_type** – The type of export job
- **export_uuid** – The export job's unique identifier.
- **chunk_id** – The identifier for the specific chunk to download.
- **version** – The export type version.

Returns

The list of objects that entail the chunk of data requested.

Example

```
>>> chunk = tio.exports.download_chunk('vulns', '{UUID}', 1)
```

initiate_export(*export_type: ~typing.Literal['vulns', 'assets', 'compliance', 'was'], *, version: str | None = None, when_done: bool = False, iterator: ~typing.Type[~tenable.io.exports.iterator.ExportsIterator] | None = <class 'tenable.io.exports.iterator.ExportsIterator'>, timeout: int | None = None, export_uuid: ~uuid.UUID | None = None, uuid: ~uuid.UUID | None = None, adopt_existing: bool = True, **payload*) → UUID

Initiate an export job of the specified export type, and return the export UUID.

This method accepts the key-value arguments supported by the methods assets(), vulns(), and compliance() for the matching export_type. For example, when the export_type is "assets", this function will only support the kwargs supported by the assets() method; if export_type is "vulns", the method will accept only those supported by the vulns() method, and so forth.

Deprecated since version 1.9.0: This method duplicates functionality that has existed within the bespoke export methods since 1.4.x. Thereforce this method has been flagged for removal. Please switch to using the appropriate export method and pass `iterator=None` in order to return a UUID instead of `ExportIterator`.

Parameters

- **export_type** – The datatype of export to get the jobs for.
- **version** – The export type version.

Examples

Initiating an assets export with no extra params.

```
>>> export_uuid = tio.exports.initiate_export("assets")
```

Initiating a vulns export with the params supported by vulns()

```
>>> export_uuid = tio.exports.initiate_export("vulns", timeout=10)
```

jobs(*export_type: Literal['vulns', 'assets', 'was'], version: str | None = None*) → dict[str, Any]

Returns the list of jobs available for a given datatype.

API Documentation for the job listing APIs for assets, and vulnerabilities datatypes.

Parameters

- **export_type** (*str*) – The datatype of export to get the jobs for.
- **version** – The export type version.

Examples

```
>>> jobs = tio.exports.jobs('vulns')
```

status(*export_type: Literal['vulns', 'assets', 'compliance', 'was']*, *export_uuid: UUID*, *version: str | None = None*) → dict[str, Any]

Gets the status of the export job.

API Documentation for the status of an export job for the assets, compliance, and vulnerabilities datatypes.

Parameters

- **export_type** (*str*) – The datatype of the export job.
- **export_uuid** (*str*) – The UUID of the export job.
- **version** – The export type version.

Examples

```
>>> status = tio.exports.status('vulns', '{UUID}')
```

vulns(***, *num_assets: int = 500*, *include_unlicensed: bool = True*, *since: ~datetime.datetime | int | None = None*, *first_found: ~datetime.datetime | int | None = None*, *first_seen: ~datetime.datetime | int | None = None*, *last_found: ~datetime.datetime | int | None = None*, *last_seen: ~datetime.datetime | int | None = None*, *last_fixed: ~datetime.datetime | int | None = None*, *resurfaced_date: ~datetime.datetime | int | None = None*, *indexed_at: ~datetime.datetime | int | None = None*, *time_taken_to_fix: dict[str, int] | None = None*, *cvss4_base_score: dict[str, float] | None = None*, *epss_score: dict[str, float] | None = None*, *cidr_range: ~ipaddress.IPv4Network | str | None = None*, *cve_id: list[str] | None = None*, *cve_category: list[~typing.Literal['cisa known exploitable', 'emerging threats', 'in the news', 'persistently exploited', 'ransomware', 'recent active exploitation', 'top 50 vpr']] | None = None*, *exploit_maturity: list[~typing.Literal['high', 'functional', 'poc', 'unproven']] | None = None*, *initiative_id: ~uuid.UUID | str | None = None*, *network_id: ~uuid.UUID | str | None = None*, *plugin_family: list[str] | None = None*, *plugin_id: list[int] | None = None*, *plugin_type: str | None = None*, *scan_uuid: ~uuid.UUID | str | None = None*, *severity: list[~typing.Literal['info', 'low', 'medium', 'high', 'critical']] | None = None*, *severity_modification_type: list[~typing.Literal['NONE', 'ACCEPTED', 'RECASTED']] | None = None*, *state: list[~typing.Literal['OPEN', 'REOPENED', 'FIXED']] | None = None*, *source: list[str] | None = None*, *vpr_score: dict[str, float] | None = None*, *vpr_v2_score: dict[str, float] | None = None*, *vpr_threat_intensity: list[~typing.Literal['very high', 'high', 'medium', 'low', 'very low']] | None = None*, *weaponization: list[~typing.Literal['apt', 'botnet', 'malware', 'ransomware', 'rootkit']] | None = None*, *tags: list[tuple[str, list[str] | str]] | None = None*, *uuid: ~uuid.UUID | str | None = None*, *timeout: int | None = None*, *when_done: bool = False*, *adopt_existing: bool = True*, *iterator: ~typing.Type[~tenable.io.exports.iterator.ExportsIterator] | None = <class 'tenable.io.exports.iterator.ExportsIterator'>*) → *ExportsIterator* | UUID

Initiate a vulnerability export.

API Documentation

Parameters

- **first_found** – Findings first discovered after this timestamp will be returned.
- **indexed_at** – Findings indexed into Tenable Vulnerability Management after this timestamp will be returned.
- **last_fixed** – Findings fixed after this timestamp will be returned. Note that this filter only applies to fixed data and should not be used when searching for active findings.

- **last_found** – Findings last observed after this timestamp will be returned.
- **since** – Findings last observed in any state after this timestamp will be returned. Cannot be used with `last_found`, `first_found`, or `last_fixed`.
- **resurfaced_date** – Returns findings that have been resurfaced on or after this datetime.
- **time_taken_to_fix** – Returns findings based on how long that it took the organization to resolve. The export will only inclucled `FIXED` findings when this filter is applied. Supported keys are `lte` for *Less Than or Equal to* or `gte` for *Greater Than or Equal to*. The values should be in seconds.
- **plugin_family** – Only return findings from the specified plugin families.
- **plugin_id** – Only return findings from the specified plugin ids.
- **plugin_type** – Only return findings with the specified plugin type.
- **scan_uuid** – Only return findings with the specified scan UUID.
- **source** – Only return vulnerabilities for assets that have the specified scan source.
- **severity_modification_type** – Only return vulnerabilities with the specified severity modification type.
- **severity** – Only return findings with the specified severities.
- **state** – Only return findings with the specified states.
- **vpr_score** – Only returns findings that meet the specified VPR criteria. The filter is formatted as a dictionary with the mathematical operation as the key. Supported operations are:

Table 1: Supported Operations

Operation	Type	Description
`eq`	list[float]	List of VPR scores that the findings must match.
`neq`	list[float]	List of VPR scores that the findings can not match.
`gt`	float	VPR scores must be greater than the specified value.
`gte`	float	VPR scores must be greater than or equal to the specified value.
`lt`	float	VPR scores must be less than the specified value.
`lte`	float	VPR scores must be less than or equal to the specified value.

- **network_id** – Only findings within the specified network UUID will be returned.
- **cidr_range** – Restrict the export to only vulns assigned to assets within the CIDR range specified.
- **tags** – A list of tag pairs to filter the results on. The tag pairs should be presented as `('CATEGORY', 'VALUE')`.
- **include_unlicensed** – Should findings for unlicensed assets that be included in the results?
- **num_assets** – As findings are grouped by asset, how many assets's findings should exist within each data chunk?
- **uuid** – A predefined export UUID to use for generating an ExportIterator. Using this parameter will ignore all of the filter arguments.

- **when_done** – When creating the iterator, setting this flag to true will tell the iterator to wait until the export job has completed before processing the first chunk. The default behaviour is to start processing chunks of data as soon as they become available.
- **timeout** – If specified, determines a timeout in seconds to wait for the export job to sit in the queue before cancelling the job and raising a `TioExportsTimeout` error. Once a job has started to be processed, the timeout is ignored.
- **iterator** – Supports overloading the iterator class to be used to process the datachunks. If set to None, then the job ID will be returned instead of an iterator.
- **adopt_existing** – Should we automatically adopt an existing Job UUID with we receive a 409 conflict?
- **cve_id** – Returns findings that match the specified CVE IDs.
- **cve_category** – Returns findings the match the specified CVE category. For more information about categories, see the *Vulnerability Categories* section in the Tenable Vulnerability Management User Guide.
- **exploit_maturity** – Returns findings that match the specified exploit maturity. Tenable assigns exploit maturity values to vulnerabilities based on the availability and sophistication of exploit code. Supported values are `high`, `functional`, `poc`, `unproven`.
- **vpr_threat_intensity** – Returns findings that match the specified threat intensity. The threat intensity of a vulnerability is based onf the number and frequency of recently observed events. Supported values are `very high`, `high`, `medium`, `low`, `very low`.
- **weaponization** – Returns findings that match the specified weaponizations. Weaponized vulnerabilities are vulnerabilities that are ready for use in a particular type of attack. Supported values are `apt`, `botnet`, `malware`, `ransomware`, `rootkit`.

Examples

Examples:

Iterating over the results of an vuln export:

```
>>> for vuln in tio.exports.vulns():
...     print(vuln)
```

Getting findings that have been observed within the last 24 hours

```
>>> vulns = tio.exports.vulns(
...     since=int(arrow.now().shift(days=-1).timestamp())
... )
```

Getting findings that have the the `Region:Chicago` tag:

```
>>> vulns = tio.exports.vulns(
...     tags=[('Region', 'Chicago')]
... )
```

was(*, *num_assets: int = 500*, *include_unlicensed: bool = True*, *since: ~datetime.datetime | int | None = None*, *first_found: ~datetime.datetime | int | None = None*, *last_fixed: ~datetime.datetime | int | None = None*, *last_found: ~datetime.datetime | int | None = None*, *indexed_at: ~datetime.datetime | int | None = None*, *asset_uuid: list[~uuid.UUID | str] | None = None*, *asset_name: str | None = None*, *cvss4_base_score: dict[str, float] | None = None*, *epss_score: dict[str, float] | None = None*, *ipv4s: list[~ipaddress.IPv4Address | str] | None = None*, *owasp_2010: list[~typing.Literal['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10']] | None = None*, *owasp_2013: list[~typing.Literal['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10']] | None = None*, *owasp_2017: list[~typing.Literal['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10']] | None = None*, *owasp_2021: list[~typing.Literal['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10']] | None = None*, *owasp_api_2019: list[~typing.Literal['API1', 'API2', 'API3', 'API4', 'API5', 'API6', 'API7', 'API8', 'API9', 'API10']] | None = None*, *plugin_ids: list[int] | None = None*, *severity: list[~typing.Literal['info', 'low', 'medium', 'high', 'critical']] | None = None*, *severity_modification_type: list[~typing.Literal['NONE', 'ACCEPTED', 'RECASTED']] | None = None*, *state: list[~typing.Literal['OPEN', 'REOPENED', 'FIXED']] | None = None*, *vpr_score: dict[str, float] | None = None*, *vpr_v2_score: dict[str, float] | None = None*, *uuid: ~uuid.UUID | str | None = None*, *timeout: int | None = None*, *when_done: bool = False*, *adopt_existing: bool = True*, *iterator: ~typing.Type[~tenable.io.exports.iterator.ExportsIterator] | None = <class 'tenable.io.exports.iterator.ExportsIterator'>*) → *ExportsIterator* | UUID

Initiate a WAS vulnerability export. API Documentation

Parameters

- **first_found** – Findings first discovered after this timestamp will be returned.
- **indexed_at** – Findings indexed after this timestamp will be returned.
- **last_fixed** – Findings fixed after this timestamp will be returned. Note that this filter only applies to fixed data and should not be used when searching for active findings.
- **last_found** – Findings last observed after this timestamp will be returned.
- **since** – Findings last observed in any state after this timestamp will be returned. Cannot be used with `last_found`, `first_found`, or `last_fixed`.
- **plugin_ids** – Only return findings from the specified plugin ids.
- **asset_uuid** – Only return findings for the assets with specific asset-uuids.
- **asset_name** – Only return findings for the asset with given name.
- **owasp_2010** – A list of chapters from the OWASP Categories 2010 report for which you want to filter findings returned in the findings export.
- **owasp_2013** – A list of chapters from the OWASP Categories 2013 report for which you want to filter findings returned in the findings export.
- **owasp_2017** – A list of chapters from the OWASP Categories 2017 report for which you want to filter findings returned in the findings export.
- **owasp_2021** – A list of chapters from the OWASP Categories 2021 report for which you want to filter findings returned in the findings export.
- **owasp_api_2019** – A list of chapters from the OWASP Categories API 2019 report for which you want to filter findings returned in the findings export.
- **severity_modification_type** – Only return vulnerabilities with the specified modification type.
- **severity** – Only return findings with the specified severities.
- **state** – Only return findings with the specified states.

- **vpr_score** – Only returns findings that meet the specified VPR criteria. The filter is formatted as a dictionary with the mathematical operation as the key. Supported operations are:

Table 2: Supported Operations

Operation	Type	Description
`eq`	list[float]	List of VPR scores that the findings must match.
`neq`	list[float]	List of VPR scores that the findings can not match.
`gt`	float	VPR scores must be greater than the specified value.
`gte`	float	VPR scores must be greater than or equal to the specified value.
`lt`	float	VPR scores must be less than the specified value.
`lte`	float	VPR scores must be less than or equal to the specified value.

- **ipv4s** – Restrict the export to only vulns assigned to assets with specific ip-addresses.
- **include_unlicensed** – Should findings for assets that are not licensed be included in the results?
- **num_assets** – As findings are grouped by asset, how many assets' findings should exist within each data chunk.
- **uuid** – A predefined export UUID to use for generating an ExportIterator. Using this parameter will ignore all the filter arguments.
- **when_done** – When creating the iterator, setting this flag to true will tell the iterator to wait until the export job has completed before processing the first chunk. The default behaviour is to start processing chunks of data as soon as they become available.
- **timeout** – If specified, determines a timeout in seconds to wait for the export job to sit in the queue before cancelling the job and raising a `TioExportsTimeout` error. Once a job has started to be processed, the timeout is ignored.
- **iterator** – Supports overloading the iterator class to be used to process the datachunks. If set to None, then the job ID will be returned instead of an iterator.
- **adopt_existing** – Should we automatically adopt an existing Job UUID with we receive a 409 conflict?

Examples

Iterating over the results of a WAS vuln export: >>> from tenable.io import TenableIO >>> tio = TenableIO("<apiKey>", "secret") >>> for vuln in tio.exports.was_vulns(): … print(vuln)

Getting findings that have been observed within the last 24 hours >>> import arrow >>> vulns = tio.exports.was(… since=int(arrow.now().shift(days=-1).timestamp()) …)

As exports are asynchronous, pyTenable by default will return an iterator to handle the state tracking, data chunking, and presentation of the data in order to reduce the amount of boilerplate code that would otherwise have to be created. These iterators support both serial iteration and threaded handling of data depending on how the data is accessed.

class ExportsIterator(*api*, ***kwargs*)

The export iterator can be used to handle the downloading and processing of the data chunks from an export request.

boxify: bool = False

Should items be returned as box objects?

cancel() → None

Cancels the current export

chunk_id: int | None = None

Id of the chunk currently being processed.

chunks: list[int]

Current list of available chunks.

count: int = 0

Number of objects processed so far within the current page (chunk).

next()

Get the next item in the current page

page_count: int = 0

Number of pages (chunks) processed so far across the entire export job.

processed: list[int]

Chunks that have already been processed.

run_threaded(*func: Callable*, *kwargs: dict[str, Any] | None = None*, *num_threads: int = 2*) → list[Future]

Initiate a multi-threaded export using the provided function and keyword arguments. The following field names are reserved and must be accepted either as an optional keyword argument, or as a named param.

- data (list[dict]): Receiver of the data-chunk.
- export_uuid (str): Receiver for the export job UUID.
- export_type (str): Receiver for the export data-type.
- export_chunk_id (int): Receiver for the export chunk id.
- version (int): Receiver for the export version.

Parameters

- **func** – The function to pass to the thread executor.
- **kwargs** – Any additional keyword arguments that are to be passed to the function as part of execution.
- **num_threads** – How many concurrent threads should be run. The default is 2.

Examples

A simple example to download the chunks and write them to disk.

```
>>> def write_chunk(data,
...                 export_uuid: str,
...                 export_type: str,
...                 export_chunk_id: int,
...                 version: int
...                 ):
...     fn = f'{export_type}-{export_uuid}-{export_chunk_id}-{version}.json'
...     with open(fn, 'w') as fobj:
```

(continues on next page)

(continued from previous page)

```
...         json.dump(data, fobj)
>>>
>>> export = tio.exports.vulns()
>>> export.run_threaded(write_chunk, num_threads=4)
```

start_time: *int*

Export job start time (unix timestamp).

status: *str*

Export job status.

timeout: *int | None = None*

Max seconds to wait in pending before cancelling the job.

type: *str*

Export job type.

uuid: *str*

Export job id.

version: *str*

Export job version.

CHAPTER

FOURTEEN

FILES

The following methods allow for interaction into the Tenable Vulnerability Management file API endpoints.

Methods available on `tio.files`:

class FileAPI(*api: APISession*)

upload(*fobj: BinaryIO*, *encrypted: bool = False*)

Uploads a file into Tenable Vulnerability Management.

file: upload

Parameters

- **fobj** (*FileObject*) – The file object intended to be uploaded into Tenable Vulnerability Management.
- **encrypted** (*bool, optional*) – If the file is encrypted, set the flag to True.

Returns

The fileuploaded attribute

Return type

str

Examples

```
>>> with open('file.txt') as fobj:
...     file_id = tio.files.upload(fobj)
```

CHAPTER

FIFTEEN

FILTERS

The following methods allow for interaction into the Tenable Vulnerability Management filters API endpoints.

Methods available on `tio.filters`:

class FiltersAPI(*api: APISession*)

This will contain all methods related to filters

access_group_asset_rules_filters(*normalize: bool = True*, *expire_age: int = 60*) → dict[str, Any]

Returns access group rules filters.

filters: access-control-rules-filters

Parameters

- **normalize** – Should the response be converted into the same structure used for the filter cache?
- **expire_age** – How many seconds old can the cache be before forcing a refresh?

Returns
Filter resource dictionary

Return type
dict

Examples

```
>>> filters = tio.filters.access_group_rules_filters()
```

access_group_asset_rules_filters_v2(*normalize: bool = True*, *expire_age: int = 60*) → dict[str, Any]

Returns access group rules filters v2.

filters: access_group_asset_rules_filters_v2

Parameters

- **normalize** – Should the response be converted into the same structure used for the filter cache?
- **expire_age** – How many seconds old can the cache be before forcing a refresh?

Returns
Filter resource dictionary

Return type
dict

Examples

```
>>> filters = tio.filters.access_group_rules_filters_v2()
```

access_group_filters(*normalize: bool = True*, *expire_age: int = 60*) → dict[str, Any]

Returns access group filters.

filters: access-group-filters

Parameters

- **normalize** – Should the response be converted into the same structure used for the filter cache?
- **expire_age** – How many seconds old can the cache be before forcing a refresh?

Returns
Filter resource dictionary

Return type
dict

Examples

```
>>> filters = tio.filters.access_group_filters()
```

access_group_filters_v2(*normalize=True*)

Returns access group filters v2.

filters: access_group_filters_v2

Parameters

- **normalize** – Should the response be converted into the same structure used for the filter cache?
- **expire_age** – How many seconds old can the cache be before forcing a refresh?

Returns
Filter resource dictionary

Return type
dict

Examples

```
>>> filters = tio.filters.access_group_filters_v2()
```

agents_filters(*normalize: bool = True*, *expire_age: int = 60*) → dict[str, Any]

Returns agent filters.

filters: agents-filters

Parameters

- **normalize** – Should the response be converted into the same structure used for the filter cache?
- **expire_age** – How many seconds old can the cache be before forcing a refresh?

Returns
Filter resource dictionary

Return type
dict

Examples

```
>>> filters = tio.filters.agents_filters()
```

asset_tag_filters(*normalize: bool = True*, *expire_age: int = 60*) → dict[str, Any]

Returns a list of filters that you can use to create the rules for applying dynamic tags.

tag: list asset tag filters

Parameters

- **normalize** – Should the response be converted into the same structure used for the filter cache?
- **expire_age** – How many seconds old can the cache be before forcing a refresh?

Returns
Filter resource dictionary

Return type
dict

Examples

```
>>> tio.filters.asset_tag_filters()
```

credentials_filters(*normalize: bool = True*, *expire_age: int = 60*) → dict[str, Any]

Returns the individual scan filters.

filters: credentials

Parameters

- **normalize** – Should the response be converted into the same structure used for the filter cache?
- **expire_age** – How many seconds old can the cache be before forcing a refresh?

Returns
Filter resource dictionary

Return type
dict

Examples

```
>>> filters = tio.filters.scan_filters()
```

networks_filters() → dict[str, Any]

Returns the networks filters.

Returns
Filter resource dictionary

Return type
dict

Examples

```
>>> filters = tio.filters.network_filters()
```

scan_filters(*normalize: bool = True*, *expire_age: int = 60*) → dict[str, Any]

Returns the individual scan filters.

Parameters

- **normalize** – Should the response be converted into the same structure used for the filter cache?
- **expire_age** – How many seconds old can the cache be before forcing a refresh?

Returns
Filter resource dictionary

Return type
dict

Examples

```
>>> filters = tio.filters.scan_filters()
```

workbench_asset_filters(*normalize: bool = True*, *expire_age: int = 60*) → dict[str, Any]

Returns the asset workbench filters.

workbenches: assets-filters

Parameters

- **normalize** – Should the response be converted into the same structure used for the filter cache?
- **expire_age** – How many seconds old can the cache be before forcing a refresh?

Returns
Filter resource dictionary

Return type
dict

Examples

```
>>> filters = tio.filters.workbench_asset_filters()
```

workbench_vuln_filters(*normalize: bool = True*, *expire_age: int = 60*) → dict[str, Any]

Returns the vulnerability workbench filters

workbenches: vulnerabilities-filters

Parameters

- **normalize** – Should the response be converted into the same structure used for the filter cache?
- **expire_age** – How many seconds old can the cache be before forcing a refresh?

Returns
Filter resource dictionary

Return type
dict

Examples

```
>>> filters = tio.filters.workbench_vuln_filters()
```

CHAPTER

SIXTEEN

FOLDERS

The following methods allow for interaction into the Tenable Vulnerability Management folders API endpoints.

Methods available on `tio.folders`:

class FoldersAPI(*api: APISession*)

create(*name*)

Create a folder.

folders: create

Parameters
name (*str*) – The name of the new folder.

Returns
The new folder id.

Return type
int

Examples

```
>>> folder = tio.folders.create('New Folder Name')
```

delete(*id*)

Delete a folder.

folders: delete

Parameters
id (*int*) – The unique identifier for the folder.

Returns
None

Examples

```
>>> tio.folders.delete(1)
```

edit(*id*, *name*)

Edit a folder.

folders: edit

Parameters

- **id** (*int*) – The unique identifier for the folder.
- **name** (*str*) – The new name for the folder.

Returns

The folder was successfully renamed.

Return type

None

Examples

```
>>> tio.folders.edit(1, 'Updated Folder Name')
```

list()

Lists the available folders.

folders: list

Returns

List of folder resource records.

Return type

list

Examples

```
>>> for folder in tio.folders.list():
...     pprint(folder)
```

CHAPTER

SEVENTEEN

GROUPS

The following methods allow for interaction into the Tenable Vulnerability Management groups API.

Methods available on `tio.groups`:

class GroupsAPI(*api: APISession*)

add_user(*group_id*, *user_id*)

Add a user to a user group.

groups: add-user

Parameters

- **group_id** (*int*) – The unique identifier of the group to add the user to.
- **user_id** (*int*) – The unique identifier of the user to add.

Returns

The user was successfully added to the group.

Return type

None

Examples

```
>>> tio.groups.add_user(1, 1)
```

create(*name*)

Create a new user group.

groups: create

Parameters

name (*str*) – The name of the group that will be created.

Returns

The group resource record of the newly minted group.

Return type

dict

Examples

```
>>> group = tio.groups.create('Group Name')
```

delete(*id*)

Delete a user group.

groups: delete

Parameters
id (*int*) – The unique identifier for the group to be deleted.

Returns
The group was successfully deleted.

Return type
None

Examples

```
>>> tio.groups.delete(1)
```

delete_user(*group_id*, *user_id*)

Delete a user from a user group.

groups: delete-user

Parameters

- **group_id** (*int*) – The unique identifier for the group to be modified.
- **user_id** (*int*) – The unique identifier for the user to be removed from the group.

Returns
The user was successfully removed from the group.

Return type
None

Examples

```
>>> tio.groups.delete_user(1, 1)
```

edit(*id*, *name*)

Edit a user group.

groups: edit

Parameters

- **id** (*int*) – The unique identifier for the group to be modified.
- **name** (*str*) – The new name for the group.

Returns
The group resource record.

Return type
dict

Examples

```
>>> tio.groups.edit(1, 'Updated name')
```

list()

Lists all of the available user groups.

groups: list

Returns
List of the group resource records

Return type
list

Examples

```
>>> for group in tio.groups.list():
...     pprint(group)
```

list_users(*id*)

List the user memberships within a specific user group.

groups: list-users

Parameters
id (*int*) – The unique identifier of the group requested.

Returns
List of user resource records based on membership to the specified group.

Return type
list

Example

```
>>> for user in tio.groups.list_users(1):
...     pprint(user)
```

CHAPTER

EIGHTEEN

NETWORKS

The following methods allow for interaction into the Tenable Vulnerability Management networks API endpoints.

Methods available on `tio.networks`:

class NetworksAPI(*api: APISession*)

This will contain all methods related to networks

assign_scanners(*network_id*, **scanner_uuids*)

Assigns one or many scanners to a network.

networks: assign-scanner networks: bulk-assign-scanner

Parameters

- **network_id** (*str*) – The UUID of the network.
- ***scanner_uuids** (*str*) – Scanner UUID(s) to assign to the network.

Examples

Assign a single scanner:

```
>>> tio.networks,assign_scanners(
...     '00000000-0000-0000-0000-000000000000', # Network UUID
...     '00000000-0000-0000-0000-000000000000') # Scanner UUID
```

Assign multiple scanners:

```
>>> tio.networks,assign_scanners(
...     '00000000-0000-0000-0000-000000000000', # Network UUID
...     '00000000-0000-0000-0000-000000000000', # Scanner1 UUID
...     '00000000-0000-0000-0000-000000000000') # Scanner2 UUID
```

create(*name*, *description=None*, *assets_ttl_days=None*)

Creates a new network within Tenable Vulnerability Management

networks: create

Parameters

- **name** (*str*) – The name of the new network.
- **description** (*str, optional*) – Description of the network.
- **assets_ttl_days** (*int, optional*) – The number of days to wait before assets age out.

- **days.** (*Assets will be permanently deleted if they are not seen on a scan within the specified number of*)
- **value** (*Maximum*) – 90
- **value** – 365

Returns
The resource record of the newly created network.

Return type
dict

Examples

```
>>> nw = tio.networks.create('Example')
```

delete(*network_id*)

Deletes the specified network.

networks: delete

Parameters
network_id (*str*) – The UUID of the network to remove.

Examples

```
>>> tio.networks.delete('00000000-0000-0000-0000-000000000000')
```

details(*network_id*)

Retrieves the details of the specified network.

networks: details

Parameters
network_id (*str*) – The UUID of the network.

Examples

```
>>> nw = tio.networks.details('00000000-0000-0000-0000-000000000000')
```

edit(*network_id*, *name*, *description=None*, *assets_ttl_days=None*)

Updates the specified network resource.

networks: update

Parameters

- **network_id** (*str*) – The UUID of the network resource to update.
- **name** (*str*) – The new name of the network resource.
- **description** (*str, optional*) – The new description of the network resource.
- **assets_ttl_days** (*int, optional*) – The number of days to wait before assets age out.
- **days.** (*Assets will be permanently deleted if they are not seen on a scan within the specified number of*)

- **value** (*Maximum*) – 90
- **value** – 365

Returns
The updates network resource.

Return type
dict

Examples

```
>>> nw = tio.networks.edit('00000000-0000-0000-0000-000000000000',
...     'Updated Network', 'Updated Description', 180)
```

list(**filters*, ***kw*)

Get the listing of configured networks from Tenable Vulnerability Management.

networks: list

Parameters

- ***filters** (*tuple, optional*) – Filters are tuples in the form of ('NAME', 'OPERATOR', 'VALUE'). Multiple filters can be used and will filter down the data being returned from the API.

  ### Examples

  ```
  – ('name', 'eq', 'example')
  ```

  As the filters may change and sortable fields may change over time, it's highly recommended that you look at the output of the `tio.networks.network_filters()` endpoint to get more details.
- **filter_type** (*str, optional*) – The filter_type operator determines how the filters are combined together. `and` will inform the API that all of the filter conditions must be met for an access group to be returned, whereas `or` would mean that if any of the conditions are met, the access group record will be returned.
- **include_deleted** (*bool, optional*) – Indicates whether deleted network objects should be included in the response. If left unspecified, the default is `False`.
- **limit** (*int, optional*) – The number of records to retrieve. Default is 50
- **offset** (*int, optional*) – The starting record to retrieve. Default is 0.
- **sort** (*tuple, optional*) – A tuple of tuples identifying the the field and sort order of the field.
- **wildcard** (*str, optional*) – A string to pattern match against all available fields returned.
- **wildcard_fields** (*list, optional*) – A list of fields to optionally restrict the wildcard matching to.

Returns
An iterator that handles the page management of the requested records.

Return type
NetworksIterator

Examples

Getting the listing of all agents:

```
>>> for nw in tio.networks.list():
...     pprint(nw)
```

Retrieving all of the windows agents:

```
>>> for nw in tio.access_groups.list(('name', 'match', 'win')):
...     pprint(nw)
```

list_scanners(*network_id*)

Retrieves the list of scanners associated to a given network.

networks: list-scanners

Parameters
network_id (*str*) – The UUID of the network.

Returns
List of scanner resources associated to this network.

Return type
list

Examples

```
>>> network = '00000000-0000-0000-0000-000000000000'
>>> for scanner in tio.networks.list_scanners(network):
...     pprint(scanner)
```

network_asset_count(*network_id: str*, *num_days: int*)

get the total number of assets in the network along with the number of assets that have not been seen for the specified number of days.

networks: network_asset_count

Parameters

- **network_id** (*str*) – The UUID of the network.
- **num_days** (*int*) – count of assets that have not been seen for the specified number of days

Returns
Returns the total number of assets in the network along with the number of assets that have not been seen for the specified number of days.

Return type
dict

Examples

```
>>> network = '00000000-0000-0000-0000-000000000000'
>>> count = tio.networks.network_asset_count(network, 180)
```

unassigned_scanners(*network_id*)

Retrieves the list of scanners that are currently unassigned to the given network. This will include scanners and scanner groups that are currently assigned to the default network.

networks: list-assignable-scanners

Parameters
id (*str*) – The UUID of the network.

Returns
The list of unassigned scanner resources

Return type
list

Examples

```
>>> network = '00000000-0000-0000-0000-000000000000'
>>> for scanner in tio.networks.unassigned_scanners(network):
...     pprint(scanner)
```

CHAPTER

NINETEEN

PERMISSIONS

The following methods allow for interaction into the Tenable Vulnerability Management :devportal`permissions <permissions-1>` API endpoints.

Methods available on `tio.permissions`:

class PermissionsAPI(*api: APISession*)

change(*otype*, *id*, **acls*)

Modify the permission of a specific object.

permissions: change

Parameters

- **otype** (*str*) – The type of object to change.
- **id** (*int*) – The unique identifier of the object.
- ***acls** (*dict*) – ACL dictionaries inform Tenable Vulnerability Management how to handle permissions of the various objects within Tenable Vulnerability Management. Please refer to the permissions documentation for more details.

Returns
The object permissions were successfully changed.

Return type
None

list(*otype*, *id*)

List the permissions of a specific object.

permissions: list

Parameters

- **otype** (*str*) – The type of object being queried.
- **id** (*int*) – The unique identifier of the object.

Returns
The permission recourse record listings for the specified object.

Return type
list

CHAPTER

TWENTY

PLUGINS

The following methods allow for interaction into the Tenable Vulnerability Management plugins API endpoints.

Methods available on `tio.plugins`:

class PluginsAPI(*api: APISession*)

This will contain all methods related to plugins

families()

List the available plugin families.

plugins: families

Returns
List of plugin family resource records.

Return type
list

Examples

```
>>> for family in tio.plugins.families():
...     pprint(family)
```

family_details(*family_id*)

Retrieve the details for a specific plugin family.

plugins: family-details

Parameters
family_id (*int*) – The plugin family unique identifier.

Returns
Returns a dictionary stating the id, name, and plugins that are housed within the plugin family.

Return type
dict

Examples

```
>>> family = tio.plugins.family_details(1)
```

list(*page=None*, *size=None*, *last_updated=None*, *num_pages=None*)

Get the listing of plugin details from Tenable Vulnerability Management.

plugins: list

Parameters

- **size** (*int, optional*) – The number of records to retrieve. Default is 1000
- **page** (*int, optional*) – The starting page to retrieve. Default is 0.
- **last_updated** (*date, optional*) – A datetime.date object stating when the threshold for the last updated field can be for a plugin.
- **num_pages** (*int, optional*) – The total number of pages to request before stopping the iterator.

Returns
An iterator that handles the page management of the requested records.

Return type
PluginsIterator

Examples

Getting the listing of all plugins:

```
>>> for plugin in tio.plugins.list():
...     pprint(plugin)
```

Retrieving all of the plugins updated since 2019-01-01:

```
>>> for plugin in tio.plugins.list(last_updated=date(2019, 1, 1)):
...     pprint(plugin)
```

Informing the iterator to cache the plugin family data for injection into each item:

```
>>> plugins = tio.plugins.list(last_updated=date(2019, 1, 1))
>>> plugins.populate_maptable = True
>>> for plugin in plugins:
...     pprint(plugin)
```

plugin_details(*plugin_id*)

Retrieve the details for a specific plugin.

plugins: plugin-details

Parameters
plugin_id (*int*) – The plugin id for the requested plugin.

Returns
A dictionary stating the id, name, family, and any other relevant attributes associated to the plugin.

Return type
dict

Examples

```
>>> plugin = tio.plugins.plugin_details(19506)
>>> pprint(plugin)
```

CHAPTER

TWENTYONE

POLICIES

The following methods allow for interaction into the Tenable Vulnerability Management policies API.

Methods available on `tio.policies`:

class PoliciesAPI(*api: APISession*)

configure(*id*, *policy*)

Configures an existing policy.

policies: configure

Parameters

- **id** (*int*) – The policy unique identifier.
- **policy** (*dict*) – The updated policy definition to push into Tenable Vulnerability Management. As these policies can be quite complex, please refer to the documentation in the policies: configure page (linked above).

Returns

Policy successfully modified.

Return type

None

Examples

```
>>> policy = tio.policies.details(1)
>>> policy['settings']['name'] = 'Updated Policy Name'
>>> tio.policies.configure(policy)
```

copy(*id*)

Duplicates a scan policy and returns the copy.

policies: copy

Parameters

id (*int*) – The unique identifier of the policy you wish to copy.

Returns

A dictionary containing the name and id of the policy copy.

Return type

dict

Example

```
>>> policy = tio.policies.copy(1)
```

create(*policy*)

Creates a new scan policy based on the policy dictionary passed.

policies: configure

Parameters
policy (*dict*) – The policy definition to push into Tenable Vulnerability Management. As these policies can be quite complex, please refer to the documentation in the policies: configure page (linked above).

Returns
A dictionary containing the name and id of the new policy.

Return type
dict

Examples

```
>>> policy = tio.policies.template_details('basic')
>>> policy['settings']['name'] = 'New Scan Policy'
>>> info = tio.policies.create(policy)
```

delete(*id*)

Delete a custom policy.

policies: delete

Parameters
id (*int*) – The unique identifier of the policy to delete.

Returns
The policy was successfully deleted.

Return type
None

Examples

```
>>> tio.policies.delete(1)
```

details(*id*)

Retrieve the details for a specific policy.

policies: details

Parameters
id (*int*) – The unique identifier of the policy.

Returns
The dictionary definition of the policy.

Return type
dict

Examples

```
>>> policy = tio.policies.details(1)
```

list()

List the available custom policies.

policies: list

Returns
List of policy resource documents.

Return type
list

Examples

```
>>> for policy in tio.policies.list():
...     pprint(policy)
```

policy_export(*id*, *fobj=None*)

Exports a specified policy from Tenable Vulnerability Management.

policies: export

Parameters

- **id** (*int*) – The unique identifier of the policy to export.
- **fobj** (*FileObject, optional*) – A file-like object to write the contents of the policy to. If none is provided a BytesIO object will be returned with the policy.

Returns
A file-like object containing the contents of the policy in XML format.

Return type
FileObject

Examples

```
>>> with open('example.nessus', 'wb') as policy:
...     tio.policies.policy_export(1, policy)
```

policy_import(*fobj*)

Imports a policy into Tenable Vulnerability Management.

policies: import

Parameters
fobj (*FileObject*) – The file object of the scan policy you wish to import.

Returns
The dictionary of the imported policy.

Return type
dict

Examples

```
>>> with open('example.nessus') as policy:
...     tio.policies.policy_import(policy)
```

template_details(*name*)

Calls the editor API and parses the policy template config to return a document that closely matches what the API expects to be POSTed or PUTed via the policy create and configure methods. The compliance audits and credentials are populated into the 'current' sub-document for the relevant resources.

Parameters
name (*str*) – The name of the scan template.

Returns
The policy configuration resource.

Return type
dict

Examples

```
>>> template = tio.policies.template_details('basic')
>>> pprint(template)
```

Please note that template_details is reverse-engineered from the responses from the editor API and isn't guaranteed to work.

templates()

returns a dictionary of the scan policy templates using the format of dict['name'] = 'UUID'. This is useful for being able to define scan policy templates w/o having to remember the UUID for each individual one.

CHAPTER

TWENTYTWO

REMEDIATION SCANS

The following methods allow for interaction into the Tenable Vulnerability Management Remediation scan create API endpoints. Remediation scan list API endpoints.

Methods available on `tio.remediation_scans`:

class RemediationScansAPI(*api: APISession*)

create_remediation_scan(***kwargs*)

Create a new remediation scan.

scans: create_remediation_scan # noqa: E501

Parameters

- **uuid** (*str, optional*) – UUID of Remediation scan template
- **name** (*str*) – The name of the remediation scan to create.
- **description** (*str, optional*) – The name of the scan to create.
- **policy** (*int, optional*) – The id or title of the scan policy to use (if not using one of the pre-defined templates). Specifying a a policy id will override the the template parameter.
- **scanner_id** (*str, optional*) – The unique id of the scanner to use. Use the GET /scanners endpoint to find the scanner ID. You can use the special value AUTO-ROUTED to assign scan targets to scanner groups based on the groups' configured scan routes.
- **target_network_uuid** (*str, optional*) – For remediation scans, enter a valid target network UUID from a previous scan you wish to remediate.
- **scan_time_window** (*int, optional*) – The time frame, in minutes, during which agents must transmit scan results to Tenable Vulnerability Management in order to be included in dashboards and reports. If your request omits this parameter, the default value is 180 minutes. For non-agent scans, this attribute is null.
- **text_targets** (*str, optional*) – The List of targets to scan
- **targets** (*list, optional*) – If defined, then a list of targets can be specified and will be formatted to an appropriate text_target attribute. A list of targets to scan
- **target_groups** (*list[int]*) – For remediation scans, enter a valid target group ID from a previous scan you wish to remediate.
- **file_targets** (*string, optional*) – The name of a file containing the list of targets to scan.
- **tag_targets** (*list[str], optional*) – The list of asset tag identifiers that the scan uses to determine which assets it evaluates
- **agent_group_id** (*list[str], optional*) – An array of agent group UUIDs to scan.

- **emails** (*list[str], optional*) – A comma-separated list of accounts that receive the email summary report.
- **acls** (*list[dict], optional*) – A list of dictionaries containing permissions to apply to the scan.
- **credentials** (*dict, optional*) – A list of credentials to use.
- **enabled_plugins** (*list, optional*) – A list of plugins IDs to add to a remediation scan.
- ****kw** (*dict, optional*) – The various parameters that can be passed to the scan creation API. Examples would be *name*, *email*, *scanner_id*, etc. For more detailed information, please refer to the API documentation linked above. Further, any keyword arguments passed that are not explicitly documented will be automatically appended to the settings document. There is no need to pass settings directly.

Returns
The scan resource record of the newly created remediation scan.

Return type
dict

Examples

Create remediation scan:

```
>>> scan = tio.remediationscans.create_remediation_scan(
...     uuid='76d67790-2969-411e-a9d0-667f05e8d49e',
...     name='Create Remediation Scan',
...     description='Remediation scan created',
...     scanner_id='10167769',
...     scan_time_window=10,
...     targets=['127.0.0.1:3000'],
...     template='advanced')
```

For further information on credentials, what settings to use, etc, refer to this doc on the developer portal.

list_remediation_scan(*limit=50*, *offset=0*, *sortval='scan_creation_date:desc'*)

Retrieve the list of Remediation scans.

scans: list_remediation_scan

Parameters

- **limit** (*int, optional*) – This value needs to be between 0 and 200
- **offset** (*int, optional*) – This value needs to be > 0
- **sort** (*str, optional*) – scan_creation_date:desc/scan_creation_date:asc Returns the remediation scan list with the ascending or descending order with offset and limit

Returns
An iterator that handles the page management of the requested records.

Return type
RemediationScansIteratorV2

Examples

```
>>> for remediation_scan in tio.scans.list():
...     pprint(remediation_scan)
```

For further information on credentials, what settings to use, etc, refer to the doc linked above on the developer portal.

CHAPTER

TWENTYTHREE

SCANNER GROUPS

The following methods allow for interaction into the Tenable Vulnerability Management scanner-groups API endpoints.

Methods available on `tio.scanner_groups`:

class ScannerGroupsAPI(*api: APISession*)

This will contain all methods related to scanner groups

add_scanner(*group_id*, *scanner_id*)

Add a scanner to a scanner group.

scanner-groups: add-scanner

Parameters

- **group_id** (*int*) – The unique identifier of the scanner group.
- **scanner_id** (*int*) – The unique identifier of the scanner.

Returns

Scanner successfully added to the scanner group.

Return type

None

Examples

```
>>> tio.scanner_groups.add_scanner(1, 1)
```

create(*name*, *group_type=None*)

Create a scanner group.

scanner-groups: create

Parameters

- **name** (*str*) – The name of the scanner group to create
- **group_type** (*str, optional*) – The type of scanner group to create. Currently the only supported type is "load_balancing"

Returns

The scanner group resource record for the created group.

Return type

dict

Example

```
>>> group = tio.scanner_groups.create('Scanner Group')
```

delete(*group_id*)

Deletes a scanner group.

scanner-groups: delete

Parameters
group_id (*int*) – The unique identifier for the scanner group to delete.

Returns
The scanner group has been successfully deleted.

Return type
None

Examples

```
>>> tio.scanner_groups.delete(1)
```

delete_scanner(*group_id*, *scanner_id*)

Removes a scanner from a scanner group.

scanner-groups: delete-scanner

Parameters

- **group_id** (*int*) – The unique identifier of the scanner group.
- **scanner_id** (*int*) – The unique identifier of the scanner to remove from the requested scanner group.

Returns
The scanner was successfully removed from the scanner group.

Return type
None

Examples

```
>>> tio.scanner_groups.delete_scanner(1, 1)
```

details(*group_id*)

Retrieves the details about a scanner group.

scanner-groups: details

Parameters
group_id (*int*) – The unique identifier for the scanner group.

Returns
The scanner group resource record.

Return type
dict

Examples

```
>>> group = tio.scanner_groups.details(1)
>>> pprint(group)
```

edit(*group_id*, *name*)

Modifies a scanner group.

scanner-groups: edit

Parameters

- **group_id** (*int*) – The unique identifier for the scanner group.
- **name** (*str*) – The new name for the scanner group.

Returns
The scanner group has been successfully updated.

Return type
None

Examples

```
>>> tio.scanner_groups.edit(1, 'New Group Name')
```

edit_routes(*group_id*, *routes*)

Updates the host-names, hostname wildcards, IP addresses, and IP address ranges that Tenable Vulnerability Management matches against targets in auto-routed scans

scanner-groups: edit-routes

Parameters

- **group_id** (*int*) – The unique identifier of the scanner group
- **routes** (*list*) – The list of routes for scanner group

Returns
The scanner group routes has been successfully updated

Return type

None

Examples:

```
>>> tio.scanner_groups.edit_routes(1, ['127.0.0.1'])
```

list()

Lists the configured scanner groups.

scanner-groups: list

Returns
List of scanner group resource records.

Return type
list

Examples

```
>>> for group in tio.scanner_groups.list():
...     pprint(group)
```

list_routes(*group_id*)

List the host-names, wildcards, IP addresses, and IP address ranges that Tenable Vulnerability Management matches against targets in auto-routed scans

scanner-groups: list-routes

Parameters
group_id (*int*) – The unique identifier of the scanner group

Returns
List of routes associated to the scanner group.

Return type
list

Examples:

```
>>> for scanner in tio.scanner_groups.list_routes(1):
...     pprint(scanner)
```

list_scanners(*group_id*)

List the scanners within a specific scanner group.

scanner-groups: list-scanners

Parameters
group_id (*int*) – The unique identifier of the scanner group.

Returns
List of scanner resource records associated to the scanner group.

Return type
list

Examples

```
>>> for scanner in tio.scanner_groups.list_scanners(1):
...     pprint(scanner)
```

CHAPTER

TWENTYFOUR

SCANNERS

The following methods allow for interaction into the Tenable Vulnerability Management scanners API.

Methods available on `tio.scanners`:

class `ScannersAPI`(*api: APISession*)

allowed_scanners()

A simple convenience function that returns the list of scanners that the current user is allowed to use.

Returns
List of scanner documents.

Return type
list

Examples

```
>>> for scanner in tio.scanners.allowed_scanners():
...     pprint(scanner)
```

control_scan(*scanner_id*, *scan_uuid*, *action*)

Perform actions against scans on a given scanner.

scanners: control-scans

Parameters

- **scanner_id** (*int*) – The unique identifier for the scanner.
- **scan_uuid** (*uuid*) – The unique identifier for the scan.
- **action** (*str*) – The action to take upon the scan. Valid actions are *stop*, *pause*, and *resume*.

Returns
The action was sent to the scan successfully.

Return type
None

Examples

Stop a scan running on the scanner:

```
>>> tio.scanners.control_scan(1, '00000000-0000-0000-0000-000000000000', 'stop')
```

delete(*id*)

Delete a scanner from Tenable Vulnerability Management.

scanners: delete

Parameters
id (*int*) – The unique identifier for the scanner to delete.

Returns
The scanner was successfully deleted.

Return type
None

Examples

```
>>> tio.scanners.delete(1)
```

details(*id*)

Retrieve the details for a specified scanner.

scanners: details

Parameters
id (*int*) – The unique identifier for the scanner

Returns
The scanner resource record.

Return type
dict

Examples

```
>>> scanner = tio.scanners.details(1)
>>> pprint(scanner)
```

edit(*id*, ***kwargs*)

Modify the scanner.

scanners: edit

Parameters

- **id** (*int*) – The unique identifier for the scanner.
- **force_plugin_update** (*bool, optional*) – Force the scanner to perform a plugin update .
- **force_ui_update** (*bool, optional*) – Force the scanner to perform a UI update.
- **finish_update** (*bool, optional*) – Force the scanner to reboot to complete the update process. This action is only valid when automatic updates are disabled.

- **registration_code** (*str, optional*) – Sets the registration code for the scanner.
- **aws_update_interval** (*int, optional*) – For AWS scanners this will inform the scanner how often to check into Tenable Vulnerability Management.

Returns
The operation was requested successfully.

Return type
None

Examples

Force a plugin update on a scanner:

```
>>> tio.scanners.edit(1, force_plugin_update=True)
```

edit_permissions(*id*, **acls*)

Modifies the permissions list for the given scanner.

Parameters

- **id** (*int*) – The unique identifier for the scanner.
- ***acls** (*dict*) – The permissions record(s) for the scanner.

Returns
The permissions have been updated successfully.

Return type
None

Examples

```
>>> tio.scanners.edit_permissions(1,
...     {'type': 'default, 'permissions': 16},
...     {'type': 'user', 'id': 2, 'permissions': 16})
```

get_aws_targets(*id*)

Returns the list of AWS targets the scanner can reach.

scanners: get-aws-targets

Parameters
id (*int*) – The unique identifier for the scanner.

Returns
List of aws target resource records.

Return type
list

Examples

```
>>> for target in tio.scanners.get_aws_targets(1):
...     pprint(target)
```

get_permissions(*id*)

Returns the permission list for a given scanner.

Parameters
id (*int*) – The unique identifier for the scanner.

Returns
The permissions resource for the scanner

Return type
dict

Examples

```
>>> tio.scanners.get_permissions(1)
```

get_scanner_key(*id*)

Return the key associated with the scanner.

scanners: get-scanner-key

Parameters
id (*int*) – The unique identifier for the scanner.

Returns
The scanner key

Return type
str

Examples

```
>>> print(tio.scanners.get_scanner_key(1))
```

get_scans(*id*)

Retrieves the scans associated to the scanner.

scanners: get-scans

Parameters
id (*int*) – The unique identifier for the scanner.

Returns
List of scan resource records associated to the scanner.

Return type
list

Examples

```
>>> for scan in tio.scanners.get_scans(1):
...     pprint(scan)
```

linking_key()

The linking key for the Tenable Vulnerability Management instance.

Returns
The linking key

Return type
str

Examples

```
>>> print(tio.scanners.linking_key())
```

list()

Retrieves the list of scanners.

scanners: list

Returns
List of scanner resource records.

Return type
list

Examples

```
>>> for scanner in tio.scanners.list():
...     pprint(scanner)
```

toggle_link_state(*id*, *linked*)

Toggles the scanner's activated state.

scanners: toggle-link-state

Parameters

- **id** (*int*) – The unique identifier for the scanner
- **linked** (*bool*) – The link status of the scanner. Setting to *False* will disable the link, whereas setting to *True* will enable the link.

Returns
The status change was successful.

Return type
None

Examples

to deactivate a linked scanner:

```
>>> tio.scanners.toggle_link_state(1, False)
```

CHAPTER

TWENTYFIVE

SCANS

The following methods allow for interaction into the Tenable Vulnerability Management scans API endpoints.

Methods available on `tio.scans`:

class ScansAPI(*api: APISession*)

This will contain all methods related to scans

attachment(*scan_id: int | UUID*, *attachment_id: int*, *key: str*, *fobj: BytesIO | None = None*) → BytesIO

Retrieve an attachment associated to a scan.

scans: attachments

Parameters

- **scan_id** (*int*) – The unique identifier for the scan.
- **attachment_id** (*int*) – The unique identifier for the attachment
- **key** (*str*) – The attachment access token.
- **fobj** (*FileObject, optional*) – a file-like object you wish for the attachment to be written to. If none is specified, a BytesIO object will be returned with the contents of the attachment.

Returns

A file-like object with the attachment written into it.

Return type

FileObject

Examples

```
>>> with open('example.file', 'wb') as fobj:
...     tio.scans.attachment(1, 1, 'abc', fobj)
```

check_auto_targets(*limit: int*, *matched_resource_limit: int*, *network_uuid: UUID | None = '00000000-0000-0000-0000-000000000000'*, *tags: List[UUID] | None = None*, *targets: List[str] | None = None*) → Dict

Evaluates a list of targets and/or tags against the scan route configuration of scanner groups.

scan: check-auto-targets

Parameters

- **limit** (*int*) – Limit the number of missed targets returned in the response.

- **matched_resource_limit** (*int*) – Limit the number of matched resource UUIDs returned in the response.
- **network_uuid** (*uuid, optional*) – The UUID of the network.
- **tags** (*list, optional*) – A list of asset tags UUIDs.
- **targets** (*list, optional*) – A list of valid targets.

Returns
Return the list of missed targets (if any), and the list of matched scanner groups (if any).

Return type
dict

Examples

```
>>> scan_routes_info = tio.scans.check_auto_targets(10, 5, targets=['127.0.0.1
↪'])
```

configure(*scan_id: int | UUID*, ***kw*) → Dict

Overwrite the parameters specified on top of the existing scan record.

scans: configure

Parameters

- **scan_id** (*int*) – The unique identifier for the scan.
- **template** (*str, optional*) – The scan policy template to use. If no template is specified then the default of *basic* will be used.
- **policy** (*int, optional*) – The id or title of the scan policy to use (if not using one of the pre-defined templates). Specifying a a policy id will override the the template parameter.
- **targets** (*list, optional*) – If defined, then a list of targets can be specified and will be formatted to an appropriate text_target attribute.
- **credentials** (*dict, optional*) – A list of credentials to use.
- **compliance** (*dict, optional*) – A list of compliance audiots to use.
- **scanner** (*str, optional*) – Define the scanner or scanner group uuid or name.
- **schedule_scan** (*dict, optional*) – Define updated schedule for scan
- ****kw** (*dict, optional*) – The various parameters that can be passed to the scan creation API. Examples would be *name*, *email*, *scanner_id*, etc. For more detailed information, please refer to the API documentation linked above. Further, any keyword arguments passed that are not explicitly documented will be automatically appended to the settings document. There is no need to pass settings directly.

Returns
The scan resource record.

Return type
dict

Examples

```
>>> tio.scans.configure(1, name='New Scan Name')
```

configure a schedule for existing scan

```
>>> configure_schedule = api.scans.configure_scan_schedule(1, interval=2)
>>> tio.scans.configure(1, schedule_scan=configure_schedule)
```

configure_scan_schedule(*scan_id*, *enabled=None*, *frequency=None*, *interval=None*, *weekdays=None*, *day_of_month=None*, *starttime=None*, *timezone=None*)

Create dictionary of keys required for scan schedule

Parameters

- **scan_id** (*int*) – The id of the Scan object in Tenable Vulnerability Management
- **enabled** (*bool, optional*) – To enable/disable scan schedule
- **frequency** (*str, optional*) – The frequency of the rule. The string inputted will be up-cased. Valid values are: `ONETIME`, `DAILY`, `WEEKLY`, `MONTHLY`, `YEARLY`. Default value is `ONETIME`.
- **interval** (*int, optional*) – The interval of the rule. The default interval is 1
- **weekdays** (*list, optional*) – List of 2-character representations of the days of the week to repeat the frequency rule on. Valid values are: *SU, MO, TU, WE, TH, FR, SA* Default values: `['SU', 'MO', 'TU', 'WE', 'TH', 'FR', 'SA']`
- **day_of_month** (*int, optional*) – The day of the month to repeat a **MONTHLY** frequency rule on. The default is today.
- **starttime** (*datetime, optional*) – When the scan should start.
- **timezone** (*str, optional*) – The timezone to use for the scan. The default if none is specified is to use UTC. For the list of usable timezones, please refer to scans-timezones

Returns

Dictionary of the keys required for scan schedule.

Return type

dict

copy(*scan_id*, *folder_id=None*, *name=None*)

Duplicates a scan and returns the details of the copy.

scans: copy

Parameters

- **scan_id** (*int*) – The unique identifier for the scan.
- **folder_id** (*int, optional*) – The unique identifier for the folder.
- **name** (*str, optional*) – The name for the copied scan.

Returns

The scan resource record for the copied scan.

Return type

dict

Examples

```
>>> new_scan = tio.scans.copy(1, 'New Scan Name')
```

create(***kw*)

Create a new scan.

scans: create

Parameters

- **name** (*str*) – The name of the scan to create.
- **template** (*str, optional*) – The scan policy template to use. If no template is specified then the default of *basic* will be used.
- **policy** (*int, optional*) – The id or title of the scan policy to use (if not using one of the pre-defined templates). Specifying a a policy id will override the the template parameter.
- **targets** (*list, optional*) – If defined, then a list of targets can be specified and will be formatted to an appropriate text_target attribute.
- **credentials** (*dict, optional*) – A list of credentials to use.
- **compliance** (*dict, optional*) – A list of compliance audits to use.
- **scanner** (*str, optional*) – Define the scanner or scanner group uuid or name.
- **schedule_scan** (*dict, optional*) – Define schedule for scan
- ****kw** (*dict, optional*) – The various parameters that can be passed to the scan creation API. Examples would be *name*, *email*, *scanner_id*, etc. For more detailed information, please refer to the API documentation linked above. Further, any keyword arguments passed that are not explicitly documented will be automatically appended to the settings document. There is no need to pass settings directly.

