freshbooks-sdk

Release 1.0.1

Andrew McIntosh

Apr 12, 2022

USER GUIDE

CHAPTER

ONE

CONFIGURING THE API CLIENT

You can create an instance of the API client in one of two ways:

- By providing your application's OAuth2 `client_id` and `client_secret` and following through the auth flow, which when complete will return an access token
- Or if you already have a valid access token, you can instantiate the client directly using that token, however token refresh flows will not function without the application id and secret.

```
from freshbooks import Client

freshBooksClient = Client(
    client_id=<your application id>,
    client_secret=<your application secret>,
    redirect_uri=<your redirect uri>
)
```

and then proceed with the auth flow (see below).

Or

```
from freshbooks import Client

freshBooksClient = Client(
    client_id=<your application id>,
    access_token=<a valid token>
)
```

CHAPTER

TWO

AUTHORIZATION FLOW

This is a brief summary of the OAuth2 authorization flow and the methods in the FreshBooks API Client around them. See the FreshBooks API - Authentication documentation.

First, instantiate your Client with `client_id`, `client_secret`, and `redirect_uri` as above.

To get an access token, the user must first authorize your application. This can be done by sending the user to the FreshBooks authorization page. Once the user has clicked accept there, they will be redirected to your `redirect_uri` with an access grant code. The authorization URL can be obtained by calling `freshBooksClient.get_auth_request_url()`. This method also accepts a list of scopes that you wish the user to authorize your application for.

```
auth_url = freshBooksClient.get_auth_request_url(['user:profile:read', 'user:clients:read
↪'])
```

Once the user has been redirected to your `redirect_uri` and you have obtained the access grant code, you can exchange that code for a valid access token.

```
auth_results = freshBooksClient.get_access_token(access_grant_code)
```

This call both sets the `access_token`, `refresh_token`, and `access_token_expires_at` fields on you Client instance, and returns those values.

```
>>> auth_results.access_token
<some token>

>>> auth_results.refresh_token
<some refresh token>

>>> auth_results.access_token_expires_at
<datetime object>
```

When the token expires, it can be refreshed with the `refresh_token` value in the Client:

```
>>> auth_results = freshBooksClient.refresh_access_token()
>>> auth_results.access_token
<a new token>
```

or you can pass the refresh token yourself:

```
>>> auth_results = freshBooksClient.refresh_access_token(stored_refresh_token)
>>> auth_results.access_token
<a new token>
```

2.1 Current User

FreshBooks users are uniquely identified by their email across our entire product. One user may act on several Businesses in different ways, and our Identity model is how we keep track of it. Each unique user has an Identity, and each Identity has Business Memberships which define the permissions they have.

See FreshBooks API - Business, Roles, and Identity and FreshBooks API - The Identity Model.

The current user can be accessed by:

```
>>> current_user = freshBooksClient.current_user()
>>> current_user.email
<some email>

>>> current_user.business_memberships
<list of businesses>
```

CHAPTER

THREE

MAKING API CALLS

Each resource in the client provides calls for `get`, `list`, `create`, `update` and `delete` calls. Please note that some API resources are scoped to a FreshBooks `account_id` while others are scoped to a `business_id`. In general these fall along the lines of accounting resources vs projects/time tracking resources, but that is not precise.

```
client = freshBooksClient.clients.get(account_id, client_user_id)
project = freshBooksClient.projects.get(business_id, project_id)
```

3.1 Get and List

API calls which return a single resource return a `Result` object with the returned data accessible via attributes. The raw json-parsed dictionary can also be accessed via the `data` attribute.

```
client = freshBooksClient.clients.get(account_id, client_user_id)

assert client.organization == "FreshBooks"
assert client.userid == client_user_id

assert client.data["organization"] == "FreshBooks"
assert client.data["userid"] == client_user_id
```

`vis_state` returns an Enum. See FreshBooks API - Active and Deleted Objects for details.

```
from freshbooks import VisState

assert client.vis_state == VisState.ACTIVE
assert client.vis_state == 0
assert client.data['vis_state'] == VisState.ACTIVE
assert client.data['vis_state'] == 0
```

API calls which return a list of resources return a `ListResult` object. The resources in the list can be accessed by index and iterated over. Similarly, the raw dictionary can be accessed via the `data` attribute.

```
clients = freshBooksClient.clients.list(account_id)

assert clients[0].organization == "FreshBooks"

assert clients.data["clients"][0]["organization"] == "FreshBooks"

for client in clients:
```

(continues on next page)

(continued from previous page)

```
    assert client.organization == "FreshBooks"
    assert client.data["organization"] == "FreshBooks"
```

3.2 Create, Update, and Delete

API calls to create and update take a dictionary of the resource data. A successful call will return a `Result` object as if a `get` call.

Create:

```
payload = {"email": "john.doe@abcorp.com"}
new_client = FreshBooksClient.clients.create(account_id, payload)

client_id = new_client.userid
```

Update:

```
payload = {"email": "john.doe@abcorp.ca"}
client = freshBooksClient.clients.update(account_id, client_id, payload)

assert client.email == "john.doe@abcorp.ca"
```

Delete:

```
client = freshBooksClient.clients.delete(account_id, client_id)

assert client.vis_state == VisState.DELETED
```

3.3 Error Handling

Calls made to the FreshBooks API with a non-2xx response are wrapped in a `FreshBooksError` exception. This exception class contains the error message, HTTP response code, FreshBooks-specific error number if one exists, and the HTTP response body.

Example:

```
from freshbooks import FreshBooksError

try:
    client = freshBooksClient.clients.get(account_id, client_id)
except FreshBooksError as e:
    assert str(e) == "Client not found."
    assert e.status_code == 404
    assert e.error_code == 1012
    assert e.raw_response ==  ("{'response': {'errors': [{'errno': 1012, "
                               "'field': 'userid', 'message': 'Client not found.', "
                               "'object': 'client', 'value': '134'}]}}")
```

Not all resources have full CRUD methods available. For example expense categories have `list` and `get` calls, but are not deletable. If you attempt to call a method that does not exist, the SDK will raise a

`FreshBooksNotImplementedError` exception, but this is not something you will likely have to account for outside of development.

