
cachet-client

Release 1.0.0

Sep 24, 2019

Contents

1	Guide	3
1.1	Install	3
1.2	Basic Usage	4
2	Reference	7
2.1	Client	7
2.2	enums	7
2.3	Ping	9
2.4	Version	9
2.5	Subscribers	10
2.6	Components	12
2.7	Component Groups	15
2.8	Incidents	17
2.9	IncidentUpdates	20
2.10	Metrics	22
2.11	Metric Points	22
2.12	Schedules	22
3	Indices and tables	23
	Python Module Index	25
	Index	27

cachet-client is a python 3.6+ client library for the open source status page system [Cachet](#) .

1.1 Install

A package is available on PyPI:

```
pip install cachet-client
```

Building from source:

```
git clone https://github.com/ZettaIO/cachet-client.git (or use ssh)
python setup.py bdist_wheel
# .whl will be located in dist/ directory and can be installed later with pip
```

1.1.1 Development Setup

Development install:

```
git clone https://github.com/ZettaIO/cachet-client.git (or use ssh)
cd cachet-client
python -m virtualenv .venv
. .venv/bin/activate
pip install -e .
```

Building docs:

```
pip install -r docs/requirements.txt
python setup.py build_sphinx
```

Running unit tests:

```
pip install -r tests/requirements.txt
tox
```

(continues on next page)

(continued from previous page)

```
# Optionally
tox -e py36 # tests only
tox -e pep8 # for pep8 run only

# Running tests with pytest also works, but this works poorly in combination
# with environment variables for the live test script (tox separates environments)
pytest tests/
```

1.1.2 Testing with real Cachet service

Do not run this script against a system in production. This is only for a test service. Cachet can easily be set up locally with docker: <https://github.com/CachetHQ/Docker>

You need to set the following environment variables:

```
CACHET_ENDPOINT
CACHET_API_TOKEN
```

Running tests:

```
python extras/live_run.py
...
=====
Nuner of tests      : 10
Succesful           : 10
Failure             : 0
Percentage passed   : 100.0%
=====
```

1.2 Basic Usage

Creating a client:

```
import cachetclient

client = cachetclient.Client(
    endpoint='https://status.test/api/v1',
    api_token='secrettoken',
)
```

Add a new subscriber with email verification:

```
sub = client.subscribers.create(email='user@example.test', verify=False)
```

List subscribers paginated:

```
# Pagination under the hood scaling better with large numbers of subscribers
for sub in client.subscribers.list(page=1, per_page=100):
    print(sub.id, sub.email, sub.verify_code)
```

Creating a component issue:


```
from cachetclient.v1 import enums

# Issue signaling to a component there is a major outage
client.incidents.create(
    name="Something blew up!",
    message="We are looking into it",
    status=enums.INCIDENT_INVESTIGATING,
    component_id=1,
    component_status=enums.COMPONENT_STATUS_MAJOR_OUTAGE,
)
```

Creating component group with components:

```
from cachetclient.v1 import enums

group = client.component_groups.create(name="Global Services")
component = client.components.create(
    name="Public website",
    status=enums.COMPONENT_STATUS_OPERATIONAL,
    description="This is a test",
    tags="test, web, something",
    group_id=group.id,
)
```


2.1 Client

`cachetclient.Client(endpoint: str = None, api_token: str = None, version: str = None, verify_tls: bool = True) → cachetclient.v1.client.Client`
Creates a cachet client. Use this function to create clients to ensure compatibility in the future.

Parameters

- **endpoint** (*str*) – The api endpoint. for example ‘`https://status.examples.test/api/v1`’. The endpoint can also be specified using the `CACHET_ENDPOINT` env variable.
- **api_token** (*str*) – The api token. Can also be specified using `CACHET_API_TOKEN` env variable.
- **version** (*str*) – The api version. If not specified the version will be derived from the endpoint url. The value “1” will create a v1 cachet client.
- **verify_tls** (*bool*) – Enable/disable tls verify. When using self signed certificates this has to be `False`.

2.2 enums

Constants / enums for various resources in cachet like component and incident status value.

2.2.1 Component Status

`cachetclient.v1.enums.COMPONENT_STATUS_OPERATIONAL`
[1] Operational. The component is working.

`cachetclient.v1.enums.COMPONENT_STATUS_PERFORMANCE_ISSUES`
[2] Performance Issues. The component is experiencing some slowness.

`cachetclient.v1.enums.COMPONENT_STATUS_PARTIAL_OUTAGE`

[3] Partial Outage. The component may not be working for everybody. This could be a geographical issue for example.

`cachetclient.v1.enums.COMPONENT_STATUS_MAJOR_OUTAGE`

[4] Major Outage. The component is not working for anybody.

