apiron Documentation

Release 8.0.0

Ithaka Harbors, Inc.

Dec 12, 2023

CONTENTS

`apiron` ("API" + "iron") is a Python wrapper for interacting with RESTful APIs

Tip: Read the *upgrade guide* to make sure you're prepared for the latest version of apiron.

Welcome to the documentation for `apiron`! You can use this documentation to orient yourself to the codebase, search broadly for existing functionality, or look up the expectations for specific classes and methods.

As with all documentation, this is a living, breathing beast. If you notice something undocumented or misdocumented, feel free to change it in a pull request!

CHAPTER

ONE

GETTING STARTED

The goal of `apiron` is to get you up and running quickly, consuming a service with little initial configuration while allowing for granular customization. The declarative nature of this setup makes the shape of services and their endpoints more obvious than placing those details in one-off calls.

The minimum possible configuration requires a bit of information about the service.

1.1 Defining a service

A service definition requires a domain and one or more endpoints with which to interact:

```
from apiron import JsonEndpoint, Service

class GitHub(Service):
    domain = 'https://api.github.com'
    user = JsonEndpoint(path='/users/{username}')
    repo = JsonEndpoint(path='/repos/{org}/{repo}')
```

1.2 Interacting with a service

Once your service definition is in place, you can interact with its endpoints in an SDK-like manner:

```
response = GitHub.user(username='defunkt')
# {"name": "Chris Wanstrath", ...}

response = GitHub.repo(org='github', repo='hub')
# {"description": "hub helps you win at git.", ...}
```

1.3 Next steps

Now that you're a seasoned pro, explore the *advanced usage*!

CHAPTER

TWO

ADVANCED USAGE

httpbin.org is a great testing tool for various situations you may run into when interacting with RESTful services. Let's define a `Service` that points at httpbin.org and hook up a few interesting endpoints.

2.1 Service and endpoints

```
# test_service.py
from apiron import Endpoint, JsonEndpoint, Service, StreamingEndpoint

class HttpBin(Service):
    domain = 'https://httpbin.org'

    getter = JsonEndpoint(path='/get')
    poster = JsonEndpoint(path='/post', default_method='POST')
    status = Endpoint(path='/status/{status_code}/')
    anything = JsonEndpoint(path='/anything/{anything}')
    slow = JsonEndpoint(path='/delay/5')
    streamer = StreamingEndpoint(path='/stream/{num_lines}')
```

2.2 Using all the features

```
import requests

from apiron import Timeout

from test_service import HttpBin

# A normal old GET call
HttpBin.getter(params={'foo': 'bar'})

# A normal old POST call
HttpBin.poster(data={'foo': 'bar'})

# A GET call with parameters formatted into the path
HttpBin.anything(anything=42)

# A GET call with a 500 response, raises RetryError since we successfully tried but got␣
```

(continues on next page)

(continued from previous page)

```
↪a bad response
try:
    HttpBin.status(status_code=500)
except requests.exceptions.RetryError:
    pass

# A GET call to a slow endpoint, raises ConnectionError since our connection failed
try:
    HttpBin.slow()
except requests.exceptions.ConnectionError:
    pass

# A GET call to a slow endpoint with a longer timeout
HttpBin.slow(
    timeout_spec=Timeout(connection_timeout=1, read_timeout=6)
)

# A streaming response
response = HttpBin.streamer(num_lines=20)
for chunk in response:
    print(chunk)
```

2.3 Service discovery

You may want to interact with a service whose name is known but whose hosts are resolved via another application. Here is an example where the resolver application always resolves to `https://www.google.com` for the host.

```
from apiron import DiscoverableService

class Eureka:
    @staticmethod
    def resolve(service_name):
        hosts = ...  # get host list from Eureka

class AuthenticationService(DiscoverableService):
    service_name = 'authentication-service'
    host_resolver_class = Eureka

    auth = Endpoint(path='/auth')


response = AuthenticationService.auth(data={'user': 'Gandalf', 'password': 'Mellon'})
```

An application may wish to use a load balancer application or a more complex service discovery mechanism (like Netflix's Eureka) to resolve the hostnames of a given service.

2.4 Workflow consistency

It's common to have an existing `requests.Session` object you'd like to use to make additional requests. This is enabled in `apiron` with the `session` argument to an endpoint call. The passed in session object will be used to send the request. This is useful for workflows where cookies or other information need to persist across multiple calls.

It's often more useful in logs to know which module initiated the code doing the logging. `apiron` allows for an existing logger object to be passed to an endpoint call using the `logger` argument so that logs will indicate the caller module rather than *`apiron.client`*.