3.4 Pagination, Filters, and Includes

`list` calls take a list of builder objects that can be used to paginate, filter, and include optional data in the response. See FreshBooks API - Parameters documentation.

3.4.1 Pagination

Pagination results are included in `list` responses in the `pages` attribute:

```
>>> clients = freshBooksClient.clients.list(account_id)
>>> clients.pages
PageResult(page=1, pages=1, per_page=30, total=6)

>>> clients.pages.total
6
```

To make a paginated call, first create a `PaginateBuilder` object that can be passed into the `list` method.

```
>>> from freshbooks import PaginateBuilder

>>> paginator = PaginateBuilder(2, 4)
>>> paginator
PaginateBuilder(page=2, per_page=4)

>>> clients = freshBooksClient.clients.list(account_id, builders=[paginator])
>>> clients.pages
PageResult(page=2, pages=3, per_page=4, total=9)
```

`PaginateBuilder` has methods `page` and `per_page` to return or set the values. When setting the values the calls can be chained.

```
>>> paginator = PaginateBuilder(1, 3)
>>> paginator
PaginateBuilder(page=1, per_page=3)

>>> paginator.page()
1

>>> paginator.page(2).per_page(4)
>>> paginator
PaginateBuilder(page=2, per_page=4)
```

ListResults can be combined, allowing your to use pagination to get all the results of a resource.

```
paginator = PaginateBuilder(1, 100)
clients = freshBooksClient.clients.list(self.account_id, builders=[paginator])
while clients.pages.page < clients.pages.pages:
    paginator.page(clients.pages.page + 1)
```

(continues on next page)

(continued from previous page)

```
    new_clients = freshBooksClient.clients.list(self.account_id, builders=[paginator])
    clients = clients + new_clients
```

3.4.2 Filters

To filter which results are return by `list` method calls, construct a `FilterBuilder` and pass that in the list of builders to the `list` method.

```
>>> from freshbooks import FilterBuilder

>>> filter = FilterBuilder()
>>> filter.equals("userid", 123)

>>> clients = freshBooksClient.clients.list(account_id, builders=[filter])
```

Filters can be built with the methods: `equals`, `in_list`, `like`, `between`, and `boolean`, which can be chained together.

```
>>> f = FilterBuilder()
>>> f.like("email_like", "@freshbooks.com")
FilterBuilder(&search[email_like]=@freshbooks.com)

>>> f = FilterBuilder()
>>> f.in_list("clientids", [123, 456]).boolean("active", False)
FilterBuilder(&search[clientids][]=123&search[clientids][]=456&active=False)

>>> f = FilterBuilder()
>>> f.boolean("active", False).in_list("clientids", [123, 456])
FilterBuilder(&active=False&search[clientids][]=123&search[clientids][]=456)

>>> f = FilterBuilder()
>>> f.between("amount", 1, 10)
FilterBuilder(&search[amount_min]=1&search[amount_max]=10)

>>> f = FilterBuilder()
>>> f.between("start_date", date.today())
FilterBuilder(&search[start_date]=2020-11-21)
```

3.4.3 Includes

To include additional relationships, sub-resources, or data in a response an `IncludesBuilder` can be constructed.

```
>>> from freshbooks import IncludesBuilder

>>> includes = IncludesBuilder()
>>> includes.include("outstanding_balance")
IncludesBuilder(&include[]=outstanding_balance)
```

Which can then be passed into `list` or `get` calls:

```
>>> clients = freshBooksClient.clients.list(account_id, builders=[includes])
>>> clients[0].outstanding_balance
[{'amount': {'amount': '100.00', 'code': 'USD'}}]

>>> client = freshBooksClient.clients.get(account_id, client_id, includes=includes)
>>> client.outstanding_balance
[{'amount': {'amount': '100.00', 'code': 'USD'}}]
```

Includes can also be passed into `create` and `update` calls to include the data in the response of the updated resource:

```
>>> payload = {"email": "john.doe@abcorp.com"}
>>> new_client = FreshBooksClient.clients.create(account_id, payload, includes=includes)
>>> new_client.outstanding_balance
[]  # New client has no balance
```

3.5 Dates and Times

For historical reasons, some resources in the FreshBooks API (mostly accounting-releated) return date/times in "US/Eastern" timezone. Some effort is taken to return `datetime` objects as zone-aware and normalized to UTC. In these cases, the raw response string will differ from the attribute. For example:

```
from datetime import datetime, timezone

assert client.data["updated"] == "2021-04-16 10:31:59"  # Zone-naive string in "US/
↪Eastern"
assert client.updated.isoformat() == '2021-04-16T14:31:59+00:00'  # Zone-aware datetime␣
↪in UTC
assert client.updated == datetime(year=2021, month=4, day=16, hour=14, minute=31,␣
↪second=59, tzinfo=timezone.utc)
```

CHAPTER

FOUR

EXAMPLES AND SAMPLE CODE

If you checkout the project, these files should be runnable locally after installing.

```
pip install .

python ./examples/create_invoice.py
```

Be sure to update the example files with your own credentials in place of <your account id> and <your access token>.