`cachetclient.v1.enums.COMPONENT_STATUS_LIST`

List of all component statuses

Can be used for:

```
>> status in enums.COMPONENT_STATUS_LIST
True
```

2.2.2 Component Group Collapsed

`cachetclient.v1.enums.COMPONENT_GROUP_COLLAPSED_FALSE`

[0] No

`cachetclient.v1.enums.COMPONENT_GROUP_COLLAPSED_TRUE`

[1] Yes

`cachetclient.v1.enums.COMPONENT_GROUP_COLLAPSED_NOT_OPERATIONAL`

[2] Component is not Operational

2.2.3 Incident Status

`cachetclient.v1.enums.INCIDENT_SCHEDULED`

[0] Scheduled. This status is reserved for a scheduled status.

`cachetclient.v1.enums.INCIDENT_INVESTIGATING`

[1] Investigating. You have reports of a problem and you're currently looking into them.

`cachetclient.v1.enums.INCIDENT_IDENTIFIED`

[2] Identified. You've found the issue and you're working on a fix.

`cachetclient.v1.enums.INCIDENT_WATCHING`

[3] Watching. You've since deployed a fix and you're currently watching the situation.

`cachetclient.v1.enums.INCIDENT_FIXED`

[4] Fixed. The fix has worked, you're happy to close the incident.

`cachetclient.v1.enums.incident_status_human (status: int)`

Get human status from incident status id

Example:

```
>> incident_status_human(enums.INCIDENT_FIXED)
Fixed
```

Parameters `status (int)` – Incident status id

Returns Human status

Return type str

2.2.4 Schedule Status

`cachetclient.v1.enums.SCHEDULE_STATUS_UPCOMING`
[0] Upcoming

`cachetclient.v1.enums.SCHEDULE_STATUS_IN_PROGRESS`
[1] In progress

`cachetclient.v1.enums.SCHEDULE_STATUS_COMPLETE`
[2] Completed

2.3 Ping

2.3.1 Methods

`PingManager.get()` → bool
Check if the cachet api is responding.

Example:

```
>> client.ping.get()
True
```

Returns True if a successful response. Otherwise False.

Return type bool

`PingManager.__call__()` → bool
Shotcut for the `get` method.

Example:

```
>> client.ping()
True
```

2.4 Version

2.4.1 Resource

`Version.value`
Version string from Cachet service

Type str

`Version.on_latest`
Are we on latest version? Requires beacon enabled on server.

Type bool

`Version.latest`
Obtains info dict about latest version. Requires beacon enabled on server.

Dict format is:

```
{
  "tag_name": "v2.3.10",
  "prelease": false,
  "draft": false
}
```

Type dict

2.4.2 Manager

`VersionManager.get()` → `cachetclient.v1.version.Version`

Get version info from the server

Example:

```
>> version = client.version.get()
>> version.value
v2.3.10
```

Returns Version instance

`VersionManager.__call__()` → `cachetclient.v1.version.Version`

Shortcut to *get*

Example:

```
>> version = client.version()
>> version.value
v2.3.10
```

2.5 Subscribers

2.5.1 Resource

Methods

`Subscriber.update()`

Posts the values in the resource to the server.

Example:

```
# Change an attribute and save the resource
>> resource.value = something
>> updated_resource = resource.update()
```

Returns The updated resource from the server

`Subscriber.get(name)` → `Union[int, str, float, bool]`

Safely obtain any attribute name for the resource

Parameters `name` (*str*) – Key name in json response

Returns Value from the raw json response. If the key doesn't exist `None` is returned.

`Subscriber.delete()` → None
Deletes the resource from the server

Attributes

`Subscriber.attrs`
The raw json respons from the server
Type dict

`Subscriber.id`
Resource ID
Type int

`Subscriber.email`
email address
Type str

`Subscriber.verify_code`
Auto generated unique verify code
Type str

`Subscriber.is_global`
Is the user subscribed to all components?
Type bool

`Subscriber.created_at`
When the subscription was created
Type datetime

`Subscriber.updated_at`
Last time the subscription was updated
Type datetime

`Subscriber.verified_at`
When the subscription was verified. None if not verified
Type datetime

2.5.2 Manager

`SubscriberManager.create(*, email: str, components: List[int] = None, verify: bool = True) → cachetclient.v1.subscribers.Subscriber`
Create a subscriber. If a subscriber already exists the existing one will be returned. Note that this endpoint cannot be used to edit the user.