Returns

The scan resource record of the newly created scan.

Return type

dict

Examples

Create an un-credentialed basic scan:

```
>>> scan = tio.scans.create(
...     name='Example Scan',
...     targets=['127.0.0.1'])
```

Creating a scan with a set of managed credentials:

```
>>> scan = tio.scans.create(
...     name='Example Managed Cred Scan',
...     targets=['127.0.0.1'],
...     credentials={'Host': {'SSH': [{'id': 'CREDENTIAL-UUID'}]}}
```

Creating a scan with a set of embedded credentials:

```
>>> scan = tio.scans.create(
...     name='Example Embedded Cred Scan',
...     targets=['127.0.0.1'],
...     credentials={'Host': {'Windows': [{
...         'domain': '',
...         'username': 'Administrator',
...         'password': 'sekretsquirrel',
...         'auth_method': 'Password'
...     }]}}
... )
```

Create an un-credentialed basic scheduled scan:

```
>>> schedule = api.scans.create_scan_schedule(
...     enabled=True, frequency='daily', interval=2, starttime=datetime.
↪utcnow())
>>> scan = tio.scans.create(
...     name='Example Scan',
...     targets=['127.0.0.1']
...     schedule_scan=schedule)
```

For further information on credentials, what settings to use, etc, refer to this doc on the developer portal.

create_scan_schedule(*enabled=False*, *frequency=None*, *interval=None*, *weekdays=None*, *day_of_month=None*, *starttime=None*, *timezone=None*)

Create dictionary of keys required for scan schedule

Parameters

- **enabled** (*bool, optional*) – To enable/disable scan schedule
- **frequency** (*str, optional*) – The frequency of the rule. The string inputted will be up-cased. Valid values are: `ONETIME`, `DAILY`, `WEEKLY`, `MONTHLY`, `YEARLY`. Default value is `ONETIME`.
- **interval** (*int, optional*) – The interval of the rule. The default interval is 1
- **weekdays** (*list, optional*) – List of 2-character representations of the days of the week to repeat the frequency rule on. Valid values are: *SU, MO, TU, WE, TH, FR, SA* Default values: `['SU', 'MO', 'TU', 'WE', 'TH', 'FR', 'SA']`
- **day_of_month** (*int, optional*) – The day of the month to repeat a **MONTHLY** frequency rule on. The default is today.
- **starttime** (*datetime, optional*) – When the scan should start.
- **timezone** (*str, optional*) – The timezone to use for the scan. The default if none is specified is to use UTC. For the list of usable timezones, please refer to scans-timezones

Returns

Dictionary of the keys required for scan schedule.

Return type

`dict`

delete(*scan_id: int | UUID*)

Remove a scan.

scans: delete

Parameters
scan_id (*int or uuid*) – The unique identifier for the scan.

Returns
The scan was successfully deleted.

Return type
None

Examples

```
>>> tio.scans.delete(1)
```

delete_history(*scan_id: int | UUID*, *history_id: int | UUID*)

Remove an instance of a scan from a scan history.

scans: delete-history

Parameters

- **scan_id** (*int or uuid*) – The unique identifier for the scan.
- **history_id** (*int or uuid*) – The unique identifier for the instance of the scan.

Returns
Scan history successfully deleted.

Return type
None

Examples

```
>>> tio.scans.delete_history(1, 1)
```

details(*scan_id: int | UUID*) → Dict

Calls the editor API and parses the scan config details to return a document that closely matches what the API expects to be POSTed or PUTed via the create and configure methods. The compliance audits and credentials are populated into the 'current' sub-document for the relevant resources.

Important

Please note that the details method is reverse-engineered from the responses from the editor API, and while we are reasonably sure that the response should align almost exactly to what the API expects to be pushed to it, this method by very nature of what it's doing isn't guaranteed to always work.

Note

If you're looking for the results of the most recent scan, and what matches to the `GET /scans/{id}` call, then take a look at the results method.

Parameters
scan_id (*int or uuid*) – The unique identifier for the scan.

Returns
The scan configuration resource.

Return type
dict

Examples

```
>>> scan = tio.scans.details(1)
>>> pprint(scan)
```

export(*scan_id: int | UUID*, **filters: Tuple[str, str, str]*, *stream_hook: Callable | None = None*, ***kw*)

Export the scan report.

scans: export

Parameters

- **scan_id** (*int or uuid*) – The unique identifier of the scan.
- ***filters** (*tuple, optional*) – A list of tuples detailing the filters that wish to be applied the response data. Each tuple is constructed as ('filter', 'operator', 'value') and would look like the following example: *('plugin.id', 'eq', '19506')*. For a complete list of the available filters and options, please refer to the API documentation linked above.
- **stream_hook** (*callable, optional*) – If set, send the streaming response to this callable. The callable is responsible for iterating over the stream but does *not* need to close the file object. The signature for the callable is:

```
def f(response: requests.Response,
      fobj: BytesIO,
      chunk_size: int) -> BytesIO:
```

- **history_id** (*int, optional*) – The unique identifier for the instance of the scan.
- **history_uuid** (*uuid, optional*) – The UUID for the instance of the scan.
- **format** (*str, optional*) – What format would you like the resulting data to be in. The default would be nessus output. Available options are *nessus*, *csv*, *html*, *pdf*, *db*. Default is *nessus*.
- **password** (*str, optional*) – If the export format is *db*, then what is the password used to encrypt the NessusDB file. This is a require parameter for NessusDB exports.
- **chapters** (*list, optional*) – A list of the chapters to write for the report. The chapters list is only required for PDF and HTML exports. Available chapters are *vuln_hosts_summary*, *vuln_by_host*, *compliance_exec*, *remediations*, *vuln_by_plugin*, and *compliance*. List order will denote output order. Default is *vuln_by_host*.
- **filter_type** (*str, optional*) – Are the filters exclusive (this AND this AND this) or inclusive (this OR this OR this). Valid values are *and* and *or*. The default setting is *and*.
- **scan_type** (*str, optional*) – This parameter is required only when using the API with Web Application Scanning. Available option is 'web-app'.
- **fobj** (*FileObject, optional*) – The file-like object to be returned with the exported data. If no object is specified, a BytesIO object is returned with the data. While this is an optional parameter, it is highly recommended to use this parameter as exported files can be quite large, and BytesIO objects are stored in memory, not on disk.

Returns
The file-like object of the requested export.

Return type
FileObject

Examples

Export the full report of the latest instance of the scan:

```
>>> with open('example.nessus', 'wb') as reportobj:
...     tio.scans.export(1, fobj=reportobj)
```

Export a specific instance of the scan:

```
>>> with open('example.nessus', 'wb') as reportobj:
...     tio.scans.export(1, history_id=1, fobj=reportobj)
```

history(*scan_id: int | UUID*, *limit: int = 50*, *offset: int = 0*, *pages: int | None = None*, *sort: Tuple[str, str] = None*) → ScanHistoryIterator

Get the scan history of a given scan from Tenable Vulnerability Management.

scans: history

Parameters

- **scan_id** (*int or uuid*) – The unique identifier for the scan.
- **limit** (*int, optional*) – The number of records to retrieve. Default is 50
- **offset** (*int, optional*) – The starting record to retrieve. Default is 0.
- **sort** (*tuple, optional*) – A tuple of tuples identifying the the field and sort order of the field.

Returns
An iterator that handles the page management of the requested records.

Return type
ScanHistoryIterator

Examples

```
>>> for history in tio.scans.history(1):
...     pprint(history)
```

host_details(*scan_id: int | UUID*, *host_id: int*, *history_id: int | None = None*, *history_uuid: UUID | None = None*) → Dict

Retrieve the host details from a specific scan.

scans: host-details

Parameters

- **scan_id** (*int*) – The unique identifier for the scan.
- **host_id** (*int*) – The unique identifier for the host within the scan.
- **history_id** (*int, optional*) – The unique identifier for the instance of the scan.

- **history_uuid** (*str, optional*) – The unique identifier for the scan instance.

Returns
The information related to the host requested.

Return type
dict

Examples

```
>>> host = tio.scans.host_details(1, 1)
```

import_scan(*fobj: BytesIO*, *folder_id: int | None = None*, *password: str | None = None*, *aggregate: bool = True*)

Import a scan report into Tenable Vulnerability Management.

scans: import

Parameters

- **fobj** (*FileObject*) – The File-like object of the scan to import.
- **folder_id** (*int, optional*) – The unique identifier for the folder to place the scan into.
- **password** (*str, optional*) – The password needed to decrypt the file. This is only necessary for NessusDB files uploaded.
- **aggregate** (*bool, optional*) – should the Nessus report be aggregated into the aggregate results? The default is True.

Returns
The scan resource record for the imported scan.

Return type
dict

Examples

Import a .nessusv2 report:

```
>>> with open('example.nessus', 'rb') as reportobj:
...     tio.scans.import(reportobj)
```

Import a NessusDB report.

```
>>> with open('example.db', 'rb') as reportobj:
...     tio.scans.import(reportobj, password='sekret')
```

info(*scan_id: int | UUID*, *history_uuid: UUID*) → Dict

Retrieves information about the status of the specified instance of the scan.

scan: get-scan-history

Parameters

- **scan_id** (*int or uuid*) – The unique identifier for the scan.
- **history_uuid** (*str*) – The unique identifier for the scan instance.

Returns
The metadata about the scan instance specified.

Return type
dict

Examples

```
>>> info = tio.scans.info(1, 'BA0ED610-C27B-4096-A8F4-3189279AFFE7')
```

launch(*scan_id: int | UUID*, *targets: List[str] | None = None*)

Launches a scan.

scans: launch

Parameters

- **scan_id** (*int or uuid*) – The unique identifier for the scan.
- **targets** (*list, optional*) – A list of targets to be scanned instead of the default targets in the scan.

Response:

str:
The uuid of the scan instance (history).

Examples

Launch the scan with the configured targets:

```
>>> tio.scans.launch(1)
```

Launch the scan with some custom targets:

```
>>> tio.scans.launch(1, targets=['127.0.0.1'])
```

list(*folder_id: int | None = None*, *last_modified: datetime | None = None*) → List[Dict]

Retrieve the list of configured scans.

scans: list

Parameters

- **folder_id** (*int, optional*) – Only return scans within this folder.
- **last_modified** (*datetime, optional*) – Only return scans that have been modified since the time specified.

Returns
A list containing the list of scan resource records.

Return type
list

Examples

```
>>> for scan in tio.scans.list():
...     pprint(scan)
```

pause(*scan_id: int | UUID*, *block: bool = False*)

Pauses a running scan.

scans: pause

Parameters

- **scan_id** (*int or uuid*) – The unique identifier of the scan to pause.
- **block** (*bool, optional*) – Block until the scan is actually paused. Default is False.

Returns
The scan was successfully requested to be paused.

Return type
None

Examples

```
>>> tio.scans.pause(1)
```

plugin_output(*scan_id: int | UUID*, *host_id: int*, *plugin_id: int*, *history_id: int | None = None*, *history_uuid: UUID | None = None*) → Dict

Retrieve the plugin output for a specific instance of a vulnerability on a host.

scans: plugin-output

Parameters

- **scan_id** (*int or uuid*) – The unique identifier of the scan.
- **host_id** (*int*) – The unique identifier of the scanned host.
- **plugin_id** (*int*) – The plugin id.
- **history_id** (*int, optional*) – The unique identifier of the scan instance.

Returns
The plugin resource record for that plugin on that host.

Return type
dict

Examples

```
>>> output = tio.scans.plugin_output(1, 1, 1)
>>> pprint(output)
```

progress(*scan_id: int | UUID*, *history_id: int | None = None*, *history_uuid: UUID | None = None*) → int

Get the progress of the specified scan.

scans: get-scan-progress

Parameters
scan_id (*int | UUID*) – The

results(*scan_id: int | UUID*, *history_id: int | UUID | None = None*, *history_uuid: UUID | None = None*)
Return the scan results from either the latest scan or a specific scan instance in the history.

scans: details

Parameters

- **scan_id** (*int or uuid*) – The unique identifier for the scan.
- **history_id** (*int, optional*) – The unique identifier for the instance of the scan.
- **history_uuid** (*uuid, optional*) – The UUID for the instance of the scan.

Returns
The scan result dictionary.

Return type
dict

Examples

Retrieve the latest results:

```
>>> results = tio.scans.results(419)
```

Retrieve a specific instance of the result set using history_id:

```
>>> results = tio.scans.results(419, history_id=15184619)
```

Retrieve a specific instance of the result set using history_uuid:

```
>>> results = tio.scans.results(419, history_uuid="123e4567-e89b-12d3-a456-
↪426614174000")
```

resume(*scan_id: str | UUID*)
Resume a paused scan.

scans: resume

Parameters
scan_id (*int or uuid*) – The unique identifier for the scan.

Returns
The scan was successfully requested to resume.

Return type
None

Examples

```
>>> tio.scans.resume(1)
```

schedule(*scan_id: str | UUID*, *enabled: bool*) → dict

Enables or disables the scan schedule.

scans: schedule

Parameters

- **scan_id** (*int*) – The unique identifier for the scan.
- **enabled** (*bool*) – Enables or Disables the scan scheduling.

Returns
The schedule resource record for the scan.

Return type
dict

Examples

Enable a scan schedule:

```
>>> tio.scans.schedule(1, True)
```

schedule_const

alias of ScanScheduleConst

set_read_status(*scan_id: str | UUID*, *read_status: bool*)

Sets the read status of the scan. This is generally used to toggle the unread status of the scan within the UI.

scans: read-status

Parameters

- **scan_id** (*int or uuid*) – The unique identifier for the scan.
- **read_status** (*bool*) – Is the scan in a read or unread state? True would denote read, whereas False is unread.

Returns
The status of the scan was updated.

Return type
None

Examples

Set a scan to unread:

```
>>> tio.scans.set_read_status(1, False)
```

status(*scan_id: str | UUID*) → str

Get the status of the latest instance of the scan.

scans: get-latest-status

Parameters
scan_id (*int or uuid*) – The unique identifier for the scan.

Returns
The current status of the last instance.

Return type
str

Examples

```
>>> tio.scans.status(1)
u'completed'
```

stop(*scan_id: str | UUID*, *block: bool = False*)
Stop a running scan.

scans: stop

Parameters
- **scan_id** (*int*) – The unique identifier for the scan.
- **block** (*bool, optional*) – Block until the scan is actually stopped. Default is False.

Returns
The scan was successfully requested to stop.

Return type
None

Examples

Stop the scan asynchronously:

```
>>> tio.scans.stop(1)
```

Stop the scan and wait for the scan to stop:

```
>>> tio.scans.stop(1, True)
```

timezones() → List[str]
Retrieves the list of timezones.

scans: timezones

Returns
List of allowed timezone strings accepted by Tenable.IO

Return type
list

Examples

```
>>> for item in tio.scans.timezones():
...     pprint(item)
```

upsert_aws_credentials(*scan*)

Checks the credential dict of scan dict to derive operation add or edit. This function assumes there are no edit credentials in the credentials dict. If there is any edit credentials,it would override the same. :param scan: scan object to update edit credential if it matches criteria

Returns
The scan with updated credentials.

Return type
dict

CHAPTER

TWENTYSIX

SERVER

The following methods allow for interaction into the Tenable Vulnerability Management server API endpoints.

Methods available on `tio.server`:

class ServerAPI(*api: APISession*)

properties()

Retrieves the various properties used within the Tenable Vulnerability Management instance.

server: properties

Returns
The server properties.

Return type
dict

Examples

```
>>> props = tio.server.properties()
>>> pprint(props)
```

status()

Retrieves the server status of the Tenable Vulnerability Management instance.

server: status

Returns
The server status.

Return type
dict

Examples

```
>>> status = tio.server.status()
>>> pprint(status)
```

CHAPTER

TWENTYSEVEN

SESSION

The following methods allow for interaction into the Tenable Vulnerability Management session API endpoints.

Methods available on `tio.session`:

class SessionAPI(*api: APISession*)

Tenable Vulnerability Management session API is deprecated. it is recommended to use `users` endpoint instead

change_password(*old_password*, *new_password*)

Change the password of the current user.

session: password

Parameters

- **old_password** (*str*) – The current password.
- **new_password** (*str*) – The new password.

Returns

The password has been successfully changed.

Return type

None

Examples

```
>>> tio.session.change_password('old_pass', 'new_pass')
```

details()

Retrieve the current users resource record.

session: get

Returns

The user's session resource record.

Return type

dict

Examples

```
>>> user = tio.session.details()
>>> pprint(user)
```

edit(*name*, *email*)

Modify the currently logged-in user.

session: edit

Parameters

- **name** (*str*) – The full name of the user.
- **email** (*str*) – The email address of the user.

Returns
The session data for the current user.

Return type
dict

Examples

```
>>> tio.session.edit('John Doe', 'joe@company.com')
```

enable_two_factor(*phone*)

Initiate the phone-based two-factor authorization verification process.

session: two-factor-enable

Parameters
phone (*str*) – The phone number to use for two-factor auth.

Returns
One-time activation code sent to the provided phone number.

Return type
None

Examples

```
>>> tio.session.enable_two_factor('9998887766')
```

gen_api_keys()

Generate new API keys for the current user.

session: keys

Returns
A dictionary containing the new API Keypair.

Return type
dict

Examples

```
>>> keys = tio.session.gen_api_keys()
```

restore()

Restore the session to the logged-in user. This will remove any user impersonation setting that have been set.

session: restore

Returns
The session has properly been restored to the original user.

Return type
None

Example

```
>>> tio.session.restore()
```

two_factor(*email*, *sms*, *phone=None*)

Configure two-factor authorization.

session: two-factor

Parameters

- **email** (*bool*) – Whether two-factor should be additionally sent as an email.
- **sms** (*bool*) – Whether two-factor should be enabled. This will send SMS codes.
- **phone** (*str, optional*) – The phone number to use for two-factor authentication. Required when sms is set to *True*.

Returns
Setting changes were successfully updated.

Return type
None

Example

Configure email multi-factor auth:

```
>>> tio.session.two_factor(True, False)
```

Configure SMS multi-factor auth:

```
>>> tio.session.two_factor(False, True, '9998887766')
```

verify_two_factor(*code*)

Send the verification code for two-factor authorization.

session: verify-code

Parameters
code (*str*) – The verification code that was sent to the device.

Returns
The verification code was valid and two-factor is enabled.

Return type
None

Examples

```
>>> tio.session.verify_two_factor('abc123')
```

CHAPTER

TWENTYEIGHT

TAGS

The following methods allow for interaction into the Tenable Vulnerability Management tagging API endpoints.

Methods available on `tio.tags`:

class TagsAPI(*api: APISession*)

This will contain all methods related to tags

assign(*assets*, *tags*)

Assigns the tag category/value pairs defined to the assets defined.

tags: assign tags

Parameters

- **assets** (*list*) – A list of Asset UUIDs.
- **tags** (*list*) – A list of tag category/value pair UUIDs.

Returns

Job UUID of the assignment job.

Return type

str

Examples

```
>>> tio.tags.assign(
...     assets=['00000000-0000-0000-0000-000000000000'],
...     tags=['00000000-0000-0000-0000-000000000000'])
```

create(*category*, *value*, *description=None*, *category_description=None*, *filters=None*, *filter_type=None*, *all_users_permissions=None*, *current_domain_permissions=None*)

Create a tag category/value pair

tags: create-tag-value

Parameters

- **category** (*str*) – The category name, or the category UUID. If the category does not exist, then it will be created along with the new value.
- **value** (*str*) – The value for the tag.
- **category_description** (*str, optional*) – If the category is to be created, a description can be optionally provided.
- **description** (*str, optional*) – A description for the Category/Value pair.

- **filters** (*list, optional*) – Filters are list of tuples in the form of ('FIELD', 'OPERATOR', 'VALUE'). Multiple filters can be used and will filter down the data for automatically applying the tag to asset.

  #### Examples

  – `('operating_system', 'match', ['Linux'])`

  – `('name', 'nmatch', 'home')`

  Note that multiple values can be passed in list of string format

- **filter_type** (*str, optional*) – The filter_type operator determines how the filters are combined together. `and` will inform the API that all of the filter conditions must be met whereas `or` would mean that if any of the conditions are met. Default is `and`

- **all_users_permissions** (*list, optional*) – List of the minimum set of permissions all users have on the current tag. Possible values are ALL, CAN_EDIT, and CAN_SET_PERMISSIONS.

- **current_domain_permissions** (*list, optional*) – List of user and group-specific permissions for the current tag current_domain_permissions are list of tuples in the form of ('ID', 'NAME', 'TYPE', 'PERMISSIONS') the TYPE can be either *USER* or *GROUP* and the PERMISSIONS can be *ALL*, *CAN_EDIT* or *CAN_SET_PERMISSIONS* any one or all in list

  #### Examples

  – `(uuid , 'user@company.com', 'USER', ['CAN_EDIT'])`

Returns
Tag value resource record

Return type
dict

Examples

Creating a new tag & Category:

```
>>> tio.tags.create('Location', 'Chicago')
```

Creating a new Tag value in the existing Location Category:

```
>>> tio.tags.create('Location', 'New York')
```

Creating a new Tag value in the existing Location Category and apply to assets dynamically:

```
>>> tio.tags.create('Location', 'San Francisco',
...     filters=[('operating_system', 'match', ['Linux'])])
```

Creating a new Tag value in the existing Location Category and set permissions for users:

```
>>> tio.tags.create('Location', 'Washington',
...     all_users_permissions=['CAN_EDIT'],
...     current_domain_permissions=[('c2f2d080-ac2b-4278-914b-29f148682ee1',
...         'user@company.com', 'USER', ['CAN_EDIT'])
...     ])
```

Creating a new Tag Value within a Category by UUID:

```
>>> tio.tags.create('00000000-0000-0000-0000-000000000000', 'Madison')
```

create_category(*name*, *description=None*)

Creates a new category

tags: create-category

Parameters

- **name** (*str*) – The name of the category to create
- **description** (*str, optional*) – Description for the category to create.

Returns
Tag category resource record

Return type
dict

Examples

```
>>> tio.tags.create_category('Location')
```

delete(**tag_value_uuids*)

Deletes tag value(s).

tag: delete tag value

Parameters
***tag_value_uuid** (*str*) – The unique identifier for the tag value to be deleted.

Returns
None

Examples

Deleting a single tag value:

```
>>> tio.tags.delete('00000000-0000-0000-0000-000000000000')
```

Deleting multiple tag values:

```
>>> tio.tags.delete('00000000-0000-0000-0000-000000000000',
...     '10000000-0000-0000-0000-000000000001')
```

delete_category(*tag_category_uuid*)

Deletes a tag category.

tag: delete tag category

Parameters
tag_category_uuid (*str*) – The unique identifier for the tag category to be deleted.

Returns
None

Examples

```
>>> tio.tags.delete('00000000-0000-0000-0000-000000000000')
```

details(*tag_value_uuid*)

Retrieves the details for a specific tag category/value pair.

tag: tag details

Parameters
tag_value_uuid (*str*) – The unique identifier for the c/v pair

Returns
Tag value resource record

Return type
dict

Examples

```
>>> tio.tags.details('00000000-0000-0000-0000-000000000000')
```

details_category(*tag_category_uuid*)

Retrieves the details for a specific tag category.

tag: tag category details

Parameters
tag_category_uuid (*str*) – The unique identifier for the category

Returns
Tag category resource record

Return type
dict

Examples

```
>>> tio.tags.details_category('00000000-0000-0000-0000-000000000000')
```

edit(*tag_value_uuid*, *value=None*, *description=None*, *filters=None*, *filter_type=None*, *all_users_permissions=None*, *current_domain_permissions=None*)

Updates Tag category/value pair information.

tag: edit tag value

Parameters

- **tag_value_uuid** (*str*) – The unique identifier for the c/v pair to be edited.
- **value** (*str, optional*) – The new name for the category value.

- **description** (*str, optional*) – New description for the category value.
- **filters** (*list, optional*) – Filters are list of tuples in the form of ('FIELD', 'OPERATOR', 'VALUE'). Multiple filters can be used and will filter down the data for automatically applying the tag to asset.

  Examples::

  - `('operating_system', 'match', ['Linux'])`
  - `('name', 'nmatch', 'home')`

  Note that multiple values can be passed in list of string format
- **filter_type** (*str, optional*) – The filter_type operator determines how the filters are combined together. `and` will inform the API that all of the filter conditions must be met whereas `or` would mean that if any of the conditions are met. Default is `and`
- **all_users_permissions** (*list, optional*) – List of the minimum set of permissions all users have on the current tag. Possible values are ALL, CAN_EDIT, and CAN_SET_PERMISSIONS.
- **current_domain_permissions** (*list, optional*) – List of user and group-specific permissions for the current tag current_domain_permissions are list of tuples in the form of ('ID', 'NAME', 'TYPE', 'PERMISSIONS') the TYPE can be either *USER* or *GROUP* and the PERMISSIONS can be *ALL*, *CAN_EDIT* or *CAN_SET_PERMISSIONS* any one or all in list

  Examples::

  - `(uuid, 'user@company.com', 'USER', ['CAN_EDIT'])`

Returns
Tag value resource record.

Return type
dict

Examples

```
>>> tio.tags.edit('00000000-0000-0000-0000-000000000000',
...     name='NewValueName')
```

edit_category(*tag_category_uuid*, *name=None*, *description=None*)

Updates Tag category information.

tag: edit tag category

Parameters

- **tag_category_uuid** (*str*) – The unique identifier for the category to be edited.
- **name** (*str, optional*) – The new name for the category.
- **description** (*str, optional*) – New description for the category.

Returns
Tag category resource record.

Return type
dict

Examples

```
>>> tio.tags.edit_category('00000000-0000-0000-0000-000000000000',
...     name='NewValueName')
```

get_tag_uuid(*category*, *value*)

Fetches tag UUID using category/value pairs.

Parameters

- **category** (*str*) – The category name for the tag.
- **value** (*str*) – The value for the tag.

Returns
Tag UUID.

Return type
str

Examples

```
>>> tio.tags.get_tag_uuid('test_category','test_value')
```

list(**filters*, ***kw*)

Retrieves a list of tag category/value pairs based off of the filters defined within the query.

tags: list tags

Parameters

- ***filters** (*tuple, optional*) – A defined filter tuple consisting of the name, operator, and value. Example: `('category_name', 'eq', 'Location')`.
- **filter_type** (*str, optional*) – If multiple filters are defined, the filter_type toggles the behavior as to how these filters are used. Either all of the filters have to match (`AND`) or any of the filters have to match (`OR`). If not specified, the default behavior is to assume filter_type is `AND`.
- **limit** (*int, optional*) – How many records should be returned in a given page. If nothing is set, it will default to 1000 records.
- **pages** (*int, optional*) – How many pages of data would you like to request?
- **offset** (*int, optional*) – How many records to skip before returning results. If nothing is set, it will default to 0.
- **sort** (*tuple, optional*) – A tuple of tuples identifying the the field and sort order of the field.

Returns
An iterator that handles the pagination of the results

Return type
TagIterator

Examples

Return all of the Tag Values:

```
>>> for tag in tio.tags.list():
...     pprint(tag)
```

Return all of the Tags of the Location category:

```
>>> for tag in tio.tags.list(('category_name', 'eq', 'Location')):
...     pprint(tag)
```

list_categories(**filters*, ***kw*)

Retrieves a list of tag categories based off of the filters defined within the query.

tags: list categories

Parameters

- ***filters** (*tuple, optional*) – A defined filter tuple consisting of the name, operator, and value. Example: `('name', 'eq', 'Location')`.
- **filter_type** (*str, optional*) – If multiple filters are defined, the filter_type toggles the behavior as to how these filters are used. Either all of the filters have to match (`AND`) or any of the filters have to match (`OR`). If not specified, the default behavior is to assume filter_type is `AND`.
- **limit** (*int, optional*) – How many records should be returned in a given page. If nothing is set, it will default to 1000 records.
- **pages** (*int, optional*) – How many pages of data would you like to request?
- **offset** (*int, optional*) – How many records to skip before returning results. If nothing is set, it will default to 0.
- **sort** (*tuple, optional*) – A tuple of tuples identifying the the field and sort order of the field.

Returns

An iterator that handles the pagination of the results

Return type

`TagIterator`

Examples

Return all of the Tag Categories:

```
>>> for tag in tio.tags.list_categories():
...     pprint(tag)
```

Return all of the Tags of the Location category:

```
>>> for tag in tio.tags.list_categories(
...   ('name', 'eq', 'Location')):
...     pprint(tag)
```

unassign(*assets*, *tags*)

Un-assigns the tag category/value pairs defined to the assets defined.

tags: assign tags

Parameters

- **assets** (*list*) – A list of Asset UUIDs.
- **tags** (*list*) – A list of tag category/value pair UUIDs.

Returns

Job UUID of the un-assignment job.

Return type

str

Examples

```
>>> tio.tags.unassign(
...     assets=['00000000-0000-0000-0000-000000000000'],
...     tags=['00000000-0000-0000-0000-000000000000'])
```

CHAPTER

TWENTYNINE

USERS

The following methods allow for interaction into the Tenable Vulnerability Management users API endpoints.

Methods available on `tio.users`:

class UsersAPI(*api: APISession*)

This will contain all methods related to Users

change_password(*user_id*, *old_password*, *new_password*)

Change the password for a specific user.

users: password

Parameters

- **user_id** (*int*) – The unique identifier for the user.
- **old_password** (*str*) – The current password.
- **new_password** (*str*) – The new password.

Returns

The password has been successfully changed.

Return type

None

Examples

```
>>> tio.users.change_password(1, 'old_pass', 'new_pass')
```

create(*username*, *password*, *permissions*, *name=None*, *email=None*, *account_type=None*)

Create a new user.

users: create

Parameters

- **username** (*str*) – The username for the new user.
- **password** (*str*) – The password for the new user.
- **permissions** (*int*) – The permissions role for the user. The permissions integer is derived based on the desired role of the user. For details describing what permissions values mean what roles, please refer to the User Roles table to see what permissions are accepted.
- **name** (*str, optional*) – The human-readable name of the user.

- **email** (*str, optional*) – The email address of the user.
- **account_type** (*str, optional*) – The account type for the user. The default is *local*.

Returns
The resource record of the new user.

Return type
dict

Examples

Create a standard user:

```
>>> user = tio.users.create('jsmith@company.com', 'password1', 32)
```

Create an admin user and add the email and name:

```
>>> user = tio.users.create('jdoe@company.com', 'password', 64,
...     name='Jane Doe', email='jdoe@company.com')
```

delete(*user_id*)

Removes a user from Tenable Vulnerability Management.

users: delete

Parameters
user_id (*int*) – The unique identifier of the user.

Returns
The user was successfully deleted.

Return type
None

Examples

```
>>> tio.users.delete(1)
```

details(*user_id*)

Retrieve the details of a user.

users: details

Parameters
user_id (*int*) – The unique identifier for the user.

Returns
The resource record for the user.

Return type
dict

Examples

```
>>> user = tio.users.details(1)
```

edit(*user_id*, *permissions=None*, *name=None*, *email=None*, *enabled=None*)

Modify an existing user.

users: edit

Parameters

- **user_id** (*int*) – The unique identifier for the user.
- **permissions** (*int, optional*) – The permissions role for the user. The permissions integer is derived based on the desired role of the user. For details describing what permissions values mean what roles, please refer to the User Roles table to see what permissions are accepted.
- **name** (*str, optional*) – The human-readable name of the user.
- **email** (*str, optional*) – The email address of the user.
- **enabled** (*bool, optional*) – Is the user account enabled?

Returns

The modified user resource record.

Return type

dict

Examples

```
>>> user = tio.users.edit(1, name='New Full Name')
```

edit_auths(*user_id*, *api_permitted=None*, *password_permitted=None*, *saml_permitted=None*)

update user authorizations for accessing a Tenable Vulnerability Management instance.

users: edit-auths

Parameters

- **user_id** (*int*) – The unique identifier for the user.
- **api_permitted** (*bool*) – Indicates whether API access is authorized for the user.
- **password_permitted** (*bool*) – Indicates whether user name and password login is authorized for the user.
- **saml_permitted** (*bool*) – Indicates whether SSO with SAML is authorized for the user.

Returns

Returned if Tenable Vulnerability Management successfully updates the user's authorizations.

Return type

None

Examples

```
>>> tio.users.edit_auths(1, True, True, False)
```

enable_two_factor(*user_id*, *phone*, *password*)

Enable phone-based two-factor authorization for a specific user.

users: two-factor-enable

Parameters

- **user_id** (*int*) – The user id
- **phone** (*str*) – The phone number to use for two-factor auth.
- **password** (*str*) – The user password.

Returns
One-time activation code sent to the provided phone number.

Return type
None

Examples

```
>>> tio.users.enable_two_factor(1, '9998887766')
```

enabled(*user_id*, *enabled*)

Enable the user account.

users: enabled

Parameters

- **user_id** (*int*) – The unique identifier for the user.
- **enabled** (*bool*) – Is the user enabled?

Returns
The modified user resource record.

Return type
dict

Examples

Enable a user:

```
>>> tio.users.enabled(1, True)
```

Disable a user:

```
>>> tio.users.enabled(1, False)
```

gen_api_keys(*user_id*)

Generate the API keys for a specific user.

users: keys

Parameters
user_id (*int*) – The unique identifier for the user.

Returns
A dictionary containing the new API Key-pair.

Return type
dict

Examples

```
>>> keys = tio.users.gen_api_keys(1)
```

impersonate(*name*)

Impersonate as a specific user.

users: impersonate

Parameters
name (*str*) – The user-name of the user to impersonate.

Returns
Impersonation successful.

Return type
None

Examples

```
>>> tio.users.impersonate('jdoe@company.com')
```

list()

Retrieves a list of users.

users: list

Returns
List of user resource records.

Return type
list

Examples

```
>>> for user in tio.users.list():
...     pprint(user)
```

list_auths(*user_id*)

list user authorizations for accessing a Tenable Vulnerability Management instance.

users: list-auths

Parameters
user_id (*int*) – The unique identifier for the user.

Returns
Returns authorizations for the user.

Return type
dict

Examples

```
>>> auth = tio.users.list_auths(1)
```

two_factor(*user_id*, *email*, *sms*, *phone=None*)

Configure two-factor authorization for a specific user.

users: two-factor

Parameters

- **user_id** (*int*) – The unique identifier for the user.
- **email** (*bool*) – Whether two-factor should be additionally sent as an email.
- **sms** (*bool*) – Whether two-factor should be enabled. This will send SMS codes.
- **phone** (*str, optional*) – The phone number to use for two-factor authentication. Required when sms is set to *True*.

Returns
Setting changes were successfully updated.

Return type
None

Examples

Enable email authorization for a user:

```
>>> tio.users.two_factor(1, True, False)
```

Enable SMS authorization for a user:

```
>>> tio.users.two_factor(1, False, True, '9998887766')
```

verify_two_factor(*user_id*, *code*)

Send the verification code for two-factor authorization.

users: two-factor-enable-verify

Parameters
code (*str*) – The verification code that was sent to the device.

Returns
The verification code was valid and two-factor is enabled.

Return type
None

Examples

```
>>> tio.users.verify_two_factor(1, 'abc123')
```

CHAPTER

THIRTY

WAS

The following methods allow for interaction into the Tenable Vulnerability Management WAS API endpoints.

Methods available on `tio.was`:

class WasAPI(*api: APISession*)

This class contains methods related to WAS.

download_scan_report(*scan_uuid: str*) → Dict

Downloads the individual target scan results.

Parameters

scan_uuid (*str*) – UUID of the scan whose report to download.

export(***kwargs*) → WasIterator

Export Web Application Scan Results based on filters applied.

Parameters

- **single_filter** (*tuple*) – A single filter to apply to the scan configuration search. This is a tuple with three elements - field, operator, and value in that order.
- **and_filter** (*list*) – An array of filters that must all be satisfied. This is a list of tuples with three elements - field, operator, and value in that order.
- **or_filter** (*list*) – An array of filters where at least one must be satisfied. This is a list of tuples with three elements - field, operator, and value in that order.

Returns

WasIterator

Examples

Passing AND filter to the API

```
>>> was_iterator = tio.was.export(
...     and_filter=[
...         ("scans_started_at", "gte", "2023/03/24"),
...         ("scans_status", "contains", ["completed"])
...     ]
... )
...
... for finding in was_iterator:
...     print(finding)
```

CHAPTER

THIRTYONE

WORKBENCHES

The following methods allow for interaction into the Tenable Vulnerability Management workbenches API endpoints.

> **Note**
>
> Workbenches API endpoints have an upper bound on the amount of data that they will return, so for larger result sets, it may make more sense to use the exports API.

Methods available on `tio.workbenches`:

class `WorkbenchesAPI`(*api: APISession*)

asset_activity(*uuid*)

Query for the asset activity (when was the asset was seen, were there changes, etc.).

workbenches: asset-activity

Parameters
uuid (*str*) – The asset unique identifier.

Returns
The activity list of the asset specified.

Return type
list

Examples

```
>>> asset_id = '00000000-0000-0000-0000-000000000000'
>>> for entry in tio.workbenches.asset_activity(asset_id):
...     pprint(entry)
```

asset_delete(*asset_uuid*)

Deletes the asset.

workbenches: asset-delete

Parameters
asset_uuid (*str*) – The unique identifier for the asset.

Return type
None

Examples

```
>>> asset_id = '00000000-0000-0000-0000-000000000000'
>>> tio.workbenches.asset_delete(asset_id)
```

asset_info(*uuid*, *all_fields=True*)

Query for the information for a specific asset within the asset workbench.

workbenches: asset-info

Parameters

- **id** (*str*) – The unique identifier (UUID) of the asset.
- **all_fields** (*bool, optional*) – If all_fields is set to true (the default state), then an expanded dataset is returned as defined by the API documentation (linked above).

Returns
The resource record for the asset.

Return type
dict

Examples

```
>>> asset = tio.workbenches.asset_info('00000000-0000-0000-0000-000000000000')
```

asset_vuln_info(*uuid*, *plugin_id*, **filters*, ***kw*)

Retrieves the vulnerability information for a specific plugin on a specific asset within Tenable Vulnerability Management.

workbenches: asset-vulnerability-info

Parameters

- **uuid** (*str*) – The unique identifier of the asset to query.
- **plugin_id** (*int*) – The unique identifier of the plugin.
- **age** (*int, optional*) – The maximum age of the data to be returned.
- ***filters** (*list, optional*) – A list of tuples detailing the filters that wish to be applied the response data. Each tuple is constructed as ('filter', 'operator', 'value') and would look like the following example: *('host.hostname', 'match', 'asset.com')*. For a complete list of the available filters and options, please refer to the API documentation linked above.
- **filter_type** (*str, optional*) – Are the filters exclusive (this AND this AND this) or inclusive (this OR this OR this). Valid values are *and* and *or*. The default setting is *and*.

Returns
List of vulnerability resource records.

Return type
list

Examples

```
>>> asset_id = '00000000-0000-0000-0000-000000000000'
>>> vuln = tio.workbenches.asset_vuln_info(asset_id, 19506)
>>> pprint(vuln)
```

asset_vuln_output(*uuid*, *plugin_id*, **filters*, ***kw*)

Retrieves the vulnerability output for a specific vulnerability on a specific asset within Tenable Vulnerability Management.

workbenches: asset-vulnerability-output

Parameters

- **uuid** (*str*) – The unique identifier of the asset to query.
- **plugin_id** (*int*) – The unique identifier of the plugin.
- **age** (*int, optional*) – The maximum age of the data to be returned.
- ***filters** (*list, optional*) – A list of tuples detailing the filters that wish to be applied the response data. Each tuple is constructed as ('filter', 'operator', 'value') and would look like the following example: *('host.hostname', 'match', 'asset.com')*. For a complete list of the available filters and options, please refer to the API documentation linked above.
- **filter_type** (*str, optional*) – Are the filters exclusive (this AND this AND this) or inclusive (this OR this OR this). Valid values are *and* and *or*. The default setting is *and*.

Returns

List of vulnerability resource records.

Return type

list

Examples

```
>>> asset_id = '00000000-0000-0000-0000-000000000000'
>>> output = tio.workbenches.asset_vuln_output(asset_id, 19506)
>>> pprint(output)
```

asset_vulns(*uuid*, **filters*, ***kw*)

Return the vulnerabilities for a specific asset.

workbenches: asset-vulnerabilities

Parameters

- **uuid** (*str*) – The unique identifier of the asset to query.
- **age** (*int, optional*) – The maximum age of the data to be returned.
- ***filters** (*list, optional*) – A list of tuples detailing the filters that wish to be applied the response data. Each tuple is constructed as ('filter', 'operator', 'value') and would look like the following example: *('host.hostname', 'match', 'asset.com')*. For a complete list of the available filters and options, please refer to the API documentation linked above.

- **filter_type** (*str, optional*) – Are the filters exclusive (this AND this AND this) or inclusive (this OR this OR this). Valid values are *and* and *or*. The default setting is *and*.

Returns
List of vulnerability resource records.

Return type
list

Examples

```
>>> asset_id = '00000000-0000-0000-0000-000000000000'
>>> for vuln in tio.workbenches.asset_vulns(asset_id):
...     pprint(vuln)
```

assets(**filters*, ***kw*)

The assets workbench allows for filtering and interactively querying the asset data stored within Tenable Vulnerability Management. There are a wide variety of filtering options available to find specific pieces of data.

workbenches: assets

Parameters

- **age** (*int, optional*) – The maximum age of the data to be returned.
- ***filters** (*list, optional*) – A list of tuples detailing the filters that wish to be applied the response data. Each tuple is constructed as ('filter', 'operator', 'value') and would look like the following example: *('host.hostname', 'match', 'asset.com')*. For a complete list of the available filters and options, please refer to the API documentation linked above.
- **filter_type** (*str, optional*) – Are the filters exclusive (this AND this AND this) or inclusive (this OR this OR this). Valid values are *and* and *or*. The default setting is *and*.
- **all_fields** (*bool, optional*) – Should all of the available fields be returned for each returned asset, or just the default fields represented in the UI. The default is set to *True* which will return the same level of detail as the workbenches: asset-info endpoint.

Returns
List of asset resource records.

Return type
list

Examples

Query for all of the asset information:

```
>>> for asset in tio.workbenches.assets():
...     pprint(asset)
```

Query for just the windows assets:

```
>>> for asset in tio.workbenches.assets(
...     ('operating_system', 'match', 'Windows')):
...     pprint(asset)
```

export(**filters*, ***kw*)

Export data from the vulnerability workbench. These exports can be in a number of different formats, however the defaults are set to export a Nessusv2 report.

workbenches: export

Parameters

- ***filters** (*tuple, optional*) – A list of tuples detailing the filters that wish to be applied the response data. Each tuple is constructed as ('filter', 'operator', 'value') and would look like the following example: *('plugin.id', 'eq', '19506')*. For a complete list of the available filters and options, please refer to the API documentation linked above.
- **asset_uuid** (*uuid, optional*) – Restrict the output to the asset identifier specified.
- **plugin_id** (*int, optional*) – Restrict the output to the plugin identifier specified.
- **format** (*str, optional*) – What format would you like the resulting data to be in. The default would be nessus output. Available options are *nessus*, *csv*, *html*, *pdf*. Default is 'nessus'
- **chapters** (*list, optional*) – A list of the chapters to write for the report. The chapters list is only required for PDF, CSV, and HTML exports. Available chapters are `vuln_hosts_summary`, `vuln_by_host`, `vuln_by_plugin`, and `vuln_by_asset`. List order will denote output order. In the case of CSV reports, only `vuln_by_asset` and `vuln_by_plugin` are available and only a singular chapter can be specified.
- **filter_type** (*str, optional*) – Are the filters exclusive (this AND this AND this) or inclusive (this OR this OR this). Valid values are *and* and *or*. The default setting is *and*.
- **fobj** (*FileObject, optional*) – The file-like object to be returned with the exported data. If no object is specified, a BytesIO object is returned with the data. While this is an optional parameter, it is highly recommended to use this parameter as exported files can be quite large, and BytesIO objects are stored in memory, not on disk.

Returns
The file-like object of the requested export.

Return type
`FileObject`

Examples

```
>>> with open('example.nessus', 'wb') as exportobj:
...     tio.workbenches.export(fobj=exportobj)
```

vuln_assets(**filters*, ***kw*)

Retrieve assets based on the vulnerability data.

workbenches: assets-vulnerabilities

Parameters

- **age** (*int, optional*) – The maximum age of the data to be returned.
- ***filters** (*list, optional*) – A list of tuples detailing the filters that wish to be applied the response data. Each tuple is constructed as ('filter', 'operator', 'value') and would look like the following example: *('host.hostname', 'match', 'asset.com')*. For a complete list of the available filters and options, please refer to the API documentation linked above.
- **filter_type** (*str, optional*) – Are the filters exclusive (this AND this AND this) or inclusive (this OR this OR this). Valid values are *and* and *or*. The default setting is *and*.

Returns
List of asset resource records.

Return type
list

Examples

```
>>> for asset in tio.workbenches.vuln_assets():
...     pprint(asset)
```

vuln_info(*plugin_id*, **filters*, ***kw*)

Retrieve the vulnerability information for a specific vulnerability.

workbenches: vulnerability-info

Parameters

- **age** (*int, optional*) – The maximum age of the data to be returned.
- ***filters** (*list, optional*) – A list of tuples detailing the filters that wish to be applied the response data. Each tuple is constructed as ('filter', 'operator', 'value') and would look like the following example: *('host.hostname', 'match', 'asset.com')*. For a complete list of the available filters and options, please refer to the API documentation linked above.
- **filter_type** (*str, optional*) – Are the filters exclusive (this AND this AND this) or inclusive (this OR this OR this). Valid values are *and* and *or*. The default setting is *and*.

Returns
Vulnerability info resource

Return type
dict

Examples

```
>>> info = tio.workbenches.vuln_info(19506)
>>> pprint(info)
```

vuln_outputs(*plugin_id*, **filters*, ***kw*)

Retrieve the vulnerability output for a given vulnerability.

workbenches: vulnerability-output

Parameters

- **age** (*int, optional*) – The maximum age of the data to be returned.
- ***filters** (*list, optional*) – A list of tuples detailing the filters that wish to be applied the response data. Each tuple is constructed as ('filter', 'operator', 'value') and would look like the following example: *('host.hostname', 'match', 'asset.com')*. For a complete list of the available filters and options, please refer to the API documentation linked above.
- **filter_type** (*str, optional*) – Are the filters exclusive (this AND this AND this) or inclusive (this OR this OR this). Valid values are *and* and *or*. The default setting is *and*.

Returns
Vulnerability outputs resource

Return type
dict

Examples

```
>>> outputs = tio.workbenches.vuln_outputs(19506)
>>> pprint(outputs)
```

vulns(**filters*, ***kw*)

The vulnerability workbench allows for filtering and interactively querying the vulnerability data stored within Tenable Vulnerability Management. There are a wide variety of filtering options available to find specific pieces of data.

workbenches: vulnerability-info

Parameters

- **age** (*int, optional*) – The maximum age of the data to be returned.
- **authenticated** (*bool, optional*) – If set to true will only return authenticated vulnerabilities.
- **exploitable** (*bool, optional*) – If set to true will only return exploitable vulnerabilities.
- ***filters** (*list, optional*) – A list of tuples detailing the filters that wish to be applied the response data. Each tuple is constructed as ('filter', 'operator', 'value') and would look like the following example: *('host.hostname', 'match', 'asset.com')*. For a complete list of the available filters and options, please refer to the API documentation linked above.

- **filter_type** (*str, optional*) – Are the filters exclusive (this AND this AND this) or inclusive (this OR this OR this). Valid values are *and* and *or*. The default setting is *and*.
- **resolvable** (*bool, optional*) – If set to true will only return vulnerabilities with a remediation path.
- **severity** (*str, optional*) – Only return results of a specific severity (critical, high, medium, or low).

Returns
Vulnerability info resource

Return type
dict

CHAPTER

THIRTYTWO

TENABLE SECURITY CENTER

Note

Please refer to the common themes section for TenableSC for details on how these methods are written from an overall concept. Not all attributes are explicitly documented, only the ones that pyTenable is augmenting, validating, or modifying. For a complete listing of the attributes that can be passed to most APIs, refer to the official API documentation that each method calls, which is conveniently linked in each method's docs.

class TenableSC(*host: str | None = None*, *access_key: str | None = None*, *secret_key: str | None = None*, ***kwargs*)

TenableSC API Wrapper The Tenable Security Center object is the primary interaction point for users to interface with Tenable Security Center via the pyTenable library. All of the API endpoint classes that have been written will be grafted onto this class.

Parameters

- **host** (*str*) – The address of the Tenable Security Center instance to connect to. (NOTE: The *host* parameter will be deprecated in favor of the *url* parameter in future releases).
- **access_key** (*str, optional*) – The API access key to use for sessionless authentication.
- **adapter** (*requests.Adaptor, optional*) – If a requests session adaptor is needed to ensure connectivity to the Tenable Security Center host, one can be provided here.
- **backoff** (*float, optional*) – If a 429 response is returned, how much do we want to backoff if the response didn't send a Retry-After header. The default backoff is 1 second.
- **cert** (*tuple, optional*) – The client-side SSL certificate to use for authentication. This format could be either a tuple or a string pointing to the certificate. For more details, please refer to the Requests Client-Side Certificates documentation.
- **p12_cert** (*str, optional*) – The client-side PKCS12 certificate to use for certificate-based authentication. A password must be provided to descrypt the password along with the certificate file (see password).
- **password** (*str, optional*) – The password to use for session authentication.
- **port** (*int, optional*) – The port number to connect to on the specified host. The default is port 443. (NOTE: The *port* parameter will be deprecated in favor of the unified *url* parameter in future releases).
- **retries** (*int, optional*) – The number of retries to make before failing a request. The default is 5.

- **scheme** (*str, optional*) – What HTTP scheme should be used for URI path construction. The default is `https`. (NOTE: The *scheme* parameter will be deprecated in favor of the unified *url* parameter in future releases).
- **secret_key** (*str, optional*) – The API secret key to use for sessionless authentication.
- **session** (*requests.Session, optional*) – If a requests Session is provided, the provided session will be used instead of constructing one during initialization.
- **ssl_verify** (*bool, optional*) – Should the SSL certificate on the Tenable Security Center instance be verified? Default is False.
- **username** (*str, optional*) – The username to use for session authentication.
- **timeout** (*int, optional*) – The connection timeout parameter informing the library how long to wait in seconds for a stalled response before terminating the connection. If unspecified, the default is 300 seconds.

Examples

A direct connection to Tenable Security Center:

```
>>> from tenable.sc import TenableSC
>>> sc = TenableSC(url='https://sc.company.tld')
```

A connection to Tenable Security Center using SSL certificates:

```
>>> sc = TenableSC(url='https://sc.company.tld',
...                cert=('/path/client.cert', '/path/client.key')
                   )
```

Using a PKCS12 Certificate:

```
>>> sc = TenableSC(url='https://sc.company.tld',
...                p12_cert='/path/client.p12',
...                password='s3kr3tsqu1rr3l',
...                )
```

Using API Keys to communicate to Tenable Security Center:

```
>>> sc = TenableSC(url='https://sc.company.tld',
...                access_key='abcdef1234567890',
...                secret_key='abcdef1234567890'
...                )
```

For more information, please See Tenable's SC API documentation and the SC API Best Practices Guide.

property accept_risks

The interface object for the *Tenable Security Center Accept Risks APIs*.

property alerts

The interface object for the *Tenable Security Center Alerts APIs*.

property analysis

The interface object for the *Tenable Security Center Analysis APIs*.

property asset_lists

The interface object for the *Tenable Security Center Asset Lists APIs*.

property audit_files

The interface object for the *Tenable Security Center Audit Files APIs*.

property credentials

The interface object for the *Tenable Security Center Credentials APIs*.

property current

The interface object for the *Tenable Security Center Current Session APIs*.

property feeds

The interface object for the *Tenable Security Center Feeds APIs*.

property files

The interface object for the *Tenable Security Center Files APIs*.

property groups

The interface object for the *Tenable Security Center Groups APIs*.

property hosts

The interface object for the *Tenable Security Center Hosts APIs*.

property license

The interface object for the *Tenable Security Center License APIs*.

login(*username=None*, *password=None*, *access_key=None*, *secret_key=None*)

Logs the user into Tenable Security Center

Parameters

- **username** (*str, optional*) – Username
- **password** (*str, optional*) – Password
- **access_key** (*str, optional*) – API Access Key
- **secret_key** (*str, optional*) – API Secret Key

Returns

None

Examples

Using a username && password:

```
>>> sc = TenableSC('127.0.0.1', port=8443)
>>> sc.login('username', 'password')
```

Using API Keys:

```
>>> sc = TenableSC('127.0.0.1', port=8443)
>>> sc.login(access_key='ACCESSKEY', secret_key='SECRETKEY')
```

logout()

Logs out of Tenable Security Center and resets the session.

Examples

```
>>> sc.logout()
```

property organizations

The interface object for the *Tenable Security Center Organization APIs*.

property plugins

The interface object for the *Tenable Security Center Plugins APIs*.

property policies

The interface object for the *Tenable Security Center Policies APIs*.

property queries

The interface object for the *Tenable Security Center Queries APIs*.

property recast_risks

The interface object for the *Tenable Security Center Recast Risks APIs*.

property report_definition

The interface object for the *Tenable.sc ReportDefinition APIs*.

property repositories

The interface object for the *Tenable Security Center Repositories APIs*.

property roles

The interface object for the *Tenable Security Center Roles APIs*.

property scan_instances

The interface object for the *Tenable Security Center Scan Instances APIs*.

property scan_zones

The interface object for the *Tenable Security Center Scan Zones APIs*.

property scanners

The interface object for the *Tenable Security Center Scanners APIs*.

property scans

The interface object for the *Tenable Security Center Scans APIs*.

property status

The interface object for the *Tenable Security Center Status APIs*.

property system

The interface object for the *Tenable Security Center System APIs*.

property tickets

The interface object for the *Tenable Security Center Ticket APIs*.

property users

The interface object for the *Tenable Security Center Users APIs*.

property version

The version of the SecurityCenter instance

32.1 Common Themes

32.1.1 Tenable Security Center CRUD within pyTenable

pyTenable allows for the ability to leverage both the naturalized inputs as well as passing the raw parameters within the same structure. In some cases this doesn't seem immediately obvious, however allows for the ability to pass parameters that either haven't yet been, or in some cases, may never be interpreted by the library.

For example, in the alerts API, you could pass the snake_cased `always_exec_on_trigger` or you could pass what the API endpoint itself expects, which is `executeOnEveryTrigger`. The snake-cased version expects a boolean value, which will be converted into the string value that camelCased variant expects. You'll see this behavior a lot throughout the library, and is intended to allow you to sidestep most things should you need to. For example, in the alerts API again, you may not want to pass a trigger as `trigger=('sumip', '>=', '100')` and instead pass as the parameters that are to be written into the JSON request: `triggerName='sumip', triggerOperator='>=', triggerValue='100'`. Both of these methods will produce the same JSON request, and the the option is yours to use the right way for the job.

Along these same lines, its possible to see how the JSON documents are being constructed by simply looking at the `_constructor` methods for each APIEndpoint class. If pyTenable is getting in your way, you can almost always sidestep it and pass the exact dictionary you wish to pass on to the API.

32.1.2 Schedule Dictionaries

A dictionary detailing the repeating schedule within Tenable Security Center. This dictionary consists of 1 or 3 parameters, depending on the type of schedule. In all of the definitions except `ical`, a single parameter of `type` is passed with lone of the following values: `ical`, `never`, `rollover`, and `template`. If no document is specified, then the default of `never` is assumed. For repeating scans, you'll have to use the type of `ical` and also specify the `start` and `repeatRule` parameters as well. The `start` parameter is an iCal DateTime Form #3 formatted string specifying the date and time in which to start the repeating event. The `repeatRule` parameter is an iCal Recurrence Rule formatted string.