4.1 Authorization Flow

```
# This is an example where we run through the OAuth flow,
# select a business, and display a client from that business.

from types import SimpleNamespace
from freshbooks import Client as FreshBooksClient

fb_client_id = "<your client id>"
secret = "<your client secret>"
redirect_uri = "<your redirect uri>"

freshBooksClient = FreshBooksClient(
    client_id=fb_client_id,
    client_secret=secret,
    redirect_uri=redirect_uri
)

authorization_url = freshBooksClient.get_auth_request_url(
    scopes=['user:profile:read', 'user:clients:read']
)
print(f"Go to this URL to authorize: {authorization_url}")

# Going to that URL will prompt the user to log into FreshBooks and authorize the␣
↪application.
# Once authorized, FreshBooks will redirect the user to your `redirect_uri` with the␣
↪authorization
# code will be a parameter in the URL.
auth_code = input("Enter the code you get after authorization: ")

```

(continues on next page)

(continued from previous page)

```
# This will exchange the authorization code for an access token
token_response = freshBooksClient.get_access_token(auth_code)
print(f"This is the access token the client is now configurated with: {token_response.access_token}")
print(f"It is good until {token_response.access_token_expires_at}")
print()

# Get the current user's identity
identity = freshBooksClient.current_user()
businesses = []

# Display all of the businesses the user has access to
for num, business_membership in enumerate(identity.business_memberships, start=1):
    business = business_membership.business
    businesses.append(
        SimpleNamespace(name=business.name, business_id=business.id, account_id=business.account_id)
    )
    print(f"{num}: {business.name}")
business_index = int(input("Which business do you want to use? ")) - 1
print()

business_id = businesses[business_index].business_id  # Used for project-related calls
account_id = businesses[business_index].account_id  # Used for accounting-related calls

# Get a client for the business to show successful access
client = freshBooksClient.clients.list(account_id)[0]
print(f"'{client.organization}' is a client of {businesses[business_index].name}")
```

4.2 Create Invoice

```
# This is an example where we create a new client and an invoice for them.

from datetime import date
from freshbooks import Client as FreshBooksClient
from freshbooks import FreshBooksError

fb_client_id = "<your client id>"
access_token = "<your access token>"
account_id = "<your account id>"

freshBooksClient = FreshBooksClient(client_id=fb_client_id, access_token=access_token)

# Create the client
print("Creating client...")
try:
    client_data = {"organization": "Python SDK Test Client"}
    client = freshBooksClient.clients.create(account_id, client_data)
except FreshBooksError as e:
    print(e)
```

(continues on next page)

(continued from previous page)

```
    print(e.status_code)
    exit(1)

print(f"Created client {client.id}")

# Create the invoice
line1 = {
    "name": "Fancy Dishes",
    "description": "They're pretty swanky",
    "qty": 6,
    "unit_cost": {
        "amount": "27.00",
        "code": "CAD"
    }
}
line2 = {
    "name": "Regular Glasses",
    "description": 'They look "just ok"',
    "qty": 8,
    "unit_cost": {
        "amount": "5.95",
        "code": "CAD"
    }
}
invoice_data = {
    "customerid": client.id,
    "create_date": date.today().isoformat(),
    "lines": [line1, line2],
}
print("Creating invoice...")
try:
    invoice = freshBooksClient.invoices.create(account_id, invoice_data)
except FreshBooksError as e:
    print(e)
    print(e.status_code)
    exit(1)

print(f"Created invoice {invoice.id}")
print(f"Invoice total is {invoice.amount.amount} {invoice.amount.code}")

# Invoices are created in draft status, so we need to mark it as sent
print("Marking invoice as sent...")
invoice_data = {
    "action_mark_as_sent": True
}
invoice = freshBooksClient.invoices.update(account_id, invoice.id, invoice_data)
```

CHAPTER

FIVE

CLIENT

class freshbooks.client.**Client**(*client_id*, *client_secret=None*, *redirect_uri=None*, *access_token=None*, *refresh_token=None*, *user_agent=None*, *timeout=30*, *auto_retry=True*)

Bases: `object`

property bill_payments: *freshbooks.api.accounting.AccountingResource*

FreshBooks bill_payments resource with calls to get, list, create, update, delete

Return type :py:class:~`freshbooks.api.accounting.AccountingResource`

property bill_vendors: *freshbooks.api.accounting.AccountingResource*

FreshBooks bill_vendors resource with calls to get, list, create, update, delete

Return type :py:class:~`freshbooks.api.accounting.AccountingResource`

property bills: *freshbooks.api.accounting.AccountingResource*

FreshBooks bills resource with calls to get, list, create, update, delete

Return type :py:class:~`freshbooks.api.accounting.AccountingResource`

property callbacks: *freshbooks.api.events.EventsResource*

FreshBooks callbacks (webhook callbacks) resource with calls to get, list, create, update, delete, resend_verification, verify

Return type :py:class:~`freshbooks.api.events.EventsResource`

property clients: *freshbooks.api.accounting.AccountingResource*

FreshBooks clients resource with calls to get, list, create, update, delete

Return type :py:class:~`freshbooks.api.accounting.AccountingResource`

property credit_notes: *freshbooks.api.accounting.AccountingResource*

FreshBooks credit_notes resource with calls to get, list, create, update, delete

Return type :py:class:~`freshbooks.api.accounting.AccountingResource`

current_user()

The identity details of the currently authenticated user.

See FreshBooks API - Business, Roles, and Identity

Return type :py:class:~`freshbooks.models.Identity`

property estimates: *freshbooks.api.accounting.AccountingResource*

FreshBooks estimates resource with calls to get, list, create, update, delete

Return type :py:class:~`freshbooks.api.accounting.AccountingResource`

property expenses: *freshbooks.api.accounting.AccountingResource*

FreshBooks expenses resource with calls to get, list, create, update, delete

Return type :py:class:~freshbooks.api.accounting.AccountingResource

property expenses_categories: *freshbooks.api.accounting.AccountingResource*

FreshBooks expenses categories resource with calls to get and list

Return type :py:class:~freshbooks.api.accounting.AccountingResource

property gateways: *freshbooks.api.accounting.AccountingResource*

FreshBooks gateways resource with calls to list, delete

Return type :py:class:~freshbooks.api.accounting.AccountingResource

get_access_token(*code*)

Makes a call to the FreshBooks token URL to get an access_token.