Keyword Arguments

- **email** (*str*) – Email address to subscribe
- **components** (*List[int]*) – The components to subscribe to. If omitted all components are subscribed.
- **verify** (*bool*) – Verification status. If `False` a verification email is sent to the user

Returns `Subscriber` instance

`SubscriberManager.list` (*page: int = 1, per_page: int = 20*) → Generator[cachetclient.v1.subscribers.Subscriber, None, None]

List all subscribers

Keyword Arguments

- **page** (*int*) – The page to start listing
- **per_page** – Number of entires per page

Returns Generator of Subscriber instances

`SubscriberManager.delete` (*subscriber_id: int*) → None

Delete a specific subscriber id

Parameters **subscriber_id** (*int*) – Subscriber id to delete

Raises `requests.exceptions.HTTPError` – if subscriber do not exist

`SubscriberManager.count` () → int

Count the total number of subscribers

Returns Number of subscribers

Return type int

2.6 Components

2.6.1 Resource

Methods

`Component.add_tag` (*name: str*) → None

Add a new tag.

Parameters **name** (*str*) – Name of the tag

`Component.del_tag` (*name: str*) → None

Delete a tag.

Parameters **name** (*str*) – Name of tag to remove

Raises `KeyError` – if tag does not exist

`Component.has_tag` (*name: str*) → bool

Check if a tag exists.

Parameters **name** (*str*) – Tag name

Returns If the tag exists

Return type bool

`Component.update` ()

Posts the values in the resource to the server.

Example:

```
# Change an attribute and save the resource
>> resource.value = something
>> updated_resource = resource.update()
```


Returns The updated resource from the server

`Component.delete()` → None
Deletes the resource from the server

Attributes

`Component.id`
The unique ID of the component

Type int

`Component.name`
Get or set name of the component

Type str

`Component.description`
Get or set component description

Type str

`Component.link`
Get or set http link to the component

Type str

`Component.status`
Get or set status id of the component (see enums)

Type int

`Component.status_name`
Human readable status representation

Type str

`Component.order`
Get or set order of the component in a group

Type int

`Component.group_id`
Get or set the component group id

Type int

`Component.enabled`
Get or set enabled state

Type bool

`Component.tags`
Get or set tags for the component
Also see [`add_tag`](#), [`del_tag`](#) and [`has_tag`](#) methods.

Type set

`Component.created_at`
When the component was created

Type datetime

`Component.updated_at`

Last time the component was updated

Type `datetime`

2.6.2 Manager

`ComponentManager.create` (*, *name*: *str*, *status*: *int*, *description*: *str* = *None*, *link*: *str* = *None*, *order*: *int* = *None*, *group_id*: *int* = *None*, *enabled*: *bool* = *True*, *tags*: *Set*[*str*] = *None*)

Create a component.

Keyword Arguments

- **name** (*str*) – Name of the component
- **status** (*int*) – Status if of the component (see enums module)
- **description** (*str*) – Description of the component (required)
- **link** (*str*) – Link to the component
- **order** (*int*) – Order of the component in its group
- **group_id** (*int*) – The group it belongs to
- **enabled** (*bool*) – Enabled status
- **tags** (*list*) – String tags

Returns `Component` instance

`ComponentManager.update` (*component_id*: *int*, *, *status*: *int*, *name*: *str* = *None*, *description*: *str* = *None*, *link*: *str* = *None*, *order*: *int* = *None*, *group_id*: *int* = *None*, *enabled*: *bool* = *None*, *tags*: *Set*[*str*] = *None*, ***kwargs*) → `cachet-client.v1.components.Component`

Update a component by id.

Parameters **component_id** (*int*) – The component to update

Keyword Arguments

- **status** (*int*) – Status of the component (see enums)
- **name** (*str*) – New name
- **description** (*str*) – New description
- **link** (*str*) – Link to component
- **order** (*int*) – Order in component group
- **group_id** (*int*) – Component group id
- **enabled** (*bool*) – Enable status of component
- **tags** (*list*) – List of strings

Returns Updated `Component` from server

`ComponentManager.list` (*page*: *int* = 1, *per_page*: *int* = 20) → `Generator[cachetclient.v1.components.Component, None, None]`

List all components

Keyword Arguments

- **page** (*int*) – The page to start listing

- **per_page** (*int*) – Number of entires per page

Returns Generator of Component instances

`ComponentManager.get (component_id: int) → cachetclient.v1.components.Component`

Get a component by id

Parameters `component_id` (*int*) – Id of the component

Returns Component instance

Raises `HttpError` – if not found

`ComponentManager.count () → int`

Count the number of components

Returns Total number of components

Return type `int`

`ComponentManager.delete (component_id: int) → None`

Delete a component

Parameters `component_id` (*int*) – Id of the component

Raises `HTTPError` – if component do not exist

2.7 Component Groups

2.7.1 Resource

Methods

`ComponentGroup.update ()`

Posts the values in the resource to the server.