- Example Never Declaration:

```
{'type': 'never'}
```

- Example daily event starting at 9am Eastern

```
{
    'type': 'ical',
    'start': 'TZID=America/New_York:20190214T090000',
    'repeatRule': 'FREQ=DAILY;INTERVAL=1'
}
```

- Example weekly event every Saturday at 8:30pm Eastern

```
{
    'type': 'ical',
    'start': 'TZID=America/New_York:20190214T203000',
    'repeatRule': 'FREQ=WEEKLY;BYDAY=SA;INTERVAL=1'
}
```

There are detailed instructions in the RFC documentation on how to construct these recurrence rules. Further there are some packages out there to aid in converting more human-readable text into recurrence rules, such as the recurrent package for example.

32.2 Accept Risks

The following methods allow for interaction into the Tenable Security Center Accept Risk API.

Methods available on `sc.accept_risks`:

class AcceptRiskAPI(*api: APISession*)

apply(*id*, *repo*)

Applies the accept risk rule for either all repositories, or the repository specified.

accept-risk: apply

Parameters

- **id** (*int*) – The identifier for the accept risk rule.
- **repo** (*int, optional*) – A specific repository to apply the rule to. The default if not specified is all repositories (**0**).

Returns

Empty string response from the API.

Return type

str

Examples

```
>>> sc.accept_risks.apply(1)
```

create(*plugin_id*, *repos*, ***kw*)

Creates a new accept risk rule. Either ips, uuids, asset_list, or host_uuids must be specified, otherwise it will apply to all.

accept-risk: create

Parameters

- **plugin_id** (*int*) – The plugin to apply the accept risk rule to.
- **repos** (*list*) – The list of repositories to apply this accept risk rule to.
- **asset_list** (*int, optional*) – The asset list id to apply the accept risk rule to. Please note that `asset_list`, `ips`, `uuids`, and `host_uuids` are mutually exclusive.
- **comments** (*str, optional*) – The comment associated to the accept risk rule.
- **expires** (*int, optional*) – Timestamp. When should the rule expire? if no expiration is set, the rule will never expire. If not mentioned, value is -1 (-1 represents December 31st 1969 23:59:59 hours GMT)
- **ips** (*list, optional*) – A list of IPs to apply the accept risk rule to. Please note that `asset_list`, `ips`, `uuids`, and `host_uuids` are mutually exclusive.
- **port** (*int, optional*) – The port to restrict this accept risk rule to. The default is unrestricted.
- **protocol** (*int, optional*) – The protocol to restrict the accept risk rule to. The default is unrestricted.
- **uuids** (*list, optional*) – The agent uuids to apply the accept risk rule to. Please note that `asset_list`, `ips`, `uuids`, and `host_uuids` are mutually exclusive.

- **host_uuids** (*list[str], optional*) – The hostUUIDs to apply the accept risk rule to. Please note that `asset_list`, `ips`, `uuids`, and `host_uuids` are mutually exclusive.

Returns
The newly created accept risk rule definition.

Return type
dict

Examples

Create a rule to accept 97737 on 2 IPs till Aug 25th 2021 00:00 Hrs GMT.

```
>>> rule = sc.accept_risks.create(97737, [1],
...     ips=['192.168.0.101', '192.168.0.102'], expires=1629849600)
```

Create a rule to accept 97737 on all IPs on repository 1:

```
>>> rule = sc.accept_risks.create(97737, [1])
```

delete(*id*)

Removes the accepted risk rule from Tenable Security Center

accept-risk: delete

Parameters
id (*int*) – The identifier for the accept risk rule.

Returns
Empty string response from the API.

Return type
str

Examples

```
>>> sc.accept_risks.delete(1)
```

details(*id*, *fields=None*)

Retrieves the details of an accepted risk rule.

accept-risk details

Parameters

- **id** (*int*) – The identifier for the accept risk rule.
- **fields** (*list, optional*) – A list of attributes to return for each accepted risk rule.

Returns
The accept risk rule details.

Return type
dict

Examples

```
>>> rule = sc.accept_risks.details(1)
>>> pprint(rule)
```

list(*repo_ids=None*, *plugin_id=None*, *port=None*, *org_ids=None*, *fields=None*)

Retrieves the list of accepted risk rules.

accept-risk: list

Parameters

- **fields** (*list, optional*) – A list of attributes to return for each accepted risk rule.
- **plugin_id** (*int, optional*) – Plugin id to filter the response on.
- **port** (*int, optional*) – Port number to filter the response on.
- **org_ids** (*list, optional*) – List of organization ids to filter on.
- **repo_ids** (*list, optional*) – List of repository ids to filter the response on.

Returns
A list of accepted risk rules.

Return type
list

Examples

```
>>> for rule in sc.accept_risks.list():
...     pprint(rule)
```

32.3 Alerts

The following methods allow for interaction into the Tenable Security Center Alert API.

Methods available on `sc.alerts`:

class AlertAPI(*api: APISession*)

create(**filters*, ***kw*)

Creates a new alert. The fields below are explicitly checked, however any additional parameters mentioned in the API docs can be passed to the document constructor.

alert: create

Parameters

- ***filters** (*tuple*) – A filter expression. Refer to the detailed description within the analysis endpoint documentation for more details on how to formulate filter expressions.
- **data_type** (*str*) – The type of filters being used. Must be of type `lce`, `ticket`, `user`, or `vuln`. If no data-type is specified, then the default of `vuln` will be set.
- **name** (*str*) – The name of the alert.
- **description** (*str, optional*) – A description for the alert.

- **trigger** (*tuple*) – A tuple in the filter-tuple format detailing what would constitute a trigger. For example: `('sumip', '=', '1000')`.
- **always_exec_on_trigger** (*bool, optional*) – Should the trigger always execute when the trigger fires, or only execute when the returned data changes? Default is `False`.
- **schedule** (*dict, optional*) – This is the schedule dictionary that will inform Tenable Security Center how often to run the alert. If left unspecified then we will default to `{'type': 'never'}`.
- **action** (*list*) – The action(s) that will be performed when the alert trigger fires. Each action is a dictionary detailing what type of action to take, and the details surrounding that action. The supported type of actions are `email`, `notifications`, `report`, `scan`, `syslog`, and `ticket`. The following examples lay out each type of action as an example:

  – Email action type:

```
{'type': 'email',
 'subject': 'Example Email Subject',
 'message': 'Example Email Body'
 'addresses': 'user1@company.com\nuser2@company.com',
 'users': [{'id': 1}, {'id': 2}],
 'includeResults': 'true'}
```

  – Notification action type:

```
{'type': 'notification',
 'message': 'Example notification',
 'users': [{'id': 1}, {'id': 2}]}
```

  – Report action type:

```
{'type': 'report',
 'report': {'id': 1}}
```

  – Scan action type:

```
{'type': 'scan',
 'scan': {'id': 1}}
```

  – Syslog action type:

```
{'type': 'syslog',
 'host': '127.0.0.1',
 'port': '514',
 'message': 'Example Syslog Message',
 'severity': 'Critical'}
```

  – Ticket action type:

```
{'type': 'ticket',
 'assignee': {'id': 1},
 'name': 'Example Ticket Name',
 'description': 'Example Ticket Description',
 'notes': 'Example Ticket Notes'}
```

Returns
The alert resource created.

Return type
dict

Examples

```
>>> sc.alerts.create(
...     ('severity', '=', '3,4'),
...     ('exploitAvailable', '=', 'true'),
...     trigger=('sumip', '>=', '100'),
...     name='Too many High or Critical and Exploitable',
...     action=[{
...         'type': 'notification',
...         'message': 'Too many High or Crit Exploitable Vulns',
...         'users': [{'id': 1}]
...     }])
```

delete(*id*)

Deletes the specified alert.

alert: delete

Parameters
id (*int*) – The alert identifier.

Returns
The response code of the action.

Return type
str

Examples

```
>>> sc.alerts.delete(1)
```

details(*id*, *fields=None*)

Returns the details for a specific alert.

alert: details

Parameters

- **id** (*int*) – The identifier for the alert.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The alert resource record.

Return type
dict

Examples

```
>>> alert = sc.alerts.detail(1)
>>> pprint(alert)
```

edit(*id*, **filters*, ***kw*)

Updates an existing alert. All fields are optional and will overwrite the existing value.

alert: update

Parameters

- **id** (*int*) – The alert identifier.
- ***filters** (*tuple*) – A filter expression. Refer to the detailed description within the analysis endpoint documentation for more details on how to formulate filter expressions.
- **data_type** (*str*) – The type of filters being used. Must be of type `lce`, `ticket`, `user`, or `vuln`. If no data-type is specified, then the default of `vuln` will be set.
- **name** (*str, optional*) – The name of the alert.
- **description** (*str, optional*) – A description for the alert.
- **trigger** (*tuple, optional*) – A tuple in the filter-tuple format detailing what would constitute a trigger. For example: `('sumip', '=', '1000')`.
- **always_exec_on_trigger** (*bool, optional*) – Should the trigger always execute when the trigger fires, or only execute when the returned data changes? Default is `False`.
- **schedule** (*dict, optional*) – This is the schedule dictionary that will inform Tenable Security Center how often to run the alert. If left unspecified then we will default to `{'type': 'never'}`.
- **action** (*list*) – The action(s) that will be performed when the alert trigger fires. Each action is a dictionary detailing what type of action to take, and the details surrounding that action.

Returns
The modified alert resource.

Return type
dict

Examples

```
>>> sc.alerts.update(1, name='New Alert Name')
```

execute(*id*)

Executes the specified alert.

alert: execute

Parameters
id (*int*) – The alert identifier.

Returns
The alert resource.

Return type
dict

list(*fields=None*)

Retrieves the list of alerts.

alert: list

Parameters
fields (*list, optional*) – A list of attributes to return for each alert.

Returns
A list of alert resources.

Return type
dict

Examples

```
>>> for alert in sc.alerts.list()['manageable']:
...     pprint(alert)
```

32.4 Analysis

The following methods allow for interaction into the Tenable Security Center analysis API. The analysis area in Tenable Security Center is highly complex and allows for a wide range of varied inputs and outputs. This single endpoint has been broken down in pyTenable to several methods in order to apply some defaults to the expected data-types and options most likely to be returned. As the filters are dependent on the tool and data-type that is being referenced, the best solution to understanding what filters are available when getting started is to simply pass a known bad filter string and use the resulting error as an indicator of what's available. For example, you could perform the following action below while attempting to see the available filters for the mobile data-type when using the `vulndetails` tool:

```
>>> x = sc.analysis.mobile(('something', '=', ''))
>>> x.next()
Traceback (most recent call last):
  File "<input>", line 1, in <module>
    x.next()
  File "tenable/base.py", line 75, in next
    self._get_page()
```

(continues on next page)

(continued from previous page)

```
  File "tenable/sc/analysis.py", line 43, in _get_page
    resp = self._api.post('analysis', json=query).json()
  File "tenable/base.py", line 436, in post
    return self._request('POST', path, **kwargs)
  File "tenable/base.py", line 379, in _request
    raise self._error_codes[status](resp)
ForbiddenError: 00000000-0000-0000-0000-000000000000:403 {"type":"regular",
"response":"","error_code":146,"error_msg":"Invalid parameters specified for
mobile vuln query.  The filter 'something' is invalid (valid filters:
repositoryIDs, port, pluginID, familyID, pluginOutput, lastSeen,
lastMitigated, severity, protocol, pluginName, baseCVSSScore,
exploitAvailable, pluginPublished, pluginModified, vulnPublished,
patchPublished, deviceID, mdmType, deviceModel, serialNumber, deviceUser,
deviceVersion, osCPE).","warnings":[],"timestamp":1545060739}
```

The resulting error details specifically what filters can be set.

When it comes to constructing filters, TenableSC uses a common filter structure for the collapsed filter-set. This format is in the form of a 3 entry tuple consisting of ('filtername', 'operator', 'value'). For example, if you're looking to set the `pluginID` filter to `19506` the filter would look like `('pluginID', '=', '19506')`. Severities are in level of criticality, from 0 (informational) to 4 (critical). Filters like these can be a string of comma- separated values to indicate multiple items. So for high and critical vulns, `('severity', '=', '3,4')` would return only what your looking for.

Asset list calculations in filters are a bit more complex, but still shouldn't be too difficult. Tenable Security Center leverages nested pairs for the asset calculations combined with a operator to define how that pair are to be combined. Each of the elements within the pair can further be nested, allowing for some quite complex asset list math to happen.

On the simple side, if you just want to look for The the combined results of asset lists 1 or 2, you would perform: `('asset', '~', ('or', 1, 2))`. Note the tilda, informing the filtering engine that it will need to perform some sort of calculation first. The tilda is only used when using the asset filter.

Now for a more complex calculation, you could look for the IPs that exist in both 1 or 2, but not 3: `('asset', '~', ('and', ('or', 1, 2), ('not', 3)))` As you can see it's just a matter of nesting out from "1 or 2". The only new concept here is the paired tuple for not. asking for the inverse of an asset list requires that you wrap it in a tuple with the not operator.

Methods available on `sc.analysis`:

class AnalysisAPI(*api: APISession*)

console(**filters*, ***kw*)

Queries the analysis API for log data from the Tenable Security Center Console itself.

analysis: sclog-type

Parameters

- **filters** (*tuple, optional*) – The analysis module provides a more compact way to write filters to the analysis endpoint. The purpose here is to aid in more readable code and reduce the amount of boilerplate that must be written to support a filtered call to analysis. The format is simply a list of tuples. Each tuple is broken down into (field, operator, value).
- **date** (*str, optional*) – A date in YYYYMM format. the default is simply "all".
- **pages** (*int, optional*) – The number of pages to query. Default is all.
- **limit** (*int, optional*) – How many entries should be in each page? Default is 200.

- **offset** (*int, optional*) – How many entries to skip before processing. Default is 0.
- **sort_field** (*str, optional*) – The field to sort the results on.
- **sort_direction** (*str, optional*) – The direction in which to sort the results. Valid settings are `asc` and `desc`. The default is `asc`.

Returns
An iterator object handling data pagination.

Return type
:obj:` AnalysisResultsIterator`

events(**filters*, ***kw*)

Queries the analysis API for event data from the Log Correlation Engine

analysis: event-type

Parameters

- **filters** (*tuple, optional*) – The analysis module provides a more compact way to write filters to the analysis endpoint. The purpose here is to aid in more readable code and reduce the amount of boilerplate that must be written to support a filtered call to analysis. The format is simply a list of tuples. Each tuple is broken down into (field, operator, value).
- **pages** (*int, optional*) – The number of pages to query. Default is all.
- **limit** (*int, optional*) – How many entries should be in each page? Default is 200.
- **offset** (*int, optional*) – How many entries to skip before processing. Default is 0.
- **source** (*str, optional*) – The data source location. Allowed sources are `lce` and `archive`. Defaults to `lce`.
- **silo_id** (*int, optional*) – If a silo id is specified, then the results fetched will be from the lce silo specified and not from the cumulative result set.
- **sort_field** (*str, optional*) – The field to sort the results on.
- **sort_direction** (*str, optional*) – The direction in which to sort the results. Valid settings are `asc` and `desc`. The default is `asc`.
- **tool** (*str, optional*) – The analysis tool for formatting and returning a specific view into the information. If no tool is specified, the default will be `vulndetails`. Available tools are: `listdata`, `sumasset`, `sumclassa`, `sumclassb`, `sumclassc`, `sumconns`, `sumdate`, `sumdstip`, `sumevent`, `sumevent2`, `sumip`, `sumport`, `sumprotocol`, `sumsrcip`, `sumtime`, `sumtype`, `sumuser`, `syslog`, `timedist`

Returns
An iterator object handling data pagination.

Return type
`AnalysisResultsIterator`

mobile(**filters*, ***kw*)

Queries the analysis API for mobile data collected from querying one or many MDM solutions.

analysis: mobile-type

Parameters

- **filters** (*tuple, optional*) – The analysis module provides a more compact way to write filters to the analysis endpoint. The purpose here is to aid in more readable code and reduce the amount of boilerplate that must be written to support a filtered call to analysis. The format is simply a list of tuples. Each tuple is broken down into (field, operator, value).
- **pages** (*int, optional*) – The number of pages to query. Default is all.
- **limit** (*int, optional*) – How many entries should be in each page? Default is 200.
- **offset** (*int, optional*) – How many entries to skip before processing. Default is 0.
- **sort_field** (*str, optional*) – The field to sort the results on.
- **sort_direction** (*str, optional*) – The direction in which to sort the results. Valid settings are `asc` and `desc`. The default is `asc`.
- **tool** (*str, optional*) – The analysis tool for formatting and returning a specific view into the information. If no tool is specified, the default will be `vulndetails`. Available tools are: `listvuln`, `sumdeviceid`, `summdmuser`, `summodel`, `sumoscpe`, `sumpluginid`, `sumseverity`, `vulndetails`

Returns
An iterator object handling data pagination.

Return type
`AnalysisResultsIterator`

scan(*scan_id*, **filters*, ***kw*)

Queries the analysis API for vulnerability data from a specific scan.

analysis: vuln-type

Parameters

- **scan_id** (*int*) – If a scan id is specified, then the results fetched will be from the scan specified and not from the cumulative result set.
- **filters** (*tuple, optional*) – The analysis module provides a more compact way to write filters to the analysis endpoint. The purpose here is to aid in more readable code and reduce the amount of boilerplate that must be written to support a filtered call to analysis. The format is simply a list of tuples. Each tuple is broken down into (field, operator, value).
- **pages** (*int, optional*) – The number of pages to query. Default is all.
- **limit** (*int, optional*) – How many entries should be in each page? Default is 200.
- **offset** (*int, optional*) – How many entries to skip before processing. Default is 0.
- **source** (*str, optional*) – The data source location. Allowed sources are `cumulative` and `patched`. Defaults to `cumulative`.
- **sort_field** (*str, optional*) – The field to sort the results on.
- **sort_direction** (*str, optional*) – The direction in which to sort the results. Valid settings are `asc` and `desc`. The default is `asc`.
- **tool** (*str, optional*) – The analysis tool for formatting and returning a specific view into the information. If no tool is specified, the default will be `vulndetails`. Available tools are: `cceipdetail`, `cveipdetail`, `iavmipdetail`, `iplist`, `listmailclients`, `listservices`, `listos`, `listsoftware`, `listsshservers`,

listvuln, listwebclients, listwebservers, sumasset, sumcce, sumclassa, sumclassb, sumclassc, sumcve, sumdnsname, sumfamily, sumiavm, sumid, sumip, summsbulletin, sumprotocol, sumremediation, sumseverity, sumuserresponsibility, sumport, trend, vulndetails, vulnipdetail, vulnipsummary

- **view** (*str, optional*) – The type of vulnerability slice you'd like to have returned. The returned data can be either all, new, or patched. If no view is specified, then the default will be all.

Returns
An iterator object handling data pagination.

Return type
AnalysisResultsIterator

Examples

A quick example showing how to get the information for a specific scan from SecurityCenter. As the default is for the scan method to return data from the vulndetails tool, we can handle this without actually doing anything other than calling

```
>>> for vuln in sc.analysis.scan(1):
...     pprint(vuln)
```

To ask for a specific subset of information (like only critical and exploitable vulns) you'd want to pass the filter tuples into the query like so:

```
>>> vulns = sc.analysis.scan(1
...     ('severity', '=', '4'),
...     ('exploitAvailable', '=', 'true'))
```

To request a different data format (like maybe an IP summary of vulns) you just need to specify the appropriate tool:

```
>>> ips = sc.analysis.scan(1
...     ('severity', '=', '4'),
...     ('exploitAvailable', '=', 'true'), tool='sumip')
```

vulns(**filters*, ***kw*)

Query's the analysis API for vulnerability data within the cumulative repositories.

analysis: vuln-type

Parameters

- **filters** (*tuple, optional*) – The analysis module provides a more compact way to write filters to the analysis endpoint. The purpose here is to aid in more readable code and reduce the amount of boilerplate that must be written to support a filtered call to analysis. The format is simply a list of tuples. Each tuple is broken down into (field, operator, value).
- **query_id** (*int, optional*) – The ID number of the SC Query where filters should be pulled from in place of the tuple filters. This is mutually exclusive with the tuple filters.
- **pages** (*int, optional*) – The number of pages to query. Default is all.

- **limit** (*int, optional*) – How many entries should be in each page? Default is 200.
- **offset** (*int, optional*) – How many entries to skip before processing. Default is 0.
- **source** (*str, optional*) – The data source location. Allowed sources are `cumulative` and `patched`. Defaults to `cumulative`.
- **scan_id** (*int, optional*) – If a scan id is specified, then the results fetched will be from the scan specified and not from the cumulative result set.
- **sort_field** (*str, optional*) – The field to sort the results on.
- **sort_direction** (*str, optional*) – The direction in which to sort the results. Valid settings are `asc` and `desc`. The default is `asc`.
- **tool** (*str, optional*) – The analysis tool for formatting and returning a specific view into the information. If no tool is specified, the default will be `vulndetails`. Available tools are: `cceipdetail`, `cveipdetail`, `iavmipdetail`, `iplist`, `listmailclients`, `listservices`, `listos`, `listsoftware`, `listsshservers`, `listvuln`, `listwebclients`, `listwebservers`, `sumasset`, `sumcce`, `sumclassa`, `sumclassb`, `sumclassc`, `sumcve`, `sumdnsname`, `sumfamily`, `sumiavm`, `sumid`, `sumip`, `summsbulletin`, `sumprotocol`, `sumremediation`, `sumseverity`, `sumuserresponsibility`, `sumport`, `trend`, `vulndetails`, `vulnipdetail`, `vulnipsummary`, `sumwasurl`, `wasvulndetail`, `waslistvuln`

Returns
An iterator object handling data pagination.

Return type
`AnalysisResultsIterator`

Examples

A quick example showing how to get all of the information stored in SecurityCenter. As the default is for the vulns method to return data from the vulndetails tool, we can handle this without actually doing anything other than calling

```
>>> from pprint import pprint
>>> for vuln in sc.analysis.vulns():
...     pprint(vuln)
```

To ask for a specific subset of information (like only critical and exploitable vulns) you'd want to pass the filter tuples into the query like so:

```
>>> vulns = sc.analysis.vulns(
...     ('severity', '=', '4'),
...     ('exploitAvailable', '=', 'true'))
```

To request a different data format (like maybe an IP summary of vulns) you just need to specify the appropriate tool:

```
>>> ips = sc.analysis.vulns(
...     ('severity', '=', '4'),
...     ('exploitAvailable', '=', 'true'), tool='sumip')
```

32.5 Asset Lists

The following methods allow for interaction into the Tenable Security Center Assets API. These items are typically seen under the **Assets** section of Tenable Security Center.

Methods available on `sc.asset_lists`:

class AssetListAPI(*api: APISession*)

create(*name*, *list_type*, ***kw*)

Creates an asset-list.

asset-list: create

Parameters

- **name** (*str*) – The name for the asset list to create.
- **list_type** (*str*) – The type of list to create. Supported values are `combination`, `dnsname`, `dnsnameupload`, `dynamic`, `ldapquery`, `static`, `staticeventfilter`, `staticvulnfilter`, `templates`, `upload`, `watchlist`, `watchlisteventfilter`, and `watchlistupload`.
- **combinations** (*tuple, optional*) – An asset combination tuple. For further information refer to the asset combination logic described at *tenable.sc.analysis*.
- **data_fields** (*list, optional*) – A list of data fields as required for a given asset list type. Each item within the list should be formatted in the following way: `{'fieldName': 'name', 'fieldValue': 'value'}`
- **description** (*str, optional*) – The description for the asset list being created.
- **dn** (*str, optional*) – The base DN to use for an LDAP query. Must also provide a `search_string` and an `ldap_id`.
- **dns_names** (*list, optional*) – When defining a DNS asset list, use this attribute to provide the list of DNS addresses.
- **exclude_managed_ips** (*bool, optional*) – Determines whether or not managed IPs should be excluded from the asset list.
- **filters** (*list, optional*) – A list of filter tuples to use when defining filtered asset list types. Follows the same format as filters within the rest of pyTenable.
- **fobj** (*FileObject, optional*) – A file-like object to use when uploading an asset list.
- **ips** (*list, optional*) – A list of IP Addresses, CIDRs, and/or IP Address ranges to use for the purposes of a static asset list.
- **lce_id** (*int, optional*) – When defining a event-based asset list, which LCE should be used to generate the asset list query.
- **ldap_id** (*int, optional*) – The numeric identifier pertaining to the LDAP server to use for an LDAP query. must also provide a `dn` and a `search_string`.
- **prep** (*bool, optional*) – Should asset preparation be run after the list is created? If unspecified, the default action is `True`.
- **rules** (*tuple, optional*) – For a dynamic asset list, the tuple definition of the rules to determine what Ips are associated to this asset list. Rules follow a similar pattern to the asset combination logic and are written in a way to follow the same visual methodology as the UI.

For example, a simple dynamic ruleset may look like:

```
('any', ('dns', 'contains', 'svc.company.tld'),
        ('dns', 'contains', 'prod.company.tld'))
```

Which would match all assets with either svc.company.tld or prod.company.tld in their DNS names. Rule gropups can be nested as well, by supplying a new group tuple instead of a rule:

```
('any', ('dns', 'contains', 'svc.company.tld'),
        ('dns', 'contains', 'prod.company.tld'),
        ('any', ('ip', 'contains', '192.168.140'),
                ('ip', 'contains', '192.168.141')))
```

In this example we have nested another group requiring that the ip may contain either of the values in addition to any of the DNS rules.

It's also possible to constrain the rule to a specific plugin or plugins as well by adding a 4th element in a rule tuple. Defining them would look like so:

```
# Singular Plugin ID
('plugintext', 'contains', 'credentialed', 19506)
# Multiple Plugin IDs
('plugintext', 'contains', 'stuff', [19506, 10180])
```

  - Available rules are `dns`, `exploitAvailable`, `exploitFrameworks`, `firstseen`, `mac`, `os`, `ip`, `uuid`, `lastseen`, `netbioshost`, `netbiosworkgroup`, `pluginid`, `plugintext`, `port`, `severity`, `sshv1`, `sshv2`, `tcpport`, `udpport`, and `xref`.
  - Available operators are `contains`, `eq`, `lt`, `lte`, `ne`, `gt`, `gte`, `regex`, `pcre`.
  - Group alauses are either `any` or `all`. Any is a logical or. All is a logical and.

- **scan_id** (*int, optional*) – When defining an "individual" source_type, the numeric id of the scan instance to base the query upon.
- **search_string** (*str, optional*) – The search string to use as part of an LDAP Query. Must also provide a `dn` and an `ldap_id`.
- **sort_dir** (*str, optional*) – When defining a filtered asset list type, determines the direction of the sort to use. This field must be passed when defining a sort_field.
- **sort_field** (*str, optional*) – When defining a filtered asset list type, determines what field to sort the resulting query on.
- **source_type** (*str, optional*) – The source of the data to query from when defining a filtered asset list type.
- **start_offset** (*int, optional*) – The start offset of the filter to use when defining a filtered asset list type.
- **tags** (*str, optional*) – A tag to associate to the asset list.
- **template** (*int, optional*) – The numeric id of the template to use.
- **tool** (*str, optional*) – When specifying filtered asset list types, the analysis tool to use for determining what IPs should be included within the asset list.
- **view** (*str, optional*) – When the source_type is "individual", the view defined what subset of the data to use.

Returns
The newly created asset-list.

Return type
dict

Examples

```
>>> asset-list = sc.asset_lists.create()
```

delete(*id*)

Removes a asset-list.

asset-list: delete

Parameters
id (*int*) – The numeric identifier for the asset-list to remove.

Returns
The deletion response dict

Return type
dict

Examples

```
>>> sc.asset_lists.delete(1)
```

details(*id*, *org_id=None*, *fields=None*)

Returns the details for a specific asset-list.

asset-list: details

Parameters

- **id** (*int*) – The identifier for the asset-list.
- **org_id** (*int, optional*) – The organizationID for the asset-list.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The details of asset id.

Return type
dict

Examples

```
>>> asset_id_details = sc.asset_lists.details(1,1)
>>> pprint(asset_id_details)
```

edit(*id*, ***kw*)

Edits an asset-list.

asset-list: edit

Parameters

- **id** (*int*) – The numeric id of the asset list to edit.
- **combinations** (*tuple, optional*) – An asset combination tuple. For further information refer to the asset combination logic described at *tenable.sc.analysis*.
- **data_fields** (*list, optional*) – A list of data fields as required for a given asset list type. Each item within the list should be formatted in the following way: `{'fieldName': 'name', 'fieldValue': 'value'}`
- **description** (*str, optional*) – The description for the asset list being created.
- **dn** (*str, optional*) – The base DN to use for an LDAP query. Must also provide a `search_string` and an `ldap_id`.
- **dns_names** (*list, optional*) – When defining a DNS asset list, use this attribute to provide the list of DNS addresses.
- **exclude_managed_ips** (*bool, optional*) – Determines whether or not managed IPs should be excluded from the asset list.
- **filters** (*list, optional*) – A list of filter tuples to use when defining filtered asset list types. Follows the same format as filters within the rest of pyTenable.
- **fobj** (*FileObject, optional*) – A file-like object to use when uploading an asset list.
- **ips** (*list, optional*) – A list of IP Addresses, CIDRs, and/or IP Address ranges to use for the purposes of a static asset list.
- **lce_id** (*int, optional*) – When defining a event-based asset list, which LCE should be used to generate the asset list query.
- **ldap_id** (*int, optional*) – The numeric identifier pertaining to the LDAP server to use for an LDAP query. must also provide a `dn` and a `search_string`.
- **name** (*str, optional*) – The name for the asset list to create.
- **prep** (*bool, optional*) – Should asset preparation be run after the list is created? If unspecified, the default action is `True`.
- **rules** (*tuple, optional*) – For a dynamic asset list, the tuple definition of the rules to determine what Ips are associated to this asset list. Rules follow a similar pattern to the asset combination logic and are written in a way to follow the same visual methodology as the UI.
- **scan_id** (*int, optional*) – When defining an "individual" source_type, the numeric id of the scan instance to base the query upon.
- **search_string** (*str, optional*) – The search string to use as part of an LDAP Query. Must also provide a `dn` and an `ldap_id`.
- **sort_dir** (*str, optional*) – When defining a filtered asset list type, determines the direction of the sort to use. This field must be passed when defining a sort_field.
- **sort_field** (*str, optional*) – When defining a filtered asset list type, determines what field to sort the resulting query on.
- **source_type** (*str, optional*) – The source of the data to query from when defining a filtered asset list type.
- **start_offset** (*int, optional*) – The start offset of the filter to use when defining a filtered asset list type.

- **tags** (*str, optional*) – A tag to associate to the asset list.
- **template** (*int, optional*) – The numeric id of the template to use.
- **tool** (*str, optional*) – When specifying filtered asset list types, the analysis tool to use for determining what IPs should be included within the asset list.
- **type** (*str, optional*) – The type of list to create. Supported values are `combination`, `dnsname`, `dnsnameupload`, `dynamic`, `ldapquery`, `static`, `staticeventfilter`, `staticvulnfilter`, `templates`, `upload`, `watchlist`, `watchlisteventfilter`, and `watchlistupload`.
- **view** (*str, optional*) – When the source_type is "individual", the view defined what subset of the data to use.

Returns
The newly updated asset-list.

Return type
dict

Examples

```
>>> asset-list = sc.asset_lists.edit()
```

export_definition(*id*, *fobj=None*)

Exports an asset list definition and stored the data in the file-like object that was passed.

asset-list: export

Parameters

- **id** (*int*) – The numeric identifier for the asset list to export.
- **fobj** (*FileObject*) – The file-like object to store the asset list XML definition.

Returns
The file-like object containing the XML definition.

Return type
FileObject

Examples

```
>>> with open('example.xml', 'wb') as fobj:
...     sc.asset_lists.export_definition(1, fobj)
```

import_definition(*fobj*, *name=None*)

Imports an asset list definition from an asset list definition XML file.

asset-list: import

Parameters

- **name** (*str*) – The name of the asset definition to create.
- **fobj** (*FileObject*) – The file-like object containing the XML definition.

Returns
The created asset list from the import.

Return type
dict

Examples

```
>>> with open('example.xml', 'rb') as fobj:
...     sc.asset_lists.import_definition('Example', fobj)
```

ldap_query(*ldap_id*, *dn*, *search_string*)

Performs a LDAP test query on the specified LDAP service configured.

asset-list: test-ldap-query

Parameters

- **ldap_id** (*int*) – The numeric identifier for the configured LDAP service.
- **dn** (*str*) – The valid search base to use.
- **search_string** (*str*) – The search string to query the LDAP service with.

Returns
The LDAP response.

Return type
dict

Examples

```
>>> resp = sc.asset_lists.ldap_query(1, 'domain.com', '*')
```

list(*fields=None*)

Retrieves the list of asset list definitions.

asset-list: list

Parameters
fields (*list, optional*) – A list of attributes to return for each asset-list.

Returns
A list of asset-list resources.

Return type
list

Examples

```
>>> for asset-list in sc.asset_lists.list():
...     pprint(asset-list)
```

refresh(*id*, *org_id*, **repos*)

Initiates an on-demand recalculation of the asset list. Note this endpoint requires being logged in as an admin user.

asset-list: refresh

Parameters

- **id** (*int*) – The numeric identifier of the asset list to refresh.
- **org_id** (*int*) – The organization associated to the asset list.
- ***repos** (*int*) – Repository ids to perform the recalculation on.

Returns
Response of the items that the asset list is associated to.

Return type
dict

Examples

Perform the refresh against a single repo:

```
>>> sc.asset_lists.refresh(1, 1, 1)
```

Perform the refresh against many repos:

```
>>> sc.asset_lists.refresh(1, 1, 1, 2, 3)
```

share(*id*, **groups*)

Shares the specified asset list to another user group.

asset-lists: share

Parameters

- **id** (*int*) – The numeric id for the credential.
- ***groups** (*int*) – The numeric id of the group(s) to share to.

Returns
The updated asset-list resource.

Return type
dict

Examples

```
>>> sc.asset_lists.share(1, group_1, group_2)
```

tags()

Retrieves the list of unique tags associated to asset lists.

asset-lists: tags

Returns
List of tags

Return type
list

Examples

```
>>> tags = sc.asset_lists.tags()
```

32.6 Audit Files

The following methods allow for interaction into the Tenable Security Center Audit File API and the Audit File Template API. These items are typically seen under the **Scans: Audit Files** section of Tenable Security Center.

Methods available on `sc.audit_files`:

class AuditFileAPI(*api: APISession*)

create(*name*, *audit_file=None*, *tailoring_file=None*, ***kw*)

Creates a audit file.

audit file: create

Parameters

- **name** (*str*) – The name of the audit file.
- **audit_file** (*FileObject, optional*) – The file-like object containing the audit file if uploading a custom audit file.
- **benchmark** (*str, optional*) – When the type is set to either SCAP datatype, this specifies the name of the benchmark.
- **data_stream** (*str, optional*) – When using version 1.2 of either SCAP datatype, you must specify the name of the data stream.
- **description** (*str, optional*) – A description of for the audit file.
- **profile** (*str, optional*) – When the type is set to either SCAP datatype, this specifies the name of the profile.
- **tailoring_file** (*FileObject, optional*) – When the SCAP version is set to 1.2, this tailoring file can optionally be provided.
- **template** (*int, optional*) – The audit file template it to use. If using a template, then no file is uploaded.
- **type** (*str, optional*) – The type of audit file to upload. Generally only used when uploading SCAP content as it will default to the Tenable-created audit-file format. Supported SCAP values are `scapWindows` and `scapLinux`.
- **vars** (*dict, optional*) – If a template is specified, then this dictionary specifies the parameters within the template to customize and what those values should be. The values are provided within the template definition.
- **version** (*str, optional*) – When specifying a SCAP datatype, this informs Tenable Security Center what version of SCAP this audit checklist is. Supported values are `1.0`, `1.1`, and `1.2`.

Returns
The newly created audit file.

Return type
dict

Examples

```
>>> audit = sc.audit_files.create()
```

delete(*id*)

Removes a audit file.

audit file: delete

Parameters
id (*int*) – The numeric identifier for the audit file to remove.

Returns
An empty response.

Return type
str

Examples

```
>>> sc.audit_files.delete(1)
```

details(*id*, *fields=None*)

Returns the details for a specific audit file.

audit file: details

Parameters

- **id** (*int*) – The identifier for the audit file.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The audit file resource record.

Return type
dict

Examples

```
>>> audit = sc.audit_files.details(1)
>>> pprint(audit)
```

edit(*id*, *audit_file=None*, *tailoring_file=None*, ***kw*)

Edits a audit file.

audit file: edit

Parameters

- **audit_file** (*FileObject, optional*) – The file-like object containing the audit file if uploading a custom audit file.
- **benchmark** (*str, optional*) – When the type is set to either SCAP datatype, this specifies the name of the benchmark.
- **data_stream** (*str, optional*) – When using version 1.2 of either SCAP datatype, you must specify the name of the data stream.

- **description** (*str, optional*) – A description of for the audit file.
- **name** (*str, optional*) – The name of the audit file.
- **profile** (*str, optional*) – When the type is set to either SCAP datatype, this specifies the name of the profile.
- **tailoring_file** (*FileObject, optional*) – When the SCAP version is set to 1.2, this tailoring file can optionally be provided.
- **template** (*int, optional*) – The audit file template it to use. If using a template, then no file is uploaded.
- **type** (*str, optional*) – The type of audit file to upload. Generally only used when uploading SCAP content as it will default to the Tenable-created audit-file format. Supported SCAP values are `scapWindows` and `scapLinux`.
- **vars** (*dict, optional*) – If a template is specified, then this dictionary specifies the parameters within the template to customize and what those values should be. The values are provided within the template definition.
- **version** (*str, optional*) – When specifying a SCAP datatype, this informs Tenable Security Center what version of SCAP this audit checklist is. Supported values are `1.0`, `1.1`, and `1.2`.

Returns
The newly updated audit file.

Return type
`dict`

Examples

```
>>> audit = sc.audit_files.edit()
```

export_audit(*id*, *fobj=None*)

Exports an Audit File.

audit file: export

Parameters

- **id** (*int*) – The audit file numeric identifier.
- **fobj** (*FileObject, optional*) – The file-like object to write the resulting file into. If no file-like object is provided, a BytesIO objects with the downloaded file will be returned. Be aware that the default option of using a BytesIO object means that the file will be stored in memory, and it's generally recommended to pass an actual file-object to write to instead.

Returns
The file-like object with the resulting zipped report.

Return type
`FileObject`

Examples

```
>>> with open('example.zip', 'wb') as fobj:
...     sc.audit_files.export_audit(1, fobj)
```

list(*fields=None*)

Retrieves the list of audit file definitions.

audit file: list

Parameters
fields (*list, optional*) – A list of attributes to return for each audit file.

Returns
A list of audit file resources.

Return type
list

Examples

```
>>> for audit in sc.audit_files.list():
...     pprint(audit)
```

template_categories()

Returns the audit file template categories

audit template: categories

Returns
List of audit file category listing dicts.

Return type
list

Examples

```
>>> for cat in sc.audit_files.template_categorties():
...     pprint(cat)
```

template_details(*id*, *fields=None*)

Returns the details for the specified audit file template id.

audit template: details

Parameters

- **id** (*int*) – The numeric identifier for the audit file template.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The audit file template record.

Return type
dict

Examples

```
>>> tmpl = sc.audit_files.template_details(1)
```

template_list(*category=None*, *search=None*, *fields=None*)

Returns the list of audit file templates.

audit templates: list

Parameters

- **category** (*int, optional*) – Restrict the results to only the specified category id.
- **fields** (*list, optional*) – A list of attributes to return.
- **search** (*str, optional*) – Restrict the response to only audit file names that match the search string specified.

Returns

List of audit file records.

Return type

list

Examples

```
>>> for tmpl in sc.audit_files.template_list():
...     pprint(tmpl)
```

32.7 Credentials

The following methods allow for interaction into the Tenable Security Center Scan Credentials API. These items are typically seen under the **Scan Credentials** section of Tenable Security Center.

Methods available on `sc.credentials`:

class CredentialAPI(*api: APISession*)

create(*name*, *cred_type*, *auth_type*, ***kw*)

Creates a credential.

credential: create

Parameters

- **name** (*str*) – The name for the credential.
- **cred_type** (*str*) – The type of credential to store. Valid types are `database`, `snmp`, `ssh`, and `windows`.
- **auth_type** (*str*) – The type of authentication for the credential. Valid types are `beyondtrust`, `certificate`, cyberark``, `kerberos`, `lieberman`, `lm`, `ntlm`, `password`, `publicKey`, `thycotic`.
- **beyondtrust_api_key** (*str, optional*) – The API key to use for authenticating to Beyondtrust.

- **beyondtrust_duration** (*int, optional*) – The length of time to cache the checked-out credentials from Beyondtrust. This value should be less than the password change interval within Beyondtrust.
- **beyondtrust_host** (*str, optional*) – The host address for the Beyondtrust application.
- **beyondtrust_port** (*int, optional*) – The port number associated with the Beyondtrust application.
- **beyondtrust_use_escalation** (*bool, optional*) – If enabled, informs the scanners to use Beyondtrust for privilege escalation.
- **beyondtrust_use_private_key** (*bool, optional*) – If enabled, informs the scanners to use key-based auth for SSH connections instead of password auth.
- **beyondtrust_use_ssl** (*bool, optional*) – Should the scanners communicate to Beyondtrust over SSL for credential retrieval? If left unspecified, the default is set to `True`.
- **beyondtrust_verify_ssl** (*bool, optional*) – Should the SSL certificate be validated when communicating to Beyondtrust? If left unspecified, the default is `False`.
- **community_string** (*str, optional*) – The SNMP community string to use for authentication.
- **db_type** (*str, optional*) – The type of database connection that will be performed. Valid types are `DB2`, `Informix/DRDA`, `MySQL`, `Oracle`, `PostgreSQL`, `SQL Server`.
- **description** (*str, optional*) – A description to associate to the credential.
- **domain** (*str, optional*) – The Active Directory domain to use if the user is a member of a domain.
- **escalation_path** (*str, optional*) – The path in which to run the escalation commands.
- **escalation_password** (*str, optional*) – The password to use for the escalation.
- **escalation_su_use** (*str, optional*) – If performing an SU escalation, this is the user to escalate to.
- **escalation_username** (*str, optional*) – The username to escalate to.
- **kdc_ip** (*str, optional*) – The kerberos host supplying the session tickets.
- **kdc_port** (*int, optional*) – The port to use for kerberos connections. If left unspecified the default is `88`.
- **kdc_protocol** (*str, optional*) – The protocol to use for kerberos connections. Valid options are `tcp` and `udp`. If left unspecified then the default is `tcp`.
- **kdc_realm** (*str, optional*) – The Kerberos realm to use for authentication.
- **lieberman_host** (*str, optional*) – The address for the Lieberman vault.
- **lieberman_port** (*int, optional*) – The port number where the Lieberman service is listening.
- **lieberman_pam_password** (*str, optional*) – The password to authenticate to the Lieberman RED API.
- **lieberman_pam_user** (*str, optional*) – The username to authenticate to the Lieberman RED API.

- **lieberman_system_name** (*str, optional*) – The name for the credentials in Lieberman.
- **lieberman_use_ssl** (*bool, optional*) – Should the scanners communicate to Lieberman over SSL for credential retrieval? If left unspecified, the default is set to True.
- **lieberman_verify_ssl** (*bool, optional*) – Should the SSL certificate be validated when communicating to Lieberman? If left unspecified, the default is False.
- **password** (*str, optional*) – The password for the credential.
- **port** (*int, optional*) – A valid port number for a database credential.
- **private_key** (*file, optional*) – The fileobject containing the SSH private key.
- **privilege_escalation** (*str, optional*) – The type of privilege escalation to perform once authenticated. Valid values are .k5login, Cisco 'enable', dzdo, none, pbrun, su, su+sudo, sudo. If left unspecified, the default is none.
- **public_key** (*file, optional*) – The fileobject containing the SSH public key or certificate.
- **oracle_auth_type** (*str, optional*) – The type of authentication to use when communicating to an Oracle database server. Supported values are sysdba, sysoper, and normal. If left unspecified, the default option is normal.
- **oracle_service_type** (*str, optional*) – The type of service identifier specified in the sid parameter. Valid values are either sid or service_name. If left unspecified, the default is sid.
- **sid** (*str, optional*) – The service identifier or name for a database credential.
- **sql_server_auth_type** (*str, optional*) – The type of authentication to perform to the SQL Server instance. Valid values are SQL and Windows. The default value if left unspecified is SQL.
- **tags** (*str, optional*) – A tag to associate to the credential.
- **username** (*str, optional*) – The username for the OS credential.
- **thycotic_domain** (*str, optional*) – The domain, if set, within Thycotic.
- **thycotic_organization** (*str, optional*) – The organization to use if using a cloud instance of Thycotic.
- **thycotic_password** (*str, optional*) – The password to use when authenticating to Thycotic.
- **thycotic_private_key** (*bool, optional*) – If enabled, informs the scanners to use key-based auth for SSH connections instead of password auth.
- **thycotic_secret_name** (*str, optional*) – The secret name value on the Tycotic server.
- **thycotic_url** (*str, optional*) – The absolute URL path pointing to the Thycotic secret server.
- **thycotic_username** (*str, optional*) – The username to use to authenticate to Thycotic.
- **thycotic_verify_ssl** (*bool, optional*) – Should the SSL certificate be validated when communicating to Thycotic? If left unspecified, the default is False.

- **vault_account_name** (*str, optional*) – The unique name of the credential to retrieve from CyberArk. Generally referred to as the *name* parameter within CyberArk.
- **vault_address** (*str, optional*) – The domain for the CyberArk account. SSL must be configured through IIS on the CCP before using.
- **vault_app_id** (*str, optional*) – The AppID to use with CyberArk.
- **vault_cyberark_client_cert** (*file, optional*) – The fileobject containing the CyberArk client certificate.
- **vault_cyberark_url** (*str, optional*) – The URL for the CyberArk AIM web service. If left unspecified, the default URL path of `/AIMWebservice/v1.1/AIM.asmx` will be used..
- **vault_cyberark_private_key** (*file, optional*) – The fileobject containing the CyberArk client private key.
- **vault_cyberark_private_key_passphrase** (*str, optional*) – The passhrase for the private key.
- **vault_folder** (*str, optional*) – The folder to use within CyberArk for credential retrieval.
- **vault_host** (*str, optional*) – The CyberArk Vault host.
- **vault_password** (*str, optional*) – The password to use for authentication to the vault if the CyberArk Central Credential Provider is configured for basic auth.
- **vault_policy_id** (*int, optional*) – The CyberArk PolicyID assigned to the credentials to retrieve.
- **vault_port** (*int, optional*) – The port in which the CyberArk Vault resides.
- **vault_safe** (*str, optional*) – The CyberArk safe that contains the credentials to retrieve.
- **vault_use_ssl** (*bool, optional*) – Should the scanners communicate to CyberArk over SSL for credential retrieval? If left unspecified, the default is set to `True`.
- **vault_username** (*str, optional*) – The username to use for authentication to the vault if the CyberArk Central Credential Provider is configured for basic auth.
- **vault_verify_ssl** (*bool, optional*) – Should the SSL certificate be validated when communicating to the vault? If left unspecified, the default is `False`.

Returns
The newly created credential.

Return type
dict

Examples

Creating a Windows AD credential:

```
>>> cred = sc.credentials.create(
...     'Example AD User', 'windows', 'ntlm',
...     username='scanneruser',
...     password='sekretpassword',
...     domain='Company.com')
```

Creating a root user SSH credential:

```
>>> cred = sc.credentials.create(
...     'Example SSH Cred', 'ssh', 'password',
...     username='root',
...     password='sekretpassword')
```

Creating a root user SSH cred with a private key:

```
>>> with open('privatekeyfile', 'rb') as keyfile:
...     cred = sc.credentials.create(
...         'Example SSH Keys', 'ssh', 'publickey',
...         username='root',
...         private_key=keyfile)
```

Creating a normal user SSH cred with sudo for privilege escalation:

```
>>> cred = sc.credentials.create(
...     'Example SSH Sudo', 'ssh', 'password',
...     username='user',
...     password='sekretpassword',
...     privilege_escalation='sudo',
...     escalation_password='sekretpassword')
```

Creating a SQL Server cred set:

```
>>> cred = sc.credentials.create(
...     'Example SQL Server', 'database', 'SQL Server',
...     username='sa',
...     password='sekretpassword',
...     sql_server_auth_type='SQL',
...     sid='database_name')
```

delete(*id*)

Removes a credential.

credential: delete

Parameters
id (*int*) – The numeric identifier for the credential to remove.

Returns
An empty response.

Return type
str

Examples

```
>>> sc.credentials.delete(1)
```

details(*id*, *fields=None*)

Returns the details for a specific credential.

credential: details

Parameters

- **id** (*int*) – The identifier for the credential.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The credential resource record.

Return type
dict

Examples

```
>>> cred = sc.credentials.details(1)
>>> pprint(cred)
```

edit(*id*, ***kw*)

Edits a credential.

credential: edit

Parameters

- **auth_type** (*str, optional*) – The type of authentication for the credential. Valid types are beyondtrust, certificate, cyberark``, kerberos, lieberman, lm, ntlm, password, publickey, thycotic.
- **beyondtrust_api_key** (*str, optional*) – The API key to use for authenticating to Beyondtrust.
- **beyondtrust_duration** (*int, optional*) – The length of time to cache the checked-out credentials from Beyondtrust. This value should be less than the password change interval within Beyondtrust.
- **beyondtrust_host** (*str, optional*) – The host address for the Beyondtrust application.
- **beyondtrust_port** (*int, optional*) – The port number associated with the Beyondtrust application.
- **beyondtrust_use_escalation** (*bool, optional*) – If enabled, informs the scanners to use Beyondtrust for privilege escalation.
- **beyondtrust_use_private_key** (*bool, optional*) – If enabled, informs the scanners to use key-based auth for SSH connections instead of password auth.
- **beyondtrust_use_ssl** (*bool, optional*) – Should the scanners communicate to Beyondtrust over SSL for credential retrieval? If left unspecified, the default is set to True.

- **beyondtrust_verify_ssl** (*bool, optional*) – Should the SSL certificate be validated when communicating to Beyondtrust? If left unspecified, the default is `False`.
- **community_string** (*str, optional*) – The SNMP community string to use for authentication.
- **db_type** (*str, optional*) – The type of database connection that will be performed. Valid types are `DB2`, `Informix/DRDA`, `MySQL`, `Oracle`, `PostgreSQL`, `SQL Server`.
- **description** (*str, optional*) – A description to associate to the credential.
- **domain** (*str, optional*) – The Active Directory domain to use if the user is a member of a domain.
- **escalation_path** (*str, optional*) – The path in which to run the escalation commands.
- **escalation_password** (*str, optional*) – The password to use for the escalation.
- **escalation_su_use** (*str, optional*) – If performing an SU escalation, this is the user to escalate to.
- **escalation_username** (*str, optional*) – The username to escalate to.
- **kdc_ip** (*str, optional*) – The kerberos host supplying the session tickets.
- **kdc_port** (*int, optional*) – The port to use for kerberos connections. If left unspecified the default is `88`.
- **kdc_protocol** (*str, optional*) – The protocol to use for kerberos connections. Valid options are `tcp` and `udp`. If left unspecified then the default is `tcp`.
- **kdc_realm** (*str, optional*) – The Kerberos realm to use for authentication.
- **lieberman_host** (*str, optional*) – The address for the Lieberman vault.
- **lieberman_port** (*int, optional*) – The port number where the Lieberman service is listening.
- **lieberman_pam_password** (*str, optional*) – The password to authenticate to the Lieberman RED API.
- **lieberman_pam_user** (*str, optional*) – The username to authenticate to the Lieberman RED API.
- **lieberman_system_name** (*str, optional*) – The name for the credentials in Lieberman.
- **lieberman_use_ssl** (*bool, optional*) – Should the scanners communicate to Lieberman over SSL for credential retrieval? If left unspecified, the default is set to `True`.
- **lieberman_verify_ssl** (*bool, optional*) – Should the SSL certificate be validated when communicating to Lieberman? If left unspecified, the default is `False`.
- **name** (*str, optional*) – The name for the credential.
- **password** (*str, optional*) – The password for the credential.
- **port** (*int, optional*) – A valid port number for a database credential.
- **private_key** (*file, optional*) – The fileobject containing the SSH private key.
- **privilege_escalation** (*str, optional*) – The type of privilege escalation to perform once authenticated. Valid values are `.k5login`, `Cisco 'enable'`, `dzdo`, `none`, `pbrun`, `su`, `su+sudo`, `sudo`. If left unspecified, the default is `none`.

- **public_key** (*file, optional*) – The fileobject containing the SSH public key or certificate.
- **oracle_auth_type** (*str, optional*) – The type of authentication to use when communicating to an Oracle database server. Supported values are `sysdba`, `sysoper`, and `normal`. If left unspecified, the default option is `normal`.
- **oracle_service_type** (*str, optional*) – The type of service identifier specified in the `sid` parameter. Valid values are either `sid` or `service_name`. If left unspecified, the default is `sid`.
- **sid** (*str, optional*) – The service identifier or name for a database credential.
- **sql_server_auth_type** (*str, optional*) – The type of authentication to perform to the SQL Server instance. Valid values are `SQL` and `Windows`. The default value if left unspecified is `SQL`.
- **tags** (*str, optional*) – A tag to associate to the credential.
- **type** (*str. optional*) – The type of credential to store. Valid types are `database`, `snmp`, `ssh`, and `windows`.
- **username** (*str, optional*) – The username for the OS credential.
- **thycotic_domain** (*str, optional*) – The domain, if set, within Thycotic.
- **thycotic_organization** (*str, optional*) – The organization to use if using a cloud instance of Thycotic.
- **thycotic_password** (*str, optional*) – The password to use when authenticating to Thycotic.
- **thycotic_private_key** (*bool, optional*) – If enabled, informs the scanners to use key-based auth for SSH connections instead of password auth.
- **thycotic_secret_name** (*str, optional*) – The secret name value on the Tycotic server.
- **thycotic_url** (*str, optional*) – The absolute URL path pointing to the Thycotic secret server.
- **thycotic_username** (*str, optional*) – The username to use to authenticate to Thycotic.
- **thycotic_verify_ssl** (*bool, optional*) – Should the SSL certificate be validated when communicating to Thycotic? If left unspecified, the default is `False`.
- **vault_account_name** (*str, optional*) – The unique name of the credential to retrieve from CyberArk. Generally referred to as the *name* parameter within CyberArk.
- **vault_address** (*str, optional*) – The domain for the CyberArk account. SSL must be configured through IIS on the CCP before using.
- **vault_app_id** (*str, optional*) – The AppID to use with CyberArk.
- **vault_cyberark_client_cert** (*file, optional*) – The fileobject containing the CyberArk client certificate.
- **vault_cyberark_url** (*str, optional*) – The URL for the CyberArk AIM web service. If left unspecified, the default URL path of `/AIMWebservice/v1.1/AIM.asmx` will be used..
- **vault_cyberark_private_key** (*file, optional*) – The fileobject containing the CyberArk client private key.

- **vault_cyberark_private_key_passphrase** (*str, optional*) – The passhrase for the private key.
- **vault_folder** (*str, optional*) – The folder to use within CyberArk for credential retrieval.
- **vault_host** (*str, optional*) – The CyberArk Vault host.
- **vault_password** (*str, optional*) – The password to use for authentication to the vault if the CyberArk Central Credential Provider is configured for basic auth.
- **vault_policy_id** (*int, optional*) – The CyberArk PolicyID assigned to the credentials to retrieve.
- **vault_port** (*int, optional*) – The port in which the CyberArk Vault resides.
- **vault_safe** (*str, optional*) – The CyberArk safe that contains the credentials to retrieve.
- **vault_use_ssl** (*bool, optional*) – Should the scanners communicate to CyberArk over SSL for credential retrieval? If left unspecified, the default is set to `True`.
- **vault_username** (*str, optional*) – The username to use for authentication to the vault if the CyberArk Central Credential Provider is configured for basic auth.
- **vault_verify_ssl** (*bool, optional*) – Should the SSL certificate be validated when communicating to the vault? If left unspecified, the default is `False`.

Returns
The newly updated credential.

Return type
dict

Examples

```
>>> cred = sc.credentials.edit()
```

list(*fields=None*)

Retrieves the list of credential definitions.

- credential: list

Parameters
fields (*list, optional*) – A list of attributes to return for each credential.

Returns
A list of credential resources.

Return type
list

Examples

```
>>> for cred in sc.credentials.list():
...     pprint(cred)
```

share(*id*, **groups*)

Shares the specified credential to another user group.

credential: share

Parameters

- **id** (*int*) – The numeric id for the credential.
- ***groups** (*int*) – The numeric id of the group(s) to share to.