This requires the access_grant code obtained after the user is redirected by the authorization step. See freshbooks.client.Client.get_auth_request_url.

This call sets the access_token, refresh_token, and access_token_expires_at attributes on the Client instance and also returns those values in an object.

Args: code: access_grant code from the authorization redirect

Returns: Simple namespace containing access_token, refresh_token, and access_token_expires_at

Raises: FreshBooksError: If the call fails to return a access token. FreshBooksClientConfigError: If client_secret and redirect_uri are not set on the client instance.

Return type :py:class:~types.SimpleNamespace

get_auth_request_url(*scopes=None*)

Returns the url that a client needs to request an oauth grant from the server.

To get an oauth access token, send your user to this URL. The user will be prompted to log in to FreshBooks, after which they will be redirected to the redirect_uri set on the client with the access grant as a parameter. That grant can then be used to fetch an access token by calling get_access_token.

Note: The redirect_uri must be one of the URLs your application is registered for.

If scopes are not specified, then the access token will be given the default scopes your application is registered for.

Args: scopes: List of scopes if your want an access token with only a subset of your registered scopes

Returns: The URL for the authorization request

Raises: FreshBooksClientConfigError: If redirect_uri is not set on the client instance.

Return type :py:class:str

property invoice_payment_options: *freshbooks.api.payments.PaymentsResource*

FreshBooks default payment options resource with calls to defaults, get, create

Return type :py:class:~freshbooks.api.payments.PaymentsResource

property invoice_profiles: *freshbooks.api.accounting.AccountingResource*

FreshBooks invoice_profiles resource with calls to get, list, create, update, delete

Return type :py:class:~freshbooks.api.accounting.AccountingResource

property invoices: *freshbooks.api.accounting.AccountingResource*

FreshBooks invoices resource with calls to get, list, create, update, delete

Return type :py:class:~freshbooks.api.accounting.AccountingResource

property items: *freshbooks.api.accounting.AccountingResource*

FreshBooks items resource with calls to get, list, create, update, delete

Return type :py:class:~freshbooks.api.accounting.AccountingResource

property other_income: *freshbooks.api.accounting.AccountingResource*

FreshBooks other_incomes resource with calls to get, list, create, update, delete

Return type :py:class:~freshbooks.api.accounting.AccountingResource

property payments: *freshbooks.api.accounting.AccountingResource*

FreshBooks payments resource with calls to get, list, create, update, delete

Return type :py:class:~freshbooks.api.accounting.AccountingResource

property projects: *freshbooks.api.projects.ProjectsResource*

FreshBooks projects resource with calls to get, list, create, update, delete

Return type :py:class:~freshbooks.api.projects.ProjectsResource

refresh_access_token(*refresh_token=None*)

Makes a call to the FreshBooks token URL to refresh an access_token.

If `refresh_token` is provided, it will call to refresh it, otherwise it will use the `refresh_token` on the Client instance.

This call sets the `access_token`, `refresh_token`, and `access_token_expires_at` attributes on the Client instance to the new values from the refresh call, and also returns those values in an object.

Args: refresh_token: (Optional) refresh_token from initial `get_access_token` call

Returns: Simple namespace containing `access_token`, `refresh_token`, and `access_token_expires_at`

Raises: FreshBooksClientConfigError: If refresh_token is not set on the client instance and is not provided.

Return type :py:class:~types.SimpleNamespace

property service_rates: *freshbooks.api.comments.CommentsSubResource*

FreshBooks service_rates resource with calls to get, list, create, update

Return type :py:class:~freshbooks.api.comments.CommentsSubResource

property services: *freshbooks.api.comments.CommentsResource*

FreshBooks services resource with calls to get, list, create, update, delete

Return type :py:class:~freshbooks.api.comments.CommentsResource

property staff: *freshbooks.api.accounting.AccountingResource*

FreshBooks staff resource with calls to get, list, update, delete

Return type :py:class:~freshbooks.api.accounting.AccountingResource

property systems: *freshbooks.api.accounting.AccountingResource*

FreshBooks systems resource with calls to get only

Return type :py:class:~freshbooks.api.accounting.AccountingResource

property tasks: *freshbooks.api.accounting.AccountingResource*

FreshBooks tasks resource with calls to get, list, create, update, delete

Note: There is a lot of overlap between Services and Tasks. In general services are used to add categories of work to projects, and tasks are used to add billable work to invoices.

Creating a task should create the corresponding service and vice versa.

Return type :py:class:~freshbooks.api.accounting.AccountingResource

property taxes: *freshbooks.api.accounting.AccountingResource*

FreshBooks taxes resource with calls to get, list, create, update, delete

Return type :py:class:~freshbooks.api.accounting.AccountingResource

property time_entries: *freshbooks.api.timetracking.TimetrackingResource*

FreshBooks time_entries resource with calls to get, list, create, update, delete

Return type :py:class:~freshbooks.api.timetracking.TimetrackingResource

freshbooks.client.**DEFAULT_TIMEOUT = 30**

Default request timeout to FreshBooks

CHAPTER

SIX

RESOURCES

6.1 Accounting

class freshbooks.api.accounting.**AccountingResource**(*client_config*, *accounting_path*, *single_name*, *list_name*, *delete_via_update=True*, *missing_endpoints=None*)

Handles resources under the `/accounting` endpoints.

create(*account_id*, *data*, *includes=None*)

Create a resource.

Args: account_id: The alpha-numeric account id data: Dictionary of data to populate the resource builders: (Optional) IncludesBuilder object for including additional data, sub-resources, etc.

Returns: Result: Result object with the new resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

delete(*account_id*, *resource_id*)

Delete a resource.