Example:

```
# Change an attribute and save the resource
>> resource.value = something
>> updated_resource = resource.update()
```

Returns The updated resource from the server

`ComponentGroup.delete () → None`

Deletes the resource from the server

Attributes

`ComponentGroup.id`

Id of the component group

Type `int`

`ComponentGroup.name`

Set or get name of component group

Type `str`

`ComponentGroup.order`

Get or set order value for group

Type `int`

`ComponentGroup.collapsed`

Get or set collapsed status. See `enums` module for values.

Type `int`

`ComponentGroup.lowest_human_status`

Lowest component status, human readable

Type `str`

`ComponentGroup.is_collapsed`

Does the current collapsed value indicate the group is collapsed? Note that the collapsed value may also indicate the group is not operational.

Type `bool`

`ComponentGroup.is_open`

Does the current collapsed value indicate the group is open? Note that the collapsed value may also indicate the group is not operational.

Type `bool`

`ComponentGroup.is_operational`

Does the current collapsed value indicate the group not operational?

Type `bool`

`ComponentGroup.created_at`

When the group was created

Type `datetime`

`ComponentGroup.updated_at`

Last time updated

Type `datetime`

2.7.2 Manager

`ComponentGroupManager.create` (*, *name*: *str*, *order*: *int* = 0, *collapsed*: *int* = 0) → `cachet-client.v1.component_groups.ComponentGroup`

Create a component group

Keyword Arguments

- **name** (*str*) – Name of the group
- **order** (*int*) – group order
- **collapsed** (*int*) – Collapse value (see `enums`)

Returns `ComponentGroup` instance

`ComponentGroupManager.update` (*group_id*: *int*, *, *name*: *str*, *order*: *int* = None, *collapsed*: *int* = None, ***kwargs*) → `cachet-client.v1.component_groups.ComponentGroup`

Update component group

Parameters **group_id** (*int*) – The group id to update

Keyword Arguments

- **name** (*str*) – New name for group
- **order** (*int*) – Order value of the group
- **collapsed** (*int*) – Collapsed value. See enums module.

`ComponentGroupManager.count()` → `int`

Count the number of component groups

Returns Number of component groups

Return type `int`

`ComponentGroupManager.list` (*page: int = 1, per_page: int = 20*) → `Generator[cachetclient.v1.component_groups.ComponentGroup, None, None]`

List all component groups

Keyword Arguments

- **page** (*int*) – The page to start listing
- **per_page** – Number of entires per page

Returns Generator of `ComponentGroup` instances

`ComponentGroupManager.get` (*group_id*) → `cachetclient.v1.component_groups.ComponentGroup`

Get a component group by id

Parameters `group_id` (*int*) – Id of the component group

Returns `ComponentGroup` instance

Raises `requests.exceptions.HTTPError` – if not found

`ComponentGroupManager.delete` (*group_id: int*) → `None`

Delete a component group

Parameters `group_id` (*int*) – Id of the component

Raises `requests.exceptions.HTTPError` – if not found

2.8 Incidents

2.8.1 Resource

Methods

`Incident.updates()` → `Generator[cachetclient.v1.incidents.Incident, None, None]`

`Generator['Incident', None, None]`: Incident updates for this issue

`Incident.update()`

Posts the values in the resource to the server.

Example:

```
# Change an attribute and save the resource
>> resource.value = something
>> updated_resource = resource.update()
```

Returns The updated resource from the server

`Incident.delete()` → None
Deletes the resource from the server

Attributes

`Incident.id`
unique id of the incident

Type int

`Incident.component_id`
Get or set component id for this incident

Type int

`Incident.name`
Get or set name/title of the incident

Type str

`Incident.message`
Get or set message

Type str

`Incident.notify`
Get or set notification flag

Type bool

`Incident.status`
Get or set status. See `enums`

Type int

`Incident.human_status`
Human representation of the status

Type str

`Incident.visible`
Get or set visibility of the incident

Type bool

`Incident.scheduled_at`
Scheduled time. This is used for scheduled events like maintenance in Cachet 2.3 where incident status is `INCIDENT_SCHEDULED`. 2.4 has its own schedule resource and endpoints.

Type datetime

`Incident.created_at`
When the issue was created

Type datetime

`Incident.updated_at`
Last time the issue was updated

Type datetime

`Incident.deleted_at`

When the issue was deleted

Type datetime

2.8.2 Manager

`IncidentManager.create` (*, *name: str, message: str, status: int, visible: bool = True, component_id: int = None, component_status: int = None, notify: bool = True, created_at: datetime.datetime = None, template: str = None, template_vars: List[str] = None*) → `cachetclient.v1.incidents.Incident`

Create and general issue or issue for a component. `component_id` and `component_status` must be supplied when making a component issue.