Returns
The updated credential resource.

Return type
dict

Examples

```
>>> sc.credentials.share(1, group_1, group_2)
```

tags()

Retrieves the list of unique tags associated to credentials.

credential: tags

Returns
List of tags

Return type
list

Examples

```
>>> tags = sc.credentials.tags()
```

32.8 Current Session

The following methods allow for interaction with the Tenable Security Center CurrentOrganization API and the CurrentUser API.

Methods available on `sc.current`:

class CurrentSessionAPI(*api: APISession*)

associate_cert()

Associates the certificate passed to the server with the current user's account. This allows for authentication via certificate in subsequent logins.

Returns
The updated user record.

Return type
dict

Examples

```
>>> sc.current.associate_cert()
```

org(*fields=None*)

Returns the organization of the current session.

current-organization

Parameters
fields (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the organization list API doc.

Returns
The organization record.

Return type
dict

Examples

```
>>> org = sc.current.org()
```

user(*fields=None*)

Returns the user of the current session.

current-user

Parameters
fields (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the organization list API doc.

Returns
The user record.

Return type
dict

Examples

```
>>> user = sc.current.user()
```

32.9 Feeds

The following methods allow for interaction into the Tenable Security Center Feed API.

Methods available on `sc.feeds`:

class FeedAPI(*api: APISession*)

process(*feed_type*, *fobj*)

Initiates an off-line feed update based on the specified feed_type using the file object passed as the update file.

feed: process

Parameters

- **feed_type** (*str*) – The feed type to specifically return. Valid types are *active*, *passive*, *lce*, *sc*, or *all*.
- **fobj** (*FileObject*) – The file object to upload into SecurityCenter and use as the update package.

Returns

Update successfully requested.

Return type

None

Examples

updating the active plugins:

```
>>> with open('sc-plugins-diff.tar.gz', 'rb') as plugfile:
...     sc.feeds.process('active', plugfile)
```

status(*feed_type=None*)

Returns the status of either a specific feed type (if requested) or all of the feed types if nothing is specifically asked.

feed

feed: feed-type

Parameters

feed_type (*str, optional*) – The feed type to specifically return. Valid types are *active*, *passive*, *lce*, *sc*, or *all*.

Returns

If no specific feed type is specified, then a dictionary with each type listed with a sub-dictionary detailing the status is returned. If a specific feed type is requested, then only the status information for that feed type is returned.

Return type

dict

Examples

Getting all of the feed types returned:

```
>>> status = sc.feed.status()
```

Getting the feed status for a specific type (e.g. *active*).

```
>>> status = sc.feeds.status('active')
```

update(*feed_type=None*)

Initiates an on-line feed update based on the specified feed_type. If no feed type is specified, then it will default to initiating an update for all feed types.

feed: update

Parameters
feed_type (*str, optional*) – The feed type to specifically return. Valid types are *active*, *passive*, *lce*, *sc*, or *all*.

Returns
Update successfully requested.

Return type
None

32.10 Files

The following methods allow for interaction into the Tenable Security Center File API.

Methods available on `sc.files`:

class FileAPI(*api: APISession*)

clear(*filename*)

Removes the requested file from Tenable Security Center.

file: clear

Parameters
filename (*str*) – The file identifier associated to the file.

Returns
The file location on disk that was removed.

Return type
str

upload(*fobj*)

Uploads a file into SecurityCenter and returns the file identifier to be used for subsequent calls.

file: upload

Parameters
fobj (*FileObj*) – The file object to upload into SecurityCenter.

Returns
The filename identifier to use for subsequent calls in Tenable Security Center.

Return type
str

32.11 Groups

The following methods allow for interaction into the Tenable Security Center Group API. These items are typically seen under the **User Groups** section of Tenable Security Center.

Methods available on `sc.groups`:

class GroupAPI(*api: APISession*)

create(*name*, ***kw*)

Creates a group.

group: create

Parameters

- **name** (*str*) – The name of the user group
- **asset_lists** (*list, optional*) – List of asset list ids to allow this group to access.
- **audit_files** (*list, optional*) – List of audit file ids to allow this group to access.
- **dashboards** (*list, optional*) – List of dashboard ids to allow this group to access.
- **lce_ids** (*list, optional*) – List of LCE ionstance ids to allow this group to access.
- **query_ids** (*list, optional*) – List of query ids to allow this group to access.
- **report_cards** (*list, optional*) – List of report card ids to allow this group to access.
- **repos** (*list, optional*) – List of repository ids to allow this group to access.
- **scan_creds** (*list, optional*) – List of scanning credential ids to allow this group to access.
- **scan_policies** (*list, optional*) – List of scan policy ids to allow this group to access.
- **viewable** (*list, optional*) – List of asset list ids to use for the purposes of restricting what members of this group can see within Tenable Security Center.

Returns
The newly created group.

Return type
dict

Examples

```
>>> group = sc.groups.create('New Group')
```

delete(*id*)

Removes a group.

group: delete

Parameters
id (*int*) – The numeric identifier for the group to remove.

Returns
An empty response.

Return type
str

Examples

```
>>> sc.groups.delete(1)
```

details(*id*, *fields=None*)

Returns the details for a specific group.

group: details

Parameters

- **id** (*int*) – The identifier for the group.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The group resource record.

Return type
dict

Examples

```
>>> group = sc.groups.details(1)
>>> pprint(group)
```

edit(*id*, ***kw*)

Edits a group.

group: edit

Parameters

- **asset_lists** (*list, optional*) – List of asset list ids to allow this group to access.
- **audit_files** (*list, optional*) – List of audit file ids to allow this group to access.
- **dashboards** (*list, optional*) – List of dashboard ids to allow this group to access.
- **lce_ids** (*list, optional*) – List of LCE ionstance ids to allow this group to access.

- **name** (*str, optional*) – The name of the user group
- **query_ids** (*list, optional*) – List of query ids to allow this group to access.
- **report_cards** (*list, optional*) – List of report card ids to allow this group to access.
- **repos** (*list, optional*) – List of repository ids to allow this group to access.
- **scan_creds** (*list, optional*) – List of scanning credential ids to allow this group to access.
- **scan_policies** (*list, optional*) – List of scan policy ids to allow this group to access.
- **viewable** (*list, optional*) – List of asset list ids to use for the purposes of restricting what members of this group can see within Tenable Security Center.

Returns
The newly updated group.

Return type
dict

Examples

```
>>> group = sc.groups.edit()
```

list(*fields=None*)

Retrieves the list of group definitions.

group: list

Parameters
fields (*list, optional*) – A list of attributes to return for each group.

Returns
A list of group resources.

Return type
list

Examples

```
>>> for group in sc.groups.list():
...     pprint(group)
```

32.12 Hosts

The following methods allow for interaction with the Tenable Security Center Hosts API. These items are typically seen under the **Hosts** section of Tenable Security Center.

Methods available on `sc.hosts`:

class HostsAPI(*api: APISession*)

list(*fields: List[str] | None = None*, *limit: int = 10000*, *offset: int = 0*, *pages: int | None = None*, *pagination: bool = True*, *return_json: bool = False*) → HostsResultsIterator | Dict

Retrieve the list of hosts from the system.

Parameters

- **fields** (*list[str], optional*) – What fields should be returned in the response.
- **limit** (*int, 1000*) – How many hosts should be returned?
- **offset** (*int, 0*) – At what index should
- **pages** (*int, optional*) – The maximum number of pages to return for the iterator.
- **pagination** (*bool, False*) – Should pagination be used?
- **return_json** (*bool, False*) – Should we return the json response instead of an iterator?

Response:

The response will be either the HostResultsIterator to handle pagination of the data (preferred) or the raw response from the API (if return_json is set to *True*).

Examples

```
>>> for host in sc.hosts.list():
...     print(host)
```

search(**filters: Tuple[str, str, str]*, *filter_type: Literal['and', 'or'] = 'and'*, *fields: List[str] | None = None*, *limit: int = 10000*, *offset: int = 0*, *pages: int | None = None*, *pagination: bool = True*, *return_json: bool = False*) → HostsResultsIterator | Dict

Retrieve the list of hosts from the system.

Parameters

- **filters** (*list[tuple[str, str, str]], optional*) – List of search filter tuples.
- **filter_type** (*Literal['and', 'or'], optional*) – The filtering boolean logic to use for multiple filters. If left unspecified it defaults to *and*.
- **fields** (*list[str], optional*) – What fields should be returned in the response.
- **limit** (*int, 1000*) – How many hosts should be returned?
- **offset** (*int, 0*) – At what index should
- **pages** (*int, optional*) – The maximum number of pages to return for the iterator.
- **pagination** (*bool, False*) – Should pagination be used?

- **return_json** (*bool, False*) – Should we return the json response instead of an iterator?

Response:
The response will be either the HostResultsIterator to handle pagination of the data (preferred) or the raw response from the API (if return_json is set to *True*).

Examples

```
>>> for host in sc.hosts.search(filters=[('ip', 'eq', '1.2.3.4')]):
...     print(host)
```

update_acr(*host_uuid: str*, *reasoning: List[int] | None = None*, *score: int | None = None*, *notes: str | None = None*, *overwritten: bool = True*) → Dict

Override the Asset Criticality Rating (ACR) score and the reasons for the specified host.

Parameters

- **host_uuid** (*str*) – The Host UUID to modify
- **reasonings** (*list[int], optional*) – The list of reasoning objects noting why the score was changed
- **score** (*int, optional*) – The updated ACR score
- **notes** (*str, optional*) – Notes detailing why the score was changed
- **overwritten** (*bool*) – Should we use the overwritten score or the default one?

Returns
The updated host object.

Example

```
>>> sc.host.update_acr(
...     host_uuid='12345678-1234-1234-123456789012',
...     score=7,
...     reasonings=[4],
...     notes='Why we changed this score...',
... )
```

32.13 License API

The following methods allow for interaction into the Tenable Security Center Configuration API. These items are typically seen under the **License Configuration** section of the Tenable Security Center Settings UI.

This API is simple and utilitarian. No translation of the data returned from the raw API is done.

Methods available on `sc.license`:

class LicenseAPI(*api: APISession*)

The way this works is broken for a number of reasons. The first is that we have to submit the file AND bind the license in one HTTP(S) session. Otherwise the file gets deleted and the reference to the file is lost. This is why we have to do the file read and upload in a single session in the set() method.

details()

Retrieves the current license information.

license: get

Returns
The license information.

Return type
dict

update(*file*)

Sets the license file.

license: set

Parameters
file (*str*) – The path to the license file to upload.

We are using the internal API to do this:

32.14 Organizations

The following methods allow for interaction with the Tenable Security Center Organization API. These items are typically seen under the **Organization** section of Tenable Security Center.

Methods available on `sc.organizations`:

class OrganizationAPI(*api: APISession*)

accept_risk_rules(*organization_id*, *repos=None*, *plugin=None*, *port=None*)

Retrieves the accepted risk rules for the organization and optionally will filter based on the parameters specified.

organization: accept-risk-rule

Parameters

- **organization_id** (*int*) – The organization id.
- **repos** (*list, optional*) – A list of repository ids to restrict the search to.
- **plugin** (*int, optional*) – A plugin id to restrict the search to.
- **port** (*int, optional*) – A port number to restrict the search to.

Returns
A list of rules that match the request.

Return type
list

Examples

```
>>> for rule in sc.organizations.accept_risk_rules(1):
...     pprint(rule)
```

create(*name*, ***kwargs*)

Create a new organization

SC Organization Create

Parameters

- **name** (*str*) – The name for organization.
- **info_links** (*list, optional*) – A list of custom analysis links provided to users within the host vulnerability details when analyzing data outside of SecurityCenter is desired. Links shall be described in a tuple format with `(name, link)` format. For example: `('SANS', 'https://isc.sans.edu/ipinfo.html?ip=%IP%')`
- **lce_ids** (*list, optional*) – What Log Correlation Engines (if any) should this organization be allowed to access? If left unspecified no LCE engined will be granted to this organization.
- **ldap_ids** (*list, optional*) – What ldap server configuration ids should be used with this organization?
- **nessus_managers** (*list, optional*) – Nessus Manager scanner for Nessus Agent scan imports.
- **pub_sites** (*list, optional*) – A list of publishing site ids to associate this organization.
- **repos** (*list, optional*) – A list of Repository ids to associate to this organization.
- **restricted_ips** (*list, optional*) – A list of IP addresses, CIDRs, and/or IP ranges that should never be scanned.
- **vuln_score_low** (*int*) – The vulnerability weighting to apply to low criticality vulnerabilities for scoring purposes. (Default: 1)
- **vuln_score_medium** (*int*) – The vulnerability weighting to apply to medium criticality vulnerabilities for scoring purposes. (Default: 3)
- **vuln_score_high** (*int*) – The vulnerability weighting to apply to high criticality vulnerabilities for scoring purposes. (Default: 10)
- **vuln_score_critical** (*int*) – The vulnerability weighting to apply to critical criticality vulnerabilities for scoring purposes.(Default: 40)
- **zone_selection** (*str*) – What type of scan zone selection should be performed? Available selection types are as follows: `auto_only`, `locked`, `selectable+auto`, `selectable+auto_restricted`. If left unspecified, the default is `auto_only`.
- **zones** (*list, optional*) – When `zone_selection` is not `auto_only`, this field must be filled with list of ids from available scan zone(s).

Returns

The organization resource record for the newly created Org.

Return type

dict

Examples

Creating a new organization with automatic scan zone selection:

```
>>> org = sc.organization.create('Sample Organization')
```

Creating a new organization with custom analysis links:

```
>>> org = sc.organization.create(
...     'Sample Organization',
...     info_links=[
...         ('SANS', 'https://isc.sans.edu/ipinfo.html?ip=%IP%')
... ])
```

delete(*organization_id*)

Remove the specified organization from Tenable Security Center

SC organization Delete

Parameters
organization_id (*int*) – The numeric id of the organization to delete.

Returns
Empty response string

Return type
str

Examples

```
>>> sc.organization.delete(1)
```

details(*organization_id*, *fields=None*)

Retrieves the details for the specified organization.

SC Organization Details

Parameters

- **organization_id** (*int*) – The numeric id of the organization.
- **fields** (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the organization details API doc.

Returns
The organization resource record.

Return type
dict

Examples

```
>>> org = sc.organization.details(1)
```

edit(*organization_id*, ***kwargs*)

Updates an existing organization

SC Organization Edit

Parameters

- **organization_id** – The numeric id of the organization.
- **info_links** (*list, optional*) – A list of custom analysis links provided to users within the host vulnerability details when analyzing data outside of SecurityCenter is desired.
- **lce_ids** (*list, optional*) – What Log Correlation Engines (if any) should this organization be allowed to access? If left unspecified no LCE engined will be granted to this organization.
- **ldap_ids** (*list, optional*) – What ldap server configuration ids should be used with this organization?
- **name** (*str, optional*) – The name for organization.
- **nessus_managers** (*list, optional*) – Nessus Manager scanner for Nessus Agent scan imports.
- **pub_sites** (*list, optional*) – A list of publishing site ids to associate this organization.
- **repos** (*list, optional*) – A list of Repository ids to associate to this organization.
- **restricted_ips** (*list, optional*) – A list of IP addresses, CIDRs, and/or IP ranges that should never be scanned.
- **vuln_score_low** (*int*) – The vulnerability weighting to apply to low criticality vulnerabilities for scoring purposes. (Default: 1)
- **vuln_score_medium** (*int*) – The vulnerability weighting to apply to medium criticality vulnerabilities for scoring purposes. (Default: 3)
- **vuln_score_high** (*int*) – The vulnerability weighting to apply to high criticality vulnerabilities for scoring purposes. (Default: 10)
- **vuln_score_critical** (*int*) – The vulnerability weighting to apply to critical criticality vulnerabilities for scoring purposes.(Default: 40)
- **zone_selection** (*str*) – What type of scan zone selection should be performed? Available selection types are as follows: `auto_only`, `locked`, `selectable+auto`, `selectable+auto_restricted`. If left unspecified, the default is `auto_only`.
- **zones** (*list, optional*) – When `zone_selection` is not `auto_only`, this field must be filled with list of ids from available scan zone(s).

Returns

The updated organization resource record.

Return type

dict

Examples

```
>>> sc.organization.edit(1, name='New Name')
```

list(*fields=None*)

Retrieves a list of organizations.

SC organization List

Parameters

fields (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the organization list API doc.

Returns

List of organization definitions.

Return type

list

Examples

Retrieve all of all of the organizations:

```
>>> repos = sc.organizations.list()
```

manager_create(*org_id*, *username*, *password*, *role*, ***kwargs*)

Creates a new security manager for the given org. For a complete list of parameters that are supported for this call, please refer to `tio.users.create()` for more details.

organization-security-manager: create

Parameters

- **org_id** – (int): The numeric identifier for the organization.
- **username** (*str*) – The username for the account
- **password** (*str*) – The password for the user to create
- **role** (*int*) – The role that should be assigned to this user.
- ****kwargs** (*dict*) – The keyword args to pass to the user constructor.

Returns

The newly created security manager.

Return type

dict

Examples

```
>>> secmngr = sc.organizations.manager_create(1,
...     'username', 'password', 1)
```

manager_delete(*org_id*, *user_id*, *migrate_to=None*)

Removes the user specified.

organization-security-manager: delete

Parameters

- **org_id** – (int): The numeric identifier for the organization.
- **user_id** – (int): The numeric identifier for the user.

Examples

```
>>> sc.organizations.manager_delete(1, 1)
```

manager_details(*org_id*, *user_id*, *fields=None*)

Retrieves the details of a specified security manager within a specified organization.

organization-security-manager: details

Parameters

- **org_id** – (int): The numeric identifier for the organization.
- **user_id** – (int): The numeric identifier for the user.
- **fields** (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the organization list API doc.

Returns
The user resource record.

Return type
dict

Examples

```
>>> secmngr = sc.organizations.manager_details(1, 1)
```

manager_edit(*org_id*, *user_id*, ***kwargs*)

Edits the specified security manager within the specified organization. For details on the supported arguments that may be passed, please refer to `tio.users.edit()` for more details.

organization-security-manager: edit

Parameters

- **org_id** – (int): The numeric identifier for the organization.
- **user_id** – (int): The numeric identifier for the user.
- ****kwargs** (*dict*) – The keyword args to pass to the user constructor.

Returns
The updated user record.

Return type
dict

Examples

```
>>> secmngr = sc.organizations.manager_edit(1, 1,
...     username='updated')
```

managers_list(*org_id*, *fields=None*)

Retrieves a list of security managers.

organization-security-manager: list

Parameters

- **org_id** – (int): The numeric identifier for the organization.
- **fields** (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the organization list API doc.

Returns
List of user definitions.

Return type
list

Examples

Retrieve all of the security managers for a given org.: >>> repos = sc.organizations.managers_list()

recast_risk_rules(*organization_id*, *repos=None*, *plugin=None*, *port=None*)

Retrieves the recasted risk rules for the organization and optionally will filter based on the parameters specified.

organization: recast-risk-rule

Parameters

- **organization_id** (*int*) – The organization id.
- **repos** (*list, optional*) – A list of repository ids to restrict the search to.
- **plugin** (*int, optional*) – A plugin id to restrict the search to.
- **port** (*int, optional*) – A port number to restrict the search to.

Returns
A list of rules that match the request.

Return type
list

Examples

```
>>> for rule in sc.organizations.recast_risk_rules(1):
...     pprint(rule)
```

32.15 Plugins

The following methods allow for interaction with the Tenable Security Center Plugins API. These items are typically seen under the **Plugins** section of Tenable Security Center.

Methods available on `sc.plugins`:

class PluginAPI(*api: APISession*)

details(*plugin_id*, *fields=None*)

Returns the details for a specific plugin.

plugins: details

Parameters

- **plugin_id** (*int*) – The identifier for the plugin.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The plugin resource record.

Return type
dict

Examples

```
>>> plugin = sc.plugins.detail(19506)
>>> pprint(plugin)
```

family_details(*plugin_id*, *fields=None*)

Returns the details for the specified plugin family.

plugin-family: details

Parameters

- **plugin_id** (*int*) – The plugin family numeric identifier.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The plugin family resource.

Return type
dict

Examples

```
>>> family = sc.plugins.family_details(10)
>>> pprint(family)
```

family_list(***kwargs*)

Returns the list of plugin families.

plugin-families: list

Parameters

- **fields** (*list, optional*) – A list of attributes to return.
- **filter** (*tuple, optional*) – A filter tuple for which to filter the plugins. Filter tuples must be `('name', 'operator', 'value')` and follow a similar yet different format to the analysis filters.
- **sort_field** (*str, optional*) – The field to sort the results on.
- **sort_direction** (*str, optional*) – The direction in which to sort the results. Valid settings are `asc` and `desc`. The default is `asc`.
- **type** (*str, optional*) – The type of plugins to return. Available types are `active`, `all`, `compliance`, `custom`, `lce`, `notPassive`, and `passive`. If nothing is specified, then `all` is assumed.

Returns
List of plugin family records.

Return type
list

Examples

```
>>> for fam in sc.plugins.family_list():
...     pprint(fam)
```

family_plugins(*plugin_id*, ***kwargs*)

Retrieves the plugins for the specified family.

plugin-family: plugins

Parameters

- **plugin_id** (*int*) – The numeric identifier for the plugin family.
- **fields** (*list, optional*) – A list of attributes to return.
- **filter** (*tuple, optional*) – A filter tuple for which to filter the plugins. Filter tuples must be `('name', 'operator', 'value')` and follow a similar yet different format to the analysis filters.
- **limit** (*int, optional*) – How many records should be returned in each page of data. If none is specified, the default is 1000 records.
- **offset** (*int, optional*) – At what offset within the data should we start returning data. If none is specified, the default is 0.
- **pages** (*int, optional*) – How many pages of data should we return. If none is specified then all pages will return.

- **sort_field** (*str, optional*) – The field to sort the results on.
- **sort_direction** (*str, optional*) – The direction in which to sort the results. Valid settings are `asc` and `desc`. The default is `asc`.
- **type** (*str, optional*) – The type of plugins to return. Available types are `active`, `all`, `compliance`, `custom`, `lce`, `notPassive`, and `passive`. If nothing is specified, then `all` is assumed.

Returns
an iterator object handling data pagination.

Return type
PluginResultsIterator

Examples

```
>>> plugins = sc.plugins.family_plugins(10)
>>> for plugin in plugins:
...     pprint(plugin)
```

list(***kwargs*)

Retrieves the list of plugins.

plugins: list

Parameters

- **fields** (*list, optional*) – A list of attributes to return.
- **filter** (*tuple, optional*) – A filter tuple for which to filter the plugins. Filter tuples must be `('name', 'operator', 'value')` and follow a similar yet different format to the analysis filters.
- **filters** (*list[tuple], optional*) – A list of filter tuples. Filters are treated as a logical `OR`.
- **limit** (*int, optional*) – How many records should be returned in each page of data. If none is specified, the default is 1000 records.
- **offset** (*int, optional*) – At what offset within the data should we start returning data. If none is specified, the default is 0.
- **pages** (*int, optional*) – How many pages of data should we return. If none is specified then all pages will return.
- **sort_field** (*str, optional*) – The field to sort the results on.
- **sort_direction** (*str, optional*) – The direction in which to sort the results. Valid settings are `asc` and `desc`. The default is `asc`.
- **type** (*str, optional*) – The type of plugins to return. Available types are `active`, `all`, `compliance`, `custom`, `lce`, `notPassive`, and `passive`. If nothing is specified, then `all` is assumed.

Returns
an iterator object handling data pagination.

Return type
PluginResultsIterator

Examples

To retrieve all of the plugins, you'll simply need to call the list method like so:

```
>>> plugins = sc.plugins.list()
>>> for plugin in plugins:
...     pprint(plugin)
```

If you only want the plugins with java in the name, you'd run a query similar to this one:

```
>>> plugins = sc.plugins.list(
...     filter=('name', 'like', 'java'))
```

For just the active plugins, we'd run:

```
>>> plugins = sc.plugins.list(type='active')
```

32.16 Policies

The following methods allow for interaction into the Tenable Security Center Scan Policies API. These items are typically seen under the **Scan Policies** section of Tenable Security Center.

Methods available on `sc.policies`:

class ScanPolicyAPI(*api: APISession*)

copy(*id*, *name=None*)

Clones the specified scan policy

scan-policy: copy

Parameters

- **id** (*int*) – The unique identifier for the source policy to clone.
- **name** (*str, optional*) – The name of the new policy.

Returns

The scan policy resource record for the newly created policy.

Return type

dict

Examples

```
>>> policy = sc.policies.copy(10001)
>>> pprint(policy)
```

create(***kw*)

Creates a new scan policy

scan-policy: create

Parameters

- **name** (*str*) – The Name of the new scan policy

- **audit_files** (*list, optional*) – A list of audit files (by integer id) to be used for the scan policy.
- **description** (*str, optional*) – An optional description for the policy
- **preferences** (*dict, optional*) – A dictionary of settings that override the defaults within a policy template.
- **profile_name** (*str, optional*) – The profile of the scan. Default is an empty string.
- **owner_id** (*int, optional*) – Define who shall own the policy by that user's integer identifier
- **tags** (*str, optional*) – An optional tag identifier for the policy
- **template_id** (*int, optional*) – The identifier of the policy template to use. If none is specified, the default id for the "Advanced Policy" will be used.
- **xccdf** (*bool, optional*) – Should XCCDF results be generated? The default is False.

Returns
The created scan policy resource.

Return type
dict

Examples

An example advanced policy with all of the default preferences.

```
>>> sc.policies.create(
...     name='Example Advanced Policy')
```

An example policy where we want to modify

delete(*id*)

Removes a configured scan policy.

scan-policy: delete

Parameters
id (*int*) – The unique identifier for the policy to remove.

Returns
The empty response from the API.

Return type
str

Examples

```
>>> sc.policies.delete(10001)
```

details(*id*, *fields=None*)

Retrieves the details for a specified policy.

scan-policy: details

Parameters

- **id** (*int*) – The unique identifier for the policy
- **fields** (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the policy details API doc.

Returns

Details about the scan policy template

Return type

dict

Examples

```
>>> policy = sc.policies.details(2)
>>> pprint(policy)
```

edit(*id*, ***kw*)

Edits an existing scan policy

scan-policy: edit

Parameters

- **id** (*int*) – The unique identifier to the scan policy to edit
- **audit_files** (*list, optional*) – A list of audit files (by integer id) to be used for the scan policy.
- **description** (*str, optional*) – An optional description for the policy
- **name** (*str, optional*) – The Name of the new scan policy
- **preferences** (*dict, optional*) – A dictionary of settings that override the defaults within a policy template.
- **profile_name** (*str, optional*) – The profile of the scan. Default is an empty string.
- **remove_prefs** (*list, optional*) – A list of preferences to remove from the policy.
- **owner_id** (*int, optional*) – Define who shall own the policy by that user's integer identifier
- **tags** (*str, optional*) – An optional tag identifier for the policy
- **template_id** (*int, optional*) – The identifier of the policy template to use. If none is specified, the default id for the "Advanced Policy" will be used.
- **xccdf** (*bool, optional*) – Should XCCDF results be generated? The default is False.

Returns
The updated scan policy resource.

Return type
dict

Examples

An example advanced policy with all of the default preferences.

```
>>> sc.policies.edit(10001,
...     name='Updated Example Advanced Policy')
```

To remove a preference, you would perform the following:

```
>>> sc.policies.edit(10001,
...     remove_prefs=['scan_malware'])
```

export_policy(*id*, *fobj=None*)

Export the specified scan policy

scan-policy: export

Parameters

- **id** (*int*) – The unique identifier for the scan policy to export.
- **fobj** (*FileObject, optional*) – The file-like object to write the resulting file into. If no file-like object is provided, a BytesIO objects with the downloaded file will be returned. Be aware that the default option of using a BytesIO object means that the file will be stored in memory, and it's generally recommended to pass an actual file-object to write to instead.

Returns
The file-like object with the resulting export.

Return type
FileObject

Examples

```
>>> with open('example_policy.xml', 'wb') as fobj:
...     sc.policies.export_policy(1001, fobj)
```

import_policy(*name*, *fobj*, *description=None*, *tags=None*)

Imports a scan policy into Tenable Security Center

scan-policy: import

Parameters

- **name** (*str*) – The name of the imported scan policy.
- **fobj** (*FileObject*) – The file-like object containing the scan policy.
- **description** (*str, optional*) – A description for the scan policy.
- **tags** (*str, optional*) – A tag for the scan policy.

Returns
An empty response from the API.

Return type
str

Examples

```
>>> with open('example_policy.xml', 'rb') as fobj:
...     sc.policies.import_policy('Example Policy', fobj)
```

list(*fields=None*)

Retrieved the list of Scan policies configured.

scan-policy: list

Parameters
fields (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the policy list API doc.

Returns
usable & manageable scan policies.

Return type
dict

Examples

```
>>> policies = sc.policies.list()
>>> for policy in policies['manageable']:
...     pprint(policy)
```

share(*id*, **groups*)

Shares the policy with other user groups.

scan-policy: share

Parameters

- **id** (*int*) – The unique identifier for the scan policy to share.
- ***groups** (*int*) – The list of user group ids to share the policy to.

Returns
The updated scan policy resource.

Return type
dict

Examples

Share the scan policy with groups 1, 2, and 3:

```
>>> sc.policies.share(10001, 1, 2, 3)
```

tags()

Returns the list of unique tags associated to scan policies.

scan-policy: tags

Returns
The list of unique tags

Return type
list

Examples

```
>>> tags = sc.policies.tags()
>>> pprint(tags)
```

template_details(*id*, *fields=None*, *remove_editor=True*)

Retrieves the details for a specified policy template.

scan-policy: template-details

Parameters

- **id** (*int*) – The unique identifier for the policy template
- **fields** (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the policy template details API doc.
- **remove_editor** (*bol, optional*) – Should the response have the raw editor string removed? The default is yes.

Returns
Details about the scan policy template

Return type
dict

Examples

```
>>> template = sc.policies.template_details(2)
>>> pprint(template)
```

template_list(*fields=None*)

Retrieved the list of scan policy templates.

scan-policy: template-list

Parameters
fields (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the policy template list API doc.

Returns
List of available policy templates

Return type
list

Examples

```
>>> templates = sc.policies.template_list()
>>> for policy in templates:
...     pprint(policy)
```

32.17 Queries

The following methods allow for interaction into the Tenable Security Center Query API. These items are typically seen under the **Workflow -> Query** section of Tenable Security Center.

Methods available on `sc.queries`:

class QueryAPI(*api: APISession*)

create(*name*, *tool*, *data_type*, **filters*, ***kw*)

Creates a query.

query: create

Parameters

- **name** (*str*) – The name of the new query
- **tool** (*str*) – The tool to use to query the data.
- **data_type** (*str*) – The type of data to query.
- ***filters** (*tuple, optional*) – The filters to use for the query. Refer to the documentation within the :ref:'tenable.sc.analysis' for more information on how to construct these.
- **browse_cols** (*list, optional*) – What columns are set to be browsable for the analysis view.
- **browse_sort_col** (*str, optional*) – The browsable column in which to sort on.
- **browse_sort_dir** (*str, optional*) – The direction in which to sort. Valid values are `asc` and `desc`.
- **description** (*str, optional*) – The description for the query.
- **limit** (*int, optional*) – The limit to the number of records to return. If nothing is specified, the API defaults to 100 records.
- **offset** (*int, optional*) – The number of records to skip before returning results. If nothing is specified, then the default is 0.
- **owner_id** (*int, optional*) – The identifier stating the owner of the query. If left unspecified, then the default is the current user.
- **sort_direction** (*str, optional*) – The direction in which to sort. Valid values are `asc` and `desc`.

- **sort_field** (*str, optional*) – The field in which to sort the results.
- **tags** (*str, optional*) – Tags definition for the query.

Returns
The newly created query.

Return type
dict

Examples

```
>>> query = sc.queries.create('New Query', 'vulndetails', 'vuln',
...     ('pluginID', '=', '19506'))
```

delete(*id*)

Removes a query.

query: delete

Parameters
id (*int*) – The numeric identifier for the query to remove.

Returns
An empty response.

Return type
str

Examples

```
>>> sc.queries.delete(1)
```

details(*id*, *fields=None*)

Returns the details for a specific query.

query: details

Parameters

- **id** (*int*) – The identifier for the query.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The query resource record.

Return type
dict

Examples

```
>>> query = sc.queries.details(1)
>>> pprint(query)
```

edit(*id*, **filters*, ***kw*)

Edits a query.

query: edit

Parameters

- ***filters** (*tuple, optional*) – The filters to use for the query. Refer to the documentation within the :ref:'tenable.sc.analysis' for more information on how to construct these.
- **browse_cols** (*str, optional*) – What columns are set to be browsable for the analysis view.
- **browse_sort_col** (*list, optional*) – The browsable column in which to sort on.
- **browse_sort_dir** (*str, optional*) – The direction in which to sort. Valid values are `asc` and `desc`.
- **description** (*str, optional*) – The description for the query.
- **limit** (*int, optional*) – The limit to the number of records to return. If nothing is specified, the API defaults to 100 records.
- **name** (*str, optional*) – The name of the new query
- **offset** (*int, optional*) – The number of records to skip before returning results. If nothing is specified, then the default is 0.
- **owner_id** (*int, optional*) – The identifier stating the owner of the query. If left unspecified, then the default is the current user.
- **sort_direction** (*str, optional*) – The direction in which to sort. Valid values are `asc` and `desc`.
- **sort_field** (*str, optional*) – The field in which to sort the results.
- **tags** (*str, optional*) – Tags definition for the query.
- **tool** (*str, optional*) – The tool to use to query the data.
- **type** (*str, optional*) – The type of data to query.

Returns

The newly updated query.

Return type

:obj:` dict`

Examples

```
>>> query = sc.queries.edit()
```

list(*fields=None*)

Retrieves the list of query definitions.

query: list

Parameters
fields (*list, optional*) – A list of attributes to return for each query.

Returns
A list of query resources.

Return type
list

Examples

```
>>> for query in sc.queries.list():
...     pprint(query)
```

share(*id*, **groups*)

Shares the specified query to another user group.

query: share

Parameters

- **id** (*int*) – The numeric id for the query.
- ***groups** (*int*) – The numeric id of the group(s) to share to.

Returns
The updated query resource.

Return type
dict

Examples

```
>>> sc.queries.share(1, group_1, group_2)
```

tags()

Retrieves the list of unique tags associated to queries.

query: tags

Returns
List of tags

Return type
list

Examples

```
>>> tags = sc.queries.tags()
```

32.18 Recast Risks

The following methods allow for interaction into the Tenable Security Center Recast Risk API.

Methods available on `sc.recast_risks`:

class RecastRiskAPI(*api: APISession*)

apply(*risk_id*, *repo*)

Applies the recast risk rule for either all repositories, or the repository specified.

recast-risk: apply

Parameters

- **risk_id** (*int*) – The identifier for the recast risk rule.
- **repo** (*int, optional*) – A specific repository to apply the rule to. The default if not specified is all repositories (`0`).

Returns
Empty string response from the API.

Return type
str

Examples

```
>>> sc.recast_risks.apply(1)
```

create(*plugin_id*, *repos*, *severity_id*, ***kwargs*)

Creates a new recast risk rule. Either ips, uuids, asset_list, or host_uuids must be specified, otherwise it will apply to all.

recast-risk: create

Parameters

- **plugin_id** (*int*) – The plugin to apply the recast risk rule to.
- **repos** (*list*) – The list of repositories to apply this recast risk rule to.
- **severity_id** (*int*) – The new severity that vulns matching this rule should be recast to. Valid values are: `0`: Info, `1`: Low, `2`: Medium, `3`: High, and `4`: Critical.
- **asset_list** (*int, optional*) – The asset list id to apply the recast risk rule to. Please note that `asset_list`, `ips`, `uuids`, and `host_uuids` are mutually exclusive.
- **comments** (*str, optional*) – The comment associated to the recast risk rule.
- **expires** (*int, optional*) – Timestamp. When should the rule expire? if no expiration is set, the rule will never expire. If not mentioned, value is -1 (-1 represents December 31st 1969 23:59:59 hours GMT)

- **ips** (*list, optional*) – A list of IPs to apply the recast risk rule to. Please note that `asset_list`, `ips`, `uuids`, and `host_uuids` are mutually exclusive.
- **port** (*int, optional*) – The port to restrict this recast risk rule to. The default is unrestricted.
- **protocol** (*int, optional*) – The protocol to restrict the recast risk rule to. The default is unrestricted.
- **uuids** (*list, optional*) – The agent uuids to apply the recast risk rule to. Please note that `asset_list`, `ips`, `uuids`, and `host_uuids` are mutually exclusive.
- **host_uuids** (*list[str], optional*) – The hostUUIDs to apply the accept risk rule to. Please note that `asset_list`, `ips`, `uuids`, and `host_uuids` are mutually exclusive.

Returns
The newly created recast risk rule definition.

Return type
dict

Examples

Create a rule to recast 97737 on 2 IPs to informational.

```
>>> rule = sc.recast_risks.create(97737, [1], 0
...     ips=['192.168.0.101', '192.168.0.102'])
```

Create a rule to recast 97737 on all IPs on repository 1:

```
>>> rule = sc.recast_risks.create(97737, [1])
```

delete(*risk_id*)

Removes the recast risk rule from Tenable Security Center

recast-risk: delete

Parameters
risk_id (*int*) – The identifier for the recast risk rule.

Returns
Empty string response from the API.

Return type
str

Examples

```
>>> sc.recast_risks.delete(1)
```

details(*risk_id*, *fields=None*)

Retrieves the details of an recast risk rule.

recast-risk: details

Parameters

- **risk_id** (*int*) – The identifier for the recast risk rule.

- **fields** (*list, optional*) – A list of attributes to return for each recast risk rule.

Returns
The recast risk rule details.

Return type
dict

Examples

```
>>> rule = sc.recast_risks.details(1)
>>> pprint(rule)
```

list(*repo_ids=None*, *plugin_id=None*, *port=None*, *org_ids=None*, *fields=None*)
Retrieves the list of recasted risk rules.

recast-risk: list

Parameters

- **fields** (*list, optional*) – A list of attributes to return for each recast risk rule.
- **plugin_id** (*int, optional*) – Plugin id to filter the response on.
- **port** (*int, optional*) – Port number to filter the response on.
- **org_ids** (*list, optional*) – List of organization ids to filter on.
- **repo_ids** (*list, optional*) – List of repository ids to filter the response on.

Returns
A list of recast risk rules.

Return type
list

Examples

```
>>> for rule in sc.recast_risks.list():
...     pprint(rule)
```

32.19 Reports Definition

The following methods allow for interaction into the Tenable.sc ReportsDefinition API.

Methods available on `sc.report_definition`:

class ReportDefinitionAPI(*api: APISession*)

launch(*id*)
Launches a Report definition. report-definition: launch :param id: The report definition identifier to launch. :type id: int

Returns
A report ID resource for the newly launched report definition.

Return type
dict

Examples

```
>>> running = sc.report_definition.launch(1)
>>> print('The Scan Result ID is {}'.format(
...     running['scanResult']['id']))
```

32.20 Repositories

The following methods allow for interaction with the Tenable Security Center Repository API. These items are typically seen under the **Repositories** section of Tenable Security Center.

Methods available on `sc.repositories`:

class RepositoryAPI(*api: APISession*)

accept_risk_rules(*repository_id*, ***kwargs*)

Retrieves the accepted risk rules associated with the specified repository.

repository: accept rules

Parameters

- **repository_id** (*int*) – The numeric id of the repository.
- **fields** (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the repository accept risk rules API doc.

Returns

List of the accepted risk rules that apply to the repo.

Return type

list

Examples

```
>>> rules = sc.repositories.accept_risk_rules(1)
```

asset_intersections(*repository_id*, *uuid=None*, *ip_address=None*, *dns=None*)

Retrieves the asset lists that a UUID, DNS address, or IP exists in.

repository: asst intersections

Parameters

- **repository_id** (*int*) – The numeric identifier of the repository to query.
- **dns** (*str*) – The DNS name to query
- **ip_address** (*str*) – The IP address to query
- **uuid** (*str*) – The UUID to query.

Returns

The list of assets matching the criteria.

Return type

list

Examples

```
>>> assetlists = sc.repositories.asset_intersection(1,
...     ip='192.168.0.1')
```

create(***kwargs*)

Creates a new repository

repository: create

Parameters

- **name** (*str*) – The name for the respository.
- **allowed_ips** (*list, optional*) – Allowed IPs will restrict incoming data being inserted into the repository to only the IPs that exist within the configured CIDR ranges. Accepts a list of CIDR strings based on the repository format (IPv4 or IPv6). If left unspecified, then it will default to the CIDR equivalent of "allow all" for that IP version. IPv4=0.0.0.0/0, IPv6=::/0.
- **description** (*str, optional*) – A description for the repository.
- **format** (*str, optional*) – The format of the repository. Valid choices are `agent`, `IPv4`, `IPv6`, `universal`, and `mobile`. The default if unspecified is `IPv4`.
- **fulltext_search** (*bool, optional*) – Should full-text searching be enabled? This option is used for IPv4, IPv6, universal, and agent repository formats and determins whether the plugin output is trended along with the normalized data. If left unspecified, the default is set to `False`.
- **lce_correlation** (*list, optional*) – What Log Correlation Engines (if any) should correlate against this repository. A list of configured LCE numeric IDs is supplied. This option is used on IPv4, IPv6, and agent formats and is defaulted to nothing if left unspecified.
- **nessus_sched** (*dict, optional*) – This is the .Nessus file generation schedule for IPv4, IPv6, and universal repository formats. This option should only be used if there is a need to consume the Repository in a raw Nessus XML format. If left unspecified, it will default to `{'type': 'never'}`.
- **mobile_sched** (*dict, optional*) – When using the mobile repository format, this option will inform Tenable Security Center how often to perform the MDM synchronization into the repository. If left unspecified, it will default to `{'type': 'never'}`.
- **orgs** (*list, optional*) – A list of Organization IDs used to assign the repository to 1 or many organizations.
- **preferences** (*dict, optional*) – When using a mobile repository type, this dictionary details the required preferences to inject into the backend scan needed to communicate to the MDM solution.
- **remote_ip** (*str, optional*) – When the Remote repository type is used, this is the IP address of the Tenable Security Center instance that the repository will be pulled from.
- **remote_repo** (*int, optional*) – When the Remote repository type is used, this is the numeric ID of the repository on the remote host that will be pulled.

- **remote_sched** (*dict, optional*) – When the Remote repository type is used, this is the schedule dictionary that will inform Tenable Security Center how often to synchronize with the downstream Tenable Security Center instance. If left unspecified then we will default to `{'type': 'never'}`.
- **repo_type** (*str, optional*) – What type of repository is this? Valid choices are `Local`, `Remote`, and `Offline`. The default if unspecified is `Local`.
- **scanner_id** (*int, optional*) – When using the mobile repository format, we must specify the scanner from which to query the MDM source.
- **trending** (*int, optional*) – How many days of trending snapshots should be created for this repository. This value is only used for IPv4, IPv6, universal, and agent repositories. If not supplied, the default will be 0.

Returns
The repository resource record for the newly created Repo.

Return type
dict

Examples

Creating a new IPv4 Repository leveraging the defaults:

```
>>> repo = sc.repositories.create(name='Example IPv4')
```

Creating a new IPv4 Repository with 90 days of trending and linked to the first Organization:

```
>>> repo = sc.repositories.create(
...     name='Example Trending', trending=90, orgs=[1])
```

Creating an IPv6 repository:

```
>>> repo = sc.repositories.create(
...     name='Example IPv6', format='IPv6')
```

Creating an agent repository:

```
>>> repo = sc.repositories.create(
...     name='Example Agent', format='agent')
```

Creating an MDM repository for ActiveSync that will sync every day at 6am eastern:

```
>>> repo = sc.repositories.create(
...     name='Example ActiveSync', mdm_id=1, scanner_id=1,
...     format='mobile', orgs=[1],
...     mobile_sched={
...         'repeatRule': 'FREQ=DAILY;INTERVAL=1',
...         'start': 'TZID=America/New_York:20190212T060000',
...         'type': 'ical',
...     },
...     preferences={
...         'domain': 'AD_DOMAIN',
...         'domain_admin': 'DA_ACCOUNT_NAME',
...         'domain_controller': 'dc1.company.tld',
```

(continues on next page)

(continued from previous page)

```
...         'password': 'DA_ACCOUNT_PASSWORD'
... })
```

Creating a new repository to remotely sync the downstream Tenable Security Center instance's repository 1 to this host and institute trending for 90 days:

```
>>> repo = sc.repositories.create(
...     name='Example Remote Repo',
...     repo_type='Remote',
...     remote_ip='192.168.0.101',
...     remote_repo=1,
...     trending=90,
...     orgs=[1],
...     remote_sched={
...         'type': 'ical',
...         'start': 'TZID=America/NewYork:20190212T060000',
...         'repeatRule': 'FREQ=DAILY;INTERVAL=1'
... })
```

delete(*repository_id*)

Remove the specified repository from Tenable Security Center

repository: delete

Parameters
repository_id (*int*) – The numeric id of the repository to delete.

Returns
Empty response string

Return type
str

Examples

```
>>> sc.repositories.delete(1)
```

details(*repository_id*, *fields=None*)

Retrieves the details for the specified repository.

repository: details

Parameters

- **repository_id** (*int*) – The numeric id of the repository.
- **fields** (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the repository details API doc.

Returns
The repository resource record.

Return type
dict

Examples

```
>>> repo = sc.repositories.details(1)
```

device_info(*repository_id*, *dns=None*, *ip_address=None*, *uuid=None*, *fields=None*)

Retrieves the device information for the requested device on the associated repository.

repository: device info

repository: ip info

Parameters

- **repository_id** (*int*) – The numeric id for the repository to query.
- **dns** (*str*) – The DNS name to query
- **fields** (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the repository device info API doc.
- **ip_address** (*str*) – The IP address to query
- **uuid** (*str*) – The UUID to query.

Returns
The device resource.

Return type
dict

Examples

```
>>> host = sc.repositories.device_info(1, ip_address='192.168.0.1')
```

edit(*repository_id*, ***kwargs*)

Updates an existing repository

repository: edit

Parameters

- **repository_id** (*int*) – The numeric id of the repository to edit.
- **allowed_ips** (*list, optional*) – Allowed IPs will restrict incoming data being inserted into the repository to only the IPs that exist within the configured CIDR ranges. Accepts a list of CIDR strings based on the repository format (IPv4 or IPv6).
- **description** (*str, optional*) – A description for the repository.
- **lce_correlation** (*list, optional*) – What Log Correlation Engines (if any) should correlate against this repository. A list of configured LCE numeric IDs is supplied. This option is used on IPv4, IPv6, and agent formats.
- **name** (*str, optional*) – The name for the repository.
- **nessus_sched** (*dict, optional*) – This is the .Nessus file generation schedule for IPv4 and IPv6 repository formats. This option should only be used if there is a need to consume the Repository in a raw Nessus XML format.

- **mobile_sched** (*dict, optional*) – When using the mobile repository format, this option will inform Tenable Security Center how often to perform the MDM synchronization into the repository.
- **orgs** (*list, optional*) – A list of Organization IDs used to assign the repository to 1 or many organizations.
- **preferences** (*dict, optional*) – When using a mobile repository type, this dictionary details the required preferences to inject into the backend scan needed to communicate to the MDM solution.
- **remote_ip** (*str, optional*) – When the Remote repository type is used, this is the IP address of the Tenable Security Center instance that the repository will be pulled from.
- **remote_repo** (*int, optional*) – When the Remote repository type is used, this is the numeric ID of the repository on the remote host that will be pulled.
- **remote_sched** (*dict, optional*) – When the Remote repository type is used, this is the schedule dictionary that will inform Tenable Security Center how often to synchronize with the downstream Tenable Security Center instance.
- **scanner_id** (*int, optional*) – When using the mobile repository format, we must specify the scanner from which to query the MDM source.
- **trending** (*int, optional*) – How many days of trending snapshots should be created for this repository. This value is only used for IPv4, IPv6, and agent repositories.

Returns
The repository resource record for the newly created Repo.

Return type
dict

Examples

```
>>> repo = sc.repositories.edit(1, name='Example IPv4')
```

export_repository(*repository_id*, *fobj*)

Exports the repository and writes the archive tarball into the file object passed.

repository: export

Parameters

- **repository_id** (*int*) – The numeric id associated to the repository.
- **fobj** (*FileObject*) – The file-like object for the repository archive.

Returns
The export response record.

Return type
dict

Example

```
>>> with open('repo.tar.gz', 'wb') as archive:
...     sc.repositories.export_repository(1, archive)
```

import_repository(*repository_id*, *fobj*)

Imports the repository archive for an offline repository.

repository: import

Parameters

- **repository_id** (*int*) – The numeric id associated to the offline repository.
- **fobj** (*FileObject*) – The file-like object containing the repository archive.

Returns
The import response record.

Return type
dict

Example

```
>>> with open('repo.tar.gz', 'rb') as archive:
...     sc.repositories.import_repository(1, archive)
```

list(*fields=None*, *repo_type=None*)

Retrieves a list of repositories.

repository: list

Parameters

- **fields** (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the repository list API doc.
- **repo_type** (*str, optional*) – Restrict the response to a specific type of repository. If not set, then all repository types will be returned. Allowed types are `All`, `Local`, `Remote`, and `Offline`.

Returns
List of repository definitions.

Return type
list

Examples

Retrieve all of all of the repositories:

```
>>> repos = sc.repositories.list()
```

Retrieve all of the remote repositories:

```
>>> repos = sc.repositories.list(repo_type='Remote')
```

mobile_sync(*repository_id*)

Initiates a MDM synchronization with the configured MDM source on the mobile repository specified.

repository: update mobile data

Parameters
repository_id (*int*) – The numeric id for the mobile repository to run the sync.

Returns
The sync response record.

Return type
dict

Examples

```
>>> sc.repositories.mobile_sync(1)
```

recast_risk_rules(*repository_id*, ***kwargs*)

Retrieves the recast risk rules associated with the specified repository.

repository: recast rules

Parameters

- **repository_id** (*int*) – The numeric id of the repository.
- **fields** (*list, optional*) – The list of fields that are desired to be returned. For details on what fields are available, please refer to the details on the request within the repository recast risk rules API doc.

Returns
List of the recast risk rules that apply to the repo.

Return type
list

Examples

```
>>> rules = sc.repositories.recast_risk_rules(1)
```

remote_authorize(*host*, *username*, *password*)

Authorized communication to a downstream Tenable Security Center instance with the provided username and password.

repository: authorize

Parameters

- **host** (*str*) – The downstream Tenable Security Center instance ip address.
- **username** (*str*) – The username to authenticate with.
- **password** (*str*)

Returns
Empty response object

Return type
str

Examples

```
>>> sc.repositories.remote_authorize(
...     '192.168.0.101', 'admin', 'password')
```

remote_fetch(*host*)

Retrieves the list of repositories from the specified downstream Tenable Security Center instance.

repository: fetch remote

Parameters
host (*str*) – The downstream Tenable Security Center instance ip address.

Returns
The list of repositories on the downstream Tenable Security Center instance.

Return type
list

remote_sync(*repository_id*)

Initiates a remote synchronization with a downstream Tenable Security Center instance. This action can only be performed on an offline repository.

repository: sync

Parameters
repository_id (*int*) – The numeric id for the remote repository.

Returns
The sync response record.

Return type
dict

Examples

```
>>> sc.repositories.remote_sync(1)
```

32.21 Roles

The following methods allow for interaction into the Tenable Security Center Roles API. These items are typically seen under the **User Roles** section of Tenable Security Center.

Methods available on `sc.roles`:

class RoleAPI(*api: APISession*)

create(*name*, ***kw*)

Creates a role.

role: create

Parameters

- **name** (*str*) – The name of the new role to create.
- **descrioption** (*str, optional*) – A description for the role to be created.
- **can_agent_scan** (*bool, optional*) – Are members of this role allowed to perform agent scans? If left unspecified the default is `False`.
- **can_feed_update** (*bool, optional*) – Are members of this role allowed to perform feed updates? If left unspecified, the default is `False`.
- **can_import_scan** (*bool, optional*) – Are members of this role allowed to import scans? If left unspecified, the default is `False`.
- **can_scan** (*str, optional*) – Are members of this role allowed to perform scans? Accepted values are *full*, *policy*, and *none*. If left unspecified, the default is *none*.
- **can_share** (*bool, optional*) – Are members of this role allowed to share objects with other groups? If left unspecified, the default is `False`.
- **can_view_logs** (*bool, optional*) – Are members of this role allowed to view the organizational logs from Tenable Security Center? If left unspecified, the default is `False`.
- **create_alerts** (*bool, optional*) – Are members of this role allowed to create alerts? If left unspecified, the default is `False`.
- **create_auditfiles** (*bool, optional*) – Are members of this role allowed to create their own audit files? If left unspecified, the default is `False`.
- **create_ldap_assets** (*bool, optional*) – Are members of this role allowed to create LDAP Query Asset Lists? If left unspecified, the default is `False`.
- **create_policies** (*bool, optional*) – Are members of this role allowed to create scan policies? If left unspecified, the default is `False`.
- **create_tickets** (*bool, optional*) – Are members of this role allowed to create tickets? If left unspecified, the default is `False`.
- **manage_accepted_risk_rules** (*bool, optional*) – Are members of this role allowed to manage accepted risk rules? If left unspecified, the default is `False`.
- **manage_attributes** (*bool, optional*) – Are members of this role allowed to manage attribute sets? If left unspecified, the default is `False`.
- **manage_blackout_windows** (*bool, optional*) – Are members of this role allowed to manage scanning blackout windows? If left unspecified, the default is `False`.

- **manage_groups** (*bool, optional*) – Are members of this role allowed to manage user groups? If left unspecified, the default is `False`.
- **manage_images** (*bool, optional*) – Are members of this role allowed to manage report images? If left unspecified, the default is `False`.
- **manage_recast_risk_rules** (*bool, optional*) – Are members of this role allowed to manage recast risk rules? If left unspecified, the default is `False`.
- **manage_relationships** (*bool, optional*) – Are members of this role allowed to manage the user group relationships? If left unspecified, the default is `False`.
- **manage_roles** (*bool, optional*) – Are members of this role allowed to manage group role configurations? If left unspecified, the default is `False`.

Returns
The newly created role.

Return type
dict

Examples

```
>>> role = sc.roles.create('Example Role',
...     can_scan=True, can_import_scan=True)
```

delete(*id*)

Removes a role.

role: delete

Parameters
id (*int*) – The numeric identifier for the role to remove.

Returns
An empty response.

Return type
str

Examples

```
>>> sc.roles.delete(1)
```

details(*id*, *fields=None*)

Returns the details for a specific role.

role: details

Parameters

- **id** (*int*) – The identifier for the role.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The role resource record.

Return type
dict

Examples

```
>>> role = sc.roles.details(1)
>>> pprint(role)
```

edit(*id*, ***kw*)

Edits a role.

role: edit

Parameters

- **id** (*int*) – The numeric identifier for the role.
- **name** (*str, optional*) – The name of the new role to create.
- **description** (*str, optional*) – A description for the role to be created.
- **can_agent_scan** (*bool, optional*) – Are members of this role allowed to perform agent scans? If left unspecified the default is `False`.
- **can_feed_update** (*bool, optional*) – Are members of this role allowed to perform feed updates? If left unspecified, the default is `False`.
- **can_import_scan** (*bool, optional*) – Are members of this role allowed to import scans? If left unspecified, the default is `False`.
- **can_scan** (*bool, optional*) – Are members of this role allowed to perform scans? If left unspecified, the default is `False`.
- **can_share** (*bool, optional*) – Are members of this role allowed to share objects with other groups? If left unspecified, the default is `False`.
- **can_view_logs** (*bool, optional*) – Are members of this role allowed to view the organizational logs from Tenable Security Center? If left unspecified, the default is `False`.
- **create_alerts** (*bool, optional*) – Are members of this role allowed to create alerts? If left unspecified, the default is `False`.
- **create_auditfiles** (*bool, optional*) – Are members of this role allowed to create their own audit files? If left unspecified, the default is `False`.
- **create_ldap_assets** (*bool, optional*) – Are members of this role allowed to create LDAP Query Asset Lists? If left unspecified, the default is `False`.
- **create_policies** (*bool, optional*) – Are members of this role allowed to create scan policies? If left unspecified, the default is `False`.
- **create_tickets** (*bool, optional*) – Are members of this role allowed to create tickets? If left unspecified, the default is `False`.
- **manage_accepted_risk_rules** (*bool, optional*) – Are members of this role allowed to manage accepted risk rules? If left unspecified, the default is `False`.
- **manage_attributes** (*bool, optional*) – Are members of this role allowed to manage attribute sets? If left unspecified, the default is `False`.
- **manage_blackout_windows** (*bool, optional*) – Are members of this role allowed to manage scanning blackout windows? If left unspecified, the default is `False`.
- **manage_groups** (*bool, optional*) – Are members of this role allowed to manage user groups? If left unspecified, the default is `False`.