CHAPTER

THREE

UPGRADE GUIDE

This document will guide you through upgrading from *apiron* 2.X versions to *apiron* 3.X.

3.1 Replacing `ServiceCaller` usage

As of version 2.3.0, instantiating a service class and passing it to `ServiceCaller.call` is no longer necessary. In 3.X, `ServiceCaller` has been removed altogether (though its behaviors are still in the *`apiron.client`* module). Replace calls to `ServiceCaller.call` with the more semantic `SomeService.some_endpoint`:

```
# Before
from apiron.client import ServiceCaller
from apiron.service.base import Service
from apiron.endpoint import JsonEndpoint

class GitHub(Service):
    domain = 'https://api.github.com'
    user = JsonEndpoint(path='/users/{username}')
    repo = JsonEndpoint(path='/repos/{org}/{repo}')

GITHUB = GitHub()

defunkt = ServiceCaller.call(GITHUB, GITHUB.user, path_kwargs={'username', 'defunkt'})


# After
from apiron.service.base import Service
from apiron.endpoint import JsonEndpoint

class GitHub(Service):
    domain = 'https://api.github.com'
    user = JsonEndpoint(path='/users/{username}')
    repo = JsonEndpoint(path='/repos/{org}/{repo}')

defunkt = GitHub.user(path_kwargs={'username', 'defunkt'})
```

3.2 Simplifying imports

As of version 2.4.0, most classes are available as top-level imports from the `apiron` package:

```
# Before
from apiron.service.base import Service
from apiron.endpoint import JsonEndpoint

# After
from apiron import Service, JsonEndpoint
```

3.3 Simplifying endpoint path placeholders

As of version 2.5.0, `path_kwargs` is no longer necessary; just pass path fillers as additional keyword arguments:

```
# Before
defunkt = GitHub.user(path_kwargs={'username', 'defunkt'})

# After
defunkt = GitHub.user(username='defunkt')
```

3.4 Summary

Prepare for apiron 3.X by installing apiron 2.5+ and doing the following:

- Replace `ServiceCaller.call` with the more direct `SomeService.some_endpoint`
- Import classes from `apiron` directly
- Replace `path_kwargs` with direct keyword arguments

CHAPTER

FOUR

API REFERENCE

4.1 Services

class apiron.service.base.**Service**

Bases: ServiceBase

A base class for low-level services.

A service has a domain off of which one or more endpoints stem.

classmethod get_hosts() → List[str]

The fully-qualified hostnames that correspond to this service. These are often determined by asking a load balancer or service discovery mechanism.

Returns
The hostname strings corresponding to this service

Return type
list

class apiron.service.discoverable.**DiscoverableService**

Bases: ServiceBase

A Service whose hosts are determined via a host resolver. A host resolver is any class with a `resolve()` method that takes a service name as its sole argument and returns a list of host names that correspond to that service.

classmethod get_hosts() → List[str]

The fully-qualified hostnames that correspond to this service. These are often determined by asking a load balancer or service discovery mechanism.

Returns
The hostname strings corresponding to this service

Return type
list

4.2 Endpoints

class apiron.endpoint.**Endpoint**(*path: str = '/', default_method: str = 'GET', default_params: dict[str, Any] | None = None, required_params: Iterable[str] | None = None, return_raw_response_object: bool = False, timeout_spec: Timeout | None = None, retry_spec: Retry | None = None*)

Bases: `object`

A basic service endpoint that responds with the default `Content-Type` for that endpoint

Parameters

- **path** (*str*) – The URL path for this endpoint, without the protocol or domain
- **default_method** (*str*) – (Default `'GET'`) The default method to use when calling this endpoint.
- **default_params** (*dict*) – The default parameters to use when calling this endpoint. Useful when an endpoint always or most often needs a base set of parameters supplied.
- **required_params** – An iterable of required parameter names. Calling an endpoint without its required parameters raises an exception.
- **return_raw_response_object** (*bool*) – Whether to return a `requests.Response` object or call *format_response()* on it first. This can be overridden when calling the endpoint. (Default `False`)
- **timeout_spec** (`Timeout`) – (optional) An override of the timeout behavior for calls to this endpoint. (default `None`)
- **retry_spec** (*urllib3.util.retry.Retry*) – (optional) An override of the retry behavior for calls to this endpoint. (default `None`)

format_response(*response: Response*) → str | dict[str, Any] | Iterable[bytes]

Extracts the appropriate type of response data from a `requests.Response` object

Parameters
response (*requests.Response*) – The original response from `requests`

Returns
The response's text content

Return type
str

get_formatted_path(***kwargs*) → str

Format this endpoint's path with the supplied keyword arguments

Returns
The fully-formatted path

Return type
str

get_merged_params(*supplied_params: dict[str, Any] | None = None*) → dict[str, Any]

Merge this endpoint's default parameters with the supplied parameters

Parameters
supplied_params (*dict*) – A dictionary of query parameter, value pairs

Returns
A dictionary of this endpoint's default parameters, merged with the supplied parameters. Any default parameters which have a value supplied are overridden.