Keyword Arguments

- **name** (*str*) – Name/title of the issue
- **message** (*str*) – Message body for the issue
- **status** (*int*) – Status of the incident (see enums)
- **visible** (*bool*) – Publicly visible incident
- **component_id** (*int*) – The component to update
- **component_status** (*int*) – The status to apply on component
- **notify** (*bool*) – If users should be notified
- **created_at** – when the incident was created
- **template** (*str*) – Slug of template to use
- **template_vars** (*list*) – Variables to the template

Returns Incident instance

`IncidentManager.update` (*incident_id: int, name: str = None, message: str = None, status: int = None, visible: bool = None, component_id: int = None, component_status: int = None, notify: bool = True, created_at: datetime.datetime = None, template: str = None, template_vars: List[str] = None, **kwargs*) → `cachetclient.v1.incidents.Incident`

Update an incident.

Parameters `incident_id` (*int*) – The incident to update

Keyword Arguments

- **name** (*str*) – Name/title of the issue
- **message** (*str*) – Message body for the issue
- **status** (*int*) – Status of the incident (see enums)
- **visible** (*bool*) – Publicly visible incident
- **component_id** (*int*) – The component to update
- **component_status** (*int*) – The status to apply on component
- **notify** (*bool*) – If users should be notified
- **created_at** – when the incident was created
- **template** (*str*) – Slug of template to use

- **template_vars** (*list*) – Variables to the template

Returns Updated incident Instance

`IncidentManager.list (page: int = 1, per_page: int = 1) → Generator[cachetclient.v1.incidents.Incident, None, None]`

List all incidents paginated

Keyword Arguments

- **page** (*int*) – Page to start on
- **per_page** (*int*) – Entries per page

Returns Generator of :py:data: Incident's

`IncidentManager.get (incident_id: int) → cachetclient.v1.incidents.Incident`

Get a single incident

Parameters **incident_id** (*int*) – The incident id to get

Returns Incident instance

Raises `requests.exception.HTTPError` – if incident do not exist

`IncidentManager.count () → int`

Count the number of incidents

Returns Total number of incidents

Return type `int`

`IncidentManager.delete (incident_id: int) → None`

Delete an incident

Parameters **incident_id** (*int*) – The incident id

2.9 IncidentUpdates

2.9.1 Resource

Methods

`IndicentUpdate.update () → cachetclient.v1.incident_updates.IndicentUpdate`

Update/save changes

Returns Updated IncidentUpdate instance

`IndicentUpdate.delete () → None`

Deletes the incident update

Attributes

`IndicentUpdate.id`

Resource id

Type `int`

`IndicentUpdate.incident_id`

The incident id this update belongs to

Type int

`IndicentUpdate.status`

Get or set incident status. See enums.

Type int

`IndicentUpdate.message`

Get or set message

Type str

`IndicentUpdate.user_id`

The user id creating the update

Type int

`IndicentUpdate.created_at`

when the resource was created

Type datetime

`IndicentUpdate.updated_at`

When the resource as last updated

Type datetime

`IndicentUpdate.human_status`

Human readable status

Type str

2.9.2 Manager

`IncidentUpdatesManager.create(*, incident_id: int, status: int, message: str) → cachet-client.v1.incident_updates.IndicentUpdate`

Create an incident update

Keyword Arguments

- **incident_id**(*int*) – The incident to update
- **status**(*int*) – New status id
- **message**(*str*) – Update message

Returns `IndicentUpdate` instance

`IncidentUpdatesManager.update(*, id: int, incident_id: int, status: int = None, message: str = None, **kwargs) → cachet-client.v1.incident_updates.IndicentUpdate`

Update an incident update

Parameters

- **incident_id**(*int*) – The incident
- **id**(*int*) – The incident update id to update

Keyword Arguments

- **status**(*int*) – New status id
- **message**(*str*) – New message

Returns The updated `IndicentUpdate` instance

`IncidentUpdatesManager.count(incident_id) → int`

Count the number of incident update for an incident

Parameters `incident_id(int)` – The incident

Returns Number of incident updates for the incident

Return type `int`

`IncidentUpdatesManager.list(incident_id: int, page: int = 1, per_page: int = 20) → Generator[cachetclient.v1.incident_updates.IndicentUpdate, None, None]`

List updates for an issue

Parameters `incident_id` – The incident to list updates

Keyword Arguments

- `page(int)` – The first page to request
- `per_page(int)` – Entires per page

Returns Generator of :py:data:‘IncidentUpdate’s

`IncidentUpdatesManager.get(incident_id: int, update_id: int) → cachet-client.v1.incident_updates.IndicentUpdate`

Get an incident update

Parameters

- `incident_id(int)` – The incident
- `update_id(int)` – The indicent update id to obtain

Returns `IncidentUpdate` instance

`IncidentUpdatesManager.delete(incident_id: int, update_id: int) → None`

Delete an incident update

2.10 Metrics

Needs testing and implementation

2.11 Metric Points

Needs testing and implementation.