- **manage_images** (*bool, optional*) – Are members of this role allowed to manage report images? If left unspecified, the default is `False`.
- **manage_recast_risk_rules** (*bool, optional*) – Are members of this role allowed to manage recast risk rules? If left unspecified, the default is `False`.
- **manage_relationships** (*bool, optional*) – Are members of this role allowed to manage the user group relationships? If left unspecified, the default is `False`.
- **manage_roles** (*bool, optional*) – Are members of this role allowed to manage group role configurations? If left unspecified, the default is `False`.

Returns
The newly updated role.

Return type
dict

Examples

```
>>> role = sc.roles.create()
```

list(*fields=None*)

Retrieves the list of role definitions.

role: list

Parameters
fields (*list, optional*) – A list of attributes to return for each role.

Returns
A list of role resources.

Return type
list

Examples

```
>>> for role in sc.roles.list():
...     pprint(role)
```

32.22 Scan Instances

The following methods allow for interaction into the Tenable Security Center Scan Result API. While the Tenable Security Center API refers to the model these endpoints interact with as *ScanResult*, were actually interacting with an instance of a scan definition stored within the *Scan* API endpoints. These scan instances could be running scans, stopped scans, errored scans, or completed scans. These items are typically seen under the **Scan Results** section of Tenable Security Center.

Methods available on `sc.scan_instances`:

class ScanResultAPI(*api: APISession*)

copy(*id*, **users*)

Clones the scan instance.

scan-result: copy

Parameters

- **id** (*int*) – The identifier of the scan instance to clone.
- ***users** (*int*) – A user id to associate to the scan instance.

Returns
The cloned scan instance record.

Return type
dict

Examples

```
>>> sc.scan_instances.copy(1)
```

delete(*id*)

Removes the scan instance from TenableSC.

scan-result: delete

Parameters
id (*int*) – The identifier of the scan instance to delete.

Returns
An empty string.

Return type
str

Examples

```
>>> sc.scan_instances.delete(1)
```

details(*id*, *fields=None*)

Retrieves the details for the specified scan instance.

scan-result: details

Parameters

- **id** (*int*) – The identifier for the scan instance to be retrieved.
- **fields** (*list, optional*) – List of fields to return. Refer to the API documentation referenced above for a list of available fields.

Returns
The scan instance resource record.

Return type
dict

Examples

Getting the details of a scan instance with just the default parameters:

```
>>> scan = sc.scan_instances.details(1)
>>> pprint(scan)
```

Specifying what fields you'd like to be returned:

```
>>> scan = sc.scan_instances.details(1,
...     fields=['name', 'status', 'scannedIPs', 'startTime', 'finishTime'])
>>> pprint(scan)
```

email(*id*, **emails*)

Emails the scan results of the requested scan to the email addresses defined.

scan-result: email

Parameters

- **id** (*int*) – The identifier for the specified scan instance.
- ***emails** (*str*) – Valid email address.

Returns
Empty string response.

Return type
str

Examples

```
>>> sc.scan_instances.email(1, 'email@company.tld')
```

export_scan(*id*, *fobj=None*, *export_format=None*)

Downloads the results of the scan.

scan-result: download

Parameters

- **id** (*int*) – The scan instance identifier.
- **export_format** (*str, optional*) – The format of the resulting data. Allowable values are `scap1_2` and `v2`. `v2` is the default value if none are specified.
- **fobj** (*FileObject, optional*) – The file-like object to write the resulting file into. If no file-like object is provided, a BytesIO objects with the downloaded file will be returned. Be aware that the default option of using a BytesIO object means that the file will be stored in memory, and it's generally recommended to pass an actual file-object to write to instead.

Returns
The file-like object with the resulting zipped report.

Return type
FileObject

Examples

```
>>> with open('example.zip', 'wb') as fobj:
...     sc.scan_instances.export_scan(1, fobj)
```

import_scan(*fobj*, *repo*, ***kw*)

Imports a nessus file into Tenable Security Center.

scan-result: import

Parameters

- **fobj** (*FileObject*) – The file-like object containing the Nessus file to import.
- **repo** (*int*) – The repository id for the scan.
- **auto_mitigation** (*int, optional*) – How many days to hold on to data before mitigating it? The default value is 0.
- **host_tracking** (*bool, optional*) – Should DHCP host tracking be enabled? The default is False.
- **vhosts** (*bool, optional*) – Should virtual host logic be enabled for the scan? The default is `False`.

Returns
An empty string response.

Return type
str

Examples

```
>>> with open('example.nessus', 'rb') as fobj:
...     sc.scan_instances.import_scan(fobj, 1)
```

list(*fields=None*, *start_time=None*, *end_time=None*, *optimize=True*)

Retrieves the list of scan instances.

scan-result: list

Parameters

- **fields** (*list, optional*) – A list of attributes to return.
- **start_time** (*int, optional*) – Epoch time to start search (searches against createdTime and defaults to now-30d)
- **end_time** (*int, optional*) – Epoch time to end search (searches against createdTime and defaults to now)
- **optimize** (*bool, optional*) – Informs Tenable Security Center to optimize completed scan results. If left unspecified, the default is *True*.

Returns
A list of scan instance resources.

Return type
dict

Examples

- Retrieving all of the manageable scans instances:

```
>>> for scan in sc.scan_instances.list()['manageable']:
...     pprint(scan)
```

pause(*id*)

Pauses a running scan instance. Note that this will not impact agent scan instances.

scan-result: pause

Parameters
id (*int*) – The unique identifier for the scan instance.

Returns
The Scan instance state

Return type
dict

Examples

```
>>> sc.scan_instances.pause(1)
```

reimport_scan(*id*, ***kw*)

Re-imports an existing scan into the cumulative repository.

scan-result: re-import

Parameters

- **id** (*int*) – The scan instance identifier.
- **auto_mitigation** (*int, optional*) – How many days to hold on to data before mitigating it? The default value is 0.
- **host_tracking** (*bool, optional*) – Should DHCP host tracking be enabled? The default is False.
- **vhosts** (*bool, optional*) – Should virtual host logic be enabled for the scan? The default is False.

Returns
An empty string response.

Return type
str

Examples

```
>>> sc.scan_instances.reimport_scan(1)
```

resume(*id*)

Resumes a paused scan instance. Note that this will not impact agent scan instances.

scan-result: resume

Parameters
id (*int*) – The unique identifier for the scan instance.

Returns
The Scan instance state

Return type
dict

Examples

```
>>> sc.scan_instances.resume(1)
```

stop(*id*)

Stops a running scan instance. Note that this will not impact agent scan instances.

scan-result: stop

Parameters
id (*int*) – The unique identifier for the scan instance.

Returns
The Scan instance state

Return type
dict

Examples

```
>>> sc.scan_instances.stop(1)
```

32.23 Scan Zones

The following methods allow for interaction into the Tenable Security Center Scan Zone API. These items are typically seen under the **Scan Zones** section of Tenable Security Center.

Methods available on `sc.scan_zones`:

class ScanZoneAPI(*api: APISession*)

create(*name*, ***kw*)

Creates a scan zone.

scan-zone: create

Parameters

- **name** (*str*) – The name of the scan zone
- **description** (*str, optional*) – A description for the scan zone.
- **ips** (*list, optional*) – The list of IP addresses, CIDRs, or IP ranges that encompass the scan zone.
- **scanner_ids** (*list, optional*) – A list of scanner ids to associate to the scan zone.

Returns
The newly created scan zone.

Return type
dict

Examples

```
>>> zone = sc.scan_zones.create('Example Scan Zone',
...     ips=['127.0.0.1'], scanner_ids=[1])
```

delete(*id*)

Removes the specified scan zone.

scan-zone: delete

Parameters
id (*int*) – The numeric identifier for the scan-zone to remove.

Returns
An empty response.

Return type
str

Examples

```
>>> sc.scan_zones.delete(1)
```

details(*id*, *fields=None*)

Returns the details for a specific scan zone.

scan-zone: details

Parameters

- **id** (*int*) – The identifier for the scan.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The scan zone resource record.

Return type
dict

Examples

```
>>> zone = sc.scan_zones.details(1)
>>> pprint(zone)
```

edit(*id*, ***kw*)

Edits a scan zone.

scan-zone: edit

Parameters

- **description** (*str, optional*) – A description for the scan zone.
- **ips** (*list, optional*) – The list of IP addresses, CIDRs, or IP ranges that encompass the scan zone.
- **name** (*str, optional*) – The name of the scan zone
- **scanner_ids** (*list, optional*) – A list of scanner ids to associate to the scan zone.

Returns
The newly updated scan zone.

Return type
dict

Examples

```
>>> zone = sc.scan_zones.create(1,
...     ips=['127.0.0.1'], scanner_ids=[1])
```

list(*fields=None*)

Retrieves the list of scan zone definitions.

scan-zone: list

Parameters
fields (*list, optional*) – A list of attributes to return for each scan.

Returns
A list of scan zone resources.

Return type
list

Examples

```
>>> for zone in sc.scan_zones.list():
...     pprint(zone)
```

32.24 Scanners

The following methods allow for interaction into the Tenable Security Center Scanner API. These items are typically seen under the **Scanners** section of Tenable Security Center.

Methods available on `sc.scanners`:

class ScannerAPI(*api: APISession*)

agent_scans(*id*, *search*, *results=None*)

Retrieves the list of agent scans that meed the specified search criteria.

scanner: test-scans-query

Parameters

- **id** (*int*) – The numeric id of the scanner.
- **search** (*str*) – The search string to send to the scanner.
- **results** (*list, optonal*) – The list of results ids to test.

Returns
The list of scans that match the search criteria.

Return type
list

Examples

```
>>> scans = sc.scanners.agent_scans('*')
```

create(*name*, *address*, ***kw*)

Creates a scanner.

scanner: create

Parameters

- **address** (*str*) – The address of the scanner
- **name** (*str*) – The name of the scanner
- **agent_capable** (*bool, optional*) – Is this scanner an agent capable scanner? If left unspecified the default is `False`.
- **description** (*str, optional*) – The description of the scanner.
- **enabled** (*bool, optional*) – Is this scanner enabled? If left unspecified, the default is `True`.
- **managed** (*bool, optional*) – Is the plugin set for this scanner managed? If left unspecified then the default is `False`.
- **orgs** (*list, optional*) – If the scanner is an agent capable scanner, then a list of organization ids is to be specified to attach the scanner for the purposes of agent scanning.
- **port** (*int, optional*) – What is the port that the Nessus service is running on. If left unspecified, then the default is `8834`.

- **proxy** (*bool, optional*) – Is this scanner behind a proxy? If left unspecified then the default is `False`.
- **zone_ids** (*list, optional*) – List of scan zones that this scanner is to be a member of.

Returns
The newly created scanner.

Return type
dict

Examples

```
>>> scanner = sc.scanners.create('Example Scanner', '192.168.0.1')
```

delete(*id*)

Removes the specified scanner.

scanner: delete

Parameters
id (*int*) – The numeric identifier for the scanner to remove.

Returns
An empty response.

Return type
str

Examples

```
>>> sc.scanners.delete(1)
```

details(*id*, *fields=None*)

Returns the details for a specific scanner.

scanner: details

Parameters

- **id** (*int*) – The identifier for the scanner.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The scanner resource record.

Return type
dict

Examples

```
>>> scanner = sc.scanners.details(1)
>>> pprint(scanner)
```

edit(*id*, ***kw*)

Edits a scanner.

scanner: edit

Parameters

- **id** (*int*) – The numeric identifier for the scanner.
- **address** (*str, optional*) – The address of the scanner
- **agent_capable** (*bool, optional*) – Is this scanner an agent capable scanner? If left unspecified the default is False.
- **description** (*str, optional*) – The description of the scanner.
- **enabled** (*bool, optional*) – Is this scanner enabled? If left unspecified, the default is True.
- **managed** (*bool, optional*) – Is the plugin set for this scanner managed? If left unspecified then the default is False.
- **name** (*str, optional*) – The name of the scanner
- **orgs** (*list, optional*) – If the scanner is an agent capable scanner, then a list of organization ids is to be specified to attach the scanner for the purposes of agent scanning.
- **port** (*int, optional*) – What is the port that the Nessus service is running on. If left unspecified, then the default is 8834.
- **proxy** (*bool, optional*) – Is this scanner behind a proxy? If left unspecified then the default is False.
- **zone_ids** (*list, optional*) – List of scan zones that this scanner is to be a member of.

Returns
The newly updated scanner.

Return type
dict

Examples

```
>>> scanner = sc.scanners.edit(1, enabled=True)
```

list(*fields=None*)

Retrieves the list of scanner definitions.

scanner: list

Parameters
fields (*list, optional*) – A list of attributes to return for each scanner.

Returns
A list of scanner resources.

Return type
list

Examples

```
>>> for scanner in sc.scanners.list():
...     pprint(scanner)
```

update_status()

Starts an on-demand scanner status update.

scanner: update-status

Returns
The updated scanner status for all scanners.

Return type
list

Examples

```
>>> status = sc.scanners.update_status()
```

32.25 Scans

The following methods allow for interaction into the Tenable Security Center Scan API. While the api endpoints obliquely refers to the model in which this collection of actions modifies as "Scans", Tenable Security Center is actually referring to the scan *definitions*, which are the un-launched and/or scheduled scans typically seen within the **Active Scans** section within Tenable Security Center.

Methods available on `sc.scans`:

class ScanAPI(*api: APISession*)

copy(*id*, *name*, *user_id*)

Copies an existing scan definition.

scan: copy

Parameters

- **id** (*int*) – The scan definition identifier to copy.
- **name** (*str*) – The name of the copy that's created.
- **user_id** (*int*) – The user id to assign as the owner of the new scan definition.

Returns
Scan definition resource.

Return type
dict

Examples

```
>>> sc.scans.copy(1, name='Cloned Scan')
```

create(*name*, *repo*, ***kw*)

Creates a scan definition.

scan: create

Parameters

- **name** (*str*) – The name of the scan.
- **repo** (*int*) – The repository id for the scan.
- **auto_mitigation** (*int, optional*) – How many days to hold on to data before mitigating it? The default value is 0.
- **asset_lists** (*list, optional*) – A list of asset list ids to run the scan against. A logical OR will be performed to compute what hosts to scan against.
- **creds** (*list, optional*) – A list of credential ids to use for the purposes of this scan. This list should be treated as an un-ordered list of credentials.
- **description** (*str, optional*) – A description for the scan.
- **email_complete** (*bool, optional*) – Should we notify the owner upon completion of the scan? The default is `False`.
- **email_launch** (*bool, optional*) – Should we notify the owner upon launching the scan? The default is `False`.
- **host_tracking** (*bool, optional*) – Should DHCP host tracking be enabled? The default is False.
- **max_time** (*int, optional*) – The maximum amount of time that the scan may run in hours. `0` or less for unlimited.
- **policy_id** (*int, optional*) – The policy id to use for a policy-based scan.
- **reports** (*list, optional*) – What reports should be run upon completion of the scan? Each report dictionary requires an id for the report definition and the source for which to run the report against. Example: `{'id': 1, 'reportSource': 'individual'}`.
- **rollover** (*str, optional*) – How should rollover scans be created (assuming the scan is configured to create a rollover scan with the timeout action). The available actions are to automatically start the `nextDay` at the same time the scan was originally configured to run, and to generate a rollover `template`. The default action is to generate a `template`.
- **scan_zone** (*int, optional*) – The zone identifier to use for the scan. If non is selected then the default of "0" or "All Zones" is selected.
- **schedule** (*dict, optional*) – A dictionary detailing the repeating schedule of the scan.
- **targets** (*list, optional*) – A list of valid targets. These targets could be IPs, FQDNs, CIDRs, or IP ranges.
- **timeout** (*str, optional*) – How should an incomplete scan be handled? The available actions are `discard`, `import`, and `rollover`. The default action is `import`.

- **vhosts** (*bool, optional*) – Should virtual host logic be enabled for the scan? The default is `False`.
- **inactivity_timeout** (*int*) – Inactivity Timeout in seconds. The value should be between 3600 and 432000.

Returns
The scan resource for the created scan.

Return type
dict

Examples

Creating a scan for a single host:

```
>>> sc.scans.create('Example scan', 1, policy_id=1001,
...     targets=['127.0.0.1'])
```

delete(*id*)

Removes the specified scan from SecurityCenter.

scan: delete

Parameters
id (*int*) – The identifier for the scan to delete.

Returns
The list of scan id removed.

Return type
list

Examples

```
>>> sc.scans.delete(1)
```

details(*id*, *fields=None*)

Returns the details for a specific scan.

scan: details

Parameters

- **id** (*int*) – The identifier for the scan.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The scan resource record.

Return type
dict

Examples

```
>>> scan = sc.scans.detail(1)
>>> pprint(scan)
```

edit(*id*, ***kw*)

Edits an existing scan definition.

scan: update

Parameters

- **id** (*int*) – The identifier for the scan.
- **auto_mitigation** (*int, optional*) – How many days to hold on to data before mitigating it?
- **asset_lists** (*list, optional*) – A list of asset list ids to run the scan against. A logical OR will be performed to compute what hosts to scan against.
- **creds** (*list, optional*) – A list of credential ids to use for the purposes of this scan. This list should be treated as an un-ordered list of credentials.
- **description** (*str, optional*) – A description for the scan.
- **email_complete** (*bool, optional*) – Should we notify the owner upon completion of the scan?
- **email_launch** (*bool, optional*) – Should we notify the owner upon launching the scan?
- **host_tracking** (*bool, optional*) – Should DHCP host tracking be enabled?
- **max_time** (*int, optional*) – The maximum amount of time that the scan may run in hours. `0` or less for unlimited.
- **name** (*str, optional*) – The name of the scan.
- **policy_id** (*int, optional*) – The policy id to use for a policy-based scan.
- **plugin_id** (*int, optional*) – The plugin id to use for a plugin-based scan.
- **reports** (*list, optional*) – What reports should be run upon completion of the scan? Each report dictionary requires an id for the report definition and the source for which to run the report against. Example: `{'id': 1, 'reportSource': 'individual'}`.
- **repo** (*int, optional*) – The repository id for the scan.
- **rollover** (*str, optional*) – How should rollover scans be created (assuming the scan is configured to create a rollover scan with the timeout action). The available actions are to automatically start the `nextDay` at the same time the scan was originally configured to run, and to generate a rollover `template`.
- **scan_zone** (*int, optional*) – The zone identifier to use for the scan.
- **schedule** (*dict, optional*) – A dictionary detailing the repeating schedule of the scan.
- **targets** (*list, optional*) – A list of valid targets. These targets could be IPs, FQDNs, CIDRs, or IP ranges.
- **timeout** (*str, optional*) – How should an incomplete scan be handled? The available actions are `discard`, `import`, and `rollover`.

- **vhosts** (*bool, optional*) – Should virtual host logic be enabled for the scan?

Returns
The scan resource for the created scan.

Return type
dict

Examples

Editing an existing scan's name:

```
>>> sc.scans.edit(1, name='Example scan')
```

launch(*id*, *diagnostic_target=None*, *diagnostic_password=None*)

Launches a scan definition.

scan: launch

Parameters

- **id** (*int*) – The scan definition identifier to launch.
- **diagnostic_target** (*str, optional*) – A valid IP or hostname to launch a diagnostic scan against. The `diagnostic_password` must also be specified or else this parameter will be ignored.
- **diagnostic_password** (*str, optional*) – A password to use for the diagnostic scan. The `diagnostic_target` must also be specified or else this parameter will be ignored.

Returns
A scan result resource for the newly launched scan.

Return type
dict

Examples

```
>>> running = sc.scans.launch(1)
>>> print('The Scan Result ID is {}'.format(
...     running['scanResult']['id']))
```

list(*fields=None*)

Retrieves the list of scan definitions.

scan: list

Parameters
fields (*list, optional*) – A list of attributes to return for each scan.

Returns
A list of scan resources.

Return type
list

Examples

```
>>> for scan in sc.scans.list():
...     pprint(scan)
```

32.26 Status

The following methods allow for interaction into the Tenable Security Center Status API. These API calls are typically used to understand the current job and license statuses.

Methods available on `sc.status`:

class StatusAPI(*api: APISession*)

status()

Retrieves license & status information about the Tenable Security Center instance.

status: status

Returns
The status dictionary.

Return type
dict

Examples

```
>>> status = sc.status.status()
```

32.27 System

The following methods allow for interaction into the Tenable Security Center System API. These API calls are typically used to understand timezones, system version, etc.

Methods available on `sc.system`:

class SystemAPI(*api: APISession*)

current_locale()

Retrieves the current system locale that Tenable Security Center has been set to.

system: locale

Returns
locale resource

Return type
dict

Examples

```
>>> sc.system.current_locale()
```

details()

Retrieves information about the Tenable Security Center instance. This method should only be called before authentication has occurred. As most of the information within this call already happens upon instantiation, there should be little need to call this manually.

system: get

Returns
The response dictionary

Return type
dict

Examples

```
>>> info = sc.system.details()
```

diagnostics(*task=None*, *options=None*, *fobj=None*)

Generates and downloads a diagnostic file for the purpose of troubleshooting an ailing Tenable Security Center instance.

system: diagnostics-generate

system: diagnostics-download

Parameters

- **fobj** (*FileObject, optional*) – The file-like object to write the diagnostics file to. If nothing is specified, a BytesIO object will be returnbed with the file.
- **options** (*list, optional*) – If performing a diagnostics generation, then which items should be bundled into the diagnostics file? Available options are `all`, `apacheLog`, `configuration`, `dependencies`, `dirlist`, `environment`, `installLog`, `logs`, `sanitize`, `scans`, `serverConf`, `setup`, `sysinfo`, and `upgradeLog`. If nothing is specified, it will default to `['all']`.
- **task** (*str, optional*) – Which task to perform. Available options are `appStatus` and `diagnosticsFile`. If nothing is specified, it will default to `diagnosticFile`.

Returns
A file-like object with the diagnostics file specified.

Return type
FileObject

Examples

```
>>> with open('diagnostics.tar.gz', 'wb') as fobj:
...     sc.system.diagnostics(fobj=fobj)
```

list_locales()

Retrieves the available system locales that Tenable Security Center can be set to.

system: locales

Returns
locales dictionary

Return type
dict

Examples

```
>>> sc.system.list_locales()
```

set_locale(*locale*)

Sets the system locale to be used. This requires an administrator to perform this task and will be a global change. The locale determines which pluginset language to use.

system: set-locale

Parameters
locale (*str*) – The plugin locale name

Returns
The new plugin locale.

Return type
str

Examples

Set the system locale to Japanese:

```
>>> sc.system.set_locale('ja')
```

status()

Retrieves the current system status

system: diagnostics

Returns
The status dictionary

Return type
dict

Examples

```
>>> status = sc.system.status()
```

32.28 Tickets

The following methods allow for interaction into the Tenable.sc Ticket API. These items are typically seen under the Worklow –> Tickets section of Tenable.sc. Methods available on `sc.tickets`:

class TicketAPI(*api: APISession*)

create(*name*, *assignee*, ***kw*)

Creates a ticket. ticket: create

Parameters

- **name** (*str*) – Required: The name for the ticket
- **assignee** (*dict*) – Required: A dictionary containing one key (id) for the user the ticket is being assigned to
- **status** (*str, optional*) – Optional status of the ticket: assigned, resolved, etc.
- **classification** (*str, optional*) – Optional classification of the ticket type, i.e. Information, Other, etc.
- **description** (*str, optional*) – Optional description for the ticket
- **notes** (*str, optional*) – Optional notes associated with the ticket
- **queries** (*list, optional*) – Optional list of IDs of queries to associate with the ticket
- **query** (*object, optional*) – Optional query object

Returns
The newly created ticket.

Return type
dict

Examples

```
>>> ticket = sc.tickets.create('Example Ticket', {'id':1}, status='assigned',␣
↪classification='information', description='This is an example ticket', notes=
↪'Example notes')
```

details(*id*, *fields=None*)

Returns the details for a specific ticket. ticket: details

Parameters

- **id** (*int*) – Required: the unique identifier for the ticket to be returned
- **fields** (*list*) – An optional list of attributes to return.

Returns
The ticket resource record.

Return type
dict

Examples

```
>>> ticket = sc.tickets.details(1)
>>> pprint(ticket)
```

edit(*id*, ***kw*)

Edits a ticket. ticket: edit

Parameters

- **id** (*int*) – Required: unique identifier of the ticket to be edited
- **name** (*str*) – Optional name for the ticket. Must not be blank.
- **assignee** (*dict*) – Optional dictionary containing one key (id) for the user the ticket is being assigned to
- **status** (*str, optional*) – Optional status of the ticket: assigned, resolved, etc. Must not be blank.
- **classification** (*str, optional*) – Optional classification of the ticket type, i.e. Information, Other, etc. Must not be blank.
- **description** (*str, optional*) – Optional description for the ticket
- **notes** (*str, optional*) – Optional notes associated with the ticket
- **queries** (*list, optional*) – Optional list of IDs of queries to associate with the ticket
- **query** (*object, optional*) – Optional query object

Returns
The newly updated ticket.

Return type
dict

Examples

```
>>> ticket = sc.tickets.edit(1, status='Resolved', notes='ran updates')
```

list(*fields=None*)

Outputs a dictionary of usable and manageable tickets, within which is a list of tickets. ticket: list

Parameters
fields (*list*) – Optional list of attributes to return for each ticket, e.g. [“name”,”description”]. If not specified, only a list of ticket IDs will return

Returns
A dictionary with two lists of ticket resources.

Return type
dict

Examples

```
>>> for ticket in sc.tickets.list():
...     pprint(ticket)
```

Note: you cannot delete tickets, must set them to resolved and they will be auto-purged via system configured retention period

32.29 Users

The following methods allow for interaction into the Tenable Security Center User API. These items are typically seen under the **Users** section of Tenable Security Center.

Methods available on `sc.users`:

class UserAPI(*api: APISession*)

create(*username*, *password*, *role*, ***kw*)

Creates a user.

user: create

Parameters

- **username** (*str*) – The username for the account
- **password** (*str*) – The password for the user to create
- **role** (*int*) – The role that should be assigned to this user.
- **address** (*str, optional*) – Optional street address information to associate to the user.
- **auth_type** (*str, optional*) – The Authentication type to use for the user. Valid options are `ldap`, `legacy`, `saml`, and `tns`. If left unspecified the default is `tns`.
- **city** (*str, optional*) – Optional city information to associate to the user.
- **country** (*str, optional*) – Optional country information to associate to the user.
- **default_objects** (*bool, optional*) – Should the default objects be created for members of this group?
- **email** (*str, optional*) – The email address to associate to the user.
- **email_notice** (*str, optional*) – What type of events should generate an email notification? Valid types are `id`, `password`, `both`, `none`.
- **fax** (*str, optional*) – A fax number to associate to the user.
- **fingerprint** (*str, optional*) – A fingerprint to associate to the user.
- **firstname** (*str, optional*) – A first name to associate to the user.
- **group** (*int, optional*) – A group to associate to the user. This parameter is required for users that are not Administrators.
- **is_locked** (*bool, optional*) – If the account locked? If left unspecified the default is False.
- **ldap_id** (*int, optional*) – If specifying an LDAP auth type, this is the numeric identifier for the LDAP configuration to use.

- **managed_usergroups** (*list, optional*) – A list of group ids that the user is allowed to manage users within.
- **managed_userobjs** (*list, optional*) – A list of group ids that the user is allowed to manage objects within. This includes asset lists, reports, etc.
- **org** (*int, optional*) – If logged in as an administrator, and creating a security manager account, the organization id must be passed in order to inform Tenable Security Center which organization to create the security manager within.
- **phone** (*str, optional*) – A phone number to associate to the user.
- **responsibility** (*int, optional*) – The asset list detailing what assets the user is responsible for. A value of `0` denotes all assets, any other non-zero integer must be the id of the asset list to associate to the user.
- **state** (*str, optional*) – The state to associate to the user.
- **timezone** (*str, optional*) – A timezone other than the system timezone to associate to the user. This will impact all times displayed within the user interface.
- **title** (*str, optional*) – A title to associate to the user.
- **update_password** (*bool, optional*) – Should the user be forced to update their password next login? If left unspecified, the default is False.

Returns
The newly created user.

Return type
dict

Examples

```
>>> user = sc.users.create('username', 'password', 1, group=1)
```

delete(*id: int*, *migrate_to: int | None = None*)

Removes a user.

user: delete

Parameters
id (*int*) – The numeric identifier for the user to remove.

Returns
An empty response.

Return type
str

Examples

```
>>> sc.users.delete(1)
```

details(*id: int*, *fields: List[str] = None*) → Dict

Returns the details for a specific user.

user: details

Parameters

- **id** (*int*) – The identifier for the user.
- **fields** (*list, optional*) – A list of attributes to return.

Returns
The user resource record.

Return type
dict

Examples

```
>>> user = sc.users.details(1)
>>> pprint(user)
```

edit(*id: int*, ***kw*)

Edits a user.

user: edit

Parameters

- **address** (*str, optional*) – Optional street address information to associate to the user.
- **auth_type** (*str, optional*) – The Authentication type to use for the user. Valid options are `ldap`, `legacy`, `saml`, and `tns`. If left unspecified the default is `tns`.
- **city** (*str, optional*) – Optional city information to associate to the user.
- **country** (*str, optional*) – Optional country information to associate to the user.
- **currentPassword** (*str, optional*) – Optional, requirement when updating password for current user in addition to password kwarg.
- **default_dashboards** (*bool, optional*) – Should the default dashboards be created for the user? If left unspecified, the default is True.
- **default_reportcards** (*bool, optional*) – Should the default report cards be created for the user? If left unspecified, the default is True.
- **default_reports** (*bool, optional*) – Should the default reports be created for the user? If left unspecified, the default is True.
- **email** (*str, optional*) – The email address to associate to the user.
- **email_notice** (*str, optional*) – What type of events should generate an email notification? Valid types are `id`, `password`, `both`, `none`.
- **fax** (*str, optional*) – A fax number to associate to the user.

- **fingerprint** (*str, optional*) – A fingerprint to associate to the user.
- **firstname** (*str, optional*) – A first name to associate to the user.
- **group** (*int, optional*) – A group to associate to the user. This parameter is required for users that are not Administrators.
- **is_locked** (*bool, optional*) – If the account locked? If left unspecified the default is False.
- **ldap_id** (*int, optional*) – If specifying an LDAP auth type, this is the numeric identifier for the LDAP configuration to use.
- **managed_usergroups** (*list, optional*) – A list of group ids that the user is allowed to manage users within.
- **managed_userobjs** (*list, optional*) – A list of group ids that the user is allowed to manage objects within. This includes asset lists, reports, etc.
- **org** (*int, optional*) – If logged in as an administrator, and creating a security manager account, the organization id must be passed in order to inform Tenable Security Center which organization to create the security manager within.
- **password** (*str, optional*) – The user password, currentPassword should be used with this if updating password for logged in user.
- **phone** (*str, optional*) – A phone number to associate to the user.
- **responsibility** (*int, optional*) – The asset list detailing what assets the user is responsible for. A value of `0` denotes all assets, any other non-zero integer must be the id of the asset list to associate to the user.
- **role** (*int, optional*) – The role that should be assigned to this user.
- **state** (*str, optional*) – The state to associate to the user.
- **timezone** (*str, optional*) – A timezone other than the system timezone to associate to the user. This will impact all times displayed within the user interface.
- **title** (*str, optional*) – A title to associate to the user.
- **update_password** (*bool, optional*) – Should the user be forced to update their password next login? If left unspecified, the default is False.
- **username** (*str, optional*) – The username for the account

Returns
The newly updated user.

Return type
dict

Examples

```
>>> user = sc.users.edit(1, username='newusername')
```

list(*fields: List[str] = None*) → Dict

Retrieves the list of user definitions.

user: list

Parameters
fields (*list, optional*) – A list of attributes to return for each user.

Returns
A list of user resources.

Return type
list

Examples

```
>>> for user in sc.users.list():
...     pprint(user)
```

CHAPTER

THIRTYTHREE

TENABLE OT SECURITY

This package covers the Tenable OT Security interface.

class TenableOT(***kwargs*)

The Tenable OT Security object is the primary interaction point for users to interface with Tenable.OT via the pyTenable library. All the API endpoint classes that have been written will be grafted onto this class.

Parameters

- **api_key** (*str, optional*) – The user's API key for Tenable OT Security. If an api key isn't specified, then the library will attempt to read the environment variable `TOT_API_KEY` to acquire the key.
- **url** (*str, optional*) – The base URL used to connect to the Tenable OT Security service. If a URL isn't specified, then the library will attempt to read the environment variable `TOT_URL` to acquire the URL.
- ****kwargs** – arguments passed to *tenable.base.platform.APIPlatform* for ConnectionAbortedErrorn management.

Examples

Basic Example:

```
>>> from tenable.ot import TenableOT
>>> ot = TenableOT(api_key='SECRET_KEY',
..                 url='https://ot.example.com')
```

Example with proper identification:

```
>>> ot = TenableOT(api_key='SECRET_KEY',
...                url='https://ot.example.com',
...                vendor='Company Name',
...                product='My Awesome Widget',
...                build='1.0.0')
```

Example with proper identification leveraging environment variables for the connection parameters:

```
>>> ot = TenableOT(vendor='Company', product='Widget', build='1.0.0')
```

property assets

The interface object for the *Tenable OT Security Assets APIs*.

property events

The interface object for the *Tenable OT Security Events APIs*.

property exports

The interface object for the *Tenable OT Security Exports*.

graphql(***kwargs*)

GraphQL Endpoint

This singular method exposes the GraphQL API to the library. As all keyword arguments are passed directly to the JSON body, it allows for a freeform interface into the GraphQL API.

Parameters

****kwargs** (*dict, optional*) – The key/values that should be passed to the body of the GraphQL request.

Example

```
>>> ot.graphql(
...     variables={'asset': 'b64 id string'},
...     query='''
...         query getAssetDetails($asset: ID!) {
...             asset(id: $asset) {
...                 id
...                 type
...                 name
...                 criticality
...                 location
...             }
...         }
... ''')
```

property plugins

The interface object for the *Tenable OT Security Plugins APIs*.

33.1 Data Export Helper

The methods exposed within the export module are designed to mimick the same structure that Tenable Vulnerability Management uses for exporting data. These methods ultimately translate to GraphQL queries and then are fed to the Tenable OT API.

class ExportsAPI(*api: APISession*)

assets(*filters: List[Dict] | None = None, sort: List[Dict] | None = None, search: str | None = None, start_at: str | None = None, limit: int = 200, filter_type: str = 'And', return_json: bool = False*) → OTExportsIterator | Dict

Assets Export

Parameters

- **limit** (*int, optional*) – The number of objects to be returned per page.
- **sort** (*list[dict], optional*) – A list of sort parameters to be passed to sort the responses.

- **search** (*str, optional*) – Search string
- **start_at** (*str, optional*) – Start returning data after this object id.
- **filters** (*list[dict], optional*) – List of filters to apply to restict the response to only the desired items.
- **filter_type** (*str, optional*) – When passing multiple filters, how should the filters be applied to the dataset? Acceptable values are *And* and *Or*.
- **return_json** (*bool, optional*) – If *True*, then the instead of an iterator, the json response will be returned instead.

Returns
By default the method will return an iterator that will handle pagination and return a single item at a time. If return_json is set to *True* however, then the JSON response will be returned instead for that page.

Return type
Union[OTExportsIterator, dict]

Example

```
>>> for asset in tot.exports.assets():
...     print(asset)
```

findings(*filters: List[Dict] | None = None, sort: List[Dict] | None = None, search: str | None = None, start_at: str | None = None, limit: int = 200, filter_type: str = 'And', return_json: bool = False*) → OTExportsIterator | Dict

Findings Export

Parameters

- **sorted** (*list[dict], optional*) – A list of asset sort parameters to be passed to sort the responses.
- **search** (*str, optional*) – Asset Search string
- **filters** (*list[dict], optional*) – List of asset filters to apply to restict the response to only the desired assets.
- **filter_type** (*str, optional*) – When passing multiple filters, how should the filters be applied to the dataset? Acceptable values are *And* and *Or*.

Returns
The Iterable that hadles the more complex logic of gathering the findings for each asset and presenting them to the caller.

Return type
OTFindingsIterator

Example

```
>>> for finding in tot.exports.findings():
...     print(finding)
```

plugins(*filters: List[Dict] | None = None*, *sort: List[Dict] | None = None*, *search: str | None = None*, *start_at: str | None = None*, *limit: int = 200*, *filter_type: str = 'And'*, *return_json: bool = False*) → OTExportsIterator | Dict

Plugin Export

Parameters

- **limit** (*int, optional*) – The number of objects to be returned per page.
- **sort** (*list[dict], optional*) – A list of sort parameters to be passed to sort the responses.
- **search** (*str, optional*) – Search string
- **start_at** (*str, optional*) – Start returning data after this object id.
- **filters** (*list[dict], optional*) – List of filters to apply to restict the response to only the desired items.
- **filter_type** (*str, optional*) – When passing multiple filters, how should the filters be applied to the dataset? Acceptable values are *And* and *Or*.
- **return_json** (*bool, optional*) – If *True*, then the instead of an iterator, the json response will be returned instead.

Returns

By default the method will return an iterator that will handle pagination and return a single item at a time. If return_json is set to *True* however, then the JSON response will be returned instead for that page.

Return type

Union[OTExportsIterator, dict]

Example

```
>>> for plugin in tot.exports.plugins():
...     print(plugin)
```

33.2 Assets

Methods described in this section relate to the assets API. These methods can be accessed at `TenableOT.assets`.

class AssetsAPI(*api: APISession*)

asset(*asset_id: int*, ***kwargs*) → OTGraphIterator

Retrieve a specific asset by ID.

Parameters

asset_id (*int*)

Returns

An iterator object handling data pagination.

Return type
OTGraphIterator

Example

```
>>>     tot.plugins.plugin(1)
```

list(*query_filter: dict | None = None*, *search: str | None = None*, *sort: List[dict] | None = None*, *start_at: str | None = None*, *limit: int | None = 200*, ***kwargs*) → OTGraphIterator

Retrieves a list of assets via the GraphQL API.

Parameters

- **query_filter** (*dict, optional*) – A document as defined by Tenable OT Security online documentation.
- **search** (*str, optional*) – A search string to further limit the response.
- **sort** (*List[dict], optional*) – A list of order documents, each of which must contain both the `field` and `direction` keys and may also contain the optional `function` key. Default sort is by descending id order. Please refer to Tenable OT Security online documentation for more information.
- **start_at** (*str, optional*) – The cursor to start the scan from (the default is an empty cursor).
- **limit** (*int, optional*) – Max number of objects that get retrieved per page (the default is 200).

Returns
An iterator object that will handle pagination of the data.

Return type
OTGraphIterator

Example

```
>>>     for asset in tot.assets.list(limit=500):
            print(asset)
```

schema_class

alias of `AssetsSchema`

33.3 Events

Methods described in this section relate to the events API. These methods can be accessed at `TenableOT.events`.

class EventsAPI(*api: APISession*)

list(*query_filter: dict | None = None*, *search: str | None = None*, *sort: List[dict] | None = None*, *start_at: str | None = None*, *limit: int | None = 200*, ***kwargs*) → OTGraphIterator

Retrieves a list of events via the GraphQL API.

Parameters

- **query_filter** (*dict, optional*) – A document as defined by Tenable OT Security online documentation.
- **search** (*str, optional*) – A search string to further limit the response.
- **sort** (*List[dict], optional*) – A list of order documents, each of which must contain both the `field` and `direction` keys and may also contain the optional `function` key. Default sort is by descending id order. Please refer to Tenable OT Security online documentation for more information.
- **start_at** (*str, optional*) – The cursor to start the scan from (the default is an empty cursor).
- **limit** (*int, optional*) – Max number of objects that get retrieved per page (the default is 200).

Returns
An iterator object that will handle pagination of the data.

Return type
`OTGraphIterator`

Example

```
>>>     for event in tot.events.list(limit=500):
            print(event)
```

schema_class

alias of `EventsSchema`

33.4 Plugins

Methods described in this section relate to the plugins API. These methods can be accessed at `TenableOT.plugins`.

class PluginsAPI(*api: APISession*)

list(*query: str = '\nquery plugins(\n $filter: PluginExpressionsParams\n $search: String\n $sort: [PluginSortParams!]!\n $limit: Int\n $startAt: String\n) {\n plugins(\n filter: $filter\n search: $search\n sort: $sort\n first: $limit\n after: $startAt\n) {\n pageInfo {\n endCursor\n }\n nodes {\n id\n name\n source\n family\n severity\n vprScore\n comment\n owner\n totalAffectedAssets\n affectedAssets {\n nodes {\n id\n name\n }\n }\n }\n}\n', query_filter: dict | None = None, search: str | None = None, sort: List[dict] | None = None, start_at: str | None = None, limit: int | None = 200, **kwargs*) → OTGraphIterator

Retrieves a list of plugins via the GraphQL API.

Parameters

- **query** (*str*) – A GraphQL query .
- **query_filter** (*dict, optional*) – A document as defined by Tenable OT Security online documentation.
- **search** (*str, optional*) – A search string to further limit the response.
- **sort** (*List[dict], optional*) – A list of order documents, each of which must contain both the `field` and `direction` keys and may also contain the optional

function key. Default sort is by descending id order. Please refer to Tenable OT Security online documentation for more information.

- **start_at** (*str, optional*) – The cursor to start the scan from (the default is an empty cursor).
- **limit** (*int, optional*) – Max number of objects that get retrieved per page (the default is 200).

Returns
An iterator object that will handle pagination of the data.

Return type
OTGraphIterator

Example

```
>>> for plugin in tot.plugins.list(limit=500):
        print(plugin)
```

plugin(*plugin_id: int*, ***kwargs*) → OTGraphIterator

Retrieve a specific plugin with additionals details by ID.

Parameters
plugin_id (*int*)

Returns
An iterator object handling data pagination.

Return type
OTGraphIterator

Example

```
>>> tot.plugins.plugin(1)
```

schema_class

alias of PluginsSchema

CHAPTER

THIRTYFOUR

TENABLE CLOUD SECURITY

class CloudSecurity(*url: str | None = None*, *api_key: str | None = None*, *verify: bool = True*, *schema_validation: bool = True*, *retries: int = 3*, *timeout: int = 300*, *vendor: str = 'unknown'*, *product: str = 'unknown'*, *build: str = 'unknown'*)

Tenable Cloud Security The CloudSecurity class is the primary interaction point for the Cloud Security GraphQL API within pyTenable. The most commonly used and supported GraphQL APIs have also been wrapped into the endpoints below.

Parameters

- **url** (*str*) – The url that points to the cloud security site to interact with. If not specified, the SDK will attempt to pull the parameter from the environment variable *TCS_URL*
- **api_key** (*str*) – The API Key to use for authentication to the API. If not specied, the SDK will attempt to pull the parameter from the environment variable *TCS_API_KEY*.
- **verify** (*bool, optional*) – Should SSL verification be performed? The default is *True*.
- **retries** (*int, optional*) – How many retries should be attempted before giving up on the current call? The default is *3*.
- **timeout** (*int, optional*) – How long to wait in seconds for the API to respond before timing out the call and throwing an error? The default is *300*.
- **vendor** (*str, optional*) – Identifies the vendor of the integration making the call to the API. This is used as part of the User-Agent construction.
- **product** (*str, optional*) – Identifies the product of the integration making the call to the API. This is used as part of the User-Agent construction.
- **build** (*str, optional*) – Identifies the build of the integration making the call to the API. This is used as part of the User-Agent construction.

query(*query: str | None = None*, *stored_file: str | None = None*, *keyword_arguments: dict[str, Any] | None = None*, *iterator: GraphQLIterator | None = None*, *graphql_model: str | None = None*, ***variables: Any*) → dict[str, Any]

Query the GraphQL API

Parameters

- **query** (*str | StringIO | DocumentNode, optional*) – The GraphQL query to pass to the remote API.
- **stored_file** (*str, optional*) – The filename of a vendored (stored) graphql query to construct.

> **Note**
>
> This parameter should not need to be used for outside of the library itself. All of the queries available with this parameter are also wrapped within the endpoint classes.

- **iterator** (GraphQLIterator, *optional*) – If specified, the response will be an instance of this iterable instead of the dictionary response. Useful for when the response data is expected to be larger datasets that would require multiple pages to collect all of the data.
- **graphql_model** (*str*, *optional*) – When using the iterator, we need to specify the base entity that is returned from the GraphQL response.
- **keyword_argument** (*dict*, *optional*) – Anything specified within this dictionary will be passed on to the gql libraries query method. While not expected to be commonly used, we're exposing this here just incase we need it.
- ****variables** (*dict*, *optional*) – Any variable declarations that need to be passed along with the query.

Returns

If no iterator is passed, then the response dictionary is
returned to the caller.

GraphQLIterator:
If an iterator class was passed, then the query is generated and the passed to the iterator nefore returning an instance of the iterator class.

Return type
Dict

Example

A very basic example:

```
>>> session.query('{ hero { name } }')
```

An example using a variable within the query:

```
>>> query = '''
... query HeroNameAndFriends($episode: Episode) {
...   hero(episode: $episode) {
...     name
...     friends {
...       name
...     }
...   }
... }
>>> session.query(query, episode='JEDI')
```

validate(*query: str | StringIO*) → list[GraphQLError]

Validates the query against the schema and returns any validation errors that may have occured.

Parameters
query (*str | StringIO*) – The query to validate against

Returns:

property assets

Interface object for the *Tenable Cloud Security Assets Queries*.

property vulns

Interface object for the *Tenable Cloud Security Vulnerability Queries*.

34.1 Assets

The following methods leverage stored queries relating to asset metadata within Cloud Security.

class AssetsAPI(*api:* GraphQLSession)

compute(*page_size: int = 10*) → *CloudSecurityAssetIterator*

Requests compute asset metadata.

Parameters
page_size (*int, optional*) – How many items should be returned per page?

Returns
An iterable is returned that handles paging of the data.

Example

```
>>> for asset in cloudsecurity.assets.compute():
...     print(asset)
```

container(*page_size: int = 10*)

Requests container asset metadata.

Parameters
page_size (*int, optional*) – How many items should be returned per page?

Returns
An iterable is returned that handles paging of the data.

Example

```
>>> for asset in cloudsecurity.assets.container():
...     print(asset)
```

class CloudSecurityAssetIterator(*api*, ***kw*)

34.2 Vulns

The following methods leverage stored queries relating to vulnerability metadata within Cloud Security.

class VulnsAPI(*api:* GraphQLSession)

containerimages(*page_size: int = 10*)

Requests container image vulnerability metadata.

Parameters
page_size (*int, optional*) – How many items should be returned per page?

Returns
An iterable is returned that handles paging of the data.

Example

```
>>> for vuln in cloudsecurity.vulns.containerimages():
...     print(vuln)
```

virtualmachines(*page_size: int = 10*)

Requests virtual machine vulnerability metadata.

Parameters
page_size (*int, optional*) – How many items should be returned per page?

Returns
An iterable is returned that handles paging of the data.

Example

```
>>> for vuln in cloudsecurity.vulns.virtualmachines():
...     print(vuln)
```

class CloudSecurityVulnIterator(*api*, ***kw*)

CHAPTER

THIRTYFIVE

PRODUCT DOWNLOADS

class Downloads(*api_token=None*, ***kwargs*)

The Downloads object is the primary interaction point for users to interface with Downloads API via the pyTenable library. All of the API endpoint classes that have been written will be grafted onto this class.

Parameters

- **api_token** (*str, optional*) – The user's API access key for Tenable Vulnerability Management If an access key isn't specified, then the library will attempt to read the environment variable `TDL_API_TOKEN` to acquire the key.
- **retries** (*int, optional*) – The number of retries to make before failing a request. The default is `5`.
- **backoff** (*float, optional*) – If a 429 response is returned, how much do we want to backoff if the response didn't send a Retry-After header. The default backoff is 1 second.
- **vendor** (*str, optional*) – The vendor name for the User-Agent string.
- **product** (*str, optional*) – The product name for the User-Agent string.
- **build** (*str, optional*) – The version or build identifier for the User-Agent string.
- **timeout** (*int, optional*) – The connection timeout parameter informing the library how long to wait in seconds for a stalled response before terminating the connection. If unspecified, the default is 120 seconds.

Examples

Basic Example:

```
>>> from tenable.dl import Downloads
>>> dl = Downloads(api_token='API_TOKEN')
```

Example with proper identification:

```
>>> dl = Downloads('API_TOKEN',
>>>     vendor='Company Name',
>>>     product='My Awesome Widget',
>>>     build='1.0.0')
```

Example with proper identification leveraging environment variables for access and secret keys:

```
>>> dl = Downloads(
>>>     vendor='Company Name', product='Widget', build='1.0.0')
```

details(*page*)

Retrieves the specific download items for the page requested.

API Endpoint Documentation

Parameters
page (*str*) – The name of the page to request.

Returns
The page details.

Return type
dict

Examples

```
>>> details = dl.details('nessus')
```

download(*page*, *package*, *fobj=None*)

Retrieves the requested package and downloads the file.

API Endpoint Documentation

Parameters

- **page** (*str*) – The name of the page
- **package** (*str*) – The package filename
- **fobj** (*FileObject, optional*) – The file-like object to write the package to. If nothing is specified, then a BytesIO object will be used.

Returns
The binary package

Return type
FileObject

Examples

```
>>> with open('Nessus-latest.x86_64.rpm', 'wb') as pkgfile:
...     dl.download('nessus',
...         'Nessus-8.3.0-es7.x86_64.rpm', pkgfile)
```

list()

Lists the available content pages.

API Endpoint Documentation

Returns
The list of page resources.

Return type
list

Examples

```
>>> pages = dl.list()
>>> for page in pages:
...     pprint(page)
```

CHAPTER

THIRTYSIX

TENABLE IDENTITY EXPOSURE

This package covers the Tenable Identity Exposure interface.

class TenableIE(***kwargs*)

property about
The interface object for the *Tenable Identity Exposure About APIs*.

property ad_object
The interface object for the *Tenable Identity Exposure AD Object APIs*.

property alerts
The interface object for the *Tenable Identity Exposure Alerts APIs*.

property api_keys
The interface object for the *Tenable Identity Exposure API-Keys APIs*.

property application_settings
The interface object for the *Tenable Identity Exposure Application Settings APIs*.

property attack_type_options
The interface object for the *Tenable Identity Exposure Attack Type Options APIs*.

property attack_types
The interface object for the *Tenable Identity Exposure Attack Types APIs*.

property attacks
The interface object for the *Tenable Identity Exposure Attacks APIs*.

property category
The interface object for the *Tenable Identity Exposure Category APIs*.

property checker
The interface object for the *Tenable Identity Exposure Checker APIs*.

property checker_option
The interface object for the *Tenable Identity Exposure Checker option APIs*.

property dashboard
The interface object for the *Tenable Identity Exposure Dashboard APIs*.

property deviance
The interface object for the *Tenable Identity Exposure Deviance APIs*.

property directories
The interface object for the *Tenable Identity Exposure Directories APIs*.

property email_notifiers
The interface object for the *Tenable Identity Exposure Email Notifiers APIs*.

property event
The interface object for the *Tenable Identity Exposure Event APIs*.

property infrastructure
The interface object for the *Tenable Identity Exposure Infrastructure APIs*.

property ldap_configuration
The interface object for the *Tenable Identity Exposure LDAP Configuration APIs*.

property license
The interface object for the *Tenable Identity Exposure License APIs*.

property lockout_policy
The interface object for the *Tenable Identity Exposure Lockout Policy APIs*.

property preference
The interface object for the *Tenable Identity Exposure Preference APIs*.

property profiles
The interface object for the *Tenable Identity Exposure Profiles APIs*.

property reason
The interface object for the *Tenable Identity Exposure Reason APIs*.

property roles
The interface object for the *Tenable Identity Exposure Roles APIs*.

property saml_configuration
The interface object for the *Tenable Identity Exposure SAML configuration APIs*.

property score
The interface object for the *Tenable Identity Exposure Score APIs*.

property syslog
The interface object for the *Tenable Identity Exposure Syslog APIs*.

property topology
The interface object for the *Tenable Identity Exposure Topology APIs*.

property users
The interface object for the *Tenable Identity Exposure Users APIs*.

property widgets
The interface object for the *Tenable Identity Exposure Widget APIs*.

36.1 About

Methods described in this section relate to the About API. These methods can be accessed at `TenableIE.about`.

class AboutAPI(*api: APISession*)

version() → str

Returns the version of the connected Tenable Identity Exposure instance.

Examples

```
>>> tie.about.version()
```

36.2 AD Object

Methods described in this section relate to the ad object API. These methods can be accessed at `TenableIE.ad_object`.

class ADObjectAPI(*api: APISession*)

details(*directory_id: str*, *infrastructure_id: str*, *ad_object_id: str*) → Dict

Retrieves the details of a specific AD object.

Parameters

- **directory_id** (*str*) – The directory instance identifier.
- **infrastructure_id** (*str*) – The infrastructure instance identifier.
- **ad_object_id** (*str*) – The AD Object identifier.

Returns
The AD object.

Return type
dict

Examples

```
>>> tie.ad_object.details(
...     directory_id='1',
...     infrastructure_id='1',
...     ad_object_id='1'
...     )
```

details_by_event(*directory_id: str*, *infrastructure_id: str*, *ad_object_id: str*, *event_id: str*) → Dict

Retrieves the details of a specific AD object.

Parameters

- **directory_id** (*str*) – The directory instance identifier.
- **infrastructure_id** (*str*) – The infrastructure instance identifier.
- **ad_object_id** (*str*) – The AD Object identifier.

- **event_id** (*str*) – The event identifier.

Returns
The AD object.

Return type
dict

Examples

```
>>> tie.ad_object.details_by_event(
...     directory_id='1',
...     infrastructure_id='1',
...     ad_object_id='1',
...     event_id='1'
...     )
```

details_by_profile_and_checker(*profile_id: str*, *checker_id: str*, *ad_object_id: str*) → Dict

Retrieves an AD object details by id that have deviances for a specific profile and checker

Parameters

- **profile_id** (*str*) – The profile instance identifier.
- **checker_id** (*str*) – The checker instance identifier.
- **ad_object_id** (*str*) – The AD Object identifier.

Returns
The AD object.

Return type
dict

Examples

```
>>> tie.ad_object.details_by_profile_and_checker(
...     profile_id='1',
...     checker_id='1',
...     ad_object_id='1'
...     )
```

get_changes(*directory_id: str*, *infrastructure_id: str*, *ad_object_id: str*, *event_id: str*, ***kwargs*) → List[Dict]

Get the AD object changes between a given event and event which precedes it.

Parameters

- **directory_id** (*str*) – The directory instance identifier.
- **infrastructure_id** (*str*) – The infrastructure instance identifier.
- **ad_object_id** (*str*) – The AD Object identifier.
- **event_id** (*str*) – The event identifier.
- **wanted_values** (*optional, list[str]*) – Which values user wants to include. `before` to include the values just before the event, `after` to include the values just after the event or `current` to include the current values.

Returns
The list of AD objects.

Return type
list[dict]

Examples

```
>>> tie.ad_object.get_changes(
...     directory_id='1',
...     infrastructure_id='1',
...     ad_object_id='1',
...     event_id='1',
...     wanted_values=['current', 'after']
...     )
```

search_all(*profile_id: str*, *checker_id: str*, *expression: Mapping*, *directories: List[int]*, *reasons: List[int]*, *show_ignored: bool*, ***kwargs*) → ADObjectIterator

Search all AD objects having deviances by profile by checker

Parameters

- **profile_id** (*str*) – The profile instance identifier.
- **checker_id** (*str*) – The checker instance identifier.
- **expression** (*mapping*) – An object describing a filter for searched items.
- **directories** (*list[int]*) – The list of directory instance identifiers.
- **reasons** (*list[int]*) – The list of reasons identifiers.
- **show_ignored** (*bool*) – Whether AD Object that only have ignored deviances should be included?
- **date_start** (*optional, str*) – The date after which the AD object deviances should have been emitted.
- **date_end** (*optional, str*) – The date before which the AD object deviances should have been emitted.
- **page** (*optional, int*) – The page number user wants to retrieve.
- **per_page** (*optional, int*) – The number of records per page user wants to retrieve.
- **max_items** (*optional, int*) – The maximum number of records to return before stopping iteration.
- **max_pages** (*optional, int*) – The maximum number of pages to request before throwing stopping iteration.

