PySpeckle

Jun 21, 2020

Contents:

CHAPTER 1

PySpeckle

A Python Speckle Client

Speckle.Works

> Speckle: open digital infrastructure for designing, making and operating the built environment. We reimagine the design process from the Internet up: Speckle is an open source (MIT) initiative for developing an extensible Design & AEC data communication and collaboration platform.

1.1 Installation

PySpeckle can be installed through `pip`: `pip install speckle`

1.2 Disclaimer

This code is WIP and as such should be used with caution, on non-sensitive projects.

1.3 Description

PySpeckle is a light Python wrapper / interface for the Speckle framework. It can be used independently through Python scripts, or as a base for building various plug-ins, such as SpeckleBlender.

1.4 Quick Start

Here is how you initialise a client, authenticate and start speckling:

```
from speckle import SpeckleApiClient

# Create a client using the appropriate server
client = SpeckleApiClient('hestia.speckle.works')

# Login with your details
client.login(
    email='test@test.com',
    password='Speckle<3Python'
)

# Stream ID to get
stream_id = 'HjenwS2s'

# Get stream data using its ID
stream = client.streams.get(stream_id)

# Print the list of placeholder objects in the stream
for object in stream.objects:
  print(object)
```

To get a list of all available streams and find a particular one by name:

```
# Fetch the list of all available streams
streams = client.streams.list()
name = "JetStream"

# Go through the list and find the stream by name
stream = None
for s in streams:
    if s.name == name:
        stream = s
        break

# If the stream is found, fetch the full stream data, using an optional query dict
# to omit some data
if stream:
    stream_data = client.streams.get(stream.streamId, {'omit':['layers','comments']})
```

To get object data from a stream:

```
stream = client.streams.get(streamId)

# Fetch a single object using its placeholder ID
object = client.objects.get(stream.objects[0].id)

# Fetch the objects all at once using an optional query dict
objects = client.objects.get_bulk([o.id for o in stream.objects], {'omit':'base64',
↪'displayValue'})

# Print out some object info
for o in objects:
    print("Object {} is type {}".format(o.id, o.type)
```

To create some data and upload it to a stream:

```
import speckle.schemas
```

(continues on next page)

(continued from previous page)

```
# Create some mesh data
vertices = [[0,0,0],[1,0,0],[1,1,0], [0,1,0]]
faces = [[0,1,2,3]]

# Create a Speckle Mesh object
sm = speckle.schemas.Mesh()

# Add vertices
for v in vertices:
    sm.vertices.extend(v)

# Add faces
for f in faces:
    if len(f) == 3: # if it is a triangle...
        sm.faces.append(0)
    elif len(f) == 4: # if it is a quad...
        sm.faces.append(1)
    sm.faces.extend(f)

# Give it a nice name
sm.name = "FancyMesh"

# Create the object on the server and receive a list of
# placeholders in return (with only one placeholder)
placeholders = client.objects.create(sm)

# Fetch the stream that we want to update
stream = client.streams.get(streamId)

# Set the stream object list to the created object or
# extend it to add the object to the existing list
stream.objects = placeholders
#stream.objects.extend(placeholders)

# Update the stream with the new data
client.streams.update(stream.streamId, stream)
```

Usage documentation can be found here.

1.5 Maintainers

SpeckleBlender is written and maintained by Tom Svilans (Github), Izzy Lyseggen and Antoine Dao.

1.6 Notes

Commit formatting can be found here.

1.6.1 Installation

PySpeckle can be installed through `pip`:

```
$ pip install speckle
```

The Pypi repository can be found here

1.6.2 QuickStart

Playing with a new library or API client can be daunting. We feel you! This is why we have compiled a bunch of example usage scenarios to walk you through how to use the python speckle client.

In the following set of examples we assume you don't have your own Speckle server up and running so you can test all of the steps below by connecting to the `hestia` speckle server.

First we must instantiate the `SpeckleApiClient`. You can name it whatever you want, we will use `client` for simplicity. As we are connecting to the `hestia` test server we will input the full domain name as an argument: `hestia.speckle.works`

```
from speckle import SpeckleApiClient

client = SpeckleApiClient('hestia.speckle.works')
```

Registering and Logging in

Before you can do anything with Speckle you need to register yourself to the server you are going to be using. If you have already registered then you can skip the first code section below and look at the one right after where we explain how to login.