2.12 Schedules

Needs testing and implementation. For now stick to issues using status `INCIDENT_SCHEDULED`.

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`
- `search`

C

- `cachetclient`, [7](#)
- `cachetclient.v1.component_groups`, [15](#)
- `cachetclient.v1.components`, [12](#)
- `cachetclient.v1.enums`, [7](#)
- `cachetclient.v1.incident_updates`, [20](#)
- `cachetclient.v1.incidents`, [17](#)
- `cachetclient.v1.ping`, [9](#)
- `cachetclient.v1.subscribers`, [10](#)
- `cachetclient.v1.version`, [9](#)

Symbols

`__call__()` (*cachetclient.v1.ping.PingManager* method), 9
`__call__()` (*cachetclient.v1.version.VersionManager* method), 10

A

`add_tag()` (*cachetclient.v1.components.Component* method), 12
`attrs` (*cachetclient.v1.subscribers.Subscriber* attribute), 11

C

`cachetclient` (module), 7
`cachetclient.v1.component_groups` (module), 15
`cachetclient.v1.components` (module), 12
`cachetclient.v1.enums` (module), 7
`cachetclient.v1.incident_updates` (module), 20
`cachetclient.v1.incidents` (module), 17
`cachetclient.v1.ping` (module), 9
`cachetclient.v1.subscribers` (module), 10
`cachetclient.v1.version` (module), 9
`Client()` (in module *cachetclient*), 7
`collapsed` (*cachetclient.v1.component_groups.ComponentGroup* attribute), 16
`COMPONENT_GROUP_COLLAPSED_FALSE` (in module *cachetclient.v1.enums*), 8
`COMPONENT_GROUP_COLLAPSED_NOT_OPERATIONAL` (in module *cachetclient.v1.enums*), 8
`COMPONENT_GROUP_COLLAPSED_TRUE` (in module *cachetclient.v1.enums*), 8
`component_id` (*cachetclient.v1.incidents.Incident* attribute), 18
`COMPONENT_STATUS_LIST` (in module *cachetclient.v1.enums*), 8
`COMPONENT_STATUS_MAJOR_OUTAGE` (in module *cachetclient.v1.enums*), 8

`COMPONENT_STATUS_OPERATIONAL` (in module *cachetclient.v1.enums*), 7
`COMPONENT_STATUS_PARTIAL_OUTAGE` (in module *cachetclient.v1.enums*), 7
`COMPONENT_STATUS_PERFORMANCE_ISSUES` (in module *cachetclient.v1.enums*), 7
`count()` (*cachetclient.v1.component_groups.ComponentGroupManager* method), 17
`count()` (*cachetclient.v1.components.ComponentManager* method), 15
`count()` (*cachetclient.v1.incident_updates.IncidentUpdatesManager* method), 21
`count()` (*cachetclient.v1.incidents.IncidentManager* method), 20
`count()` (*cachetclient.v1.subscribers.SubscriberManager* method), 12
`create()` (*cachetclient.v1.component_groups.ComponentGroupManager* method), 16
`create()` (*cachetclient.v1.components.ComponentManager* method), 14
`create()` (*cachetclient.v1.incident_updates.IncidentUpdatesManager* method), 21
`create()` (*cachetclient.v1.incidents.IncidentManager* method), 19
`create()` (*cachetclient.v1.subscribers.SubscriberManager* method), 11
`created_at` (*cachetclient.v1.component_groups.ComponentGroup* attribute), 16
`created_at` (*cachetclient.v1.components.Component* attribute), 13
`created_at` (*cachetclient.v1.incident_updates.IncidentUpdate* attribute), 21
`created_at` (*cachetclient.v1.incidents.Incident* attribute), 18
`created_at` (*cachetclient.v1.subscribers.Subscriber* attribute), 11