Return type
dict

Raises
apiron.exceptions.UnfulfilledParameterException – When a required parameter for this endpoint is not a default param and is not supplied by the caller

property path_placeholders: list[str]

The formattable placeholders from this endpoint's path, in the order they appear.

Example:

```
>>> endpoint = Endpoint(path='/api/{foo}/{bar}')
>>> endpoint.path_placeholders
['foo', 'bar']
```

property required_headers: dict[str, Any]

Generates the headers that must be sent to this endpoint based on its attributes

Returns
Header name, header value pairs

Return type
dict

class apiron.endpoint.JsonEndpoint(**args, path: str = '/', default_method: str = 'GET', default_params: Dict[str, Any] | None = None, required_params: Iterable[str] | None = None, preserve_order: bool = False*)

Bases: *Endpoint*

An endpoint that returns *application/json*

Parameters

- **path** (*str*) – The URL path for this endpoint, without the protocol or domain
- **default_method** (*str*) – (Default `'GET'`) The default method to use when calling this endpoint.
- **default_params** (*dict*) – The default parameters to use when calling this endpoint. Useful when an endpoint always or most often needs a base set of parameters supplied.
- **required_params** – An iterable of required parameter names. Calling an endpoint without its required parameters raises an exception.
- **return_raw_response_object** (*bool*) – Whether to return a `requests.Response` object or call *format_response()* on it first. This can be overridden when calling the endpoint. (Default `False`)
- **timeout_spec** (`Timeout`) – (optional) An override of the timeout behavior for calls to this endpoint. (default `None`)
- **retry_spec** (*urllib3.util.retry.Retry*) – (optional) An override of the retry behavior for calls to this endpoint. (default `None`)

format_response(*response*) → Dict[str, Any]

Extracts JSON data from the response

Parameters
response (*requests.Response*) – The original response from `requests`

Returns
The response's JSON content

Return type
`dict` if `preserve_order` is `False`

Return type
`collections.OrderedDict` if `preserve_order` is `True`

property required_headers: Dict[str, str]

Generates the headers that must be sent to this endpoint based on its attributes

Returns
Header name, header value pairs

Return type
dict

class apiron.endpoint.**StreamingEndpoint**(*path: str = '/', default_method: str = 'GET', default_params: dict[str, Any] | None = None, required_params: Iterable[str] | None = None, return_raw_response_object: bool = False, timeout_spec: Timeout | None = None, retry_spec: Retry | None = None*)

Bases: *Endpoint*

An endpoint that streams data incrementally

Parameters

- **path** (*str*) – The URL path for this endpoint, without the protocol or domain
- **default_method** (*str*) – (Default `'GET'`) The default method to use when calling this endpoint.
- **default_params** (*dict*) – The default parameters to use when calling this endpoint. Useful when an endpoint always or most often needs a base set of parameters supplied.
- **required_params** – An iterable of required parameter names. Calling an endpoint without its required parameters raises an exception.
- **return_raw_response_object** (*bool*) – Whether to return a `requests.Response` object or call *format_response()* on it first. This can be overridden when calling the endpoint. (Default `False`)
- **timeout_spec** (`Timeout`) – (optional) An override of the timeout behavior for calls to this endpoint. (default `None`)
- **retry_spec** (*urllib3.util.retry.Retry*) – (optional) An override of the retry behavior for calls to this endpoint. (default `None`)

format_response(*response*) → Iterable[bytes]

Stream response in chunks

Parameters
response (*requests.Response*) – The original response from `requests`

Returns
The response's content

Return type
generator

class apiron.endpoint.**StubEndpoint**(*stub_response: Any | None = None, **kwargs*)

Bases: *Endpoint*

A stub endpoint designed to return a pre-baked response

The intent is to allow for a service to be implemented before the endpoint is complete.

Parameters

- **stub_response** – A pre-baked response or response-determining function. Pre-baked response example: `'stub response'` or `{'stub': 'response'}` A response-determining function may operate on any arguments provided to the client's `call` method. Example of a response-determining function:

```
def stub_response(**kwargs):
    if kwargs.get('params') and kwargs['params'].get('param_key') ==
↪'param_value':
        return {'stub response': 'for param_key=param_value'}
    else:
        return {'default': 'response'}
```

- ****kwargs** – Arbitrary parameters that can match the intended real endpoint. These don't do anything for the stub but streamline the interface.