Note: Most FreshBooks resources are soft-deleted, See FreshBooks API - Active and Deleted Objects

Args: account_id: The alpha-numeric account id resource_id: Id of the resource to delete

Returns: Result: An empty Result object.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

get(*account_id*, *resource_id*, *includes=None*)

Get a single resource with the corresponding id.

Args: account_id: The alpha-numeric account id resource_id: Id of the resource to return builders: (Optional) IncludesBuilder object for including additional data, sub-resources, etc. Returns: Result: Result object with the resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

headers(*method*)

Get headers required for API calls

Return type :py:class:~`typing.Dict``[:py:class:``str`, :py:class:`str`]

list(*account_id*, *builders=None*)

Get a list of resources.

Args: account_id: The alpha-numeric account id builders: (Optional) List of builder objects for filters, pagination, etc.

Returns: ListResult: ListResult object with the resources response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.ListResult

update(*account_id*, *resource_id*, *data*, *includes=None*)

Update a resource.

Args: account_id: The alpha-numeric account id resource_id: Id of the resource to update data: Dictionary of data to update the resource to builders: (Optional) IncludesBuilder object for including additional data, sub-resources, etc.

Returns: Result: Result object with the updated resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

6.2 Auth

class freshbooks.api.auth.**AuthResource**(*client_config*)

Handles resources under the /auth endpoints.

headers(*method*)

Get headers required for API calls

Return type :py:class:~typing.Dict``[:py:class:``str, :py:class:str]

me_endpoint()

Get the identity details of the currently authenticated user.

See FreshBooks API - Business, Roles, and Identity

Returns: Result: Result object with the authenticated user's identity and business details.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Identity

6.3 Projects

class freshbooks.api.projects.**ProjectsResource**(*client_config*, *list_resource_path*, *single_resource_path*, *list_name=None*, *single_name=None*, *missing_endpoints=None*)

Bases: freshbooks.api.projects.ProjectsBaseResource

Handles resources under the /projects endpoints.

`create`(*business_id*, *data*)

Create a resource.

Args: business_id: The business id data: Dictionary of data to populate the resource

Returns: Result: Result object with the new resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

`delete`(*business_id*, *resource_id*)

Delete a resource.

Note: Most FreshBooks resources are soft-deleted, See FreshBooks API - Active and Deleted Objects

Args: business_id: The business id resource_id: Id of the resource to delete

Returns: Result: An empty Result object.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

`get`(*business_id*, *resource_id*)

Get a single resource with the corresponding id.

Args: business_id: The business id resource_id: Id of the resource to return Returns: Result: Result object with the resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

`list`(*business_id*, *builders=None*)

Get a list of resources.

Args: business_id: The business id builders: (Optional) List of builder objects for filters, pagination, etc.

Returns: ListResult: ListResult object with the resources response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.ListResult`

`update`(*business_id*, *resource_id*, *data*)

Update a resource.

Args: business_id: The business id resource_id: Id of the resource to update data: Dictionary of data to update the resource to

Returns: Result: Result object with the updated resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

6.4 Comments

class freshbooks.api.comments.**CommentsResource**(*client_config*, *list_resource_path*, *single_resource_path*, *list_name=None*, *single_name=None*, *missing_endpoints=None*)

Bases: *freshbooks.api.projects.ProjectsResource*

Handles resources under the `/comments` endpoints.

These are handled identically to `/projects` endpoints. Refer to `freshbooks.api.projects.ProjectsResource`.

create(*business_id*, *data*)

Create a resource.

Args: business_id: The business id data: Dictionary of data to populate the resource

Returns: Result: Result object with the new resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

delete(*business_id*, *resource_id*)

Delete a resource.

Note: Most FreshBooks resources are soft-deleted, See FreshBooks API - Active and Deleted Objects

Args: business_id: The business id resource_id: Id of the resource to delete

Returns: Result: An empty Result object.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

get(*business_id*, *resource_id*)

Get a single resource with the corresponding id.

Args: business_id: The business id resource_id: Id of the resource to return Returns: Result: Result object with the resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

headers(*method*)

Get headers required for API calls

Return type :py:class:~`typing.Dict``[:py:class:``str`, :py:class:`str`]

list(*business_id*, *builders=None*)

Get a list of resources.

Args: business_id: The business id builders: (Optional) List of builder objects for filters, pagination, etc.

Returns: ListResult: ListResult object with the resources response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.ListResult`

update(*business_id*, *resource_id*, *data*)

Update a resource.

Args: business_id: The business id resource_id: Id of the resource to update data: Dictionary of data to update the resource to

Returns: Result: Result object with the updated resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

class freshbooks.api.comments.**CommentsSubResource**(*client_config*, *list_resource_path*, *single_resource_path*, *single_resource_sub_path=None*, *list_name=None*, *single_name=None*, *missing_endpoints=None*)

Bases: freshbooks.api.projects.ProjectsBaseResource

Handles sub-resources under the /comments endpoints.

Eg. /comments/business/{business_id}/services/{service_id}/rate

These are handled similarly to /projects endpoints. Refer to freshbooks.api.projects.ProjectsResource.

create(*business_id*, *resource_id*, *data*)

Create a resource.

Args: business_id: The business id resource_id: Id of the parent resource to create this resource under data: Dictionary of data to populate the resource

Returns: Result: Result object with the new resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

delete(*business_id*, *resource_id*)

Delete a resource.

Note: Most FreshBooks resources are soft-deleted, See FreshBooks API - Active and Deleted Objects

Args: business_id: The business id resource_id: Id of the resource to delete

Returns: Result: An empty Result object.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

get(*business_id*, *resource_id*)

Get a single resource with the corresponding id.

Args: business_id: The business id resource_id: Id of the resource to return Returns: Result: Result object with the resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

headers(*method*)

Get headers required for API calls

Return type :py:class:~typing.Dict``[:py:class:``str, :py:class:str]

`list`(*business_id*, *builders=None*)

Get a list of resources.