D

del_tag() (cachetclient.v1.components.Component method), 12

delete() (cachetclient.v1.component_groups.ComponentGroup attribute), 21

delete() (cachetclient.v1.component_groups.ComponentGroupManager attribute), 18

delete() (cachetclient.v1.components.Component method), 13

delete() (cachetclient.v1.components.ComponentManager attribute), 15

delete() (cachetclient.v1.components.ComponentManager method), 15

delete() (cachetclient.v1.incident_updates.IncidentUpdatesManager attribute), 17

delete() (cachetclient.v1.incident_updates.IncidentUpdatesManager method), 22

delete() (cachetclient.v1.incident_updates.IndicentUpdate attribute), 20

delete() (cachetclient.v1.incident_updates.IndicentUpdate method), 20

delete() (cachetclient.v1.incidents.Incident attribute), 18

delete() (cachetclient.v1.incidents.IncidentManager attribute), 20

delete() (cachetclient.v1.incidents.IncidentManager method), 20

delete() (cachetclient.v1.subscribers.Subscriber attribute), 11

delete() (cachetclient.v1.subscribers.SubscriberManager attribute), 12

delete() (cachetclient.v1.subscribers.SubscriberManager method), 12

deleted_at (cachetclient.v1.incidents.Incident attribute), 18

description (cachetclient.v1.components.Component attribute), 13

E

email (cachetclient.v1.subscribers.Subscriber attribute), 11

enabled (cachetclient.v1.components.Component attribute), 13

G

get() (cachetclient.v1.component_groups.ComponentGroupManager attribute), 17

get() (cachetclient.v1.component_groups.ComponentGroupManager method), 17

get() (cachetclient.v1.components.ComponentManager attribute), 15

get() (cachetclient.v1.components.ComponentManager method), 15

get() (cachetclient.v1.incident_updates.IncidentUpdatesManager attribute), 16

get() (cachetclient.v1.incident_updates.IncidentUpdatesManager method), 22

get() (cachetclient.v1.incidents.IncidentManager attribute), 20

get() (cachetclient.v1.incidents.IncidentManager method), 20

get() (cachetclient.v1.ping.PingManager method), 9

get() (cachetclient.v1.subscribers.Subscriber method), 10

get() (cachetclient.v1.version.VersionManager attribute), 10

get() (cachetclient.v1.version.VersionManager method), 10

group_id (cachetclient.v1.components.Component attribute), 13

get() (cachetclient.v1.components.ComponentManager attribute), 15

get() (cachetclient.v1.components.ComponentManager method), 15

get() (cachetclient.v1.incident_updates.IncidentUpdatesManager attribute), 17

get() (cachetclient.v1.incident_updates.IncidentUpdatesManager method), 22

get() (cachetclient.v1.incidents.IncidentManager attribute), 20

get() (cachetclient.v1.incidents.IncidentManager method), 20

get() (cachetclient.v1.subscribers.Subscriber attribute), 11

get() (cachetclient.v1.subscribers.SubscriberManager attribute), 12

get() (cachetclient.v1.subscribers.SubscriberManager method), 12

get() (cachetclient.v1.version.VersionManager attribute), 10

get() (cachetclient.v1.version.VersionManager method), 10

group_id (cachetclient.v1.components.Component attribute), 13

H

has_tag() (cachetclient.v1.components.Component

method), 12

human_status (cachetclient.v1.incident_updates.IndicentUpdate attribute), 21

human_status (cachetclient.v1.incidents.Incident attribute), 18

id (cachetclient.v1.component_groups.ComponentGroup attribute), 15

id (cachetclient.v1.components.Component attribute), 13

id (cachetclient.v1.incident_updates.IncidentUpdatesManager attribute), 17

id (cachetclient.v1.incident_updates.IncidentUpdatesManager method), 22

id (cachetclient.v1.incident_updates.IndicentUpdate attribute), 20

id (cachetclient.v1.incident_updates.IndicentUpdate method), 20

id (cachetclient.v1.incidents.Incident attribute), 18

id (cachetclient.v1.subscribers.Subscriber attribute), 11

INCIDENT_FIXED (in module cachetclient.v1.enums), 8

incident_id (cachetclient.v1.incident_updates.IndicentUpdate attribute), 20

INCIDENT_IDENTIFIED (in module cachetclient.v1.enums), 8

INCIDENT_INVESTIGATING (in module cachetclient.v1.enums), 8

INCIDENT_SCHEDULED (in module cachetclient.v1.enums), 8

incident_status_human() (in module cachetclient.v1.enums), 8

INCIDENT_WATCHING (in module cachetclient.v1.enums), 8

is_collapsed (cachetclient.v1.component_groups.ComponentGroup attribute), 16