Note: When you register using a PySpeckle client this client is also authenticated and you can start using it to make API calls immediately without needing to login.

```
client.register(
    email='test@test.com',
    password='Speckle<3Python',
    company='SnakeySnakes',
    name='Snakey',
    surname='Snake'
)
```

If you are already registered then you can simply login your client as follows:

```
client.login(
    email='test@test.com',
    password='Speckle<3Python'
)
```

Sending and Retrieving Things

At the moment the api client takes dictionary payloads as data inputs (a `Speckle Object` for example) and returns data classes in most cases (essentially a class instance). If a data class is not returned then the object return is a dictionary.

Warning: Whether the client returns a data class or a dictionary is not yet very well segmented. To be sure you are getting back what you expect from the python client we reccomend reading through the *API referemce*

Attention: PySpeckle doesn't yet support any of the *websockets* data transfer methods that the SpeckleServer offers. We're working on it. Feel free to contribute if you've got an idea on how to implement it.

Common Methods

Speckle mostly follows a bog standard REST API setup where a user can `POST`, `GET`, `PUT` and `DELETE` different resources. As such the following verbs can be used to make standard REST calls to Speckle.

Warning: For most resources that support a *get*, *update*, *delete* call the *id* value to use to retrieve them is the *object.id* or Mongo *ObjectId*. The exception to this rule is the *Stream* resource which can only be retrieved by using the *StreamId* property of that stream.

Create

Equivalent of `POST / -d '{"key1":"value1", "key2":"value2"}'`

```
project = {
    'name': 'test project',
    'description': 'a project made for testing purposes',
    'tags': [],
    'streams': [],
}

project = client.projects.create(data=project)

print(project.id)
```

List

Equivalent of `GET /`

```
projects = client.projects.list()

for project in projects:
  print(project.dict())
```

Get

Equivalent of `GET /{id}`

```
project_id = '507f1f77bcf86cd799439011'

project = client.projects.get(project_id)

print(project.dict())
```

Update

Equivalent of `PUT /{id} -d '{"key1":"value1", "key2":"value2"}'`

Note: The update method returns a payload response dictionary rather than a resource data class. The response payload should look something like this

```
{
  "success": True,
  "message": "Patched ['description'] for 507f1f77bcf86cd799439011"
}
```

Warning: The *update* method will replace all the values in the input dictionary payload even if they are *None*. Be careful if you are trying to run *upsert* commands with dictionaries that contain *None* values

```
project_id = '507f1f77bcf86cd799439011'
project_update = {
    'description': 'a project updated for testing purposes',
}

response = client.projects.update(id=project_id, data=project_update)

assert response['success'], response['message']
```

Delete

Equivalent of `DELETE /{id}`

Note: The delete method returns a payload response dictionary rather than a resource data class. The response payload should look something like this

```
{
  "success": True,
  "message": "Project was deleted."
}
```

```
project_id = '507f1f77bcf86cd799439011'

response = client.projects.delete(id=project_id)

assert response['success'], response['message']
```

API Objects

If you've made it this far down you're probably thinking to yourself:

> Great I can login and mess with a project... anything else I can do?
>
> —Some Ambitious User

Glad you asked! There are a bunch of different types of Speckle objects or resources you can mess with using this python client. Here is a list of them with links to their specific documentation:

- Accounts
- Projects
- Streams
- API Clients
- Comments
- SpeckleObjects

Note: Each resource supports some or all of the common methods described above. Additionally each resource might support some extra methods which are specific to that resource. A good example is the *stream* resource which supports the *clone* method

```
streamId = 'GBexG1'

clone_stream, parent_stream = client.streams.clone(streamId)
```

Parting Notes

Hope you've found this quickstart guide useful. Please do contribute any proposed changes/improvements on the main projects github repo.

1.6.3 Client API Reference

This section offers an in-depth description of SpeckleClient API calls.

The `SpeckleClient` class

Speckle Client documentation

The SpeckleClient class is used to manage authentication and dispatch request to a given SpeckleServer.