Returns
An iterator that handles the page management of the requested records.

Return type
ADObjectIterator

Examples

```
>>> for ado in tie.ad_object.search_all(
...     profile_id='1',
...     checker_id='1',
...     show_ignored=False,
...     reasons=[1, 2],
...     directories=[1],
...     expression={'OR': [{
...         'whencreated': '2021-07-29T12:27:50.0000000Z'
...     }]},
...     date_end='2022-12-31T18:30:00.000Z',
...     date_start='2021-12-31T18:30:00.000Z',
...     page=1,
...     per_page=20,
...     max_pages=10,
...     max_items=200
...     ):
...     pprint(ado)
```

36.3 Alerts

Methods described in this section relate to the alerts API. These methods can be accessed at `TenableIE.alerts`.

class **AlertsAPI**(*api: APISession*)

details(*alert_id: str*) → Dict

Retrieves the details of a specific alert.

Parameters
alert_id (*str*) – The alert instance identifier.

Returns
the alert object.

Return type
dict

Examples

```
>>> tie.alerts.details(
...     alert_id='1'
...     )
```

list_by_profile(*profile_id: str*, ***kwargs*) → AlertIterator

Retrieve all alert instances

Parameters

- **profile_id** (*str*) – The profile instance identifier.
- **archived** (*optional, bool*) – is alert archived?
- **read** (*optional, bool*) – is alert read?

- **page** (*optional, int*) – The page number user wants to retrieve.
- **per_page** (*optional, int*) – The number of records per page user wants to retrieve.
- **max_items** (*optional, int*) – The maximum number of records to return before stopping iteration.
- **max_pages** (*optional, int*) – The maximum number of pages to request before throwing stopping iteration.

Returns
An iterator that handles the page management of the requested records.

Return type
AlertIterator

Examples

```
>>> for alert in tie.alerts.list_by_profile(
...     profile_id='1',
...     archived=False,
...     read=False,
...     page=1,
...     per_page=20,
...     max_pages=10,
...     max_items=200
...     ):
...     pprint(alert)
```

update(*alert_id: str*, ***kwargs*) → Dict

Update alert instance

Parameters

- **alert_id** (*str*) – The alert instance identifier.
- **archived** (*optional, bool*) – is alert archived?
- **read** (*optional, bool*) – is alert read?

Returns
The updated alert object.

Return type
dict

Example

```
>>> tie.alerts.update(
...     alert_id='1',
...     archived=False,
...     read=False
...     )
```

update_on_profile(*profile_id: str*, ***kwargs*) → None

Update alerts for one profile

Parameters

- **profile_id** (*str*) – The alert instance identifier.
- **archived** (*optional, bool*) – is alert archived?
- **read** (*optional, bool*) – is alert read?

Return type
None

Example

```
>>> tie.alerts.update_on_profile(
...     profile_id='1',
...     archived=False,
...     read=False
...     )
```

36.4 APIKeys

Methods described in this section relate to the APIKeys API. These methods can be accessed at `TenableIE.api_keys`.

class APIKeyAPI(*api: APISession*)

get() → str

Gets the API Key of the current user.

Examples

```
>>> tie.api_keys.get()
```

refresh() → str

Creates or renews an API for the current user. Will also refresh the API Key used in the current TenableIE session.

Examples

```
>>> tie.api_keys.refresh()
```

36.5 Attacks

Methods described in this section relate to the attacks API. These methods can be accessed at `TenableIE.attacks`.

class AttacksAPI(*api: APISession*)

list(*profile_id: str*, ***kwargs*) → List[Dict]

Retrieve all attacks

Parameters

- **profile_id** (*str*) – The attack profile identifier.
- **resource_type** (*str*) – The type of resource. possible values are `infrastructure`, `directory`, `hostname`, `ip`.
- **resource_value** (*str*) – The value of resource.
- **attack_type_ids** (*optional, list[str]*) – The list of attack type ids.
- **date_end** (*optional, str*) – The date before which the attack occurence should be considered.
- **date_start** (*optional, str*) – The date after which the attack occurence should be considered.
- **include_closed** (*optional, str*) – Whether closed attacks should be included? Accepted values are `true` or `false`
- **limit** (*optional, str*) – The number of records user wants to return.
- **order** (*optional, str*) – The order of response. Accepted values are `asc` or `desc`.
- **search** (*optional, str*) – Search a value in response.

Returns
The list of attacks objects

Return type
list

Examples

```
>>> tie.attacks.list(
...     profile_id='1',
...     resource_type='infrastructure',
...     resource_value='1',
...     attack_type_ids=[1, 2],
...     include_closed='false',
...     limit='10',
...     order='asc',
...     search='value',
...     date_end='2022-12-31T18:30:00.000Z',
...     date_start='2021-12-31T18:30:00.000Z'
...     )
```

36.6 Attack Type

Methods described in this section relate to the the attack type API. These methods can be accessed at `TenableIE.attack_types`.

class AttackTypesAPI(*api: APISession*)

list() → List[Dict]
Retrieve all attack types

Returns
The list of attack types objects

Return type
list

Examples

```
>>> tie.attack_types.list()
```

36.7 Attack Type Options

Methods described in this section relate to the attack type options API. These methods can be accessed at `TenableIE.attack_type_options`.

class `AttackTypeOptionsAPI`(*api: APISession*)

create(*profile_id: str*, *attack_type_id: str*, ***kwargs*) → List

Create attack type options related to a profile and attack type.

Parameters

- **profile_id** (*str*) – The attack profile identifier.
- **attack_type_id** (*str*) – The attack type identifier.
- **codename** (*str*) – The codename of attack type option.
- **value** (*str*) – The new value of the option.
- **value_type** (*str*) – The type of option. possible values are `string`, `regex`, `float`, `integer`, `boolean`, `date`, `object`, `array/string`, `array/regex`, `array/integer`, `array/boolean`, `array/select`, `array/object`
- **directory_id** (*optional, int*) – The directory identifier.

Returns
The newly created attack type options.

Return type
list

Example

```
>>> tie.attack_type_options.create(
...     profile_id='1',
...     attack_type_id='1',
...     codename='codename',
...     value='Some value',
...     value_type='string',
...     directory_id=None
...     )
```

list(*profile_id: str*, *attack_type_id: str*, ***kwargs*) → List[Dict]

Get all attack type options related to a profile and attack type.

Parameters

- **profile_id** (*str*) – The attack profile identifier.

- **attack_type_id** (*str*) – The attack type identifier.
- **staged** (*optional, bool*) – Get only objects that are staged. Accepted values are `True`, `False`.

Returns
The list of attack type options objects

Return type
list

Examples

```
>>> tie.attack_type_options.list(
...     profile_id='1',
...     attack_type_id='1',
...     staged=False
...     )
```

36.8 Application Settings

Methods described in this section relate to the application settings API. These methods can be accessed at `TenableIE.application_settings`.

class ApplicationSettingsAPI(*api: APISession*)

details() → Dict

Get the application settings

Returns
The application settings objects

Return type
dict

Examples

```
>>> tie.application_settings.get_settings()
```

update(***kwargs*) → Dict

Update the application settings

Parameters

- **smtp_server_address** (*optional, str*) – The IP address of the SMTP server to use to send mails.
- **smtp_server_port** (*optional, int*) – The port of SMTP server to use to send mails.
- **smtp_account** (*optional, str*) – The login to use to authenticate against SMTP server.
- **smtp_account_password** (*optional, str*) – The password to use to authenticate against SMTP server.

- **smtp_use_start_tls** (*optional, bool*) – Whether the startTLS SMTP command should be used to secure the connection to the SMTP server?
- **tls** (*optional, bool*) – Whether the configured server should connect using TLS?
- **email_sender** (*optional, str*) – The email address to display as the sender in the emails sent.
- **default_role_ids** (*optional, list[int]*) – The default role identifiers.
- **default_profile_id** (*optional, int*) – The default profile identifier.
- **internal_certificate** (*optional, str*) – The certificate chain to use to verify certificates on TLS connections.

Returns
The application settings objects

Return type
dict

Example

```
>>> tie.application_settings.update_settings(
...     smtp_use_start_tls=True,
...     tls=False,
...     default_profile_id=1,
...     )
```

36.9 Category

Methods described in this section relate to the category API. These methods can be accessed at `TenableIE.category`.

class CategoryAPI(*api: APISession*)

details(*category_id: str*) → Dict

Retrieves the details of particlar category bases on category_id.

Parameters
category_id (*str*) – The category instance identifier.

Returns
Returns the details of a given `category_id`.

Return type
dict

Examples

```
>>> tie.category.details(category_id='5')
```

list() → List[Dict]

Retrieves the list of categories in the instance.

Returns
Returns a list of categories.

Return type
list

Examples

```
>>> tie.category.list()
```

36.10 Checker

Methods described in this section relate to the checker API. These methods can be accessed at `TenableIE.checker`.

class CheckerAPI(*api: APISession*)

details(*checker_id: str*) → Dict

Gets the details of the particular checker based on checker identifier.

Parameters
checker_id (*str*) – The checker instance identifier.

Returns
Details of the given `checker_id`.

Return type
dict

Examples

```
>>> tie.checker.details(checker_id='1')
```

list() → List[Dict]

Retrieves the list of checkers.

Returns
A list of checkers.

Return type
list

Examples

```
>>> tie.checker.list()
```

36.11 Checker Option

Methods described in this section relate to the checker option API. These methods can be accessed at `TenableIE.checker_option`.

class `CheckerOptionAPI`(*api: APISession*)

create(*profile_id: str*, *checker_id: str*, ***kwargs*) → List[Dict]

Creates the new checker-option.

Parameters

- **profile_id** (*str*) – The profile instance identifier.
- **checker_id** (*str*) – The checker instance identifier.
- **codename** (*str*) – The codename of the checker option.
- **value** (*str*) – The value of the checker option.
- **value_type** (*str*) – The type of the checker option. Accepted values are: `string`, `regex`, `float`, `integer`, `boolean`, `date`, `object`, `array/string`, `array/regex`, `array/integer`, `array/boolean`, `array/select`, `array/object`.
- **directory_id** (*optional, int*) – The directory instance identifier.

Returns
Created checker option instance.

Return type
list

Examples

```
>>> tie.checker_option.create(
...     profile_id='9',
...     checker_id='2',
...     codename='codename',
...     value='false',
...     value_type='boolean'
...     directory_id=3
...     )
```

list(*profile_id: str*, *checker_id: str*, ***kwargs*) → List[Dict]

Retrieves the list of checker-options.

Parameters

- **profile_id** (*str*) – The profile instance identifier.
- **checker_id** (*str*) – The checker instance identifier.

- **staged** (*optional, bool*) – Get only the checker-options that are staged. Accepted values are `True` and `False`. Added checker options are first staged until the profile is commited. The staged profile options are not activated and don't affect yet the IOE and the exposure detection.

Returns
List of checker options.

Return type
list

Examples

```
>>> tie.checker_option.list(
...     profile_id='9',
...     checker_id='1',
...     staged=True
...     )
```

36.12 Dashboard

Methods described in this section relate to the dashboard API. These methods can be accessed at `TenableIE.dashboard`.

class DashboardAPI(*api: APISession*)

create(*name: str*, *order: int*) → Dict

Create a new dashboard instance.

Parameters

- **name** (*str*) – The name of the new dashboard.
- **order** (*int*) – order of the dashboard.

Returns
The created dashboard instance.

Return type
dict

Examples

```
>>> tie.dashboard.create(
...     name='new_dashboard',
...     order=10)
```

delete(*dashboard_id: str*) → None

Deletes the dashboard instance

Parameters
dashboard_id (*str*) – The dashboard instance identifier.

Examples

```
>>> tie.dashboard.delete(dashboard_id='22')
```

details(*dashboard_id: str*) → Dict

Retrieves the details for a specific dashboard instance.

Parameters
dashboard_id (*str*) – The dashboard instance identifier.

Returns
The details of the dashboard object of specified dashboard_id.

Return type
dict

Examples

```
>>> tie.dashboard.details(dashboard_id='1')
```

list() → List[Dict]

Retrieve all dashboard instances.

Returns
The list of dashboard objects.

Return type
list

Examples

```
>>> tie.dashboard.list()
```

update(*dashboard_id: str*, ***kwargs*)

Updates the dashboard instance based on `dashboard_id`.

Parameters

- **dashboard_id** (*str*) – The dashboard instance identifier.
- **name** (*optional, str*) – The updated name.
- **order** (*optional, int*) – The order of the dashboard.

Examples

```
>>> tie.dashboard.update(
...     dashboard_id='23',
...     name='updated_dashboard_name',
...     order=1)
```

36.13 Deviance

Methods described in this section relate to the deviance API. These methods can be accessed at `TenableIE.deviance`.

class DevianceAPI(*api: APISession*)

get_history_details(*infrastructure_id: str*, *directory_id: str*, *deviance_id: str*) → Dict

Retrieve ad-object-deviance-history instance by id.

Parameters

- **infrastructure_id** (*str*) – The infrastructure instance identifier.
- **directory_id** (*str*) – The directory instance identifier.
- **deviance_id** (*str*) – The deviance identifier.

Returns

The deviance object.

Return type

dict

Example

```
>>> tie.deviance.history_details(
...     infrastructure_id='1',
...     directory_id='1',
...     deviance_id='1'
...     )
```

list(*infrastructure_id: str*, *directory_id: str*, ***kwargs*) → List[Dict] | DevianceIterator

Retrieve all deviances for a directory

Parameters

- **infrastructure_id** (*str*) – The infrastructure instance identifier.
- **directory_id** (*str*) – The directory instance identifier.
- **page** (*optional, int*) – The page number user wants to retrieve.
- **per_page** (*optional, int*) – The number of records per page user wants to retrieve.
- **batch_size** (*optional, int*) – The total number of records user wants to retrieve.
- **last_identifier_seen** (*optional, int*) – The deviance identifier after which the deviance should be considered.
- **resolved** (*optional, bool*) – is the deviance resolved?
- **max_items** (*optional, int*) – The maximum number of items to return before stopping iteration.
- **max_pages** (*optional, int*) – The maximum number of pages to request before throwing stopping iteration.

Returns

An iterator that handles the page management of the requested records.

Return type
list[dict] or DevianceIterator

Examples

return an iterator to loop through all records

```
>>> for deviance in tie.deviance.list(
...     infrastructure_id='1',
...     directory_id='1',
...     resolved=True,
...     last_identifier_seen=1,
...     page=1,
...     per_page=10,
...     max_pages=11,
...     max_items=100
...     ):
...     pprint(deviance)
```

return a list of requested records using batch_size

```
>>> tie.deviance.list(
...     infrastructure_id='1',
...     directory_id='1',
...     resolved=True,
...     last_identifier_seen=1,
...     batch_size=100
...     )
```

list_by_checker(*profile_id: str*, *checker_id: str*, *expression: Mapping*, ***kwargs*) → List[Dict] | DevianceIterator

Retrieve all deviances by checker

Parameters

- **profile_id** (*str*) – The profile instance identifier.
- **checker_id** (*str*) – The checker instance identifier.
- **expression** (*mapping*) – An object describing a filter for searched items.
- **batch_size** (*optional, int*) – The total number of records user wants to retrieve.
- **last_identifier_seen** (*optional, int*) – The deviance identifier after which the deviance should be considered.
- **page** (*optional, int*) – The page number user wants to retrieve.
- **per_page** (*optional, int*) – The number of records per page user wants to retrieve.
- **max_items** (*optional, int*) – The maximum number of items to return before stopping iteration.
- **max_pages** (*optional, int*) – The maximum number of pages to request before throwing stopping iteration.

Returns
An iterator that handles the page management of the requested records.

Return type
list[dict] or DevianceIterator

Examples

return an iterator to loop through all records

```
>>> for deviance in tie.deviance.list_by_checker(
...     profile_id='1',
...     checker_id='1',
...     expression={'OR': [{
...         'whencreated': '2021-07-29T12:27:50.0000000Z'
...     }]},
...     last_identifier_seen=1,
...     page=1,
...     per_page=10,
...     max_pages=11,
...     max_items=100
...     ):
...     pprint(deviance)
```

return a list of requested records using batch_size

```
>>> tie.deviance.list_by_checker(
...     profile_id='1',
...     checker_id='1',
...     expression={'OR': [{
...         'whencreated': '2021-07-29T12:27:50.0000000Z'
...     }]},
...     last_identifier_seen=1,
...     batch_size=100
...     )
```

list_by_directory_and_checker(*profile_id: str*, *infrastructure_id: str*, *directory_id: str*, *checker_id: str*, ***kwargs*) → DevianceIterator

Retrieve all deviances related to a single directory and checker

Parameters

- **profile_id** (*str*) – The profile instance identifier.
- **infrastructure_id** (*str*) – The infrastructure instance identifier.
- **directory_id** (*str*) – The directory instance identifier.
- **checker_id** (*str*) – The checker instance identifier.
- **page** (*optional, str*) – The page number user wants to retrieve.
- **per_page** (*optional, str*) – The number of records per page user wants to retrieve.
- **max_items** (*optional, int*) – The maximum number of items to return before stopping iteration.
- **max_pages** (*optional, int*) – The maximum number of pages to request before throwing stopping iteration.

Returns
An iterator that handles the page management of the requested records.

Return type
DevianceIterator

Examples

```
>>> for deviance in tie.deviance.list_by_directory_and_checker(
...     profile_id='1',
...     infrastructure_id='1',
...     dashboard_id='1',
...     checker_id='1',
...     page=1,
...     per_page=10,
...     max_pages=11,
...     max_items=100
...     ):
...     pprint(deviance)
```

search(*profile_id: str*, *checker_id: str*, *ad_object_id: str*, *show_ignored: bool*, ***kwargs*) → DevianceIterator

Search all deviances by profile by checker by AD object.

Parameters

- **profile_id** (*str*) – The profile instance identifier.
- **checker_id** (*str*) – The checker identifier.
- **ad_object_id** (*str*) – The AD object identifier.
- **show_ignored** (*bool*) – Whether ignored deviances should be included?
- **date_start** (*optional, str(datetime)*) – The date after which the deviances should have been emitted.
- **date_end** (*optional, str(datetime)*) – The date before which the deviances should have been emitted.
- **page** (*optional, int*) – The page number user wants to retrieve.
- **per_page** (*optional, int*) – The number of records per page user wants to retrieve.
- **max_items** (*optional, int*) – The maximum number of items to return before stopping iteration.
- **max_pages** (*optional, int*) – The maximum number of pages to request before throwing stopping iteration.

Returns
An iterator that handles the page management of the requested records.

Return type
DevianceIterator

Example

```
>>> for deviance in tie.deviance.search(
...     profile_id='1',
...     checker_id='1',
...     ad_object_id='1',
...     show_ignored=True,
...     page=1,
...     per_page=10,
...     max_pages=11,
...     max_items=100
...     ):
...     pprint(deviance)
```

update_by_checker(*profile_id: str*, *checker_id: str*, *ignore_until: str*) → None

Update instances matching a checker id.

Parameters

- **profile_id** (*str*) – The profile instance identifier.
- **checker_id** (*str*) – The checker instance identifier.
- **ignore_until** (*str(datetime)*) – Ignore deviance until defined date.

Return type
None

Example

```
>>> tie.deviance.update_by_checker(
...     profile_id='1',
...     checker_id='1',
...     ignore_until='2022-01-27T23:59:59.999Z'
...     )
```

update_history_details(*infrastructure_id: str*, *directory_id: str*, *deviance_id: str*, ***kwargs*) → Dict

Retrieve ad-object-deviance-history instance by id.

Parameters

- **infrastructure_id** (*str*) – The infrastructure instance identifier.
- **directory_id** (*str*) – The directory instance identifier.
- **deviance_id** (*str*) – The deviance identifier.
- **ignore_until** (*optional, str(datetime)*) – Ignore deviance until defined date.

Returns
The deviance object.

Return type
dict

Example

```
>>> tie.deviance.history_details(
...     infrastructure_id='1',
...     directory_id='1',
...     deviance_id='1',
...     ignore_until='2022-01-27T23:59:59.999Z'
...     )
```

update_on_ado_and_checker(*profile_id: str*, *checker_id: str*, *ad_object_id: str*, *ignore_until: str*) → None

Update the deviances emitted on a specific AD object and for specific checker.

Parameters

- **profile_id** (*str*) – the profile instance identifier.
- **checker_id** (*str*) – The checker instance identifier.
- **ad_object_id** (*str*) – The AD object instance identifier.
- **ignore_until** (*str(datetime)*) – Ignore deviance until defined date.

Return type
None

Example

```
>>> tie.deviance.update_on_ado_and_checker(
...     profile_id='1',
...     checker_id='1',
...     ad_object_id='1',
...     ignore_until='2022-01-27T23:59:59.999Z'
...     )
```

36.14 Directory

Methods described in this section relate to the directory API. These methods can be accessed at `TenableIE.directories`.

class DirectoriesAPI(*api: APISession*)

create(*infrastructure_id: int*, *name: str*, *ip: str*, *dns: str*, ***kwargs*) → List[Dict]

Creates a new directory instance.

Parameters

- **infrastructure_id** (*int*) – The infrastructure object to bind this directory to.
- **name** (*str*) – Name of the directory instance.
- **ip** (*str*) – The IP Address of the directory server.
- **dns** (*str*) – The DNS domain that this directory is tied to.
- **directory_type** (*optional, str*) – The directory's type.

- **ldap_port** (*optional, str*) – The port number associated to the LDAP service on the directory server.
- **global_catalog_port** (*optional, str*) – The port number associated to the Global Catalog service running on the directory server.
- **smb_port** (*optional, str*) – The port number associated to the Server Messaging Block (SMB) service running on the directory server.

Returns
The created directory instance.

Return type
dict

Examples

```
>>> tie.directories.create(
...     infrastructure_id=1,
...     name='ExampleServer',
...     ip='172.16.0.1',
...     directory_type='????',
...     dns='company.tld',
...     )
```

delete(*infrastructure_id: int*, *directory_id: int*) → None

Deletes the directory instance.

Parameters

- **infrastructure_id** (*int*) – The infrastructure instance identifier.
- **directory_id** (*int*) – The directory instance identifier.

Return type
None

Examples

```
>>> tie.directories.delete(
...     infrastructure_id=2,
...     directory_id='12'
...     )
```

details(*directory_id: str*) → Dict

Retrieves the details for a specific directory instance.

Parameters
directory_id (*str*) – The directory instance identifier.

Returns
the directory object.

Return type
dict

Examples

```
>>> tie.directories.details(directory_id='1')
```

list() → List[Dict]

Retrieves all directory instances.

Returns
The list of directory objects

Return type
list

Examples

```
>>> tie.directories.list()
```

update(*infrastructure_id: int*, *directory_id: int*, ***kwargs*) → Dict

Updates the directory instance based on infrastrcture_id and directory_id.

Parameters

- **infrastructure_id** (*int*) – The infrastructure instance identifier.
- **directory_id** (*int*) – The directory instance identifier.
- **name** (*optional, str*) – Name of the directory instance.
- **ip** (*optional, str*) – The IP Address of the directory server.
- **directory_type** (*optional, str*) – The directory's type.
- **dns** (*optional, str*) – The DNS domain that this directory is tied to.
- **ldap_port** (*optional, int*) – The port number associated to the LDAP service on the directory server.
- **global_catalog_port** (*optional, str*) – The port number associated to the Global Catalog service running on the directory server.
- **smb_port** (*optional, str*) – The port number associated to the Server Messaging Block (SMB) service running on the directory server.

Returns
The updated directory object.

Return type
dict

Examples

```
>>> tie.directories.update(
...     infrastructure_id=2,
...     directory_id=9,
...     name='updated_new_name'
...     )
```

```
>>> tie.directories.update(
...     infrastructure_id=2,
...     directory_id=9,
...     name='updated_new_name',
...     ldap_port=390
...     )
```

36.15 Email Notifiers

Methods described in this section relate to the email-notifier API. These methods can be accessed at `TenableIE.email_notifiers`.

class EmailNotifiersAPI(*api: APISession*)

create(***kwargs*) → List[Dict]

Create email notifiers

Parameters

- **input_type** (*optional, str*) – The type of input. possible values are `deviances` and `attacks`
- **checkers** (List[int], required_for=[`deviances`]) – The list of checker identifiers.
- **attack_types** (List[int], required_for=[`attacks`]) – The list of attack type identifiers.
- **profiles** (*List[int]*) – The list of profile identifiers.
- **address** (*str*) – The email address.
- **should_notify_on_initial_full_security_check** (*bool*) – Whether alerts should be send when deviances are detected during the initial analysis phase?
- **directories** (*list[str]*) – The list of directory identifiers.
- **criticity_threshold** (*int*) – Threshold at which indicator alerts will be sent.
- **description** (*optional, str*) – The description for notifier.

Returns

The created email notifiers instance objects

Return type

list[dict]

Example

Create email notifier with input_type as deviances

```
>>> tie.email_notifiers.create(
...     input_type='deviances',
...     checkers=[1, 2],
...     profiles=[1],
...     address='test@domain.com',
...     should_notify_on_initial_full_security_check=False,
...     directories=[1],
...     criticity_threshold=100,
...     description='test alert'
...     )
```

Create email notifier with input_type as attacks

```
>>> tie.email_notifiers.create(
...     input_type='attacks',
...     attack_types=[1, 2],
...     profiles=[1],
...     address='test@domain.com',
...     should_notify_on_initial_full_security_check=False,
...     directories=[1],
...     criticity_threshold=100,
...     description='test alert'
...     )
```

delete(*email_notifier_id: str*) → None

Delete an Email-Notifier instance

Parameters
email_notifier_id (*str*) – The profile instance identifier.

Return type
None

Examples

```
>>> tie.email_notifiers.delete(
...     email_notifier_id='1'
...     )
```

details(*email_notifier_id: str*) → Dict

Retrieves the details for a specific email-notifier

Parameters
email_notifier_id (*str*) – The email-notifier instance identifier.

Returns
the email-notifier object.

Return type
dict

Examples

```
>>> tie.email_notifiers.details(
...     email_notifier_id='1'
...     )
```

list() → List[Dict]

Retrieve all email notifiers instances

Returns
The list of email notifier objects

Return type
list

Examples

```
>>> tie.email_notifiers.list()
```

send_test_email(***kwargs*) → None

Send a test Email notification

Parameters

- **input_type** (*optional, str*) – The type of input. possible values are `deviances` and `attacks`
- **checkers** (List[int], required_for=[`deviances`]) – The list of checker identifiers.
- **attack_types** (List[int], required_for=[`attacks`]) – The list of attack type identifiers.
- **profiles** (*List[int]*) – The list of profile identifiers.
- **address** (*str*) – The email address.
- **directories** (*list[str]*) – The list of directory identifiers.
- **criticity_threshold** (*int*) – Threshold at which indicator alerts will be sent.
- **description** (*optional, str*) – The description for notifier.

Return type
None

Examples

Send test email notifier with input_type as deviances

```
>>> tie.email_notifiers.create(
...     input_type='deviances',
...     checkers=[1, 2],
...     profiles=[1],
...     address='test@domain.com',
...     directories=[1],
...     criticity_threshold=100,
```

(continues on next page)

(continued from previous page)

```
...     description='test alert'
...     )
```

Send test email notifier with input_type as attacks

```
>>> tie.email_notifiers.create(
...     input_type='attacks',
...     attack_types=[1, 2],
...     profiles=[1],
...     address='test@domain.com',
...     directories=[1],
...     criticity_threshold=100,
...     description='test alert'
...     )
```

send_test_email_by_id(*email_notifier_id: str*) → None

Send a test Email notification by id

Parameters

email_notifier_id (*str*) – The profile instance identifier.

Return type

None

Examples

```
>>> tie.email_notifiers.send_test_email_by_id(
...     email_notifier_id='1'
...     )
```

update(*email_notifier_id: str*, ***kwargs*) → Dict

Update an existing profile

Parameters

- **email_notifier_id** (*str*) – The email-notifier instance identifier.
- **address** (*optional, str*) – The email address.
- **criticity_threshold** (*optional, int*) – Threshold at which indicator alerts will be sent.
- **directories** (*optional, List[int]*) – The list of directory identifiers.
- **description** (*optional, str*) – The description for notifier.
- **checkers** (*optional, List[int]*) – The list of checker identfiers.
- **attack_types** (*optional, List[int]*) – The list of attack type identifiers.
- **profiles** (*optional, List[int]*) – The list of profile identifiers.
- **input_type** (*optional, str*) – The type of input. possible values are `deviances`, `attacks`
- **should_notify_on_initial_full_security_check** (*bool*) – Whether alerts should be send when deviances are detected during the initial analysis phase?

Returns
The updated email-notifier instance object.

Return type
dict

Examples

```
>>> tie.email_notifiers.update(
...     email_notifier_id='1',
...     input_type='attacks',
...     attack_types=[1, 2]
...     )
```

36.16 Event

Methods described in this section relate to the the event API. These methods can be accessed at `TenableIE.event`.

class EventAPI(*api: APISession*)

details(*event_id: str*, *infrastructure_id: str*, *directory_id: str*) → Dict
Retrieves the details of specific event instance.

Parameters

- **event_id** (*str*) – The event instance identifier.
- **infrastructure_id** (*str*) – The infrastructure instance identifier.
- **directory_id** (*str*) – The directory instance identifier.

Returns
Details of the event object.

Return type
dict

Examples

```
>>> tie.event.details(
...     event_id='1',
...     infrastructure_id='1',
...     directory_id='1'
...     )
```

search_events(*expression: Mapping*, *profile_id: int*, *date_start: str*, *date_end: str*, *directory_ids: List[int]*, ***kwargs*) → List[Dict]
Searches the events.

Parameters

- **expression** (*mapping*) – An object describing a filter for searched items.
- **profile_id** (*int*) – The profile instance identifier.
- **date_start** (*str*) – The starting date from where the events are expected.

- **date_end** (*str*) – The date till which the events are expected.
- **directory_ids** (*List[int]*) – List of directory instance identifiers.
- **order** (*optional, str*) – The desired sorting order of the event identifier. Default is desc

Returns
The search result object.

Return type
list[dict]

Examples

```
>>> tie.event.search_events(
...     expression={'AND': [{'systemOnly': 'True'}]},
...     profile_id=5,
...     date_start='2022-01-05T00:00:00.000Z',
...     date_end='2022-01-12T23:59:59.999Z',
...     directory_ids=[1,2,3],
...     order='asc'
...     )
```

36.17 Infrastructure

Methods described in this section relate to the infrastructure API. These methods can be accessed at `TenableIE.infrastructure`.

class InfrastructureAPI(*api: APISession*)

create(*name: str*, *login: str*, *password: str*) → List[Dict]

Creates a new infrastructure instance with inputs of name, username and password.

Parameters

- **name** (*str*) – The new name for the infrastructure instance.
- **login** (*str*) – The login name for the infrastructure instance.
- **password** (*str*) – The password for the infrastructure instance.

Returns
Newly created infrastructure instance.

Return type
list

Examples

```
>>> tie.infrastructure.create(
...     name='test_user',
...     login='test_user@gmail.com',
...     password='tenable.ad'))
```

delete(*infrastructure_id: str*)

Deletes the particular infrastructure instance.

Parameters
infrastructure_id (*str*) – The infrastructure instance identifier.

Return type
None

Examples

```
>>> tie.infrastructure.delete(infrastructure_id='1')
```

details(*infrastructure_id: str*) → Dict

Gets the details of particular infrastructure instance.

Parameters
infrastructure_id (*str*) – The infrastructure instance identifier.

Returns
Details of particular `infrastructure_id`.

Return type
dict

Examples

```
>>> tie.infrastructure.details(infrastructure_id='1')
```

list() → List[Dict]

Retrieves the list of infrastructures.

Returns
List of infrastructure instances.

Return type
list

Examples

```
>>> tie.infrastructure.list()
```

update(*infrastructure_id: str*, ***kwargs*) → Dict

Updates the infrastructure of the specific infrastructure instance.

Parameters

- **infrastructure_id** (*str*) – The infrastructure instance identifier.
- **name** (*optional, str*) – New name to be updated.
- **login** (*optional, str*) – New login name to be updated.
- **password** (*optional, str*) – New password to be updated.

Returns
Updated infrastructure instance.

Return type
dict

Examples

```
>>> tie.infrastructure.update(
...     infrastructure_id='1',
...     login='updated_login@tenable.com',
...     name='updated_user')
```

36.18 LDAP Configuration

Methods described in this section relate to the ldap configuration API. These methods can be accessed at `TenableIE.ldap_configuration`.

class LDAPConfigurationAPI(*api: APISession*)

details() → Dict

Get LDAP configuration singleton

Returns
The LDAP configuration object

Return type
dict

Examples

```
>>> tie.ldap_configuration.details()
```

update(***kwargs*) → Dict

Update LDAP configuration singleton

Parameters

- **enabled** (*optional, bool*) – Whether the ldap configuration enabled?
- **url** (*optional, str*) – The URL of authentication provider server.
- **search_user_dn** (*optional, str*) – The DN of service account to use to authenticate the user.
- **search_user_password** (*optional, str*) – The password of the service account used for authentication.
- **user_search_base** (*optional, str*) – The DN of the param object for items in LDAP server.
- **user_search_filter** (*optional, str*) – Used to change on what attribute the LDAP query is made to authenticate the user.
- **allowed_groups** (*optional, List[Dict]*) – The LDAP group a member need to be a member of so he can authenticate. The below listed params are expected in allowed groups dict.
- **name** (*required, str*) – The name of group.
- **default_role_ids** (*required, List[int]*) – The list default role identifiers.
- **default_profile_id** (*required, int*) – The default profile identifier.

Returns

The LDAP configuration object

Example

```
>>> tie.ldap_configuration.update(
...     enabled=True,
...     allowed_groups=[{
...         'name': 'group name',
...         'default_role_ids': [1, 2],
...         'default_profile_id': 1
...     }]
... )
```

36.19 License

Methods described in this section relate to the license API. These methods can be accessed at `TenableIE.license`.

class `LicenseAPI`(*api: APISession*)

create(*license: str*) → Dict

Create new license singleton

Parameters
license (*str*) – The license string object.

Returns
The license object

Example

```
>>> tie.license.create(
...     license='license'
... )
```

details() → Dict

Get license singleton

Returns
The license object

Return type
dict

Examples

```
>>> tie.license.details()
```

36.20 Lockout Policy

Methods described in this section relate to the lockout policy API. These methods can be accessed at `TenableIE.lockout_policy`.

class `LockoutPolicyAPI`(*api: APISession*)

details() → Dict

Get the lockout policy

Returns
The lockout policy object

Return type
dict

Examples

```
>>> tie.lockout_policy.details()
```

update(***kwargs*) → None

Update the lockout policy

Parameters

- **enabled** (*optional, bool*) – Whether the lockout policy enabled?
- **lockout_duration** (*optional, int*) – The time duration for which user will be locked out after several failed login attempts.
- **failed_attempt_threshold** (*optional, int*) – The number of failed login attempts to trigger lockout.
- **failed_attempt_period** (*optional, int*) – The time to wait before the login attempts count is reseted.

Returns
None

Example

```
>>> tie.lockout_policy.update(enabled=True)
```

36.21 Preference

Methods described in this section relate to the preferences API. These methods can be accessed at `TenableIE.preference`.

class PreferenceAPI(*api: APISession*)

details() → Dict

Get the user's preferences

Returns
The user's preferences object

Return type
dict

Examples

```
>>> tie.preference.details()
```

update(***kwargs*) → Dict

Update the user's preferences

Parameters

- **language** (*optional, str*) – The language of product for the current user.

- **preferred_profile_id** (*optional, int*) – The profile identifier to use after login for the current user.

Returns
The user's preferences object

Example

```
>>> tie.preference.update(
...     language='en',
...     preferred_profile_id=1
... )
```

36.22 Profiles

Methods described in this section relate to the profiles API. These methods can be accessed at `TenableIE.profiles`.

class ProfilesAPI(*api: APISession*)

commit(*profile_id: str*) → None
Commits change of the related profile

Parameters
profile_id (*str*) – The profile instance identifier.

Returns
None

Example

```
>>> tie.profiles.commit('1')
```

copy_profile(*from_id: str*, *name: str*, *directories: List[int]*) → Dict
Creates a new profile from another profile

Parameters

- **from_id** (*str*) – The profile instance identifier user wants to copy.
- **name** (*str*) – The name of new profile.
- **directories** (*List[int]*) – The list of directory ids.

Returns
The copied role object.

Return type
dict

Examples

```
>>> tie.profiles.copy_profile(
...     from_id='1',
...     name='Copied name',
...     directories=[1, 2]
...     )
```

create(*name: str*, *directories: List[int]*) → List[Dict]

Create a profile

Parameters

- **name** (*str*) – The name of new profile.
- **directories** (*List[int]*) – The list of directory identifiers.

Returns
The created profile objects

Return type
list[dict]

Example

```
>>> tie.profiles.create(
...     name='ExampleProfile',
...     directories=[1, 2]
...     )
```

delete(*profile_id: str*) → None

Delete an existing profile

Parameters
profile_id (*str*) – The profile instance identifier.

Return type
None

Examples

```
>>> tie.profiles.delete(profile_id='1')
```

details(*profile_id: str*) → Dict

Retrieves the details for a specific profile

Parameters
profile_id (*str*) – The profile instance identifier.

Returns
The profile object.

Return type
dict

Examples

```
>>> tie.profiles.details('1')
```

list() → List[Dict]

Retrieve all profiles

Returns
The list of profile objects

Return type
list[dict]

Examples

```
>>> tie.profiles.list()
```

unstage(*profile_id: str*) → None

Unstages changes of the related profile

Parameters
profile_id (*str*) – The profile instance identifier.

Returns
None

Example

```
>>> tie.profiles.unstage('1')
```

update(*profile_id: str*, ***kwargs*) → Dict

Update an existing profile

Parameters

- **profile_id** (*str*) – The profile instance identifier.
- **name** (*optional, str*) – The name of profile.
- **deleted** (*optional, bool*) – is the profile deleted?
- **directories** (*optional, List[int]*) – The list of directory ids.

Returns
The updated profile object.

Return type
dict

Examples

```
>>> tie.profiles.update(
...     profile_id='1',
...     name='EDITED'
...     )
```

36.23 Reason

Methods described in this section relate to the reason API. These methods can be accessed at `TenableIE.reason`.

class ReasonAPI(*api: APISession*)

details(*reason_id: str*) → Dict

Retrieves the details of the reason based on reason_id

Parameters
reason_id (*str*) – The reason instance identifier.

Returns
Details of the reason object .

Return type
dict

Examples

```
>>> tie.reason.details(reason_id='1')
```

list() → List[Dict]

Retrieves the list of reason instances.

Returns
The list of reason instances.

Return type
list

Examples

```
>>> tie.reason.list()
```

list_by_checker(*profile_id: str*, *checker_id: str*) → List[Dict]

Retrieves the details of the reason based on profile_id and checker_id.

Parameters

- **profile_id** (*str*) – The profile instance identifier
- **checker_id** (*str*) – The checker instance identifier

Returns
Details of the reason object.

Return type
list

Examples

```
>>> tie.reason.list_by_checker(
...     profile_id='1',
...     checker_id='1'
...     )
```

list_by_directory_and_event(*profile_id: str*, *infrastructure_id: str*, *directory_id: str*, *event_id: str*) → List[Dict]

Retrieves the details of the reason object based on profile_id, directory_id and event_id.

Parameters

- **profile_id** (*str*) – The profile instance identifier.
- **infrastructure_id** (*str*) – The infrastructure instance identifier.
- **directory_id** (*str*) – The directory instance identifier.
- **event_id** (*str*) – The event instance identifier.

Returns
Details of the reason object.

Return type
list

Examples

```
>>> tie.reason.list_by_directory_and_event(
...     profile_id='1',
...     infrastructure_id='1',
...     directory_id='1',
...     event_id='1'
...     )
```

36.24 Roles

Methods described in this section relate to the roles API. These methods can be accessed at `TenableIE.roles`.

class RolesAPI(*api: APISession*)

copy_role(*from_id: str*, *name: str*) → Dict

Creates a new role from another role

Parameters

- **from_id** (*str*) – The role instance identifier user wants to copy.
- **name** (*str*) – The name of new role.

Returns
the copied role object.

Return type
dict

Examples

```
>>> tie.roles.copy_role(
...     from_id='1',
...     name='Copied name'
...     )
```

create(*name: str*, *description: int*) → List[Dict]

Create a new role

Parameters

- **name** (*str*) – The name of role.
- **description** (*str*) – The description of role.

Returns
The created role object.

Return type
list[dict]

Examples

```
>>> tie.roles.create(
...     name='Admin',
...     description="all privileges"
...     )
```

default_roles() → List[Dict]

Return the default roles for user creation

Returns
The default roles object.

Return type
list[dict]

Examples

```
>>> tie.roles.default_roles()
```

delete(*role_id: str*) → None

Delete an existing role

Parameters
role_id (*str*) – The role instance identifier.

Return type
None

Examples

```
>>> tie.roles.delete(
...     role_id='1',
...     )
```

details(*role_id: str*) → Dict

Retrieves the details of a specific role.

Parameters
role_id (*str*) – The role instance identifier.

Returns
the role object.

Return type
dict

Examples

```
>>> tie.roles.details(
...     role_id='1'
...     )
```

list() → List[Dict]

Retrieve all roles

Returns
The list of roles objects

Return type
list[dict]

Examples

```
>>> tie.roles.list()
```

replace_role_permissions(*role_id: str*, *permissions: List[Dict]*) → Dict

Replace permission list for a role

Parameters

- **role_id** (*str*) – The role instance identifier.
- **permissions** (*List[Dict]*) – The list of permissions dictionaries. Below are the values expected in dictionaries
- **entity_name** (*str*) – The name of entity.
- **action** (*str*) – The code of action to perform.
- **entity_ids** (*List[int]*) – The list of entity identifiers.
- **dynamic_id** (*optional, str*) – The dynamicId to use associated with the action.

Returns
the update permissions role object.

Return type
dict

Examples

```
>>> tie.roles.replace_role_permissions(
...     role_id='1',
...     permissions=[{
...         'entity_name':'dashboard',
...         'action':'action',
...         'entity_ids':[1, 2],
...         'dynamic_id': None
...     }]
... )
```

update(*role_id: str*, ***kwargs*) → Dict

Update an existing role

Parameters

- **role_id** (*str*) – The role instance identifier.
- **name** (*optional, str*) – The name of role.
- **description** (*optional, str*) – The description of role.

Returns
The updated widget object.

Return type
dict

Examples

```
>>> tie.roles.update(
...     role_id='1',
...     name='Basic'
...     )
```

36.25 SAML Configuration

Methods described in this section relate to the SAML Configuration API. These methods can be accessed at `TenableIE.saml_configuration`.

class SAMLConfigurationAPI(*api: APISession*)

details() → Dict

Retrieves the details of the SAML-configuration singleton.

Returns
The details of saml configuration singleton.

Return type
dict

Examples

```
>>> tie.saml_configuration.details()
```

generate_saml_certificate() → Dict

Generates a SAML certificate.

Returns
Generated certificate.

Return type
dict

Examples

```
>>> tie.saml_configuration.generate_saml_certificate()
```

update(***kwargs*) → Dict

Updates the SAML-configuration.

Parameters

- **enabled** (*optional, bool*) – Whether the SAML configuration is enabled or not.
- **provider_login_url** (*optional, str*) – The URL of the identity provider to reach for SAML authentication.
- **signature_certificate** (*optional, str*) – The certificate used to sign the SAML authentication.
- **activate_created_users** (*optional, bool*) – Whether the created users through SAML authentication should be activated. If false, created users will be disabled until an admin comes and activate them.
- **allowed_groups** (*optional, List[Dict]*) – The group names from the identity provider whose members are allowed to use tenable.ie. The below listed params are expected in allowed_groups dict.
- **name** (*required, str*) – The name of SAML Configuration.
- **default_profile_id** (*required, int*) – The default profile instance identifier of SAML Configuration.
- **default_role_ids** (*required, list(int)*) – The default role instance identifier of SAML Configuration.

Returns
The updated saml-configuration.

Return type
dict

Examples

```
>>> tie.saml_configuration.update(
...     enabled=True,
...     allowed_groups=[{
...         'name': 'updated_name',
...         'default_profile_id': 1,
...         'default_role_ids': [1, 2]
...     }]
...     )
```

36.26 Score

Methods described in this section relate to the score API. These methods can be accessed at `TenableIE.score`.

class ScoreAPI(*api: APISession*)

list(*profile_id: str*, ***kwargs*) → List[Dict]

Get the list of directories score by profile.

Parameters

- **Option-1** –

  profile_id (str):
  The profile instance identifier.

  directory_ids (optional, List(int)):
  The list of directory_ids.

  checker_ids (optional, List(int)):
  The list of checker_ids.

  reason_ids (optional, List(int)):
  The list of reason_ids.

- **Option-2** –

  profile_id (str):
  The profile instance identifier.

  directory_ids (optional, str):
  The directory_id instance identifier.

  checker_ids (optional, str):
  The checker_id instance identifier.

  reason_ids (optional, str):
  The reason_id instance identifier.

Returns
List of scores of different directories in the instance.

Return type
list

Examples

With single directory_ids, checker_ids, reason_ids

```
>>> tie.score.list(profile_id='1',
...     directory_ids='3',
...     checker_ids='1',
...     reason_ids='1')
```

With multiple directory_ids, checker_ids, reason_ids

```
>>> tie.score.list(profile_id='1',
...     directory_ids=[1, 2, 3],
...     checker_ids=[1, 2, 3],
...     reason_ids=[1, 2, 3])
```

36.27 Syslog

Methods described in this section relate to the syslog API. These methods can be accessed at `TenableIE.syslog`.

class SyslogAPI(*api: APISession*)

create(***kwargs*) → List[Dict]

Creates a syslog object.

Parameters

- **profiles** (*List[int]*) – The list of profile identifiers.
- **checkers** (List[int], required_for=[deviances]) – The list of checker identifiers.
- **input_type** (*str*) – The type of input to send through the syslog. Allowed values are deviances or ad_object_changes or attacks.
- **description** (*optional, str*) – The description for syslog object.
- **attack_types** (List[int], required_for=[attacks]) – Filter on the types of attack that will be sent if input type is attack.
- **ip** (*str*) – The collector ip address or hostname of the syslog.
- **port** (*int*) – The port number of the collector ip address.
- **protocol** (*str*) – The protocol used by the collector. Allowed values are TCP and UDP.
- **tls** (bool, required_for=[TCP]) – Whether the configured syslog should connect using TLS. By default and if UDP is selected as the protocol, tls is False.
- **criticity_threshold** (int, required_for=[attacks, deviances]) – Threshold at which indicator alerts will be sent.
- **directories** (*list[str]*) – The list of directory identifiers.
- **should_notify_on_initial_full_security_check** (*bool*) – Whether alerts should be sent when deviances are detected during the initial analysis phase.
- **filter_expression** (*optional, mapping*) – An object describing a filter for searched items.

Returns
The created syslog object.

Return type
list[dict]

Example

Create a syslog object with input_type as `ad_object_changes`.

```
>>> tie.syslog.create(
...     description='test_syslog',
...     input_type="ad_object_changes",
...     ip='127.0.0.1',
...     port=8888,
...     protocol="TCP",
...     tls=False,
...     directories=[2],
...     should_notify_on_initial_full_security_check=False,
...     filter_expression={'OR': [{'systemOnly': 'True'}]}
...     )
```

Create syslog object with input_type as `attacks`

```
>>> tie.syslog.create(
...     description='test_syslog',
...     input_type="attacks",
...     profiles=[1],
...     attack_types=[1],
...     ip='127.0.0.1',
...     port=8888,
...     protocol="TCP",
...     tls=True,
...     criticity_threshold=55,
...     directories=[2],
...     should_notify_on_initial_full_security_check=False,
...     filter_expression={'OR': [{'systemOnly': 'True'}]}
...     )
```

Create syslog object with input_type as `deviances`

```
>>> tie.syslog.create(
...     description='test_syslog',
...     input_type="deviances",
...     profiles=[1],
...     checkers=[1],
...     ip='127.0.0.1',
...     port=8888,
...     protocol="TCP",
...     tls=True,
...     criticity_threshold=55,
...     directories=[2],
...     should_notify_on_initial_full_security_check=False,
```

(continues on next page)

(continued from previous page)

```
...     filter_expression={'OR': [{'systemOnly': 'True'}]}
...     )
```

Create syslog object with protocol as `UDP` without passing `tls`

```
>>> tie.syslog.create(
...     description='test_syslog',
...     input_type="deviances",
...     profiles=[1],
...     checkers=[1],
...     ip='127.0.0.1',
...     port=8888,
...     protocol="UDP",
...     criticity_threshold=55,
...     directories=[2],
...     should_notify_on_initial_full_security_check=False,
...     filter_expression={'OR': [{'systemOnly': 'True'}]})
```

delete(*syslog_id: str*) → None

Deletes the syslog object of given syslog identifier.

Parameters
syslog_id (*str*) – The syslog object identifier.

Return type
None

Examples

```
>>> tie.syslog.delete(syslog_id='1')
```

details(*syslog_id: str*) → Dict

Returns the details of the syslog object of the given syslog identifier.

Parameters
syslog_id (*str*) – The syslog object identifier.

Returns
The details of the syslog object.

Return type
dict

Examples

```
>>> tie.syslog.details(syslog_id='1')
```

list() → List[Dict]

Returns all the syslog objects.

Returns
The list of syslog objects.

Return type
list

Examples

```
>>> tie.syslog.list()
```

send_notification(***kwargs*) → None

Send a test syslog notification.

Parameters

- **profiles** (*List[int]*) – The list of profile identifiers.
- **checkers** (optional, List[int], required_for=[deviances]) – The list of checker identifiers.
- **input_type** (*str*) – The type of input to send through the syslog. Allowed values for deviances or ad_object_changes or attacks.
- **description** (*optional, str*) – The description for syslog object.
- **attack_types** (optional, List[int], required_for=[attacks]) – Filter on the types of attack that will be sent if input type is attack.
- **ip** (*str*) – The collector ip address or hostname of the syslog.
- **port** (*int*) – The port number of the collector ip address.
- **protocol** (*str*) – The protocol used by the collector. Allowed values are TCP and UDP.
- **tls** (bool, required if protocol is tcp) – Whether the configured syslog should connect using TLS. By default and if UDP is selected as the protocol, tls is False.
- **criticity_threshold** (int, required_for=[attacks, deviances]) – Threshold at which indicator alerts will be sent.
- **directories** (*list[str]*) – The list of directory identifiers.

Return type
None

Examples

Send test syslog with input_type as ad_object_changes.

```
>>> tie.syslog.send_notification(
...     input_type="ad_object_changes",
...     ip='127.0.0.1',
...     port=8888,
...     protocol="TCP",
...     tls=True,
...     directories=[2],
...     )
```

Send test syslog with input_type as deviances.

```
>>> tie.syslog.send_notification(
...     checkers=[1],
...     profiles=[1],
...     input_type="deviances",
...     ip='127.0.0.1',
...     port=8888,
...     protocol="TCP",
...     tls=True,
...     criticity_threshold=10,
...     directories=[2],
...     )
```

Send test syslog with input_type as `attacks`

```
>>> tie.syslog.send_notification(
...     profiles=[1],
...     input_type="attacks",
...     attack_types=[1],
...     ip='127.0.0.1',
...     port=8888,
...     protocol="TCP",
...     tls=True,
...     criticity_threshold=10,
...     directories=[2],
...     )
```

send_syslog_notification_by_id(*syslog_id: str*) → None

Send a test syslog notification by syslog identifier.

Parameters
syslog_id (*str*) – The syslog object identifier.

Return type
None

Examples

```
>>> tie.syslog.send_syslog_notification_by_id(syslog_id='1')
```

update(*syslog_id: str*, ***kwargs*) → Dict

Updates the existing syslog object.

Parameters

- **syslog_id** (*str*) – The syslog object identifier.
- **profiles** (*optional, List[int]*) – The list of profile identifiers.
- **checkers** (optional, List[int], required_for=[deviances]) – The list of checker identifiers.
- **input_type** (*optional, str*) – The type of input to send through the syslog. Allowed values for `deviances` or `ad_object_changes` or `attacks`.
- **description** (*optional, str*) – The description for syslog object.

- **attack_types** (optional, List[int], required_for=[attacks]) – Filter on the types of attack that will be sent if input type is attack.
- **ip** (*optional, str*) – The collector ip address or hostname of the syslog.
- **port** (*optional, int*) – The port number of the collector ip address.
- **protocol** (*optional, str*) – The protocol used by the collector. Allowed values are TCP and UDP.
- **tls** (optional, bool, required if protocol is tcp) – Whether the configured syslog should connect using TLS. By default and if UDP is selected as the protocol, tls is False.
- **criticity_threshold** (optional, int, required_for=[attacks, deviances]) – Threshold at which indicator alerts will be sent.
- **directories** (*optional, list[str]*) – The list of directory identifiers.
- **should_notify_on_initial_full_security_check** (*bool*) – Whether alerts should be sent when deviances are detected during the initial analysis phase.
- **filter_expression** (*optional, mapping*) – An object describing a filter for searched items.

Returns
The updated syslog object.

Return type
dict

Examples

```
>>> tie.syslog.update(
...     syslog_id='1',
...     filter_expression={'OR': [{'systemOnly': 'True'}]}
...     )
```

36.28 Topology

Methods described in this section relate to the topology API. These methods can be accessed at `TenableIE.topology`.

class TopologyAPI(*api: APISession*)

details(*profile_id: str*) → Dict

Gets the representation of AD topology.

Parameters
profile_id (*str*) – The profile instance identifier.

Returns
Representation of AD topology.

Return type
dict

Examples

```
>>> tie.topology.details(profile_id='1')
```

36.29 Users

Methods described in this section relate to the users API. These methods can be accessed at `TenableIE.users`.

class `UsersAPI`(*api: APISession*)

change_password(*old_password: str*, *new_password: str*) → None

Update a user password

Parameters

- **old_password** (*str*) – old password of user.
- **new_password** (*str*) – new password of user.

Return type
None

Examples

```
>>> tie.users.change_password(
...     old_password='old_password',
...     new_password='new_password'
...     )
```

create(*name: str*, *email: str*, *password: str*, ***kwargs*) → List[Dict]

Create users

Parameters

- **name** (*str*) – The name of new user.
- **email** (*str*) – The email address of the user.
- **password** (*str*) – The password for the new user.
- **surname** (*optional, str*) – The surname of new user.
- **department** (*optional, str*) – The department of user.
- **biography** (*optional, str*) – The biography of user.
- **active** (*optional, bool*) – is the user active?
- **picture** (*optional, List[int]*) – The list of picture numbers

Returns
The created user objects

Return type
list[dict]

Example

```
>>> tie.users.create(
...     name='username',
...     email='test@domain.com',
...     password='user_password',
...     active=True
...     )
```

create_password(*email: str*) → None

Sends an email to create new password

Parameters
email (*str*) – The email address of the user.

Return type
None

Examples

```
>>> tie.users.create_password(email='test@domain.com')
```

delete(*user_id: str*) → None

Delete an existing user

Parameters
user_id (*str*) – The user instance identifier.

Return type
None

Examples

```
>>> tie.users.delete(user_id='1')
```

details(*user_id: str*) → Dict

Retrieves the details for a specific user

Parameters
user_id (*str*) – The user instance identifier.

Returns
the user object.

Return type
dict

Examples

```
>>> tie.users.details('1')
```

info() → Dict

Gets user information

Returns
The user info object

Return type
dict

Example

```
>>> tie.users.info()
```

list() → List[Dict]

Retrieve all users

Returns
The list of users objects

Return type
list

Examples

```
>>> tie.users.list()
```

retrieve_password(*token: str*, *new_password: str*) → None

Retrieves a user password

Parameters

- **token** (*str*) – user token.
- **new_password** (*str*) – new password for user.

Return type
None

Examples

```
>>> tie.users.retrieve_password(
...     token='token',
...     new_password='new_password'
...     )
```

update(*user_id: str*, ***kwargs*) → Dict

Update an existing user

Parameters

- **user_id** (*str*) – The user instance identifier.