is_global (cachetclient.v1.subscribers.Subscriber attribute), 11

is_managed (cachetclient.v1.component_groups.ComponentGroup attribute), 16

is_operational (cachetclient.v1.component_groups.ComponentGroup attribute), 16

latest (cachetclient.v1.version.Version attribute), 9

link (cachetclient.v1.components.Component attribute), 13

list() (cachetclient.v1.component_groups.ComponentGroupManager method), 17

list() (cachetclient.v1.components.ComponentManager method), 14

list() (cachetclient.v1.incident_updates.IncidentUpdatesManager method), 22

list() (cachetclient.v1.incidents.IncidentManager method), 20

list() (cachetclient.v1.subscribers.Subscriber attribute), 11

list() (cachetclient.v1.subscribers.SubscriberManager attribute), 12

list() (cachetclient.v1.version.VersionManager attribute), 10

list() (cachetclient.v1.version.VersionManager method), 10

list() (cachetclient.v1.incident_updates.IncidentUpdatesManager attribute), 17

list() (cachetclient.v1.incident_updates.IncidentUpdatesManager method), 22

list() (cachetclient.v1.incidents.IncidentManager attribute), 20

list() (cachetclient.v1.incidents.IncidentManager method), 20

list() (cachetclient.v1.subscribers.Subscriber attribute), 11

list() (cachetclient.v1.subscribers.SubscriberManager attribute), 12

list() (cachetclient.v1.version.VersionManager attribute), 10

list() (cachetclient.v1.version.VersionManager method), 10

`list()` (*cachetclient.v1.subscribers.SubscriberManager* method), 11

`lowest_human_status` (*cachet-client.v1.component_groups.ComponentGroup* attribute), 16

M

`message` (*cachetclient.v1.incident_updates.IndicentUpdate* attribute), 21

`message` (*cachetclient.v1.incidents.Incident* attribute), 18

N

`name` (*cachetclient.v1.component_groups.ComponentGroup* attribute), 15

`name` (*cachetclient.v1.components.Component* attribute), 13

`name` (*cachetclient.v1.incidents.Incident* attribute), 18

`notify` (*cachetclient.v1.incidents.Incident* attribute), 18

O

`on_latest` (*cachetclient.v1.version.Version* attribute), 9

`order` (*cachetclient.v1.component_groups.ComponentGroup* attribute), 15

`order` (*cachetclient.v1.components.Component* attribute), 13

S

`SCHEDULE_STATUS_COMPLETE` (in module *cachet-client.v1.enums*), 9

`SCHEDULE_STATUS_IN_PROGRESS` (in module *cachetclient.v1.enums*), 9

`SCHEDULE_STATUS_UPCOMING` (in module *cachet-client.v1.enums*), 9

`scheduled_at` (*cachetclient.v1.incidents.Incident* attribute), 18

`status` (*cachetclient.v1.components.Component* attribute), 13

`status` (*cachetclient.v1.incident_updates.IndicentUpdate* attribute), 21

`status` (*cachetclient.v1.incidents.Incident* attribute), 18

`status_name` (*cachet-client.v1.components.Component* attribute), 13

T

`tags` (*cachetclient.v1.components.Component* attribute), 13

U

`update()` (*cachetclient.v1.component_groups.ComponentGroup* method), 15

`update()` (*cachetclient.v1.component_groups.ComponentGroupManager* method), 16

`update()` (*cachetclient.v1.components.Component* method), 12

`update()` (*cachetclient.v1.components.ComponentManager* method), 14

`update()` (*cachetclient.v1.incident_updates.IncidentUpdatesManager* method), 21

`update()` (*cachetclient.v1.incident_updates.IndicentUpdate* method), 20

`update()` (*cachetclient.v1.incidents.Incident* method), 17

`update()` (*cachetclient.v1.incidents.IncidentManager* method), 19

`update()` (*cachetclient.v1.subscribers.Subscriber* method), 10

`updated_at` (*cachet-client.v1.component_groups.ComponentGroup* attribute), 16

`updated_at` (*cachetclient.v1.components.Component* attribute), 13

`updated_at` (*cachet-client.v1.incident_updates.IndicentUpdate* attribute), 21

`updated_at` (*cachetclient.v1.incidents.Incident* attribute), 18

`updated_at` (*cachetclient.v1.subscribers.Subscriber* attribute), 11

`updates()` (*cachetclient.v1.incidents.Incident* method), 17

`user_id` (*cachetclient.v1.incident_updates.IndicentUpdate* attribute), 21

V

`value` (*cachetclient.v1.version.Version* attribute), 9

`verified_at` (*cachetclient.v1.subscribers.Subscriber* attribute), 11

`verify_code` (*cachetclient.v1.subscribers.Subscriber* attribute), 11

`visible` (*cachetclient.v1.incidents.Incident* attribute), 18