Example

Instantiate a client to a given server and register/authenticate to it:

```
from speckle import SpeckleApiClient

client = SpeckleApiClient('myspeckle.speckle.works')
```

(continues on next page)

(continued from previous page)

```
client.register(
    email='test@test.com',
    password='Speckle<3Python',
    company='SnakeySnakes',
    name='Snakey',
    surname='Snake'
)

streams = client.streams.list()
```

For more detailed documentation on the inidividual resources available go *here*

class speckle.base.client.**ClientBase**(*host='hestia.speckle.works'*, *version='v1'*, *use_ssl=True*, *verbose=False*)

Base class for http speckle client

This class contains the basic properties required to register, authenticate and hold authentication credentials.

register(*email*, *password*, *company*, *name=None*, *surname=None*)

Register a new user to the speckle server

After a user has registered, this function will log them in and save auth token credentials in the client object.

Parameters

- **{str} -- Email address of the new user, must not exist on the server** (*email*) –
- **{str} -- Pretty self explanatory, must be at least 8 characters long** (*password*) –
- **{str} -- Company the user is registering to speckle under** (*company*) –

Keyword Arguments

- **{str} -- User name, server will default to 'Anonymous' if None (default** (*name*) – {None})
- **{str} -- User surname, server will default to '' if None (default** (*surname*) – {None})

login(*email*, *password*)

Login user to speckle server

After a user has logged in, this function will save auth token credentials in the client object.

Parameters

- **{str} -- Email address of the new user, must not exist on the server** (*email*) –
- **{str} -- Pretty self explanatory, must be at least 8 characters long. Oh, must also be the user's actual password...** (*password*) –

websockets(*stream_id*, *client_id=None*, *header=None*, *on_open=None*, *on_message=None*, *on_error=None*, *on_close=None*, *on_ping=None*, *on_pong=None*, *on_cont_message=None*, *get_mask_key=None*, *subprotocols=None*, *on_data=None*)

Connect to a specific stream on the host server through websockets

This function essentially generates the correct connection url the Speckle Server expects in order to connect to a specific stream. After that all this function does is instantiate a WebScoketApp class from the websocket-client package.

Parameters **{str} -- The id of a stream** (*stream_id*) –

Keyword Arguments

- **{str} -- The id of the client to authenticate as (default** (*client_id*) – {None})
- **{dict} -- custom header for websocket handshake (default** (*header*) – {None})
- **{function} -- callable object which is called at opening websocket. (default** (*on_open*) – {None}) This function takes 1 argument: 1. this class object
- **{function} -- callable object which is called when closed the connection (default** (*on_close*) – {None}) This function takes 1 argument: 1. this class object
- **{function} -- callable object which is called when receiving data (default** (*on_message*) – {None}) This function takes 2 arguments: 1. this class object 2. the utf-8 string sent by the server
- **{function} -- callable object which is called when an error is sent by the server (default** (*on_error*) – {None}) This function takes 2 arguments: 1. this class object 2. an exception object
- **{function} -- callable object which is called when the server pings (default** (*on_pong*) – {None}) This function takes 2 arguments: 1. this class object 2. the utf-8 string sent by the server
- **{function} -- callable object which is called when the server pings (default** – {None}) This function takes 2 arguments: 1. this class object 2. the utf-8 string sent by the server
- **{function} -- callback object which is called when receive continued frame data. (default** (*on_cont_message*) – {None}) This function takes 2 arguments: 1. this class object 2. the utf-8 string sent by the server 3. is continue flag, if 0 the data continues to the next frame
- **{function} -- callback object which is called when a message received (default** (*on_data*) – {None}) This is called before on_message or on_cont_message, and then on_message or on_cont_message is called. on_data has 4 argument. 1. this class object. 2. the utf-8 string sent by the server 3. data type. ABNF.OPCODE_TEXT or ABNF.OPCODE_BINARY will be same. 4. continue flag. if 0, the data continue
- **{function} -- a callable to produce new mask keys (default** (*get_mask_key*) – {None}) see the WebSocket.set_mask_key's docstring for more information
- **{list} -- list of available sub protocols (default** (*subprotocols*) – {None})

Returns WebSocketApp – a websocket-client instance

Example

```
from speckle import SpeckleApiClient

host = 'hestia.speckle.works'
stream_id = 'MawOwhxET'

client = SpeckleApiClient(host=host)
client.login('test@test.com', 'testestestest')

def print_message(ws, message):
    print(message)


ws = client.websockets(stream_id, on_message=print_message)

# Send a message to the stream
ws.send('Hi Speckle!!!')

# Start a listening server that will print what ever message
# is sent to it (due to the print_message function defined above)
ws.run_forever()
```

Accounts Methods

class speckle.resources.accounts.**Resource**(*session*, *basepath*, *me*)

API Access class for Accounts

The accounts resource is used to search, retrieve and manipulate user profiles. None of the methods return an instanciated data class, instead they all return dictionary payloads.