- **name** (*optional, str*) – The name of new user.
- **email** (*optional, str*) – The email address of the user.
- **password** (*optional, str*) – The password for the new user.
- **surname** (*optional, str*) – The surname of new user.
- **department** (*optional, str*) – The department of user.
- **biography** (*optional, str*) – The biography of user.
- **active** (*optional, bool*) – is the user active?
- **picture** (*optional, List[int]*) – The list of picture numbers

Returns
The updated user object.

Return type
dict

Examples

```
>>> tie.users.update(
...     user_id='1',
...     name='EDITED'
...     )
```

update_user_roles(*user_id: str*, *roles: List[int]*) → Dict

Replace role list for user

Parameters

- **user_id** (*str*) – The user instance identifier.
- **roles** (*List[int]*) – The list of user role identifiers.

Returns
updated user roles object

Return type
dict

Examples

```
>>> tie.users.update_user_roles(
...     user_id='1',
...     roles=[1, 2, 3]
...     )
```

36.30 Widget

Methods described in this section relate to the widget API. These methods can be accessed at `TenableIE.widgets`.

class `WidgetsAPI`(*api: APISession*)

create(*dashboard_id: int*, *pos_x: int*, *pos_y: int*, *width: int*, *height: int*, *title: str*) → List[Dict]

Creates a new widget.

Parameters

- **dashboard_id** (*int*) – The dashboard instance identifier.
- **pos_x** (*int*) – x-axis position for widget.
- **pos_y** (*int*) – y-axis position for widget.
- **width** (*int*) – width of widget.
- **height** (*int*) – height of widget.
- **title** (*str*) – title for widget.

Returns

The created widget object.

Return type

list[dict]

Examples

```
>>> tie.widgets.create(
...     dashboard_id=1,
...     pos_x=1,
...     pos_y=1,
...     width=2,
...     height=2,
...     title='ExampleWidget',
...     )
```

define_widget_options(*dashboard_id: int*, *widget_id: int*, *chart_type: str*, *series: List[Dict]*) → None

Defines the widget option.

Parameters

- **dashboard_id** (*int*) – The dashboard instance identifier.
- **widget_id** (*int*) – The dashboard instance identifier.
- **chart_type** (*str*) –

  The type of chart for widget. possible options
  are `BigNumber`, `LineChart`, `BarChart`, `SecurityCompliance` and `StepChart`.
- **series** (*list*) – Additional keywords passed will be added to the series list of dicts within the API call.

Return type

None

Examples

```
>>> tie.widgets.define_widget_options(
...     dashboard_id=1,
...     widget_id=1,
...     chart_type='BigNumber'
...     series=[
...     {
...         'dataOptions': {
...             'type': 'User',
...             'duration': 1,
...             'directoryIds': [1, 2, 3],
...             'active': True
...         },
...         'displayOptions': {
...             'label': 'label'
...         }
...     }
...     ]
...     )
```

delete(*dashboard_id: int*, *widget_id: int*) → None

Deletes an existing widget.

Parameters

- **dashboard_id** (*int*) – The dashboard instance identifier.
- **widget_id** (*int*) – The widget instance identifier.

Return type
None

Examples

```
>>> tie.widgets.delete(
...     dashboard_id=1,
...     widget_id=1
...     )
```

details(*dashboard_id: int*, *widget_id: int*) → Dict

Retrieves the details for a specific widget.

Parameters

- **dashboard_id** (*int*) – The dashboard instance identifier.
- **widget_id** (*int*) – The widget instance identifier

Returns
The widget object.

Return type
dict

Examples

```
>>> tie.widget.details(dashboard_id=1, widget_id=1)
```

list(*dashboard_id: int*) → List[Dict]

Retrieves all the widgets.

Parameters
dashboard_id (*int*) – The dashboard instance identifier.

Returns
The list of widget objects.

Return type
list

Examples

```
>>> tie.widgets.list(dashboard_id=13)
```

update(*dashboard_id: int*, *widget_id: int*, ***kwargs*) → Dict

Updates an existing widget.

Parameters

- **dashboard_id** (*int*) – The dashboard instance identifier.
- **widget_id** (*int*) – The dashboard instance identifier.
- **pos_x** (*optional, int*) – x-axis position for widget.
- **pos_y** (*optional, int*) – y-axis position for widget.
- **width** (*optional, int*) – width of widget.
- **height** (*optional, int*) – height of widget.
- **title** (*optional, str*) – title for widget.

Returns
The updated widget object.

Return type
dict

Examples

```
>>> tie.widgets.update(
...     dashboard_id=1,
...     widget_id=1,
...     pos_x=1,
...     pos_y=1,
...     width=3,
...     height=3,
...     title='EditedWidget'
...     )
```

widget_options_details(*dashboard_id: int*, *widget_id: int*) → Dict

Gets the details of widget options.

Parameters

- **dashboard_id** (*int*) – The dashboard instance identifier.
- **widget_id** (*int*) – The dashboard instance identifier.

Returns

The widget option object.

Return type

dict

Examples

```
>>> tie.widgets.widget_options_details(
...     widget_id=1,
...     dashboard_id=1
...     )
```

CHAPTER

THIRTYSEVEN

TENABLE ATTACK PATH ANALYSIS

This package covers the Tenable APA.

class TenableAPA(*access_key: str | None = None, secret_key: str | None = None, **kwargs*)

The Tenable Attack Path Analysis object is the primary interaction point for users to interface with Tenable Attack Path Analysis via the pyTenable library. All the API endpoint classes that have been written will be grafted onto this class.

Examples

Basic Example:

```
>>> from tenable.apa import TenableAPA
>>> tapa = TenableAPA('ACCESS_KEY', 'SECRET_KEY')
```

Example with proper identification:

```
>>> tapa = TenableAPA('ACCESS_KEY', 'SECRET_KEY',
>>>     vendor='Company Name',
>>>     product='My Awesome Widget',
>>>     build='1.0.0')
```

property findings

The interface object for the *Tenable Attack Path Analysis APA Findings APIs*.

property vectors

The interface object for the *Tenable Attack Path Analysis APA Findings APIs*.

37.1 Findings

Methods described in this section relate to the findings API. These methods can be accessed at `TenableAPA.findings`.

class FindingsAPI(*api: APISession*)

list(*page_number: int | None = None, next_token: str | None = None, limit: int = 50, filter: dict | None = None, sort_filed: str | None = None, sort_order: str | None = None, return_iterator=True*) → FindingIterator | FindingsPageSchema

Retrieve findings

Args:

page_number (optional, int):
For offset-based pagination, the requested page to retrieve. If this parameter is omitted, Tenable uses the default value of 1.

next_token (optional, str):
For cusrsor-based pagination, the cursor position for the next page. For the initial request, don't populate. For subsequent requests, set this parameter to the value found in the next property of the previous response. When getting null without specify a page number it means there are no more pages.

limit (optional, int):
The number of records to retrieve. If this parameter is omitted, Tenable uses the default value of 50. The maximum number of events that can be retrieved is 10,000. For example: limit=10000.

filter (optional, dict):
A document as defined by Tenable APA online documentation. Filters to allow the user to get to a specific subset of Findings. For a more detailed listing of what filters are available, please refer to the API documentation linked above, however some examples are as such:

- `{"operator":"==", "key":"state", "value":"open"}`
- `{"operator":">", "key":"last_updated_at", "value":"2024-05-30T12:28:11.528118"}`

sort_filed (optional, str):
The field you want to use to sort the results by. Accepted values are `last_updated_at`, `state`, `vectorCount`, `status`, `name`, `procedureName`, `priority`, and `mitre_id`.

sort_order (optional, str):
The sort order Accepted values are `desc` or `acs`

return_iterator (bool, optional):
Should we return the response instead of iterable?

Returns
List of findings records

Return type
`FindingIterator`

Examples:

```
>>> findings = tapa.findings.list()
>>> for f in findings:
...     pprint(f)
```

Examples:

```
>>> tapa.findings.list(
...     limit='10',
...     sort_filed='last_updated_at',
...     sort_order='desc',
...     filter='value',
...     return_iterator=False
...     )
```

search_attack_techniques(*filters: dict | None = None*, *offset: int | None = None*, *limit: int | None = None*, *sort: str | None = None*, *exclude_resolved: bool = True*, *return_iterator: bool = True*) → AttackTechniqueIterator | dict

Search attack techniques

Parameters

- **filters** (*optional, dict*) – Filter conditions for searching attack techniques. Supports complex filtering with AND/OR operators. Examples: - `{"operator":"==", "property":"priority", "value":"high"}` - `{"operator":"and", "value":[{"operator":"==", "property":"priority", "value":"high"}, {"operator":"==", "property":"state", "value":"open"}]}`
- **offset** (*optional, int*) – Number of items to skip for pagination. If omitted, the default value is 0.
- **limit** (*optional, int*) – Number of items per page. If omitted, the default value is 1000. The minimum value is 100 and the maximum value is 10000.
- **sort** (*optional, str*) – Sort parameter in format "{sort_field}:{sort_order}" with multiple variations: - Ascending: "asc", "ASC", "ascending", "ASCENDING", "Ascending" - Descending: "desc", "DESC", "descending", "DESCENDING", "Descending" - Examples: "priority:desc", "name:asc", "last_updated_at:ASCENDING", "state:DESCENDING"

  Supported sort fields: last_updated_at, priority, mitre_id, name, procedureName, status, state, vectorCount
- **exclude_resolved** (*bool, optional*) – When True (default), excludes techniques with status 'done' or 'accepted' and state 'archive'. Set to False to include all techniques.
- **return_iterator** (*bool, optional*) – Should we return the response instead of iterable?

Returns
List of attack technique records

Return type
`FindingIterator` or `dict`

Examples

```
>>> attack_techniques = tapa.findings.search_attack_techniques()
>>> for technique in attack_techniques:
...     pprint(technique)
```

Examples

```
>>> tapa.findings.search_attack_techniques(
...     limit=100,
...     sort='priority:desc',
...     filters={'operator': '==', 'property': 'priority', 'value': 'high'},
...     return_iterator=False
... )
```

Include resolved techniques:

```
>>> tapa.findings.search_attack_techniques(
...     exclude_resolved=False,
...     return_iterator=False
... )
```

37.2 Vectors

Methods described in this section relate to the vectors API. These methods can be accessed at `TenableAPA.vectors`.

class VectorsAPI(*api: APISession*)

list(*page_number: int | None = None, limit: int = 10, filter: dict | None = None, sort_field: str | None = None, sort_order: str | None = None, run_ai_summarization: bool | None = None, return_iterator=True*) → VectorIterator | VectorsPageSchema

Retrieve vectors

Args:

page_number (optional, int):
For offset-based pagination, the requested page to retrieve. If this parameter is omitted, Tenable uses the default value of 1.

limit (optional, int):
The number of records to retrieve. If this parameter is omitted, Tenable uses the default value of 25. The maximum number of events that can be retrieved is 25. For example: limit=25.

filter (optional, dict):
A document as defined by Tenable APA online documentation. Filters to allow the user to get to a specific subset of Findings. For a more detailed listing of what filters are available, please refer to the API documentation linked above, however some examples are as such:

- `{"operator":"==", "key":"name", "value":"nice name"}`
- `{"operator":">", "key":"critical_asset", "value":  10}`

sort_field (optional, str):
The field you want to use to sort the results by. Accepted values are `name`, `priority`

run_ai_summarization (optional, bool):
Indicates whether or not to run the AI summarization for missing paths. Note that enabling the AI summarization results in slower response times. Tenable uses the default value of false.

return_iterator (optional, bool):
Should we return the response instead of iterable?

Returns
List of vectors records

Return type

`VectorsIterator`

Examples:

```
>>> vectors = tapa.vectors.list()
>>> for f in vectors:
...     pprint(f)
```

Examples:

```
>>> tapa.vectors.list(
...     limit='10',
...     sort_field='name',
...     sort_order='desc',
...     filter={"operator":"==", "key":"name", "value":"nice␣
↪name"},
...     return_iterator=False
...     )
```

CHAPTER

THIRTYEIGHT

TENABLE ATTACK SURFACE MANAGEMENT

This package covers the Tenable ASM application.

class TenableASM(***kwargs*)

The TenableASM class is the primary interaction point for users to interface with Tenable Attack Surface Management via the pyTenable library. All the API endpoint classes that wrap the various aspects of ASM will be attached to this base class.

Parameters

- **api_key** (*str, optional*) – The user's API key to interface into Tenable ASM. If the key isn't specified, then the library will attempt to read the environment variable `TASM_API_KEY` to get the key.
- **url** (*str, optional*) – The base URL that the paths will be appended onto. If the url isn't specified, then the library will attempt to read the environment variable `TASM_URL`.
- **retries** (*int, optional*) – The number of retries to make before failing a request. The default is `5`.
- **backoff** (*float, optional*) – If a 429 response is returned, how much do we want to backoff if the response didn't send a Retry-After header. The default backoff is 1 second.
- **vendor** (*str, optional*) – The vendor name for the User-Agent string.
- **product** (*str, optional*) – The product name for the User-Agent string.
- **build** (*str, optional*) – The version or build identifier for the User-Agent string.
- **timeout** (*int, optional*) – The connection timeout parameter informing the library how long to wait in seconds for a stalled response before terminating the connection. If unspecified, the default is 120 seconds.

Examples

Basic example:

```
>>> from tenable.asm import TenableASM
>>> tasm = TenableASM(url='https://asm.cloud.tenable.com',
...                   api_key='abcdef1234567890'
...                   )
```

Another example with proper identification:

```
>>> tasm = TenableASM(url='https://asm.cloud.tenable.com',
...                   api_key='abcdef1234567890',
...                   vendor='Company Name',
...                   product='My Awesome Widget',
...                   build='1.0.0'
...                   )
```

Yet another example thats leveraging the `TASM_API_KEY` and `TASM_URL` environment variables:

```
>>> os.environ['TASM_URL'] = 'https://asm.cloud.tenable.com'
>>> os.environ['TASM_API_KEY'] = 'abcdef1234567890'
>>> tasm = TenableASM(vendor='Company Name',
...                   product='My Awesome Widget',
...                   build='1.0.0'
...                   )
```

property inventory

The interface object for the *Tenable ASM Inventory API*

property smart_folders

The interface object for the *Tenable ASM Smart Folders API*

38.1 Inventory

Methods described in this section relate to the inventory API and can be accessed at `TenableASM.inventory`.

class InventoryAPI(*api: APISession*)

list(**search: Tuple[str, str, str], columns: List[str] | None = None, size: int = 1000, sort_field: str | None = None, sort_asc: bool = True, inventory: bool = False*) → *InventoryIterator*

Lists the assets in the inventory

Parameters

- ***search** (*tuple[str, str, str], optional*) – A 3-part search tuple detailing what to search for from the ASM dataset. For example: `('bd.original_hostname', 'ends with', '.com')`
- **columns** (*list[str], optional*) – The list of columns to return in the response.
- **size** (*int, optional*) – The number of records to return with each page from the API. Must be an integer between *1* and *10000*.
- **sort_field** (*str, optional*) – What field should the results be worted by?
- **sort_asc** (*bool*) – How should the results be sorted? `True` specifies ascending sort, whereas `False` refers to descending.

Example

```
>>> for item in asm.inventory.list():
...     print(item)
```

class InventoryIterator(*api*, ***kw*)

Asset inventory iterator

38.2 Smart Folders

Methods described in this section relate to the smart folders API and can be accessed at `TenableASM.smart_folders`.

class SmartFoldersAPI(*api: APISession*)

list() → List[Dict[str, Any]]

Returns the list of smart folders from ASM.

Example

```
>>> folders = asm.smartfolders.list()
```

CHAPTER

THIRTYNINE

TENABLE ONE

This package covers the Tenable One.

class TenableOne(*access_key: str | None = None*, *secret_key: str | None = None*, ***kwargs*)

The Tenable One object is the primary interaction point for users to interface with Tenable One via the pyTenable library. All the API endpoint classes that have been written will be grafted onto this class.

Environment Variables:

TO_ACCESS_KEY: API Access Key for the Tenable One Application.

TO_SECRET_KEY: API Secret Key for the Tenable One Application.

TO_URL: The Application URL. Defaults to *https://cloud.tenable.com*.

Examples

Basic Example:

```
>>> from tenable.tenableone import TenableOne
>>> tenable_one = TenableOne('ACCESS_KEY', 'SECRET_KEY')
```

Example with proper identification:

```
>>> tenable_inventory = TenableOne('ACCESS_KEY', 'SECRET_KEY',
>>>     vendor='Company Name',
>>>     product='My Awesome Widget',
>>>     build='1.0.0')
```

property attack_path

The interface object for the *Tenable One Attack Path APIs*.

property exposure_view

The interface object for the *Tenable One Exposure View APIs*.

property inventory

The interface object for the *Tenable One Inventory APIs*.

property tags

The interface object for the *Tenable One Tags APIs*.

39.1 Attack Path

The following sub-package allows for interaction with the Tenable One Attack Path APIs.

class AttackPathAPI(*api: APISession*)

property export

The interface object for the *Tenable One Attack Path Export APIs*.

property findings

The interface object for the *Tenable One Attack Path Findings APIs*.

property vectors

The interface object for the *Tenable One Attack Path Vectors APIs*.

39.1.1 Export

Methods described in this section relate to the attack path export API for TenableOne. These methods can be accessed at `TenableOne.attack_path.export`.

class ExportAPI(*api: APISession*)

attack_paths(*file_format: FileFormat*, *sort: ExportSortParams | None = None*, *filters: ExportFilter | ExportFilterCondition | None = None*, *vector_ids: List[str] | None = None*, *columns: List[AttackPathColumnKey] | None = None*, *file_name: str | None = None*) → ExportRequestId

Export top attack paths

Parameters

- **file_format** (*FileFormat*) – The output file format (CSV or JSON).
- **sort** (*ExportSortParams, optional*) – Sort parameters for the export.
- **filters** (*ExportFilter | ExportFilterCondition, optional*) – Filters to apply to the export. Can be a single filter condition or a compound filter with multiple conditions.
- **vector_ids** (*list[str], optional*) – List of vector IDs to filter by.
- **columns** (*list[AttackPathColumnKey], optional*) – Columns to include in the export.
- **file_name** (*str, optional*) – The name of the export file.

Returns
The export request ID.

Return type
ExportRequestId

Examples

```
>>> export = tenable_one.attack_path.export.attack_paths(
...     file_format=FileFormat.CSV,
...     columns=[AttackPathColumnKey.PATH_NAME, AttackPathColumnKey.PRIORITY],
... )
>>> print(export.export_id)
```

attack_techniques(*file_format: FileFormat*, *filters: ExportFilter | ExportFilterCondition | None = None*, *sort: ExportSortParams | None = None*, *columns: List[AttackTechniqueColumnKey] | None = None*, *file_name: str | None = None*, *attack_technique_ids: List[str] | None = None*) → ExportRequestId

Export attack techniques

Parameters

- **file_format** (*FileFormat*) – The output file format (CSV or JSON).
- **filters** (*ExportFilter | ExportFilterCondition, optional*) – Filters to apply to the export. Can be a single filter condition or a compound filter with multiple conditions.
- **sort** (*ExportSortParams, optional*) – Sort parameters for the export.
- **columns** (*list[AttackTechniqueColumnKey], optional*) – Columns to include in the export.
- **file_name** (*str, optional*) – The name of the export file.
- **attack_technique_ids** (*list[str], optional*) – List of attack technique IDs to filter by.

Returns
The export request ID.

Return type
ExportRequestId

Examples

```
>>> export = tenable_one.attack_path.export.attack_techniques(
...     file_format=FileFormat.JSON,
...     columns=[
...         AttackTechniqueColumnKey.MITRE_ID,
...         AttackTechniqueColumnKey.TECHNIQUE_NAME,
...     ],
... )
>>> print(export.export_id)
```

download(*export_id: str*, *fobj: BytesIO | None = None*) → bytes | BytesIO

Download export results

Parameters

- **export_id** (*str*) – The export ID to download.

- **fobj** (*BytesIO, optional*) – A file-like object to write the downloaded data to. If not provided, the data will be returned as bytes. If provided, the data will be streamed directly to the file object and the file object will be returned.

Returns
The exported data as bytes if fobj is not provided, or the file object if fobj is provided.

Return type
Union[bytes, BytesIO]

Examples

Download to bytes:

```
>>> data = tenable_one.attack_path.export.download("export-123")
>>> with open("export.csv", "wb") as f:
...     f.write(data)
```

Stream to file object:

```
>>> fobj = BytesIO()
>>> tenable_one.attack_path.export.download("export-123", fobj)
>>> with open("export.csv", "wb") as f:
...     f.write(fobj.getvalue())
```

mitre_heatmap(*file_format: FileFormat*, *filter: MitreHeatmapFilter | None = None*, *columns: List[str] | None = None*, *file_name: str | None = None*) → ExportRequestId

Export MITRE ATT&CK heatmap

Parameters

- **file_format** (*FileFormat*) – The output file format. CSV emits a flat technique table; JSON emits a Navigator-compatible layer document.
- **filter** (*MitreHeatmapFilter, optional*) – Filter criteria for the heatmap (platform, query, severities, matrix, show_all_techniques).
- **columns** (*list[str], optional*) – Column names to include in the export.
- **file_name** (*str, optional*) – Custom file name for the export.

Returns
The export request ID.

Return type
ExportRequestId

Examples

```
>>> export = tenable_one.attack_path.export.mitre_heatmap(
...     file_format=FileFormat.JSON,
...     filter=MitreHeatmapFilter(matrix=MitreMatrix.ENTERPRISE),
... )
>>> print(export.export_id)
```

status(*export_id: str*) → ExportRequestStatus

Get export status

Parameters

export_id (*str*) – The export ID to check status for.

Returns

The export status information.

Return type

ExportRequestStatus

Examples

```
>>> status = tenable_one.attack_path.export.status("export-123")
>>> print(f"Status: {status.status}")
```

39.1.2 Findings

Methods described in this section relate to the findings API. These methods can be accessed at `TenableOne.attack_path.findings`.

class FindingsAPI(*api: APISession*)

list(*page_number: int | None = None, next_token: str | None = None, limit: int = 50, filter: dict | None = None, sort_filed: str | None = None, sort_order: str | None = None, return_iterator=True*) → FindingIterator | FindingsPageSchema

Retrieve findings

Parameters

- **page_number** (*optional, int*) – For offset-based pagination, the requested page to retrieve. If this parameter is omitted, Tenable uses the default value of 1.
- **next_token** (*optional, str*) – For cusrsor-based pagination, the cursor position for the next page. For the initial request, don't populate. For subsequent requests, set this parameter to the value found in the next property of the previous response. When getting null without specify a page number it means there are no more pages.
- **limit** (*optional, int*) – The number of records to retrieve. If this parameter is omitted, Tenable uses the default value of 50. The maximum number of events that can be retrieved is 10,000. For example: limit=10000.
- **filter** (*optional, dict*) – A document as defined by Tenable APA online documentation. Filters to allow the user to get to a specific subset of Findings. For a more detailed listing of what filters are available, please refer to the API documentation linked above, however some examples are as such:
  - `{"operator":"==", "key":"state", "value":"open"}`
  - `{"operator":">", "key":"last_updated_at", "value":"2024-05-30T12:28:11.528118"}`
- **sort_filed** (*optional, str*) – The field you want to use to sort the results by. Accepted values are `last_updated_at`, `state`, `vectorCount`, `status`, `name`, `procedureName`, `priority`, and `mitre_id`.
- **sort_order** (*optional, str*) – The sort order Accepted values are `desc` or `acs`

- **return_iterator** (*bool, optional*) – Should we return the response instead of iterable?

Returns
List of findings records

Return type
FindingIterator

Examples:

```
>>> findings = tapa.findings.list()
>>> for f in findings:
...     pprint(f)
```

Examples:

```
>>> tapa.findings.list(
...     limit='10',
...     sort_filed='last_updated_at',
...     sort_order='desc',
...     filter='value',
...     return_iterator=False
...     )
```

39.1.3 Vectors

Methods described in this section relate to the vectors API. These methods can be accessed at `TenableOne.attack_path`.

class VectorsAPI(*api: APISession*)

list(*page_number: int | None = None, limit: int = 10, filter: dict | None = None, sort_field: str | None = None, sort_order: str | None = None, run_ai_summarization: bool | None = None, return_iterator=True*) → VectorIterator | VectorsPageSchema

Retrieve vectors

Parameters

- **page_number** (*optional, int*) – For offset-based pagination, the requested page to retrieve. If this parameter is omitted, Tenable uses the default value of 1.
- **limit** (*optional, int*) – The number of records to retrieve. If this parameter is omitted, Tenable uses the default value of 25. The maximum number of events that can be retrieved is 25. For example: limit=25.
- **filter** (*optional, dict*) – A document as defined by Tenable APA online documentation. Filters to allow the user to get to a specific subset of Findings. For a more detailed listing of what filters are available, please refer to the API documentation linked above, however some examples are:
  - `{"operator":"==", "key":"name", "value":"nice name"}`
  - `{"operator":">", "key":"critical_asset", "value": 10}`
- **sort_field** (*optional, str*) – The field you want to use to sort the results by. Accepted values are `name`, `priority`

- **run_ai_summarization** (*optional, bool*) – Indicates whether or not to run the AI summarization for missing paths. Note that enabling the AI summarization results in slower response times. Tenable uses the default value of false.
- **return_iterator** (*optional, bool*) – Should we return the response instead of iterable?

Returns
List of vectors records

Return type
VectorsIterator

Examples

List all of the available vectors:

```
>>> vectors = tapa.vectors.list()
>>> for f in vectors:
...     pprint(f)
```

Filtering for a specific subset of vectors:

```
>>> tapa.vectors.list(
...     limit='10',
...     sort_field='name',
...     sort_order='desc',
...     filter={"operator":"==", "key":"name", "value":"nice name"},
...     return_iterator=False
...     )
```

top_attack_paths_search(*limit: int = 1000, sort: str | None = None, run_ai_summarization: str | None = None, filter: PublicVectorFilterType | None = None, exclude_resolved: bool = True*) → DiscoverPageTableResponse

Search top attack paths leading to critical assets.

This endpoint provides comprehensive search capabilities for attack paths with advanced filtering, sorting, and pagination options. The response includes detailed information about each attack path, including techniques, nodes, and metadata.

Note: Attack Paths now inherit their status from the underlying techniques. Paths marked as "Chain Prevented" (partially fixed), "Accepted" or "Done" are excluded by default to prioritize active threats. Set exclude_resolved=False to include all paths.

Parameters

- **limit** (*optional, int*) – Number of items per page (default: 1000, min: 100, max: 10000)
- **sort** (*optional, str*) – Sort parameter in format "{sort_field}:{sort_order}" (e.g., "name:asc", "priority:desc", "path_status:asc")
- **run_ai_summarization** (*optional, str*) – Whether to run AI summarization (default: "false"). Enabling AI summarization provides additional insights but results in slower response times. Valid values: "true", "false"
- **filter** (*optional, PublicVectorFilterType*) – Filter criteria for the search. The filter is passed as a JSON object in the request body. Supports complex filtering with AND/OR operators.

- **exclude_resolved** (*bool, optional*) – When True (default), excludes paths with path_status 'done', 'chain_prevented', or 'accepted'. Set to False to include all paths.

Returns
Response containing attack paths data with pagination information

Return type
`DiscoverPageTableResponse`

Examples

Search for high priority attack paths leading to critical assets

```
>>> filter_data = {
...     "operator": "AND",
...     "value": [
...         {"property": "priority", "operator": "gte", "value": 8},
...         {"property": "critical_asset", "operator": "eq", "value": True}
...     ]
... }
>>> response = t1.attack_path.vectors.top_attack_paths_search(
...     limit=500,
...     sort="priority:desc",
...     filter=filter_data
... )
>>> for attack_path in response.data:
...     print(f"Attack Path: {attack_path.name}, Priority: {attack_path.
↪priority}")
```

Filter by path_status to get only actionable paths

```
>>> filter_data = {
...     "property": "path_status",
...     "operator": "in",
...     "value": ["to_do", "in_progress", "in_review"]
... }
>>> response = t1.attack_path.vectors.top_attack_paths_search(
...     filter=filter_data
... )
```

Include resolved paths

```
>>> response = t1.attack_path.vectors.top_attack_paths_search(
...     exclude_resolved=False
... )
```

Simple search with default parameters

```
>>> response = t1.attack_path.vectors.top_attack_paths_search()
>>> print(f"Found {response.total} attack paths")
```

39.2 Inventory

The following sub-package allows for interaction with the Tenable One Inventory APIs.

class InventoryAPI(*api: APISession*)

property export

The interface object for the *Tenable One Inventory Unified Export APIs*.

property findings

The interface object for the *Tenable Exposure Management Inventory Finding APIs*.

property software

The interface object for the *Tenable One Inventory Software APIs*.

39.2.1 Assets

Methods described in this section relate to the assets API. These methods can be accessed at `TenableOne.inventory.assets`.

class AssetsAPI(*api: APISession*)

list(*query_text: str | None = None*, *query_mode: QueryMode | None = None*, *filters: list[PropertyFilter] | None = None*, *extra_properties: list[str] | None = None*, *offset: int | None = None*, *limit: int | None = None*, *sort_by: str | None = None*, *sort_direction: SortDirection | None = None*) → Assets

Retrieve assets

Parameters

- **query_text** (*str, optional*) – The text to search for.
- **query_mode** (*QueryMode, optional*) – The search mode. Defaults to QueryMode.SIMPLE.
- **filters** (*list, optional*) – A list of filters to apply. Defaults to None.
- **extra_properties** (*list, optional*) – Additional properties to include in the response. Defaults to None.
- **offset** (*int, optional*) – Number of records to skip. Defaults to 0.
- **limit** (*int, optional*) – Maximum number of records per page. Defaults to 1000.
- **sort_by** (*str, optional*) – Field to sort by.
- **sort_direction** (*SortDirection, optional*) – Sorting direction, either SortDirection.ASC or SortDirection.DESC.

Returns
The request assets.

Return type
`Asset`

Examples

```
>>> assets = tenable_inventory.assets.list()
>>> for asset in assets:
...     pprint(asset)
```

list_properties(*asset_classes: list[AssetClass] | None = None*) → list[Field]

Retrieve assets properties

Parameters

asset_classes (*list[str], optional*) – Getting properties for specific asset classes If this parameter is omitted, Tenable returns properties for all asset classes. For example: asset_classes=[AssetClass.DEVICE].

Returns

The asset properties.

Return type

list[Field]

Examples

```
>>> tenable_inventory_asset_properties = tenable_inventory.assets.list_
↪properties()
>>> for asset_property in asset_properties:
...     pprint(asset_property)
```

39.2.2 Software

Methods described in this section relate to the software API. These methods can be accessed at `TenableOne.inventory.software`.

class SoftwareAPI(*api: APISession*)

list(*query_text: str | None = None, query_mode: QueryMode | None = None, filters: list[PropertyFilter] | None = None, extra_properties: list[str] | None = None, offset: int | None = None, limit: int | None = None, sort_by: str | None = None, sort_direction: SortDirection | None = None*) → SoftwareValues

Retrieve software

Parameters

- **query_text** (*str, optional*) – The text to search for.
- **query_mode** (*QueryMode, optional*) – The search mode. Defaults to QueryMode.SIMPLE.
- **filters** (*list, optional*) – A list of filters to apply. Defaults to None.
- **extra_properties** (*list, optional*) – Additional properties to include in the response. Defaults to None.
- **offset** (*int, optional*) – Number of records to skip. Defaults to 0.
- **limit** (*int, optional*) – Maximum number of records per page. Defaults to 1000.
- **sort_by** (*str, optional*) – Field to sort by.

- **sort_direction** (*SortDirection, optional*) – Sorting direction, either SortDirection.ASC or SortDirection.DESC.

Returns
The request software.

Examples

```
>>> tenable_inventory_software = tenable_inventory.software.list()
>>> for software in tenable_inventory_software:
...     pprint(software)
```

list_properties() → list[Field]

Retrieve software properties

Returns
The software properties.

Examples

```
>>> tenable_inventory_software_properties = tenable_inventory.software.list_
↪properties()
>>> for software_property in software_properties:
...     pprint(software_property)
```

39.2.3 Export

Methods described in this section relate to the unified export API for TenableOne. These methods can be accessed at `TenableOne.inventory.export`.

class ExportAPI(*api: APISession*)

assets(*filters: List[PropertyFilter] | None = None, query: Query | None = None, properties: List[str] | None = None, use_readable_name: bool | None = None, max_chunk_file_size: int | None = None, sort_by: str | None = None, sort_direction: SortDirection | None = None, file_format: DatasetFileFormat | None = None, compress: bool | None = None*) → ExportRequestId

Export assets from TenableOne inventory

Parameters

- **filters** (*list[PropertyFilter], optional*) – A list of filters to apply to the export. Defaults to None.
- **query** (*Query, optional*) – The query to apply.
- **properties** (*list[str], optional*) – Properties to include about the assets returned in the search results. List of property names. Defaults to None.
- **use_readable_name** (*bool, optional*) – If true, the readable name of the property will be used instead of the internal (key) name. Defaults to True.
- **max_chunk_file_size** (*int, optional*) – Maximum size in bytes for each chunk file when exporting large datasets. Defaults to 20971520 (20 MB).
- **sort_by** (*str, optional*) – Field to sort by.

- **sort_direction** (*SortDirection, optional*) – Sorting direction, either SortDirection.ASC or SortDirection.DESC.
- **file_format** (*DatasetFileFormat, optional*) – The file format to be received. If not specified, the default format will be JSON. Supported formats include: CSV and JSON. Defaults to DatasetFileFormat.JSON.
- **compress** (*bool, optional*) – Whether to compress the export using GZIP compression. Compressed files are smaller but require decompression before processing.

Returns
The export request ID.

Return type
ExportRequestId

Examples

```
>>> export_id = tenable_one.inventory.export.assets(
...     properties=["cpe", "version", "file_location"],
...     sort_by="name",
...     sort_direction=SortDirection.ASC,
...     file_format=DatasetFileFormat.CSV
... )
>>> print(export_id.export_id)
```

assets_status(*status: str | None = None*, *limit: int | None = None*) → ExportJobsResponse

List asset export jobs submitted in the last 3 days

This endpoint only returns asset export jobs that were submitted within the last 3 days. Results are sorted by last refreshed time (newest first).

Parameters

- **status** (*str, optional*) – Filter by export status (e.g., 'FINISHED', 'PROCESSING', 'ERROR'). Multiple values can be provided as a comma-separated string (e.g., 'FINISHED,PROCESSING').
- **limit** (*int, optional*) – Maximum number of export jobs to return from the last 3 days. Note: This endpoint only returns jobs submitted within the last 3 days, so the actual number of results may be less than the limit if fewer jobs exist within that time window. Defaults to 1000 if not specified. Minimum: 1, Maximum: 1000.

Returns
Response containing list of asset export job information from the last 3 days.

Return type
ExportJobsResponse

Examples

```
>>> response = tenable_one.inventory.export.assets_status()
>>> for job in response.exports:
...     print(f"Export ID: {job.export_id}, Status: {job.status}")
```

```
>>> # Filter by status
>>> finished_jobs = tenable_one.inventory.export.assets_status(status='FINISHED
↪')
```

```
>>> # Limit results to 100 most recent jobs
>>> recent_jobs = tenable_one.inventory.export.assets_status(limit=100)
```

download(*export_id: str*, *chunk_id: str*, *fobj: BytesIO | None = None*) → bytes | BytesIO

Download export chunk

Parameters

- **export_id** (*str*) – The export ID.
- **chunk_id** (*str*) – The chunk ID to download.
- **fobj** (*BytesIO, optional*) – A file-like object to write the downloaded data to. If not provided, the data will be returned as bytes. If provided, the data will be streamed directly to the file object and the file object will be returned.

Returns

The exported data as bytes if fobj is not provided, or the file object if fobj is provided.

Return type

Union[bytes, BytesIO]

Examples

Download to bytes: >>> data = tenable_one.inventory.export.download("export-123", "0") >>> with open("export.csv", "wb") as f: … f.write(data)

Stream to file object: >>> with open("export.csv", "wb") as f: … fobj = BytesIO() … tenable_one.inventory.export.download("export-123", "0", fobj) … f.write(fobj.getvalue())

findings(*filters: List[PropertyFilter] | None = None*, *query: Query | None = None*, *properties: List[str] | None = None*, *use_readable_name: bool | None = None*, *max_chunk_file_size: int | None = None*, *sort_by: str | None = None*, *sort_direction: SortDirection | None = None*, *file_format: DatasetFileFormat | None = None*, *compress: bool | None = None*) → ExportRequestId

Export findings from TenableOne inventory

Parameters

- **filters** (*list[PropertyFilter], optional*) – A list of filters to apply to the export. Defaults to None.
- **query** (*Query, optional*) – The query to apply.
- **properties** (*list[str], optional*) – Properties to include about the findings returned in the search results. List of property names. Defaults to None.
- **use_readable_name** (*bool, optional*) – If true, the readable name of the property will be used instead of the internal (key) name. Defaults to True.

- **max_chunk_file_size** (*int, optional*) – Maximum size in bytes for each chunk file when exporting large datasets. Defaults to 20971520 (20 MB).
- **sort_by** (*str, optional*) – Field to sort by.
- **sort_direction** (*SortDirection, optional*) – Sorting direction, either SortDirection.ASC or SortDirection.DESC.
- **file_format** (*DatasetFileFormat, optional*) – The file format to be received. If not specified, the default format will be JSON. Supported formats include: CSV and JSON. Defaults to DatasetFileFormat.JSON.
- **compress** (*bool, optional*) – Whether to compress the export using GZIP compression. Compressed files are smaller but require decompression before processing.

Returns
The export request ID.

Return type
ExportRequestId

Examples

```
>>> export_id = tenable_one.inventory.export.findings(
...     properties=["severity", "status", "created_at"],
...     sort_by="severity",
...     sort_direction=SortDirection.DESC,
...     file_format=DatasetFileFormat.CSV
... )
>>> print(export_id.export_id)
```

findings_status(*status: str | None = None*, *limit: int | None = None*) → ExportJobsResponse

List finding export jobs submitted in the last 3 days

This endpoint only returns finding export jobs that were submitted within the last 3 days. Results are sorted by last refreshed time (newest first).

Parameters

- **status** (*str, optional*) – Filter by export status (e.g., 'FINISHED', 'PROCESSING', 'ERROR'). Multiple values can be provided as a comma-separated string (e.g., 'FINISHED,PROCESSING').
- **limit** (*int, optional*) – Maximum number of export jobs to return from the last 3 days. Note: This endpoint only returns jobs submitted within the last 3 days, so the actual number of results may be less than the limit if fewer jobs exist within that time window. Defaults to 1000 if not specified. Minimum: 1, Maximum: 1000.

Returns
Response containing list of finding export job information from the last 3 days.

Return type
ExportJobsResponse

Examples

```
>>> response = tenable_one.inventory.export.findings_status()
>>> for job in response.exports:
...     print(f"Export ID: {job.export_id}, Status: {job.status}")
```

```
>>> # Filter by status
>>> finished_jobs = tenable_one.inventory.export.findings_status(status=
↪'FINISHED')
```

```
>>> # Limit results to 100 most recent jobs
>>> recent_jobs = tenable_one.inventory.export.findings_status(limit=100)
```

status(*export_id: str*) → ExportRequestStatus

Get export status

Parameters
export_id (*str*) – The export ID to check status for.

Returns
The export status information.

Return type
ExportRequestStatus

Examples

```
>>> status = tenable_one.inventory.export.status("export-123")
>>> print(f"Status: {status.status}")
>>> print(f"Chunks available: {status.chunks_available}")
```

39.2.4 Findings

Methods described in this section relate to the inventory findings API. These methods can be accessed at `TenableExposureManagement.inventory.findings`.

class FindingsAPI(*api: APISession*)

list(*query_text: str | None = None, query_mode: QueryMode | None = None, filters: list[PropertyFilter] | None = None, extra_properties: list[str] | None = None, offset: int | None = None, limit: int | None = None, sort_by: str | None = None, sort_direction: SortDirection | None = None*) → Findings

Retrieve findings

Parameters

- **query_text** (*str, optional*) – The text to search for.
- **query_mode** (*QueryMode, optional*) – The search mode. Defaults to QueryMode.SIMPLE.
- **filters** (*list, optional*) – A list of filters to apply. Defaults to None.
- **extra_properties** (*list, optional*) – Additional properties to include in the response. Defaults to None.

- **offset** (*int, optional*) – Number of records to skip. Defaults to 0.
- **limit** (*int, optional*) – Maximum number of records per page. Defaults to 1000.
- **sort_by** (*str, optional*) – Field to sort by.
- **sort_direction** (*SortDirection, optional*) – Sorting direction, either SortDirection.ASC or SortDirection.DESC.

Returns
The request assets.

Example

```
>>> tenable_inventory_findings = tenable_inventory.finding.list()
>>> for finding in tenable_inventory_findings:
...     pprint(finding)
```

list_properties() → list[Field]

Retrieve finding properties

Returns
The finding properties.

Examples

```
>>> properties = tenable_inventory.finding.list_properties()
>>> for finding_property in properties:
...     pprint(finding_property)
```

39.3 Tags

The following methods allow for interaction with the Tenable One Tags APIs.

Methods available on `TenableOne.tags`:

class TagsAPI(*api: APISession*)

list(*query_text: str | None = None, query_mode: QueryMode | None = None, filters: list[PropertyFilter] | None = None, extra_properties: list[str] | None = None, offset: int | None = None, limit: int | None = None, sort_by: str | None = None, sort_direction: SortDirection | None = None*) → Tags

Retrieve tags

Parameters

- **query_text** (*str, optional*) – The text to search for.
- **query_mode** (*QueryMode, optional*) – The search mode. Defaults to QueryMode.SIMPLE.
- **filters** (*list, optional*) – A list of filters to apply. Defaults to None.
- **extra_properties** (*list, optional*) – Additional properties to include in the response. Defaults to None.
- **offset** (*int, optional*) – Number of records to skip. Defaults to 0.

- **limit** (*int, optional*) – Maximum number of records per page. Defaults to 1000.
- **sort_by** (*str, optional*) – Field to sort by.
- **sort_direction** (*SortDirection, optional*) – Sorting direction, either SortDirection.ASC or SortDirection.DESC.

Returns
The request tags.

Examples

```
>>> tags = tenable_inventory.tags.list()
>>> for tag in tags:
...     pprint(tag)
```

list_properties() → list[Field]

Retrieve tags properties

Returns
The tags properties.

Examples

```
>>> tenable_inventory_tags_properties = tenable_inventory.tags.list_
↪properties()
>>> for tag_property in tenable_inventory_tags_properties:
...     pprint(tag_property)
```

39.4 Exposure View

The following sub-package allows for interaction with the Tenable Exposure Management Exposure View APIs.

class ExposureViewAPI(*api: APISession*)

property cards
The interface object for the *Tenable Exposure Management Exposure view Cards APIs*.

39.4.1 Cards

Methods described in this section relate to the exposure view cards API. These methods can be accessed at `TenableExposureManagement.exposure_view.cards`.

class CardsAPI(*api: APISession*)

get_by_id(*card_id: str*, *trend_timeframe: Timeframe | None = None*, *sla_efficiency_timeframe: Timeframe | None = None*, *sla_breakdown_filter: SlaBreakdownFilter | None = SlaBreakdownFilter.ANY*, *include_trend_events: bool | None = False*) → CardDetails

Get a specific card by its ID with optional filters

Parameters

- **card_id** – The ID of the card to retrieve

- **trend_timeframe** – Optional timeframe for trend data
- **sla_efficiency_timeframe** – Optional timeframe for SLA efficiency data
- **sla_breakdown_filter** – Optional filter for SLA breakdown (ANY, REMEDIATED, NON_REMEDIATED)
- **include_trend_events** – Whether to include trend events in the response

Returns
CardDetails object containing the card details

list(*is_global_card: bool | None = None*, *query_text: str | None = None*, *offset: int | None = 0*, *limit: int | None = 25*, *sorting_order: SortDirection | None = SortDirection.ASC*) → Cards

List cards based on filter criteria

Parameters

- **is_global_card** – Filter cards by is_global flag
- **query_text** – Text query to filter cards
- **offset** – The number of items to skip before starting to collect the result set
- **limit** – Max number of items to return
- **sorting_order** – Sorting direction (ASC or DESC)

Returns
Cards object containing the list of cards and pagination info

CHAPTER

FORTY

TENABLE NESSUS

Important

The Nessus Package is currently a Technology Preview

40.1 Nessus

This package covers the Nessus interface.

class Nessus(***kwargs*)

The Nessus object is the primary interaction point for users to interface with Tenable Nessus via the pyTenable library. All of the API endpoint classes that have been written will be grafted onto this class.

property agent_groups

The interface object for the *Tenable Nessus Agent Groups APIs*.

property agents

The interface object for the *Tenable Nessus Agents APIs*.

property editor

The interface object for the *Tenable Nessus Editor APIs*.

property files

The interface object for the *Tenable Nessus File APIs*.

property folders

The interface object for the *Tenable Nessus Folders APIs*.

property groups

The interface object for the *Tenable Nessus Groups APIs*.

property mail

The interface object for the *Tenable Nessus Mail APIs*.

property permissions

The interface object for the *Tenable Nessus Permissions APIs*.

property plugin_rules

The interface object for the *Tenable Nessus Plugin Rules APIs*.

property plugins

The interface object for the *Tenable Nessus Plugins APIs*.

property policies
The interface object for the *Tenable Nessus Policies APIs*.

property proxy
The interface object for the *Tenable Nessus Proxy APIs*.

property scanners
The interface object for the *Tenable Nessus Scanners APIs*.

property scans
The interface object for the *Tenable Nessus Scans APIs*.

property server
The interface object for the *Tenable Nessus Server APIs*.

property session
The interface object for the *Tenable Nessus Session APIs*.

property settings
The interface object for the *Tenable Nessus Settings APIs*.

property software_update
The interface object for the *Tenable Nessus Software Update APIs*.

property tokens
The interface object for the *Tenable Nessus Tokens APIs*.

property users
The interface object for the *Tenable Nessus Users APIs*.

Important

The Nessus Package is currently a Technology Preview

40.1.1 Agent Groups

Methods described in this section relate to the the agent groups API. These methods can be accessed at `Nessus.agent_groups`.

class AgentGroupsAPI(*api: APISession*)

add_agent(*group_id: int*, *agent_id: int*) → None
Adds a singular agent to an agent group.

Parameters

- **group_id** (*int*) – The agent group id to modify
- **agent_id** (*int*) – The agent id

Example

```
>>> nessus.agent_groups.add_agent(group_id, agent_id)
```

add_agents(*group_id: int*, *agents: List[int]*) → None

Adds multiple agents to an agent group.

Parameters

- **group_id** (*int*) – The agent group id to modify
- **agents** (*list[int]*) – A list opf agent ids to add to the group.

Example

```
>>> nessus.agent_groups.add_agents(group_id, [agent1, agent2])
```

configure(*group_id: int*, *name: str*) → None

Changes the name of the given agent group.

Parameters

- **group_id** (*int*) – The agent group id to modify
- **name** (*str*) – The name of the agent group

Example

```
>>> nessus.agent_groups.configure(group_id, 'Example name')
```

create(*name: str*) → Dict

Creates an agent group.

Parameters

name (*str*) – The name of the agent group

Example

```
>>> group = nessus.agent_groups.create('Example agent group')
```

delete_agent(*group_id: int*, *agent_id: int*) → None

Deletes an agent from the agent group.

Parameters

- **group_id** (*int*) – The agent group id to modify
- **agent_id** (*int*) – The agent id to delete from the group

Example

```
>>> nessus.agent_groups.delete_agent(group_id, agent_id)
```

delete_agents(*group_id: int*, *agents: List[int]*) → None

Deletes multiple agents from the agent group.

Parameters

- **group_id** (*int*) – The agent group to modify
- **agents** (*list[int]*) – A list of agent ids to remove fromt he group

Example

```
>>> nessus.agent_groups.delete_agents(group_id, [agent1, agent2])
```

delete_group(*group_id: int*) → None

Deletes an agent group.

Parameters
group_id (*int*) – The agent group id to be deleted.

Example

```
>>> nessus.agent_groups.delete_group(group_id)
```

delete_groups(*group_ids: List[int]*) → None

Deleted multiple agent groups.

Parameters
group_ids (*list[int]*) – A list of agent group ids to delete

Example

```
>>> nessus.agent_groups.delete_groups([group1, group2, group3])
```

details(*group_id: int*) → Dict

Returns details for the given agent group

Parameters
group_id (*int*) – The agent group id to retrieve

Example

```
>>> group = nessus.agent_groups.details(group_id)
```

list() → List[Dict]

Returns a listing of the agent groups

Example

```
>>> groups = nessus.agent_groups.list()
```

Important

The Nessus Package is currently a Technology Preview

40.1.2 Agents

Methods described in this section relate to the the agents API. These methods can be accessed at `Nessus.agents`.

class AgentsAPI(*api: APISession*)

delete(*agent_id: int*) → None

Deletes an agent

Parameters

agent_id (*int*) – Id of the agent to delete

Example

```
>>> nessus.agents.delete(agent_id)
```

delete_many(*agent_ids: List[int]*) → None

Deletes multiple agents.

Parameters

agent_ids (*list[int]*) – List of agent ids to delete

Example

```
>>> nessus.agents.delete_many([agent1, agent2, agent3])
```

details(*agent_id: int*) → Dict

Returns the details for an agent.

Parameters

agent_id (*int*) – Id of the agent to retreive

Example

```
>>> agent = nessus.agents.details(agent_id)
```

list(*limit: int = 1000*, *offset: int = 0*, *sort_by: str | None = None*, *sort_order: Literal['asc', 'desc'] | None = None*, *search_type: Literal['and', 'or'] | None = None*, *filters: Dict | Tuple | None = None*, *return_json: bool = False*) → PaginationIterator | List[Dict]

Returns a list of agents.

Parameters

- **filters** (*list[tuple], optional*) – List of filters.
- **sort_by** (*str, optional*) – Field to sort by
- **sort_order** (*str, optional*) – Is the sort ascending (`asc`) or descending (`desc`)?
- **limit** (*int, optional*) – Number of records per page
- **offset** (*int, optional*) – How many records to skip before starting to return data
- **return_json** (*bool, optional*) – Should a JSON object be returned instead of an iterator?

Example

```
>>> for agent in nessus.agents.list():
...     print(agent)
```

Example with filtering:

```
>>> for agent in nessus.agents.list(
...     filters=[('name', 'match', 'lin')]
... ):
...     print(agent)
```

Example getting the JSON response instead:

```
>>> agents = nessus.agents.list(return_json=True)
```

unlink(*agent_id: int*) → None

Unlinks an agent.

Parameters

agent_id (*int*) – Id of the agent to unlink

Example

```
>>> nessus.agents.unlink(agent_id)
```

unlink_many(*agent_ids: List[int]*) → None

Unlinks multiple agents.

Parameters

agent_ids (*list[int]*) – List of agent ids to unlink

Example

```
>>> nessus.agents.unlink_many([agent1, agent2, agent3])
```

Important

The Nessus Package is currently a Technology Preview

40.1.3 Editor

Methods described in this section relate to the the editor API. These methods can be accessed at `Nessus.editor`.

class EditorAPI(*api: APISession*)

details(*object_type: Literal['scan', 'policy']*, *object_id: int*) → Dict

Returns the object requested.

Parameters

- **object_type** (*str*) – The type of the object requested (either scan or policy).
- **object_id** (*int*) – The id of the object to fetch.

Returns

The editor object for the requested item.

Return type

Dict

Example

```
>>> nessus.editor.details('scan', 1)
```

export_audit(*object_type: Literal['scan', 'policy']*, *object_id: int*, *file_id: int*, ***kwargs*) → BytesIO

Exports the given audit file. from the scan or policy

Parameters

- **object_type** (*str*) – The object type (either scan or policy).
- **object_id** (*int*) – The id of the object to export from.
- **file_id** (*int*) – The id of the audit file to export.
- **fobj** (*BytexIO, optional*) – The file object to write the exported file to. If none is specified then a BytesIO object is written to in memory.
- **chunk_size** (*int, optional*) – The chunk sizing for the download itself.
- **stream_hook** (*callable, optional*) – Overload the default downloading behavior with a custom stream hook.
- **hook_kwargs** (*dict, optional*) – keyword arguments to pass to the stream_hook callable in addition to the default passed params.

Returns

The file object of the requested audit file.

Return type
BytesIO

Example

```
>>> with open('example.audit', 'wb') as audit_file:
...     nessus.editor.export_audit('policy', 1, 1, fobj=audit_file)
```

plugin_description(*policy_id: int*, *family_id: int*, *plugin_id: int*) → Dict

Returns the plugin description.

Parameters

- **policy_id** (*int*) – The id of the policy to lookup.
- **family_id** (*int*) – The id of the plugin family to lookup.
- **plugin_id** (*int*) – The id of the plugin to lookup within the family.

Returns
The plugin description.

Return type
Dict

Example

```
>>> nessus.editor.plugin_description(1, 1, 19506)
```

template_details(*object_type: Literal['scan', 'policy']*, *template_uuid: str*) → Dict

Returns the template object requested.

Parameters

- **object_type** (*str*) – The type of the object requested (either scan, or policy).
- **template_uuid** (*str*) – The UUID of the template to fetch.

Returns
The editor object for the requested template.

Return type
Dict

Example

```
>>> nessus.editor.template_details(
...     'scan', '00000000-0000-0000-000000000000')
```

template_list(*object_type: Literal['scan', 'policy']*) → List[Dict]

Returns the list of template objects.

Parameters
object_type (*str*) – The type of templates to return (either scan or policy).

Returns
List of template summaries.

Return type
List[Dict]

Example

```
>>> for tmpl in nessus.editor.template_list('scan'):
...     print(tmpl)
```

Important

The Nessus Package is currently a Technology Preview

40.1.4 Files

Methods described in this section relate to the files API. These methods can be accessed at `Nessus.files`.

class FilesAPI(*api: APISession*)

upload(*fobj: BytesIO*, *encrypted: bool = False*) → str

Uploads a file to Tenable Nessus

Parameters

- **fobj** (*BytesIO*) – The file object to upload
- **encrypted** (*bool, optional*) – Is the file encrypted?

Returns
File identifier to be used in future calls.

Return type
str

Example

```
>>> with open('example.txt', 'rb') as fobj:
...     fn = nessus.files.upload(fobj)
```

Important

The Nessus Package is currently a Technology Preview

40.1.5 Folders

Methods described in this section relate to the folders API. These methods can be accessed at `Nessus.folders`.

class FoldersAPI(*api: APISession*)

create(*name: str*) → int

Create a new folder

Parameters
name (*str*) – The name of the new folder

Returns
The id for the created folder.

Return type
int

Example

```
>>> id = nessus.folders.create('Example')
```

delete(*folder_id: int*) → None

Deletes a user-defined folder

Parameters
folder_id (*int*) – The unique identifier for the folder to delete

Example

```
>>> nessus.folders.delete(id)
```

edit(*folder_id: int*, *name: str*) → None

Updated the name of the specified folder.

Parameters

- **folder_id** (*int*) – Unique identifier for the folder to edit
- **name** (*str*) – New name for the folder

Example

```
>>> nessus.folders.edit(id, name='New Example')
```

list() → List

Gets the list of available folders

Returns
List of folder objects

Return type
List

Example

```
>>> for folder in nessus.folders.list():
...     print(folder)
```

Important

The Nessus Package is currently a Technology Preview

40.1.6 Groups

Methods described in this section relate to the groups API. These methods can be accessed at `Nessus.groups`.

class GroupsAPI(*api: APISession*)

add_user(*group_id: int*, *user_id: int*) → None

Adds a user to the group

Parameters

- **group_id** (*int*) – The group id
- **user_id** (*int*) – The user id

Example

```
>>> nessus.groups.add_user(group_id, user_id)
```

create(*name: str*) → Dict

Creates a new group

Parameters
name (*str*) – The name of the new group

Returns
The new group object

Return type
Dict

Example

```
>>> nessus.groups.create('Example Group')
```

delete(*group_id: int*) → None

Deletes a group

Parameters
group_id (*int*) – The id of the group to delete

Example

```
>>> nessus.groups.delete(group_id)
```

delete_many(*group_ids: List[int]*) → None

Deletes many groups

Parameters
group_ids (*list[int]*) – The list of group ids to delete

Example

```
>>> nessus.groups.delete_many([group_1, group_2, group_3])
```

edit(*group_id: int*, *name: str*) → Dict

Edits the specified group

Parameters

- **group_id** (*int*) – The group to edit
- **name** (*str*) – The new name for the group

Returns
The updated group object

Return type
Dict

Example

```
>>> nessus.groups.edit(group_id, name='Updated Name')
```

list() → List[Dict]

Gets the list of groups

Returns
List of group objects

Return type
List

Example

```
>>> for group in nessus.groups.list():
...     print(group)
```

list_users(*group_id: int*) → List[Dict]

Gets the list of users associated to the specified group

Parameters
group_id (*int*) – The group to request members for

Returns
The list of users associated to the group specified.