4.3 Service client

class apiron.client.**Timeout**(*connection_timeout, read_timeout*)

Bases: `tuple`

Create new instance of Timeout(connection_timeout, read_timeout)

connection_timeout

Alias for field number 0

read_timeout

Alias for field number 1

apiron.client.**call**(*service: apiron.Service, endpoint: apiron.Endpoint, method: str | None = None, session: requests.Session | None = None, params: dict[str, Any] | None = None, data: dict[str, Any] | None = None, files: dict[str, str] | None = None, json: dict[str, Any] | None = None, headers: dict[str, Any] | None = None, cookies: dict[str, Any] | None = None, auth: Any | None = None, encoding: str | None = None, retry_spec: retry.Retry | None = None, timeout_spec: Timeout | None = None, logger: logging.Logger | None = None, allow_redirects: bool = True, return_raw_response_object: bool | None = None, **kwargs*)

Parameters

- **service** (`Service`) – The service that hosts the endpoint being called
- **endpoint** (`Endpoint`) – The endpoint being called
- **method** (`str`) – The HTTP method to use for the call
- **session** (`requests.Session`) – (optional) An existing session, useful for making many calls in a single session (default `None`)

- **params** (*dict*) – (optional) GET parameters to send to the endpoint (default None)
- **data** (*dict*) – (optional) POST data to send to the endpoint. A dict will be form-encoded, while a str will be sent raw (default None)
- **files** (*dict*) – (optional) Dictionary of 'filename': file-like-objects for multipart encoding upload. (default None)
- **json** (*dict*) – (optional) A JSON-serializable dictionary that will be sent as the POST body (default None)
- **headers** (*dict*) – HTTP Headers to send to the endpoint (default None)
- **cookies** (*dict*) – Cookies to send to the endpoint (default None)
- **auth** – An object suitable for the requests.Request object's auth argument
- **encoding** (*str*) – The codec to use when decoding the response. Default behavior is to have requests guess the codec. (default None)
- **retry_spec** (*urllib3.util.retry.Retry*) – (optional) An override of the retry behavior for this call. (default None)
- **timeout_spec** (Timeout) – (optional) An override of the timeout behavior for this call. (default None)
- **logger** (*logging.Logger*) – (optional) An existing logger for logging from the proper caller for better correlation
- **allow_redirects** (*bool*) – (optional) Enable/disable GET/OPTIONS/POST/PUT/PATCH/DELETE/HEAD redirection (default True)
- **return_raw_response_object** (*bool*) – Whether to return a requests.Response object or call format_response() on it first. (Default False)
- ****kwargs** – Arguments to be formatted into the endpoint argument's path attribute

Returns

The result of endpoint's format_response()

Return type

The type returned by endpoint's format_response()

Raises

- **requests.RetryError** – if retry threshold exceeded due to bad HTTP codes (default 500 range)
- **requests.ConnectionError** – if retry threshold exceeded due to connection or request timeouts

CHAPTER

FIVE

WRITING DOCUMENTATION

Learn about *developing these docs* themselves.

CHAPTER

SIX

INDICES

You can always find these features in the navigation menu.

- genindex
- modindex
- search

6.1 Developing the apiron documentation

These docs are built using sphinx.

6.1.1 Developing

If you have *tox* installed, you may build these docs by running *tox -e docs*. Otherwise, you can follow the instructions below to manually install dependencies and build the docs.

Automated

You can use the `docs` environment for `tox` to build the documentation. With `tox` installed, run `tox -e docs` to build the HTML documentation. After the documentation is built, you can serve the build directory, which will be located at `.tox/docs/tmp/docs`.

Manual

You can also manage the documentation manually if you want more control.

Installation

Use your favorite method to create a virtual environment and install the package with its extras for documentation:

```
$ cd /path/to/apiron/
$ pyenv virtualenv 3.8.0 apiron  # pick your favorite virtual environment tool
$ pyenv local apiron
(apiron) $ pip install -e .[docs]
```

Building

You can build or rebuild the static documentation using `make`:

```
$ cd /path/to/apiron/docs/
(apiron) $ make html
Running Sphinx v1.7.4
...
build succeeded.

The HTML pages are in _build/html.
```

If you'd instead like to have the docs rebuilt as you're changing them, you can watch for changes:

```
$ cd /path/to/apiron/docs/
(apiron) $ make watch
[...] Serving on http://127.0.0.1:8000
[...] Start watching changes
[...] Start detecting changes
```

PYTHON MODULE INDEX

a

INDEX