Example

Here is an example of what an account object looks like:

```
{
    "id": "507f1f77bcf86cd799439011"
    "name": "Test"
    "surname": "McTestyFace"
    "company": "Acme"
    "avatar": "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRDzvciiQ5-
↪P6itEjycriGb9l9sJH9E538C-tM9QRgFVTaj0Muq"
    "role": "user"
}
```

get(*id*)

Get a specific user from the SpeckleServer

Parameters **{str} -- The ID of the resource to retrieve**(*id*) –

Returns dict – The user

Example

```
>>> user_id = '507f1f77bcf86cd799439011'
>>> client.accounts.get(id=user_id)
{
    "success": True,
    "resource": {
        "id": "507f1f77bcf86cd799439011",
        "email": "test@test.com"
        "name": "Test"
        "surname": "McTestyFace"
        "company": "Acme"
        "avatar": "https://encrypted-tbn0.gstatic.com/images?
↪q=tbn:ANd9GcRDzvciiQ5-P6itEjycriGb9l9sJH9E538C-tM9QRgFVTaj0Muq"
        "role": "user"
    }
}
```

get_profile()

Get current logged in user's profile

Returns dict – The current user's profile

Example

```
>>> from speckle.SpeckleClient import SpeckleApiClient
>>> client = SpeckleApiClient(host='hestia.speckle.com')
>>> client.login(email='test@test.com', password='somesupersecret')
>>> client.accounts.get_profile()
{
    "success": True,
    "resource": {
        "id": "507f1f77bcf86cd799439011",
        "email": "test@test.com"
        "name": "Test"
        "surname": "McTestyFace"
        "company": "Acme"
        "avatar": "https://encrypted-tbn0.gstatic.com/images?
↪q=tbn:ANd9GcRDzvciiQ5-P6itEjycriGb9l9sJH9E538C-tM9QRgFVTaj0Muq"
        "role": "user"
    }
}
```

update_profile(*data*)

Update the current logged in user's profile

Parameters **{dict} -- A dictionary of profile values to be updated** (*data*) –

Returns dict – A confirmfation payload of the updated keys

Example

```
>>> from speckle.SpeckleClient import SpeckleApiClient
>>> client = SpeckleApiClient(host='hestia.speckle.com')
```

(continues on next page)

(continued from previous page)

```
>>> client.login(email='test@test.com', password='somesupersecret')
>>> client.accounts.get_profile()
{
    "success": True,
    "resource": {
        "id": "507f1f77bcf86cd799439011",
        "email": "test@test.com"
        "name": "Test"
        "surname": "McTestyFace"
        "company": "Acme"
        "avatar": "https://encrypted-tbn0.gstatic.com/images?
↪q=tbn:ANd9GcRDzvciiQ5-P6itEjycriGb919sJH9E538C-tM9QRgFVTaj0Muq"
        "role": "user"
    }
}
>>> client.accounts.update_profile({'name': 'Tester'})
{
    "success": True,
    "message": 'User profile updated.'
}
```

set_role(*id*, *role*)

Set the role of a user

> **Warning:** The client must be authenticated with an *admin* user to carry out this operation.