Args: business_id: The business id builders: (Optional) List of builder objects for filters, pagination, etc.

Returns: ListResult: ListResult object with the resources response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.ListResult`

`update`(*business_id*, *resource_id*, *data*)

Update a resource.

Args: business_id: The business id resource_id: Id of the resource to update data: Dictionary of data to update the resource to

Returns: Result: Result object with the updated resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

6.5 Time-Tracking

`class freshbooks.api.timetracking.`**`TimetrackingResource`**(*client_config*, *list_resource_path*, *single_resource_path*, *list_name=None*, *single_name=None*, *missing_endpoints=None*)

Bases: *freshbooks.api.projects.ProjectsResource*

Handles resources under the `/timetracking` endpoints.

These are handled identically to `/projects` endpoints. Refer to `freshbooks.api.projects.ProjectsResource`.

`create`(*business_id*, *data*)

Create a resource.

Args: business_id: The business id data: Dictionary of data to populate the resource

Returns: Result: Result object with the new resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

`delete`(*business_id*, *resource_id*)

Delete a resource.

Note: Most FreshBooks resources are soft-deleted, See FreshBooks API - Active and Deleted Objects

Args: business_id: The business id resource_id: Id of the resource to delete

Returns: Result: An empty Result object.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~`freshbooks.models.Result`

get(*business_id*, *resource_id*)

Get a single resource with the corresponding id.

Args: business_id: The business id resource_id: Id of the resource to return Returns: Result: Result object with the resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

headers(*method*)

Get headers required for API calls

Return type :py:class:~typing.Dict``[:py:class:``str, :py:class:str]

list(*business_id*, *builders=None*)

Get a list of resources.

Args: business_id: The business id builders: (Optional) List of builder objects for filters, pagination, etc.

Returns: ListResult: ListResult object with the resources response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.ListResult

update(*business_id*, *resource_id*, *data*)

Update a resource.

Args: business_id: The business id resource_id: Id of the resource to update data: Dictionary of data to update the resource to

Returns: Result: Result object with the updated resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

6.6 Payments

class freshbooks.api.payments.**PaymentsResource**(*client_config*, *path*, *single_name*, *sub_path=None*, *defaults_path=None*, *static_params=None*, *missing_endpoints=None*)

Handles resources under the /payments endpoints.

create(*account_id*, *resource_id*, *data*)

Create a resource.

Args: account_id: The alpha-numeric account id resource_id: Id of the resource to create payment details for data: Dictionary of data to populate the resource

Returns: Result: Result object with the new resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

defaults(*account_id*)

Get the default settings for an account resource.

Args: account_id: The alpha-numeric account id Returns: Result: Result object with the default data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

get(*account_id*, *resource_id*)

Get a single resource with the corresponding id.

Args: account_id: The alpha-numeric account id resource_id: Id of the resource to return payment details for Returns: Result: Result object with the resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

headers(*method*)

Get headers required for API calls

Return type :py:class:~typing.Dict``[:py:class:``str, :py:class:str]

6.7 Events

class freshbooks.api.events.**EventsResource**(*client_config*, *accounting_path*, *single_name*, *list_name*, *delete_via_update=True*, *missing_endpoints=None*)

Bases: *freshbooks.api.accounting.AccountingResource*

Handles resources under the /events endpoints.

These are handled almost similarly to /accounting endpoints. Refer to freshbooks.api.accounting.AccountingResource.

create(*account_id*, *data*, *includes=None*)

Create a resource.

Args: account_id: The alpha-numeric account id data: Dictionary of data to populate the resource builders: (Optional) IncludesBuilder object for including additional data, sub-resources, etc.

Returns: Result: Result object with the new resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

delete(*account_id*, *resource_id*)

Delete a resource.

Note: Most FreshBooks resources are soft-deleted, See FreshBooks API - Active and Deleted Objects

Args: account_id: The alpha-numeric account id resource_id: Id of the resource to delete

Returns: Result: An empty Result object.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

get(*account_id*, *resource_id*, *includes=None*)

Get a single resource with the corresponding id.

Args: account_id: The alpha-numeric account id resource_id: Id of the resource to return builders: (Optional) IncludesBuilder object for including additional data, sub-resources, etc. Returns: Result: Result object with the resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

headers(*method*)

Get headers required for API calls

Return type :py:class:~typing.Dict``[:py:class:``str, :py:class:str]

list(*account_id*, *builders=None*)

Get a list of resources.

Args: account_id: The alpha-numeric account id builders: (Optional) List of builder objects for filters, pagination, etc.

Returns: ListResult: ListResult object with the resources response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.ListResult

resend_verification(*account_id*, *resource_id*)

Tell FreshBooks to resend the verification webhook for the callback

Args: account_id: The alpha-numeric account id resource_id: Id of the resource to update

Returns: Result: Result object with the resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

update(*account_id*, *resource_id*, *data*, *includes=None*)

Update a resource.

Args: account_id: The alpha-numeric account id resource_id: Id of the resource to update data: Dictionary of data to update the resource to builders: (Optional) IncludesBuilder object for including additional data, sub-resources, etc.

Returns: Result: Result object with the updated resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

verify(*account_id*, *resource_id*, *verifier*)

Verify webhook callback by making a put request

Args: account_id: The alpha-numeric account id resource_id: Id of the resource to update verifier: The string verifier received by the webhook callback URI

Returns: Result: Result object with the resource's response data.

Raises: FreshBooksError: If the call is not successful.

Return type :py:class:~freshbooks.models.Result

CHAPTER

SEVEN

MODELS

class freshbooks.models.**Identity**(*data*)

Bases: *freshbooks.models.Result*

An Identity is a `freshbooks.models.Result` object with additional properties and helper methods to make accessing the current user's identity easier.