Return type
List

Example

```
>>> for user in nessus.groups.list_users(group_id):
...     print(user)
```

remove_user(*group_id: int*, *user_id: int*) → None

Removes a user from the specified group

Parameters

- **group_id** (*int*) – The group to modify
- **user_id** (*int*) – The user to remove

Example

```
>>> nessus.groups.remove_user(group_id, user_id)
```

Important

The Nessus Package is currently a Technology Preview

40.1.7 Mail

Methods described in this section relate to the mail API. These methods can be accessed at `Nessus.mail`.

class MailAPI(*api: APISession*)

details() → Dict

Retrieves the Tenable Nessus daemon's mail settings

Returns
Dictionary of SMTP settings

Return type
Dict

Example

```
>>> nessus.mail.details()
```

edit(*smtp_host: str | None = None*, *smtp_port: int | None = None*, *smtp_enc: Literal['No Encryption', 'Use TLS if available', 'Force SSLForce TLS'] | None = None*, *smtp_from: str | None = None*, *smtp_www_host: str | None = None*, *smtp_user: str | None = None*, *smtp_pass: str | None = None*, *smtp_auth: Literal['NONE', 'PLAIN', 'LOGIN', 'NTLM', 'CRAM-MD5'] | None = None*) → None

Updates the Tenable Nessus daemon's mail settings

Parameters

- **smtp_host** (*str, optional*) – DNS/IP Address of the SMTP server
- **smtp_port** (*int, optional*) – Port number for the SMTP service
- **smtp_enc** (*str, optional*) – The connection encryption for the SMTP server
- **smtp_from** (*str, optional*) – Reply email address for email sent by the Tenable Nessus daemon
- **smtp_www_host** (*str, optional*) – The host to use in email links
- **smtp_user** (*str, optional*) – The username to use when authenticating to the SMTP service
- **smtp_pass** (*str, optional*) – The password to use when authenticating to the SMTP service
- **smtp_auth** (*str, optional*) – The authentication type for the SMTP server

Example

```
>>> nessus.mail.edit(smtp_user='new_user',
...                  smtp_pass='updated_password',
...                  smtp_auth='LOGIN',
...                  )
```

Important

The Nessus Package is currently a Technology Preview

40.1.8 Permissions

Methods described in this section relate to the permissions API. These methods can be accessed at `Nessus.permissions`.

class PermissionsAPI(*api: APISession*)

details(*object_type: Literal['scanner']*, *object_id: int*) → List[Dict]

Retrieves the access control list for the specified object

Parameters

- **object_type** (*str*) – The type of permissions object
- **object_id** (*int*) – The unique id of the object to retrieve

Returns

List of ACL objects.

Return type

List

Example

```
>>> nessus.permissions.details('scanner', 1)
```

edit(*object_type: Literal['scanner']*, *object_id: int*, *acls: List[Dict]*) → None

Updates the permissions for the specified object

Parameters

- **object_type** (*str*) – The type of object to modify
- **object_id** (*int*) – The unique id of the object to modify
- **acls** (*list[dict]*) – The list of access control objects to apply

Example

```
>>> nessus.permissions.edit('scanner', 1, acls=[
...     {
...         'type': 'default',
...         'permissions': 16
...     }, {
...         'type': 'user',
...         'permissions': 64,
...         'name': 'admin',
...         'id': 1,
...         'owner': 1
...     })
```

Important

The Nessus Package is currently a Technology Preview

40.1.9 Plugin Rules

Methods described in this section relate to the plugin rules API. These methods can be accessed at `Nessus.plugin_rules`.

class PluginRulesAPI(*api: APISession*)

create(*plugin_id: int*, *type: Literal['recast_critical', 'recast_high', 'recast_medium', 'recast_low', 'recast_info', 'exclude']*, *host: str | None = None*, *date: int | None = None*) → None

Creates a new plugin rule

Parameters

- **plugin_id** (*int*) – The plugin id to modify
- **type** – (str): The type of modification to perform
- **host** (*str, optional*) – The host to apply this rule to
- **date** (*int, optional*) – The unix date for this rule to expire

Example

```
>>> nessus.plugin_rules.create(
...     plugin_id=19506,
...     type='exclude',
...     host='192.168.0.1',
...     date=1645164000
... )
```

delete(*rule_id: int*) → None

Deletes a plugin rule

Parameters
rule_id (*int*) – The rule to delete

Example

```
>>> nessus.plugin_rules.delete(1)
```

delete_many(*rule_ids: List[int]*) → None

Deletes multiple plugin rules

Parameters
rule_ids (*list[int]*) – The rules to delete

Example

```
>>> nessus.plugin_rules.delete_many([1, 2, 3])
```

details(*rule_id: int*) → Dict

Returns the details of a given plugin rule

Parameters
rule_id (*int*) – The plugin rule id

Returns
The plugin rule object requested

Return type
Dict

Example

```
>>> nessus.plugin_rules.details(1)
```

edit(*rule_id: int*, *plugin_id: int | None = None*, *type: Literal['recast_critical', 'recast_high', 'recast_medium', 'recast_low', 'recast_info', 'exclude'] | None = None*, *host: str | None = None*, *date: int | None = None*) → None

Creates a new plugin rule

Parameters

- **rule_id** (*int*) – The rule to modify

- **plugin_id** (*int, optional*) – The plugin id to modify
- **type** – (str, optional): The type of modification to perform
- **host** (*str, optional*) – The host to apply this rule to
- **date** (*int, optional*) – The unix date for this rule to expire

Example

```
>>> nessus.plugin_rules.edit(1, date=1645164000)
```

list() → List[Dict]

Lists the plugin rules

Returns
List of plugin rule objects

Return type
List[Dict]

Example

```
>>> for rule in nessus.plugin_rules.list():
...     print(rule)
```

Important

The Nessus Package is currently a Technology Preview

40.1.10 Plugins

Methods described in this section relate to the plugins API. These methods can be accessed at `Nessus.plugins`.

class PluginsAPI(*api: APISession*)

families() → List[Dict]

Returns the list of plugin families.

Example

```
>>> families = nessus.plugins.families()
```

family_details(*family_id: int*) → Dict

Returns the details for a given plugin family.

Parameters
family_id (*int*) – The id of the family to return

Example

```
>>> family = nessus.plugins.family_details(fam_id)
```

list()

Returns an iterable to walk through each plugin.

Example

```
>>> for plugin in nessus.plugins.list():
...     print(plugin)
```

plugin_details(*plugin_id: int*) → Dict

Returns the details for a given plugin id.

Parameters
plugin_id (*int*) – The id of the plugin to return

Example

```
>>> plugin = nessus.plugins.plugin_details(19506)
```

Important

The Nessus Package is currently a Technology Preview

40.1.11 Policies

Methods described in this section relate to the files API. These methods can be accessed at `Nessus.policies`.

class PoliciesAPI(*api: APISession*)

copy(*policy_id: int*) → Dict

Duplicates an existing scan policy.

Parameters
policy_id (*int*) – The id of the policy to clone.

Returns
The cloned policy object.

Return type
Dict

Example

```
>>> nessus.policies.copy(1)
```

create(*uuid: str*, ***kwargs*) → Dict

Creates a new scan policy using the provided settings.

Parameters

- **uuid** (*str*) – The UUID for the editor template to use.
- ****kwargs** (*dict*) – Additional settings to use to create the policy.

Returns
Response object with identifying information on the new policy

Return type
Dict

Example

```
>>> tmpl = '731a8e52-3ea6-a291-ec0a-d2ff0619c19d7bd788d6be818b65'
>>> nessus.policies.create(tmpl_uuid, settings={
...     'name': 'Sample Policy'
... })
```

delete(*policy_id: int*) → None

Deletes the specified scan policy.

Parameters
policy_id (*int*) – The id of the policy to delete.

Example

```
>>> nessus.policies.delete(1)
```

delete_many(*policy_ids: List[int]*) → List[int]

Deletes the specified scan policies.

Parameters
policy_ids (*list[int]*) – The list of policy ids to delete.

Example

```
>>> nessus.policies.delete_many([1, 2, 3])
```

details(*policy_id: int*) → Dict

Returns the details of the selected policy.

Parameters
policy_id (*int*) – The id of the policy to retrieve.

Returns
The policy object.

Return type
Dict

Example

```
>>> nessus.policies.details(1)
```

edit(*policy_id: int*, ***kwargs*) → None

Updates an existing scan policy.

Parameters

- **policy_id** (*int*) – The id of the policy to edit.
- ****kwargs** (*dict*) – Attributes to be passed into the JSON body.

Example

```
>>> policy = nessus.policies.details(1)
>>> policy['settings']['name'] = 'Updated Policy'
>>> nessus.policies.edit(1, **policy)
```

export_policy(*policy_id: int*, *fobj: BytesIO | None = None*, ***kwargs*) → BytesIO

Export the specified policy and download it.

Parameters

- **policy_id** (*int*) – The id of the policy to export.
- **fobj** (*BytexIO, optional*) – The file object to write the exported file to. If none is specified then a BytesIO object is written to in memory.
- **chunk_size** (*int, optional*) – The chunk sizing for the download itself.
- **stream_hook** (*callable, optional*) – Overload the default downloading behavior with a custom stream hook.
- **hook_kwargs** (*dict, optional*) – keyword arguments to pass to the stream_hook callable in addition to the default passed params.

import_policy(*fobj: BytesIO*) → Dict

Imports the policy into the nessus scanner.

Parameters
fobj (*BytesIO*) – The file object containing the policy.

Returns
The imported policy object.

Return type
Dict

Example

```
>>> with open('policy.xml', 'rb') as policy:
...     nessus.policies.import_policy(policy)
```

list() → List[Dict]

Lists the available policies.

Returns
List of policy objects.

Return type
List[Dict]

Example

```
>>> for policy in nessus.policies.list():
...     print(policy)
```

Important

The Nessus Package is currently a Technology Preview

40.1.12 Proxy

Methods described in this section relate to the proxy API. These methods can be accessed at `Nessus.proxy`.

class ProxyAPI(*api: APISession*)

details() → Dict

Retrieves the current proxy settings

Returns
The current proxy settings

Return type
Dict

Example

```
>>> nessus.proxy.details()
```

edit(*proxy: str | None = None*, *proxy_auth: Literal['auto', 'basic', 'digest', 'none', 'ntlm'] | None = None*, *proxy_password: str | None = None*, *proxy_port: int | None = None*, *proxy_username: str | None = None*, *user_agent: str | None = None*) → None

Updates the proxy settings

Parameters

- **proxy** (*str, optional*) – The proxy host
- **proxy_auth** (*str, optional*) – The proxy auth method

- **proxy_password** (*str, optional*) – The auth password
- **proxy_port** (*int, optional*) – The proxy port
- **proxy_username** (*str, optional*) – The proxy auth username
- **user_agent** (*str, optional*) – The proxy user agent.

Example

```
>>> nessus.proxy.edit(proxy='proxy.company.com',
...                   proxy_auth='none',
...                   proxy_port=3128
...                   )
```

Important

The Nessus Package is currently a Technology Preview

40.1.13 Scanners

Methods described in this section relate to the scanners API. These methods can be accessed at `Nessus.scanners`.

class ScannersAPI(*api: APISession*)

aws_targets(*scanner_id: int*) → List[Dict]

Retrieves the AWS targets from the scanner. Only applies to AWS scanners.

Parameters
scanner_id (*int*) – Id of the scanner to call

Returns
List of AWS target objects.

Return type
List

Example

```
>>> for target in nessus.scanners.aws_targets(1):
...     print(target)
```

control_scan(*scanner_id: int, scan_uuid: str, action: Literal['stop', 'pause', 'resume']*) → None

Controls a scan currently running on a scanner.

Parameters

- **scanner_id** (*int*) – Id of the scanner to control
- **scan_uuid** (*str*) – UUID of the scan to control
- **action** (*str*) – The action to perform on the scan.

Example

```
>>> nessus.scanners.control_scan(scanner_id, scan_uuid, 'pause')
```

delete(*scanner_id: int*) → None

Delete and unlink the scanner.

Parameters
scanner_id (*int*) – Id of the scanner to delete

Example

```
>>> nessus.scanners.delete(1)
```

delete_many(*scanner_ids: List[int]*) → None

Delete and unlink many scanners.

Parameters
scanner_ids (*list[int]*) – List of scanner ids to delete

Example

```
>>> nessus.scanners.delete_many([1, 2, 3])
```

details(*scanner_id: int*) → Dict

Retrieve the details for a scanner.

Parameters
scanner_id (*int*) – Id of the scanner to retrieve

Example

```
>>> nessus.scanners.details(1)
```

link_state(*scanner_id: int*, *linked: bool*) → None

Toggles the link state of the specified scanner.

Parameters

- **scanner_id** (*int*) – Id of the scanner to modify
- **linked** (*bool*) – Should the scanner be linked?

Example

```
>>> nessus.scanners.link_state(1, linked=True)
```

list() → List[Dict]

Returns a list of scanners.

Returns
List of scanners connected to this scanner.

Return type
List

Example

```
>>> for scanner in nessus.scanners.list():
...     print(scanner)
```

running_scans(*scanner_id: int*) → List[Dict]

Retrieves the list of running scans on the scanner.

Parameters
scanner_id (*int*) – Id of the scanner to call

Returns
If scans are running on the scanner, a list of scan objects will be returned. If no scans are currently running on the scanner, then a None object will be returned.

Return type
List

Example

```
>>> scans = nessus.scanners.active_scans(1)
>>> if scans:
...     for scan in scans:
...         print(scan)
```

scanner_key(*scanner_id: int*) → str

Retrieves the scanner key for the requested scanner.

Parameters
scanner_id (*int*) – Id of the scanner to call

Returns
The scanner key

Return type
str

Example

```
>>> nessus.scanners.scanner_key(1)
```

update(*scanner_id: int*, *force_plugin_update: bool | None = None*, *force_ui_update: bool | None = None*, *finish_update: bool | None = None*, *registration_code: str | None = None*, *aws_update_interval: int | None = None*) → None

Update the scanner

Parameters

- **scanner_id** (*int*) – Id of the scanner to update
- **force_plugin_update** (*bool, optional*) – Should the scanner plugins be forcibly updated?
- **force_ui_update** (*bool, optional*) – Should the scanner UI be forcibly updated?
- **finish_update** (*bool, optional*) – Should the scanner service be restarted to run the latest software update? This is only valid if automatic updates on the scanner are disabled.
- **registration_code** (*str, optional*) – Sets the registration code for the scanner.
- **aws_update_interval** (*int, optional*) – Informs the scanner how often to check into the controlling Tenable Nessus service. This is only valid for AWS scanners.

Example

```
>>> nessus.scanners.update(1,
...                        force_plugin_update=True,
...                        force_ui_update=True
...                        )
```

Important

The Nessus Package is currently a Technology Preview

40.1.14 Scans

Methods described in this section relate to the scans API. These methods can be accessed at `Nessus.scans`.

class ScansAPI(*api: APISession*)

attachment(*scan_id: int*, *attachment_id: int*, *key: str*, *fobj: BytesIO | None = None*) → BytesIO

Returns the requested attachment file

Parameters

- **scan_id** (*int*) – The Scan to fetch from
- **attachment_id** (*int*) – The id of the scan attachment
- **key** (*str*) – The access token for the attachment
- **fobj** (*BytesIO, optional*) – File object to write to

Returns
The file object requested

Return type
BytesIO

Example

```
>>> with open('example.png', 'wb') as image_file:
...     nessus.scans.attachment(1, 1, 'something', image_file)
```

configure(*scan_id: int*, ***kwargs*) → None

Reconfigures an existing scan.

Parameters

- **scan_id** (*int*) – Id of the scan to modify
- ****kwargs** – the various settings to pass

Returns
The updated scan object

Return type
Dict

Example

```
>>> nessus.scans.configure(1, settings={
...     'name': 'Example Scan',
...     'enabled': True,
...     'text_targets': '192.168.1.1'
... })
```

copy(*scan_id: int*, *folder_id: int | None = None*, *name: str | None = None*) → Dict

Copies the scan object

Parameters

- **scan_id** (*int*) – Id of the scan to copy
- **folder_id** (*int, optional*) – Id of the destination folder
- **name** (*str, optional*) – Name of the copied scan

Returns
The copied scan object

Return type
Dict

Example

```
>>> nessus.scans.copy(1)
```

create(***kwargs*) → Dict

Creates a new scan

Parameters
****kwargs** – The parameters to pass to the API to create the scan. For information on what to pass here, consult the API documentation

Returns
The created scan object

Return type
Dict

Example

```
>>> nessus.scans.create(uuid='abcdef12345667890abcdef',
                        settings={
                            'name': 'Example Scan',
                            'enabled': False,
                            'text_targets': '192.168.1.1'
                        })
```

delete(*scan_id: int*) → None

Deletes the specified scan object

Parameters
scan_id (*int*) – Id of the scan to delete

Example

```
>>> nessus.scans.delete(1)
```

delete_history(*scan_id: int*, *history_id: int*) → None

Deletes the specified history object within a scan.

Parameters

- **scan_id** (*int*) – The scan to modify
- **history_id** (*int*) – Id of the history object to remove

Example

```
>>> nessus.scans.delete_history(1, 1)
```

delete_many(*scan_ids: List[int]*) → List

Deletes multiple scan objects

Parameters
scan_ids (*List[int]*) – List of scan ids to delete

Returns
list of deleted scans

Return type
List

Example

```
>>> nessus.scans.delete_many([1, 2, 3])
```

details(*scan_id: int*) → Dict

Returns the details for the specified scan.

Parameters
scan_id (*int*) – Id of the scan to retrieve

Example

```
>>> nessus.scans.details(1)
```

export_formats(*scan_id: int*, *schedule_id: int | None = None*) → Dict

Returns the available export formats and report options.

Parameters

- **scan_id** (*int*) – The scan to export
- **schedule_id** (*int, optional*) – The schedule id associated with the scan

Returns
The available export and report options

Return type
Dict

Example

```
>>> nessus.scans.export_formats(1)
```

export_scan(*scan_id: int*, *history_id: int | None = None*, *fobj: BytesIO | None = None*, ***kwargs*) → BytesIO

Generate a scan export or report and download it.

Parameters

- **scan_id** (*int*) – The id of the scan to export.
- **history_id** (*int, optional*) – The history id of the specific point in time to export.
- **fobj** (*BytexIO, optional*) – The file object to write the exported file to. If none is specified then a BytesIO object is written to in memory.
- **filters** (*list[tuple], optional*) – The filters to apply to the exported data.
- **format** (*str, optional*) – The exported scan format. Supported values are `nessus`, `html`, `csv`, and `db`. If unspecified, the default is `nessus`.
- **password** (*str, optional*) – The password to apply to the exported data (required for db).
- **template_id** (*int, optional*) – When exporting in HTML or PDF, what report definition should the exported data be represented within.
- **chunk_size** (*int, optional*) – The chunk sizing for the download itself.
- **stream_hook** (*callable, optional*) – Overload the default downloading behavior with a custom stream hook.
- **hook_kwargs** (*dict, optional*) – keyword arguments to pass to the stream_hook callable in addition to the default passed params.

import_scan(*fobj: BytesIO | None = None, file_id: str | None = None, folder_id: int | None = None, password: str | None = None*) → Dict

Import a scan report into the Tenable Nessus scanner. Either a file object or a file_id must be specified.

Parameters

- **fobj** (*BytesIO, optional*) – The file object to import.
- **file_id** (*str, optional*) – The id of the already uploaded file object to import.
- **folder_id** (*int, optional*) – The folder that the imported scan should reside within.
- **password** (*str, optional*) – If the file object is encrypted, this password will be used to decrypt.

Example

```
>>> with open('Example.nessus', 'rb') as reportfile:
...     nessus.scans.import_scan(reportfile)
```

kill(*scan_id: int*) → None

Forcefully terminate the currently running scan.

Parameters

scan_id (*int*) – The id of the scan to terminate.

Example

```
>>> nessus.scans.kill(1)
```

launch(*scan_id: int*, *alt_targets: List[str] | None = None*) → str

Launch a configured scan.

Parameters

- **scan_id** (*int*) – The id of the scan to launch.
- **alt_targets** (*list[str], optional*) – A List of alternative targets to run the scan against.

Example

```
>>> nessus.scan.launch(1)
```

list(*folder_id: int | None = None*, *last_modification_date: int | None = None*) → Dict

List of the available scan objects.

Parameters

- **folder_id** (*int, optional*) – Restrict the results to only the specified folder id.
- **last_modification_date** (*int, optional*) – Restrict the results to only scans modified after the specified timestamp.

Example

```
>>> for scan in nessus.scans.list():
...     print(scan)
```

pause(*scan_id: int*) → None

Pauses a currently running scans.

Parameters

scan_id (*int*) – The id of the scan to pause.

Example

```
>>> nessus.scans.pause(1)
```

plugin_output(*scan_id: int*, *host_id: int*, *plugin_id: int*, *history_id: int | None = None*) → Dict

Returns the plugin output for a specific finding within a scan.

Parameters

- **scan_id** (*int*) – The id of the scan
- **host_id** (*int*) – The id of the host within the scan
- **plugin_id** (*int*) – The plugin id of the finding on the host
- **history_id** (*int, optional*) – The id of the history object within the scan.

Returns
The restuls of the specific finding specified.

Return type
Dict

Example

```
>>> nessus.scans.plugin_output(1, 1, 19506)
```

read_status(*scan_id: int*, *read: bool*) → None

Sets the read status for the given scan.

Parameters

- **scan_id** (*int*) – The id of the scan to modify
- **read** (*bool*) – Is the scan read?

Example

```
>>> nessus.scans.read_status(1, True)
```

resume(*scan_id: int*) → None

Resumes a paused scan.

Parameters
scan_id (*int*) – The id of the scan to resume.

Example

```
>>> nessus.scans.resume(1)
```

schedule(*scan_id: int*, *enabled: bool*) → Dict

Enables/Disables the scan schedule for the given scan.

Parameters

- **scan_id** (*int*) – The id of the scan to modify
- **enabled** (*bool*) – Should the scan schedule be enabled?

Returns
The scan schedule settings.

Return type
Dict

stop(*scan_id: int*) → None

Stops a running scan

Parameters
scan_id (*int*) – The id of the scan to stop.

Example

```
>>> nessus.scans.stop(1)
```

timezones() → List[Dict]

Returns the currently configured timezone data

Returns
List of timezone objects

Return type
List[Dict]

Example

```
>>> nessus.scans.timezones()
```

Important

The Nessus Package is currently a Technology Preview

40.1.15 Server

Methods described in this section relate to the server API. These methods can be accessed at `Nessus.server`.

class ServerAPI(*api: APISession*)

properties() → Dict

Retrieves the Tenable Nessus server properties.

Returns
The various properties for this server.

Return type
Dict

Example

```
>>> nessus.server.properties()
```

restart(*reason: str | None = None*, *soft: bool | None = None*, *unlink: bool | None = None*, *when_idle: bool | None = None*) → None

Initiates a restart of this Tenable Nessus service

Parameters

- **reason** (*str, optional*) – What is the reason for the restart to occur?
- **soft** (*bool, optional*) – Should we only restart the web service (soft restart) or restart the whole Tenable Nessus service?
- **unlink** (*bool, optional*) – Should the scanner be unlinked from it's upstream controller before restarting?

- **when_idle** (*bool, optional*) – Should the scanner restart once there are no running scans?

Example

```
>>> nessus.server.restart(reason='Time to restart',
...                       when_idle=True,
...                       soft=True
...                       )
```

status() → Dict

Retrieves the current server status.

Returns
The server status

Return type
Dict

Example

```
>>> nessus.server.status()
```

Important

The Nessus Package is currently a Technology Preview

40.1.16 Session

Methods described in this section relate to the session API. These methods can be accessed at `Nessus.session`.

class SessionAPI(*api: APISession*)

api_keys() → Dict

Generates a new API key pair for the current user. The API Keys for the current session will also be updated if the auth mechanism is api keys

Returns
The newly generated keypair for the user.

Return type
Dict

Example

```
>>> nessus.session.api_keys()
```

chpasswd(*current_password*, *new_password*) → None

Updated the current user's password.

Parameters

- **current_password** (*str*) – The user's current password
- **new_password** (*str*) – The new password for the user

Example

```
>>> nessus.session.chpasswd('old_pass', 'new_pass')
```

edit(*name: str | None = None*, *email: str | None = None*) → None

Updates the current user's settings.

Parameters

- **name** (*str, optional*) – Updated name for the user
- **email** (*str, optional*) – Updated email for the user

Example

```
>>> nessus.session.edit(email='user@name.com')
```

get() → Dict

Returns the current user's session data

Returns
The session details dictionary

Return type
Dict

Example

```
>>> nessus.session.get()
```

Important

The Nessus Package is currently a Technology Preview

40.1.17 Settings

Methods described in this section relate to the Settings API. These methods can be accessed at `Nessus.settings`.

class SettingsAPI(*api: APISession*)

alerts(*start_time: int | None = None*, *end_time: int | None = None*) → List[Dict]

Returns the list of health alerts generated by the scanner

Parameters

- **start_time** (*int, optional*) – Start time to query the historical data for. Defaults to 24hrs ago.
- **end_time** (*int, optional*) – End time to query the historicat data for. Defaults to now.

Returns
List of alert objects matching the specified time range

Return type
List[Dict]

Example

```
>>> nessus.settings.alerts()
```

health() → Dict

Returns the current health statistics fore the Tenable Nessus scanner

Returns
Health stats information

Return type
Dict

Example

```
>>> nessus.settings.health()
```

list() → List[Dict]

Returns the list of advanced settings

Returns
List of settings objects.

Return type
List[Dict]

Example

```
>>> nessus.settings.list()
```

modify(*settings: List[Dict]*) → Dict

Modifies the advanced settings on the Tenable Nessus scanner. Settings objects must contain an action and a name field. They may also require a value field and/or an id field depending on the nature of the change.

Parameters
settings (*list[dict]*) – List of settings change objects

Examples

Adding a new value:

```
>>> nessus.settings.modify([{
...     'action': 'add',
...     'name': 'new_value',
...     'value': 'value_contents'
... }])
```

Updating a default setting value:

```
>>> nessus.settings.modify([{
...     'action': 'edit',
...     'name': 'allow_post_scan_editing',
...     'value': 'no'
... }])
```

Removing a setting:

```
>>> nessus.settings.modify([{
...     'action': 'remove',
...     'name': 'old_setting',
...     'id': 'abcdef1234567890abcdef'
... }])
```

Important

The Nessus Package is currently a Technology Preview

40.1.18 Software Update

Methods described in this section relate to the software update API. These methods can be accessed at `Nessus.software_update`.

class SoftwareUpdateAPI(*api: APISession*)

settings(*update: Literal['all', 'plugins', 'disabled']*, *custom_host: str | None = None*, *auto_update_delay: int | None = None*) → None

Update the software update settings

Parameters

- **update** (*str*) – What components should be updated? Expected values are `all`, `plugins`, and `disabled`.
- **custom_host** (*str, optional*) – URL of the custom plugin feed host
- **auto_update_delay** (*int, optional*) – How often should the plugin feed attempt to update (in hours)

Example

```
>>> nessus.software_update.settings(update='all',
...                                 auto_update_delay=24
...                                 )
```

update() → None

Schedules a software update for all components

Example

```
>>> nessus.software_update.update()
```

Important

The Nessus Package is currently a Technology Preview

40.1.19 Tokens

Methods described in this section relate to the tokens API. These methods can be accessed at `Nessus.tokens`.

class TokensAPI(*api: APISession*)

download(*token: str, fobj: BytesIO | None = None, chunk_size: int = 1024, stream_hook: Callable[[Response, BytesIO, int], BytesIO] | None = None, hook_kwargs: Dict | None = None*) → BytesIO

Downloads the specified token download

Parameters

- **token** (*str*) – The token to download
- **fobj** (*BytesIO, optional*) – The file object to write to. If unspecified, an in-memory object will be created and returned
- **chunk_size** (*int, optional*) – The chunk size to use when writing to the file object. The default is unspecified is `1024` bytes.
- **stream_hook** (*Callable[Response, BytesIO, int], optional*) – If specified, the output will be passed to the stream hook instead of processing ourselves.
- **hook_kwargs** (*Dict, optional*) – Any additional keyword arguments that should be passed on to the stream hook.

Returns
The file object

Return type
BytesIO

Example

```
>>> with open('file.ext', 'wb') as fobj:
...     nessus.tokens.download('1234567890', fobj=fobj)
```

status(*token: str*) → Dict

Retrieves the status of the specified token

Parameters
token (*str*) – The token to check the status of

Returns
The status response

Return type
Dict

Example

```
>>> nessus.tokens.status('1234567890')
```

Important

The Nessus Package is currently a Technology Preview

40.1.20 Users

Methods described in this section relate to the users API. These methods can be accessed at `Nessus.users`.

class UsersAPI(*api: APISession*)

api_keys(*user_id: int*) → Dict

Generates API Keys for the specified user

Parameters
user_id (*int*) – The id of the user to generate keys for

Returns
The new API Keys

Return type
Dict

Example

```
>>> nessus.users.api_keys(1)
```

chpasswd(*user_id: int*, *password: str*) → None

Change the specified user's password

Parameters

- **user_id** (*int*) – The user to update
- **password** (*str*) – The new password

Example

```
>>> nessus.users.chpasswd(1, 'n3wp@ssw0rd')
```

create(*username: str*, *password: str*, *permissions: int*, *type: Literal['local', 'ldap'] = 'local'*, *name: str | None = None*, *email: str | None = None*) → Dict

Creates a new user

Parameters

- **username** (*str*) – The unique username
- **password** (*str*) – The user's password
- **permissions** (*int*) – The permission level for the user. Basic users are 16, regular Users are 32, and administrators are 64.
- **type** (*str, optional*) – The type of user account to create. The default is `local`
- **name** (*str, optional*) – A friendly name for the user
- **email** (*str, optional*) – The user's email address

Returns
The created user object

Return type
Dict

Example

```
>>> nessus.users.create(username='example',
...                     password='s3cr3tsqu1rr3l',
...                     permissions=32,
...                     name='Example User',
...                     email='example@company.com'
...                     )
```

delete(*user_id: int*) → None

Deletes the specified user

Parameters
user_id (*int*) – The id of the user to delete

Example

```
>>> nessus.users.delete(1)
```

delete_many(*user_ids: List[int]*) → None

Deletes many users

Parameters
user_ids (*list[int]*) – The list of user ids to delete

Example

```
>>> nessus.users.delete_many([1, 2, 3])
```

details(*user_id: int*) → Dict

Retrieves the specified user object

Parameters
user_id (*int*) – The id of the user to fetch

Returns
The user object

Return type
Dict

Example

```
>>> nessus.users.details(1)
```

edit(*user_id: int*, *permissions: int*, *name: str | None = None*, *email: str | None = None*) → Dict

Updates the specified user object

Parameters

- **user_id** (*int*) – The id of the user to update
- **permissions** (*int*) – The permissions settings for the user
- **name** (*str, optional*) – The user's friendly name
- **email** (*str, optional*) – The user's email address

Returns
The updates user object

Return type
Dict

Example

```
>>> nessus.users.edit(1, 32, name='Updated User')
```

list() → List[Dict]

Retrieves the list of users configured in the system

Returns
List of user objects

Return type
List[Dict]

Example

```
>>> for user in nessus.users.list():
...     print(user)
```

CHAPTER

FORTYONE

REPORTS

41.1 Understanding Tenable Report Formats

Nessus, SecurityCenter, Tenable.io, and other Tenable applications produce data in several formats. While most of these formats are meant to be consumable by the user directly (such as PDF, CSV, and HTML reports), some of these formats are meant to be used for machine to machine transfers. The most notable example of this is the Nessus version 2 file format.

The Nessus version 2 format is a XML-based format that allows for a wide range of flexibility in providing different and varied sets of data within a singular report. While not very data dense (reports can get quite large in size), it's easily compressible and well understood.

class NessusReportv2(*fobj*)

The NessusReportv2 generator will return vulnerability items from any Nessus version 2 formatted Nessus report file. The returned data will be a python dictionary representation of the ReportItem with the relevant host properties attached. The ReportItem's structure itself will determine the resulting dictionary, what attributes are returned, and what is not.

Please note that in order to use this generator, you must install the python `lxml` package.

Parameters

fobj (*File object or string path*) – Either a File-like object or a string path pointing to the file to be parsed.

Examples

For example, if we wanted to load a Nessus report from disk and iterate through the contents, it would simply be a matter of:

```
>>> with open('example.nessus') as nessus_file:
...     report = NessusReportv2(nessus_file)
...     for item in report:
...         print(item)
```

CHAPTER

FORTYTWO

CROSS-PACKAGE TOOLING

42.1 Base Platform

The APIPlatform class is the base class that all platform packages will inherit from. Throughout pyTenable v1, packages will be transitioning to using this base class over the original APISession class.

class APIPlatform(***kwargs*)

Base class for all API Platform packages. This class handles all of the base connection logic.

Parameters

- **adaptor** (*Object, optional*) – A Requests Session adaptor to bind to the session object.
- **backoff** (*float, optional*) – If a 429 response is returned, how much do we want to backoff if the response didn't send a Retry-After header. If left unspecified, the default is 1 second.
- **box** (*bool, optional*) – Should responses be passed through Box? If left unspecified, the default is `True`.
- **box_attrs** (*dict, optional*) – Any additional attributes to pass to the Box constructor for this session? For a list of attributes that can be sent, please refer to the Box documentation for more information.
- **build** (*str, optional*) – The build number to put into the User-Agent string.
- **product** (*str, optional*) – The product name to put into the User-Agent string.
- **proxies** (*dict, optional*) – A dictionary detailing what proxy should be used for what transport protocol. This value will be passed to the session object after it has been either attached or created. For details on the structure of this dictionary, consult the proxies section of the Requests documentation.
- **retries** (*int, optional*) – The number of retries to make before failing a request. The default is 5.
- **session** (*requests.Session, optional*) – Provide a pre-built session instead of creating a requests session at instantiation.
- **squash_camel** (*bool, optional*) – Should the responses have CamelCase responses be squashed into snake_case? If left unspecified, the default value is `False`. Note that this will only work when Box is enabled.
- **ssl_verify** (*bool, optional*) – If SSL Verification needs to be disabled (for example when using a self-signed certificate), then this parameter should be set to `False` to disable verification and mask the Certificate warnings.

- **url** (*str, optional*) – The base URL that the paths will be appended onto.
- **vendor** (*str, optional*) – The vendor name to put into the User-Agent string.

delete(*path: str*, ***kwargs*) → list | dict[str, Any] | Box | BoxList | Response

Initiates an HTTP DELETE request using the specified path. Refer to the `requests.request` for more detailed information on what keyword arguments can be passed:

Parameters

- **path** (*str*) – The path to be appended onto the base URL for the request.
- ****kwargs** (*dict*) – Keyword arguments to be passed to `restfly.session.APISession._req()`.

Returns

`requests.Response` **or** `box.Box`
If the request was informed to attempt to "boxify" the response and the response was JSON data, then a Box will be returned. In all other scenarios, a Response object will be returned.

Examples

```
>>> api = APISession()
>>> resp = api.delete('/')
```

get(*path: str*, ***kwargs*) → list | dict[str, Any] | Box | BoxList | Response

Initiates an HTTP GET request using the specified path. Refer to `requests.request` for more detailed information on what keyword arguments can be passed:

Parameters

- **path** (*str*) – The path to be appended onto the base URL for the request.
- ****kwargs** (*dict*) – Keyword arguments to be passed to `restfly.session.APISession._req()`.

Returns

`requests.Response` **or** `box.Box`
If the request was informed to attempt to "boxify" the response and the response was JSON data, then a Box will be returned. In all other scenarios, a Response object will be returned.

Examples

```
>>> api = APISession()
>>> resp = api.get('/')
```

head(*path: str*, ***kwargs*) → list | dict[str, Any] | Box | BoxList | Response

Initiates an HTTP HEAD request using the specified path. Refer to the `requests.request` for more detailed information on what keyword arguments can be passed:

Parameters

- **path** (*str*) – The path to be appended onto the base URL for the request.

- ****kwargs** (*dict*) – Keyword arguments to be passed to `restfly.session.APISession._req()`.

Returns

`requests.Response` **or** `box.Box`
If the request was informed to attempt to "boxify" the response and the response was JSON data, then a Box will be returned. In all other scenarios, a Response object will be returned.

Examples

```
>>> api = APISession()
>>> resp = api.head('/')
```

patch(*path: str*, ***kwargs*) → list | dict[str, Any] | Box | BoxList | Response

Initiates an HTTP PATCH request using the specified path. Refer to the `requests.request` for more detailed information on what keyword arguments can be passed:

Parameters

- **path** (*str*) – The path to be appended onto the base URL for the request.
- ****kwargs** (*dict*) – Keyword arguments to be passed to `restfly.session.APISession._req()`.

Returns

`requests.Response` **or** `box.Box`
If the request was informed to attempt to "boxify" the response and the response was JSON data, then a Box will be returned. In all other scenarios, a Response object will be returned.

Examples

```
>>> api = APISession()
>>> resp = api.patch('/')
```

post(*path: str*, ***kwargs*) → list | dict[str, Any] | Box | BoxList | Response

Initiates an HTTP POST request using the specified path. Refer to the `requests.request` for more detailed information on what keyword arguments can be passed:

Parameters

- **path** (*str*) – The path to be appended onto the base URL for the request.
- ****kwargs** (*dict*) – Keyword arguments to be passed to `restfly.session.APISession._req()`.

Returns

`requests.Response` **or** `box.Box`
If the request was informed to attempt to "boxify" the response and the response was JSON data, then a Box will be returned. In all other scenarios, a Response object will be returned.

Examples

```
>>> api = APISession()
>>> resp = api.post('/')
```

put(*path: str*, ***kwargs*) → list | dict[str, Any] | Box | BoxList | Response

Initiates an HTTP PUT request using the specified path. Refer to the `requests.request` for more detailed information on what keyword arguments can be passed:

Parameters

- **path** (*str*) – The path to be appended onto the base URL for the request.
- ****kwargs** (*dict*) – Keyword arguments to be passed to `restfly.session.APISession._req()`.

Returns

`requests.Response` **or** `box.Box`
If the request was informed to attempt to "boxify" the response and the response was JSON data, then a Box will be returned. In all other scenarios, a Response object will be returned.

Examples

```
>>> api = APISession()
>>> resp = api.put('/')
```

42.2 GraphQL Base Module

The GraphQL module offers a simple yet flexible interface to wrap any Tenable GraphQL applications into the pyTenable SDK.

class GraphQLSession(*url: str | None = None*, *api_key: str | None = None*, *verify: bool = True*, *schema_validation: bool = True*, *retries: int = 3*, *timeout: int = 300*, *vendor: str = 'unknown'*, *product: str = 'unknown'*, *build: str = 'unknown'*)

GraphQL API Session handler using the official GQL python library instead of Restfly.

_query_folder

The location to where stored query files exist. Because of how Parent-Child relationships work within python and how that effects file locations, this MUST be set within the child class. This should almost _always_ be set to: *Path(__file__).parent.joinpath('queries')*

Type
Path

_base_path

The URI path (excluding the root /) to where the GraphQL API resides.

Type
str

_env_base

The environment variable prefix for the library.

Type
str

_client

Set as part of initialization, however is the GQL library client that will be used to interface to the GQL API.

Type
gql.Client

construct_query(*query: str | StringIO | GraphQLRequest | None = None*, *stored_file: str | None = None*) → GraphQLRequest

The query constructor takes any of the input types given and will return a DocumentNode containing the GraphQL query to be used with the query method.

As this method is called by both the query and validate methods directly, there generally isn't a need to call this outside of those two methods.

Parameters

- **query** (*str | StringIO | GraphQLRequest, optional*) – The query obj that we want to normalize into a GraphQLRequest.
- **stored_file** (*str, optional*) – The filename of a vendored (stored) graphql query to construct.

Returns
GraphQLRequest

query(*query: str | None = None*, *stored_file: str | None = None*, *keyword_arguments: dict[str, Any] | None = None*, *iterator: GraphQLIterator | None = None*, *graphql_model: str | None = None*, ***variables: Any*) → dict[str, Any]

Query the GraphQL API

Parameters

- **query** (*str | StringIO | DocumentNode, optional*) – The GraphQL query to pass to the remote API.
- **stored_file** (*str, optional*) – The filename of a vendored (stored) graphql query to construct.

> **Note**
> This parameter should not need to be used for outside of the library itself. All of the queries available with this parameter are also wrapped within the endpoint classes.

- **iterator** (*GraphQLIterator, optional*) – If specified, the response will be an instance of this iterable instead of the dictionary response. Useful for when the response data is expected to be larger datasets that would require multiple pages to collect all of the data.
- **graphql_model** (*str, optional*) – When using the iterator, we need to specify the base entity that is returned from the GraphQL response.
- **keyword_argument** (*dict, optional*) – Anything specified within this dictionary will be passed on to the gql libraries query method. While not expected to be commonly used, we're exposing this here just incase we need it.

- ****variables** (*dict, optional*) – Any variable declarations that need to be passed along with the query.

Returns

If no iterator is passed, then the response dictionary is
returned to the caller.

GraphQLIterator:
If an iterator class was passed, then the query is generated and the passed to the iterator nefore returning an instance of the iterator class.

Return type
Dict

Example

A very basic example:

```
>>> session.query('{ hero { name } }')
```

An example using a variable within the query:

```
>>> query = '''
... query HeroNameAndFriends($episode: Episode) {
...   hero(episode: $episode) {
...     name
...     friends {
...       name
...     }
...   }
... }
>>> session.query(query, episode='JEDI')
```

validate(*query: str | StringIO*) → list[GraphQLError]

Validates the query against the schema and returns any validation errors that may have occured.

Parameters
query (*str | StringIO*) – The query to validate against

Returns:

class GraphQLEndpoint(*api:* GraphQLSession)

A GraphQL Endpoint class to be used in-place of the Restfly-base endpoint adaptor.

_query(**args, **kwargs*) → Dict[str, Any] | *GraphQLIterator*

Simple helper to call the api query

class GraphQLIterator(*api, **kw*)

An iterator class to be used with GraphQL paginated/iterable datasets.

next() → Any

Ask for the next record

42.3 Base Endpoint

The APIEndpoint class is the base class that all endpoint modules will inherit from. Throughout pyTenable v1, packages will be transitioning to using this base class over the original APISession class.

class APIEndpoint(*api: APISession*)

Base API Endpoint class

42.4 Version 1 Base Classes

These classes are what pyTenable < 1.2 used for all interactions. They are here as most of the library will still use it until these have been phased out in favor of the newer RESTfly-derived classes.

As these classes exist only as a basis for the application packages, it isn't recommended to use this directly. Further if you're looking for a generic API interface to use for your own uses, take a look at the RESTfly library.

class APIResultsIterator(*api*, ***kw*)

The API iterator provides a scalable way to work through result sets of any size. The iterator will walk through each page of data, returning one record at a time. If it reaches the end of a page of records, then it will request the next page of information and then continue to return records from the next page (and the next, and the next) until the counter reaches the total number of records that the API has reported.

Note that this Iterator is used as a base model for all of the iterators, and while the mechanics of each iterator may vary, they should all behave to the user in a similar manner.

count

The current number of records that have been returned

Type
int

page

The current page of data being walked through. pages will be cycled through as the iterator requests more information from the API.

Type
list

page_count

The number of record returned from the current page.

Type
int

total

The total number of records that exist for the current request.

Type
int

next()

Ask for the next record

CHAPTER

FORTYTHREE

COMMON COMPONENTS

CHAPTER

FORTYFOUR

ERRORS

exception RestflyException(*msg*, ***kwargs*)

Base exception class that sets up logging and handles some basic scaffolding for all other exception classes. This exception should never be directly seen.

exception UnexpectedValueError(*msg*, ***kwargs*)

An unexpected value error is thrown whenever the value specified for a parameter is outside the bounds of what is expected. For example, if the parameter **a** is expected to have a value of 1, 2, or 3, and it is instead passed a value of 0, then it is an unexpected value, and this Exception should be thrown by the package.

exception RequiredParameterError(*msg*, ***kwargs*)

A Required Parameter error is thrown whenever the value specified for a parameter is required to have a value other than *None*.

exception APIError(*resp*, ***kwargs*)

The APIError Exception is a generic Exception for handling responses from the API that aren't whats expected. The APIError Exception itself attempts to provide the developer with enough information around the response to ascertain what went wrong.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception BadRequestError(*resp*, ***kwargs*)

The server cannot or will not process the request due to an apparent client error (e.g., malformed request syntax, size too large, invalid request message framing, or deceptive request routing).

Typically associated with a `400` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception UnauthorizedError(*resp*, ***kwargs*)

Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. See Basic access authentication and Digest access authentication. 401 semantically means "unauthenticated", i.e. the user does not have the necessary credentials.

Typically associated with a `401` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception ForbiddenError(*resp*, ***kwargs*)

The request was valid, but the server is refusing action. The user might not have the necessary permissions for a resource, or may need an account of some sort.

Typically associated with a `403` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception NotFoundError(*resp*, ***kwargs*)

The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible.

Typically associated with a `404` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception InvalidMethodError(*resp*, ***kwargs*)

A request method is not supported for the requested resource; for example, a GET request on a form that requires data to be presented via POST, or a PUT request on a read-only resource.

Typically associated with a 405 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception NotAcceptableError(*resp*, ***kwargs*)

The requested resource is only capable of generating content not acceptable according to the Accept headers sent in the request.

Typically associated with a 406 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception ProxyAuthenticationError(*resp*, ***kwargs*)

The client must first authenticate itself with the proxy.

Typically associated with a 407 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception RequestTimeoutError(*resp*, ***kwargs*)

The client did not produce a request within the time that the server was prepared to wait. The client MAY repeat the request without modifications at any later time.

Typically associated with a 408 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception RequestConflictError(*resp*, ***kwargs*)

Indicates that the request could not be processed because of conflict in the current state of the resource, such as an edit conflict between multiple simultaneous updates.

Typically associated with a 409 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception NoLongerExistsError(*resp*, ***kwargs*)

Indicates that the resource requested is no longer available and will not be available again. This should be used when a resource has been intentionally removed and the resource should be purged. Upon receiving a 410 status code, the client should not request the resource in the future. Clients such as search engines should remove the resource from their indices. Most use cases do not require clients and search engines to purge the resource, and a "404 Not Found" may be used instead.

Typically associated with a 410 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception LengthRequiredError(*resp*, ***kwargs*)

The request did not specify the length of its content, which is required by the requested resource.

Typically associated with a 411 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception **PreconditionFailedError**(*resp*, ***kwargs*)

The server does not meet one of the preconditions that the requester put on the request.

Typically associated with a `412` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception **PayloadTooLargeError**(*resp*, ***kwargs*)

The request is larger than the server is willing or able to process.

Typically associated with a `413` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception **URITooLongError**(*resp*, ***kwargs*)

The URI provided was too long for the server to process. Often the result of too much data being encoded as a query-string of a GET request, in which case it should be converted to a POST request.

Typically associated with a `414` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception UnsupportedMediaTypeError(*resp*, ***kwargs*)

The request entity has a media type which the server or resource does not support. For example, the client uploads an image as image/svg+xml, but the server requires that images use a different format.

Typically associated with a 415 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception RangeNotSatisfiableError(*resp*, ***kwargs*)

The client has asked for a portion of the file (byte serving), but the server cannot supply that portion. For example, if the client asked for a part of the file that lies beyond the end of the file.

Typically associated with a 416 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception ExpectationFailedError(*resp*, ***kwargs*)

The server cannot meet the requirements of the Expect request-header field.

Typically associated with a 417 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception TeapotResponseError(*resp*, ***kwargs*)

This code was defined in 1998 as one of the traditional IETF April Fools' jokes, in RFC 2324, Hyper Text Coffee Pot Control Protocol, and is not expected to be implemented by actual HTTP servers. The RFC specifies this code should be returned by teapots requested to brew coffee.

Typically associated with a 418 Status code.

code
The HTTP response code from the offending response.

Type
int

response
This is the Response object that had caused the Exception to fire.

Type
request.Response

exception MisdirectRequestError(*resp*, ***kwargs*)
The request was directed at a server that is not able to produce a response

Typically associated with a `421` Status code.

code
The HTTP response code from the offending response.

Type
int

response
This is the Response object that had caused the Exception to fire.

Type
request.Response

exception InvalidContentError(*resp*, ***kwargs*)
The request contained content that did not match the expected schema or was otherwise invalid in some way.

Typically associated with a `422` Status code.

code
The HTTP response code from the offending response.

Type
int

response
This is the Response object that had caused the Exception to fire.

Type
requests.Response

exception TooEarlyError(*resp*, ***kwargs*)
Indicates that the server is unwilling to risk processing a request that might be replayed.

Typically associated with a `425` Status code.

code
The HTTP response code from the offending response.

Type
int

response
This is the Response object that had caused the Exception to fire.

Type
request.Response

exception UpgradeRequiredError(*resp*, ***kwargs*)

The client should switch to a different protocol such as TLS/1.0, given in the Upgrade header field.

Typically associated with a 426 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception PreconditionRequiredError(*resp*, ***kwargs*)

The origin server requires the request to be conditional. Intended to prevent the 'lost update' problem, where a client GETs a resource's state, modifies it, and PUTs it back to the server, when meanwhile a third party has modified the state on the server, leading to a conflict.

Typically associated with a 428 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception TooManyRequestsError(*resp*, ***kwargs*)

The user has sent too many requests in a given amount of time. Intended for use with rate-limiting schemes.

Typically associated with a 429 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception RequestHeaderFieldsTooLargeError(*resp*, ***kwargs*)

The server is unwilling to process the request because either an individual header field, or all the header fields collectively, are too large.

Typically associated with a 431 Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception UnavailableForLegalReasonsError(*resp*, ***kwargs*)

A server operator has received a legal demand to deny access to a resource or to a set of resources that includes the requested resource.

Typically associated with a `451` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception ServerError(*resp*, ***kwargs*)

A generic error message, given when an unexpected condition was encountered and no more specific message is suitable.

Typically associated with a `500` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception MethodNotImplementedError(*resp*, ***kwargs*)

The server either does not recognize the request method, or it lacks the ability to fulfill the request. Usually this implies future availability.

Typically associated with a `501` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception **BadGatewayError**(*resp*, ***kwargs*)

The server was acting as a gateway or proxy and received an invalid response from the upstream server.

Typically associated with a `502` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception **ServiceUnavailableError**(*resp*, ***kwargs*)

The server cannot handle the request (because it is overloaded or down for maintenance). Generally, this is a temporary state.

Typically associated with a `503` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception **GatewayTimeoutError**(*resp*, ***kwargs*)

The server was acting as a gateway or proxy and did not receive a timely response from the upstream server.

Typically associated with a `504` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception **NotExtendedError**(*resp*, ***kwargs*)

Further extensions to the request are required for the server to fulfill it.

Typically associated with a `510` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

exception NetworkAuthenticationRequiredError(*resp*, ***kwargs*)

The client needs to authenticate to gain network access. Intended for use by intercepting proxies used to control access to the network

Typically associated with a `511` Status code.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

class AuthenticationWarning

An authentication warning is thrown when an unauthenticated API session is initiated.

class FileDownloadError(*resource: str*, *resource_id: str*, *filename: str*)

FileDownloadError is thrown when a file fails to download.

msg

The error message

Type
str

filename

The Filename or file id that was requested.

Type
str

resource

The resource that the file was requested from (e.g. “scans”)

Type
str

resource_id

The identifier for the resource that was requested.

Type
str

class ImpersonationError(*resp*, ***kwargs*)

An ImpersonationError exists when there is an issue with user impersonation.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

uuid

The Request UUID of the request. This can be used for the purpose of tracking the request and the response through the Tenable.io infrastructure. In the case of Non-Tenable.io products, is simply an empty string.

Type
str

class PasswordComplexityError(*resp*, ***kwargs*)

PasswordComplexityError is thrown when attempting to change a password and the password complexity is insufficient.

code

The HTTP response code from the offending response.

Type
int

response

This is the Response object that had caused the Exception to fire.

Type
request.Response

uuid

The Request UUID of the request. This can be used for the purpose of tracking the request and the response through the Tenable.io infrastructure. In the case of Non-Tenable.io products, is simply an empty string.

Type
str

class TioExportsError(*export: str*, *uuid: str*, *msg: str | None = None*)

When the exports APIs throw an error when processing an export, pyTenable will throw this error in turn to relay that context to the user.

class TioExportsTimeout(*export: str*, *uuid: str*, *msg: str | None = None*)

When an export has been cancelled due to timeout, this error is thrown.

CHAPTER

FORTYFIVE

TESTING THE LIBRARY

To run through the test suite for pyTenable, we'll need to install a few pre-requisites first. pyTenable uses py.test & VCRpy to run through the pre-recorded API calls an validate that the code is performing as intended.

```
pip install -r dev-requirements.txt
```

Once the pre-requisites are installed, it's simply a matter of running the test suite:

```
pytest --vcr-record=none --cov=tenable tests
```

If you would like to run through the test suite with a live Tenable.io container, you'll want to perform the following actions:

```
export TIO_TEST_ADMIN_ACCESS="ADMIN_API_ACCESS_KEY_HERE"
export TIO_TEST_ADMIN_SECRET="ADMIN_API_SECRET_KEY_HERE"
export TIO_TEST_STD_ACCESS="STANDARD_ACCOUNT_ACCESS_KEY_HERE"
export TIO_TEST_STD_SECRET="STANDARD_ACCOUNT_SECRET_KEY_HERE"
pytest --disable-vcr --cov=tenable.io tests/io
```

For Tenable.sc, you would run the following:

```
export SC_TEST_HOST="TENABLE_SC_IP_OR_FQDN"
export SC_TEST_USER="TENABLE_SC_SECMANAGER_USER"
export SC_TEST_PASS="TENABLE_SC_SECMANAGER_PASSWORD"
export SC_TEST_ADMIN_USER="TENABLE_SC_ADMIN_USER"
export SC_TEST_ADMIN_PASS="TENABLE_SC_ADMIN_PASSWORD"
pytest --disable-vcr --cov=tenable.sc tests/sc
```

CHAPTER

FORTYSIX

WELCOME TO PYTENABLE'S DOCUMENTATION!

pyTenable is intended to be a pythonic interface into the Tenable application APIs. Further by providing a common interface and a common structure between all of the various applications, we can ease the transition from the vastly different APIs between some of the products.

- Issue Tracker: https://github.com/tenable/pyTenable/issues
- Github Repository: https://github.com/tenable/pyTenable
- Docs: https://pytenable.readthedocs.io

46.1 Installation

To install the most recent published version to pypi, its simply a matter of installing via pip:

```
pip install pytenable
```

If you're looking for bleeding-edge, then feel free to install directly from the github repository like so:

```
pip install git+git://github.com/tenable/pytenable.git#egg=pytenable
```

46.2 Getting Started

Lets assume that we want to get the list of scans that have been run on our Tenable.io application. Performing this action is as simple as the following:

```
from tenable.io import TenableIO
tio = TenableIO(access_key='TIO_ACCESS_KEY', secret_key='TIO_SECRET_KEY')
for scan in tio.scans.list():
   print('{status}: {id}/{uuid} - {name}'.format(**scan))
```

Getting started with Tenable.sc is equally as easy:

```
from tenable.sc import TenableSC
sc = TenableSC(url='https://SC_URL', access_key='AKEY', secret_key='SKEY')
for vuln in sc.analysis.vulns():
   print('{ip}:{pluginID}:{pluginName}'.format(**vuln))
```

For more detailed information on whats available, please refer to the pyTenable Documentation

46.3 Logging

Enabling logging for pyTenable is a simple matter of enabling debug logs through the python logging package. An easy example is detailed here:

```
import logging
logging.basicConfig(level=logging.DEBUG)
```

46.4 License

The project is licensed under the MIT license.

PYTHON MODULE INDEX

t

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W