Parameters

- **{str} -- The ID of the user to be updated**(*id*) –
- **{str} -- The role to give the user**(*role*) –

Returns dict – A response payload confirming the user role update

Example

```
>>> from speckle.SpeckleClient import SpeckleApiClient
>>> client = SpeckleApiClient(host='hestia.speckle.com')
>>> client.login(email='test@test.com', password='somesupersecret')
>>> client.accounts.get_profile()
{
    "success": True,
    "resource": {
        "id": "507f1f77bcf86cd799439011",
        "email": "test@test.com"
        "name": "Test"
        "surname": "McTestyFace"
        "company": "Acme"
        "avatar": "https://encrypted-tbn0.gstatic.com/images?
↪q=tbn:ANd9GcRDzvciiQ5-P6itEjycriGb919sJH9E538C-tM9QRgFVTaj0Muq"
        "role": "admin"
    }
}
>>> user_to_modify = '9eeb71aa8b2a292512a3bf94091e2df8'
```

(continues on next page)

(continued from previous page)

```
>>> client.accounts.set_role(id=user_to_modify, role='admin')
{
    "success": True,
    "message": 'User profile updated.'
}
```

search(*search*)

Search for one or more users

Parameters **{str} -- A search string. Should be at least 3 characters long.** (*search*) –

Returns list – A list of found users

Example

```
>>> client.accounts.search('tom')
{
    "success": True,
    "resources": [{
        "id": "507f1f77bcf86cd799439011",
        "email": "tom@test.com"
        "name": "Tom"
        "surname": "Svilans"
        "company": "Acme"
        "avatar": "https://encrypted-tbn0.gstatic.com/images?
↪q=tbn:ANd9GcRDzvciiQ5-P6itEjycriGb919sJH9E538C-tM9QRgFVTaj0Muq"
        "role": "admin"
    },
    {
        "id": "397f19073b30eb9f71fc5bas",
        "email": "test@tom.com"
        "name": "Test"
        "surname": "McTestyFace"
        "company": "Tom Associated"
        "avatar": "https://encrypted-tbn0.gstatic.com/images?
↪q=tbn:ANd9GcRDzvciiQ5-P6itEjycriGb919sJH9E538C-tM9QRgFVTaj0Muq"
        "role": "user"
    },
    {
        "id": "9bfb9894c21e29014cd0e8166",
        "email": "tom@tom.com"
        "name": "Tom"
        "surname": "Tom"
        "company": "Tom"
        "avatar": "https://i.ytimg.com/vi/Dtm5nMlqq1Q/maxresdefault.jpg"
        "role": "admin"
    }]

}
```

Projects Methods

class speckle.resources.projects.**Project**(**data*)

Project Data Class

Parameters **{BaseModel} -- The BaseModel for all Speckle objects** (*ResourceBaseSchema*) –

class speckle.resources.projects.**Resource**(*session*, *basepath*, *me*)

API Access class for Projects

list()

List all projects

Returns list – A list of project data class instances

create(*data*)

Create a project from a data dictionary

Parameters **{dict} -- A dictionary describing a project**(*data*) –

Returns Project – The instance created on the Speckle Server

get(*id*)

Get a specific project from the SpeckleServer

Parameters **{str} -- The ID of the project to retrieve**(*id*) –

Returns Project – The project

update(*id*, *data*)

Update a specific project

Parameters

- **{str} -- The ID of the project to update**(*id*) –
- **{dict} -- A dict of values to update**(*data*) –

Returns dict – a confirmation payload with the updated keys

delete(*id*)

Delete a specific project

Parameters **{str} -- The ID of the project to delete**(*id*) –

Returns dict – A confirmation payload

comment_get(*id*)

Retrieve comments attached to a project

Parameters **{str} -- The ID of the project to retrieve comments from**(*id*) –

Returns list – A list of comments

comment_create(*id*, *data*)

Add a comment to a project

Parameters

- **{str} -- The ID of the project to comment on**(*id*) –
- **{dict} -- A comment dictionary object**(*data*) –

Returns CommentSchema – The comment created by the server

add_stream(*id*, *stream_id*)

Add a Stream to a project

Parameters

- **{str} -- The ID of a project**(*id*) –

- **{str} -- The StreamId of a stream** (*stream_id*) –

Returns dict – A dictionary with the stream and the project

remove_stream (*id*, *stream_id*)
Remove a stream from a project

Parameters

- **{str} -- The ID of a project** (*id*) –
- **{str} -- The StreamId of a stream** (*stream_id*) –

Returns dict – A dictionary with the stream and the project

add_user (*id*, *user_id*)
Add a user to a project.

Note: When a user is first added to a project they have read and write authorizations. If you only want to allow them to write be sure to downgrade the user right after adding them.