Example:

```
>>> current_user = freshBooksClient.current_user()
>>> current_user.email
<some email>

>>> current_user.business_memberships
<list of businesses>
```

property business_memberships: Any

The authenticated user's businesses and their role in that business.

Return type :py:data:~typing.Any

property full_name: str

The authenticated user's name

Return type :py:class:str

property identity_id: int

The authenticated user's identity_id

Return type :py:class:int

class freshbooks.models.**ListResult**(*name*, *single_name*, *data*, *include_pages=True*)

Bases: object

Result object from API calls with a list of resources returned.

Data in the API can be accessed via attributes.

Example:

```
clients = freshBooksClient.clients.list(account_id)
assert clients[0].organization == "FreshBooks"
```

The json-parsed dictionary can also be directly accessed via the `data` attribute.

Example:

```
assert clients.data["clients"][0]["organization"] == "FreshBooks"
```

The list can also be iterated over to access the individual resources as `Result` obejcts.

Example:

```
for client in clients:
    assert client.organization == "FreshBooks"
    assert client.data["organization"] == "FreshBooks"
```

Pagination results are included in the `pages` attribute:

```
>>> clients.pages
PageResult(page=1, pages=1, per_page=30, total=6)
>>> clients.pages.total
6
```

For including pagination in requests, see `freshbooks.builders.paginator.PaginateBuilder`.

class freshbooks.models.**Result**(*name*, *data*)

Bases: `object`

Result object from API calls with a single resource returned.

Data in the API can be accessed via attributes.

Example:

```
client = freshBooksClient.clients.get(account_id, user_id)
assert client.organization == "FreshBooks"
assert client.userid == user_id
```

The json-parsed dictionary can also be directly accessed via the `data` attribute.

Example:

```
assert client.data["organization"] == "FreshBooks"
assert client.data["userid"] == user_id
```

enum freshbooks.models.**VisState**(*value*)

Bases: `enum.IntEnum`

Enum of FreshBooks entity vis_status values

Member Type `int`

Valid values are as follows:

ACTIVE = <VisState.ACTIVE: 0>

DELETED = <VisState.DELETED: 1>

ARCHIVED = <VisState.ARCHIVED: 2>

CHAPTER

EIGHT

BUILDERS

8.1 Paginator

class freshbooks.builders.paginator.**PaginateBuilder**(*page=None*, *per_page=None*)

Bases: freshbooks.builders.Builder

Builder for making paginated list queries.

Has two attributes, `page` and `per_page`. When a `PaginateBuilder` object is passed to a `.list()` call, the call will fetch only the `per_page` number of results and will fetch the results offset by `page`.

```
>>> from freshbooks import PaginateBuilder

>>> paginator = PaginateBuilder(2, 4)
>>> paginator
PaginateBuilder(page=2, per_page=4)

>>> clients = freshBooksClient.clients.list(account_id, builders=[paginator])
>>> clients.pages
PageResult(page=2, pages=3, per_page=4, total=9)
```

build(*resource_name=None*)

Builds the query string parameters from the PaginateBuilder.

Args: resource_name: The type of resource to generate the query string for. Eg. AccountingResource, ProjectsResource

Returns: The built query string

Return type :py:class:`str`

page(*page=None*)

Set the page you wish to fetch in a list call, or get the currently set the page. When setting, can be chained.

```
>>> paginator = PaginateBuilder(1, 3)
>>> paginator
PaginateBuilder(page=1, per_page=3)

>>> paginator.page()
1

>>> paginator.page(2).per_page(4)
```

(continues on next page)

(continued from previous page)

```
>>> paginator
PaginateBuilder(page=2, per_page=4)
```

Args: page: (Optional) The page of results to return in the API call

Returns: The PaginateBuilder instance if a `page` value is provided, otherwise returns the currently set `page` value.

Return type :py:data:~typing.Union``[:py:class:``int, :py:obj:None, :py:class:~freshbooks.builders.Builder]

per_page(*per_page=None*)

Set the number of results you wish to fetch in a page of a list call, or get the currently set per_page. When setting, can be chained.

The page size is capped at 100.

```
>>> paginator = PaginateBuilder(1, 3)
>>> paginator
PaginateBuilder(page=1, per_page=3)

>>> paginator.per_page()
3

>>> paginator.per_page(4).page(2)
>>> paginator
PaginateBuilder(page=2, per_page=4)
```

Args: per_page: (Optional) The number of results to return in each API call

Returns: The PaginateBuilder instance if a `per_page` value is provided, otherwise returns the currently set `per_page` value.

Return type :py:data:~typing.Union``[:py:class:``int, :py:obj:None, :py:class:~freshbooks.builders.Builder]

8.2 Filters

class freshbooks.builders.filter.**FilterBuilder**

Bases: `freshbooks.builders.Builder`

Builder for making filtered list queries.

Filters can be builts with the methods: `equals`, `in_list`, `like`, `between`, and `boolean`, `date_time` which can be chained together.

```
>>> from freshbooks import FilterBuilder

>>> f = FilterBuilder()
>>> f.like("email_like", "@freshbooks.com")
FilterBuilder(&search[email_like]=@freshbooks.com)

>>> f = FilterBuilder()
>>> f.in_list("clientids", [123, 456]).boolean("active", False)
```

(continues on next page)

(continued from previous page)

```
FilterBuilder(&search[clientids][]=123&search[clientids][]=456&active=False)

>>> f = FilterBuilder()
>>> f.boolean("active", False).in_list("clientids", [123, 456])
FilterBuilder(&active=False&search[clientids][]=123&search[clientids][]=456)

>>> f = FilterBuilder()
>>> f.between("amount", 1, 10)
FilterBuilder(&search[amount_min]=1&search[amount_max]=10)

>>> f = FilterBuilder()
>>> f.between("start_date", date.today())
FilterBuilder(&search[start_date]=2020-11-21)
```

between(*field*, *min=None*, *max=None*)

Filters results where the provided field is between two values.

In general 'between' filters end in a `_min` or `_max` (as in `amount_min` or `amount_max`) or `_date` (as in `start_date`, `end_date`). If the provided field does not end in `_min`/`_max` or `_date`, then the appropriate `_min`/`_max` will be appended.

For date fields, you can pass the iso format `2020-10-17` or a `datetime` or `date` object, which will be converted to the proper string format.

Examples:

- `filter.between("amount", 1, 10)` will yield filters `&search[amount_min]=1&search[amount_max]=10`
- `filter.between("amount_min", min=1)` will yield filter `&search[amount_min]=1`
- `filter.between("amount_max", max=10)` will yield filter `&search[amount_max]=10`
- `filter.between("start_date", "2020-10-17")` will yield filter `&search[start_date]=2020-10-17`
- `filter.between("start_date", datetime.date(year=2020, month=10, day=17))` will yield filter

`&search[start_date]=2020-10-17`

Args: field: The API response field to filter on min: (Optional) The value the field should be greater than (or equal to) max: (Optional) The value the field should be less than (or equal to)

Returns: The FilterBuilder instance

Return type :py:class:~`freshbooks.builders.Builder`

boolean(*field*, *value*)

Filters results where the field is equal to true or false.

Example: `filter.boolean("active", False)` will yield the filter `&active=false`

Args: field: The API response field to filter on value: True or False

Returns: The FilterBuilder instance

Return type :py:class:~`freshbooks.builders.Builder`

build(*resource_name=None*)

Builds the query string parameters from the FilterBuilder.

Args: resource_name: The type of resource to generate the query string for. Eg. AccountingResource, ProjectsResource

Returns: The built query string

Return type :py:class:`str`

`date_time`(*field*, *value*)

Filters for entries that come before or after a particular time, as specified by the field. Eg. "updated_since" on Time Entries will return time entries updated after the provided time.

The url parameter must be in ISO 8601 format (eg. 2010-10-17T05:45:53Z)

Example:

- `filter.date_time("updated_since", "2020-10-17T13:14:07")` will yield `&updated_since=2020-10-17T13:14:07`

Args: field: The API response field to filter on value: The datetime, or ISO 8601 format string value

Returns: The FilterBuilder instance

Return type :py:class:`~freshbooks.builders.Builder`

`equals`(*field*, *value*)

Filters results where the field is equal to the provided value.

Example: `filter.equals("username", "Bob")` will yield the filter `&search[username]=Bob`

Args: field: The API response field to filter on value: The value the field should equal

Returns: The FilterBuilder instance

Return type :py:class:`~freshbooks.builders.Builder`

`in_list`(*field*, *values*)

Filters if the provided field matches a value in a list.

In general, an 'in' filter will be bound to the plural form of the field. Eg. `userid` for an equal filter, `userids` for a list filter.

Here we only append an 's' to the field name if it doesn't have one yet. This way we can be as forgiving as possible for developers by accepting: `filter.in_list("userid", [1, 2])` or `filter.in_list("userids", [1, 2])`.

Of course the FreshBooks API is not 100% consistent, so there are a couple of unique cases that may not be handled.

Args: field: The API response field to filter on values: List of values the field should one of

Returns: The FilterBuilder instance

Return type :py:class:`~freshbooks.builders.Builder`

`like`(*field*, *value*)

Filters for a match contained within the field being searched. For example, "leaf" will Like-match "aleaf" and "leafy", but not "leav", and "leafs" would not Like-match "leaf".

Args: field: The API response field to filter on value: The value the field should contain

Returns: The FilterBuilder instance

Return type :py:class:`~freshbooks.builders.Builder`

8.3 Includes

class freshbooks.builders.includes.**IncludesBuilder**

Bases: freshbooks.builders.Builder

Builder for including relationships, sub-resources, or additional data in the response.

```
>>> from freshbooks import IncludesBuilder

>>> includes = IncludesBuilder()
>>> includes.include("late_reminders")
IncludesBuilder(&include[]=late_reminders)
```

build(*resource_name=None*)

Builds the query string parameters from the IncludesBuilder.

Args: resource_name: The type of resource to generate the query string for. Eg. AccountingResource, ProjectsResource

Returns: The built query string

Return type :py:class:str

include(*key*)

Add an include key to the builder.

Example: includes.include("late_reminders") will yield the filter &include[]=late_reminders

Args: key: The key for the resource or data to include

Returns: The IncludesBuilder instance

Return type :py:class:~freshbooks.builders.Builder

CHAPTER

NINE

ERRORS

exception freshbooks.errors.**FreshBooksClientConfigError**

Bases: Exception

Exception thrown when optional client parameters are not set, but and required.

with_traceback()

Exception.with_traceback(tb) – set self.**traceback** to tb and return self.

exception freshbooks.errors.**FreshBooksError**(*status_code*, *message*, *raw_response=None*, *error_code=None*)

Bases: Exception

Exception thrown when FreshBooks returns a non-2xx response or when the response is missing expected content.

Example:

```
try:
    client = freshBooksClient.clients.get(self.account_id, client_id)
except FreshBooksError as e:
    assert str(e) == "Client not found."
    assert e.status_code == 404
    assert e.error_code == 1012
```

Attributes: message: Error message status_code: HTTP status code from the server. error_code: (Optional) FreshBooks specific error code, if available raw_response: Content response from the server.

with_traceback()

Exception.with_traceback(tb) – set self.**traceback** to tb and return self.

exception freshbooks.errors.**FreshBooksNotImplementedError**(*resource_name*, *method_name*)

Bases: Exception

Exception thrown when making a resource call that does not exist. Eg.

```
>>> freshBooksClient.staff.create()
```

with_traceback()

Exception.with_traceback(tb) – set self.**traceback** to tb and return self.

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

INDEX

G

H

I

L

M

O

P

R

S

T

U

V

W