Parameters

- **{str} -- The ID of a project** (*id*) –
- **{str} -- The ID of a user** (*user_id*) –

Returns dict – A confirmation payload

remove_user (*id*, *user_id*)
Remove a user from a project

Note: When you remove a user from a project this also removes all of their read and write access to the project's streams.

Parameters

- **{str} -- The ID of a project** (*id*) –
- **{str} -- The ID of a user** (*user_id*) –

Returns dict – A confirmation payload

upgrade_user (*id*, *user_id*)
Upgrade a user to have Write access to the project and it's streams

Parameters

- **{str} -- The ID of a project** (*id*) –
- **{str} -- The ID of a user** (*user_id*) –

Returns dict – A confirmation payload

downgrade_user (*id*, *user_id*)
Downgrade a user to have only Read access to the project and it's streams

Parameters

- **{str} -- The ID of a project** (*id*) –

- **{str} -- The ID of a user** (*user_id*) –

Returns dict – A confirmation payload

Streams Methods

class speckle.resources.streams.**Resource**(*session*, *basepath*, *me*)

API Access class for Streams

list(*query={'omit': 'objects'}*)

List all streams

Returns list – A list of Streams, without objects attached

create(*data*)

Create a stream from a data dictionary

Parameters **{dict} -- A dictionary describing a stream** (*data*) –

Returns Stream – The instance created on the Speckle Server

get(*id*, *query=None*)

Get a specific stream from the SpeckleServer

Parameters **{str} -- The StreamId of the stream to retrieve** (*id*) –

Returns Stream – The stream

update(*id*, *data*)

Update a specific stream

Parameters

- **{str} -- The StreamId of the stream to update** (*id*) –
- **{dict} -- A dict of values to update** (*data*) –

Returns dict – a confirmation payload with the updated keys

delete(*id*)

Delete a specific stream

Parameters **{str} -- The StreamId of the stream to delete** (*id*) –

Returns dict – A confirmation payload

comment_get(*id*)

Retrieve comments attached to a stream

Parameters **{str} -- The StreamId of the stream to retrieve comments from** (*id*) –

Returns list – A list of comments

comment_create(*id*, *data*)

Add a comment to a stream

Parameters

- **{str} -- The StreamId of the stream to comment on** (*id*) –
- **{dict} -- A comment dictionary object** (*data*) –

Returns CommentSchema – The comment created by the server

clone(*id*, *name=None*)
Clone a stream

Parameters **{str} -- The StreamId of the stream to clone** (*id*) –

Keyword Arguments **{str} -- The name of the new cloned stream. eg** (*name*) – stream-x-2019-06-09-backup (default: {None})

Returns tuple – The clone and parent stream as dicts

diff(*id*, *other_id*)
Runs a diff on two streams

Parameters

- **{str} -- StreamId of the main stream** (*id*) –
- **{str} -- StreamId of the stream to compare** (*other_id*) –

Returns dict – A response payload with objects and layers as keys

list_objects(*id*)
Return the list of objects in a stream

Parameters **{str} -- StreamId of the stream to list objects from** (*id*) –

Returns list – A list of Speckle objects

list_clients(*id*)
Return the list of api clients connected to the stream

Parameters **{str} -- StreamId of the stream to list objects from** (*id*) –

Returns list – A list of API clients

Api Client Methods

class speckle.resources.api_clients.**Resource**(*session*, *basepath*, *me*)
API Access class for API Clients

The *api_client* resource mostly returns an API Client data class instance.

Example

Here is an example of what an API Client object looks like in dict form and when converted to a data class:

```
>>> from speckle.resources.api_clients import ApiClient
>>> api_client_dict =  {
        'role': 'Hybrid',
        'documentName': 'Test',
        'documentType': 'DataServer',
        'documentLocation': 'http://some.server.com',
        'online': True,
    }
>>>  api_client = ApiClient.parse_obj(api_client_dict)
>>> api_client.role
'Hybrid'
```

(continues on next page)

(continued from previous page)

```
>>> api_client.documentGuid # Automatically created when document is instantiated␣
↪if not specified
'e991c923-cbb7-45f4-9488-e0f61ba006c0'
```

list()

List all API clients

Returns: list – A list of API client data class instances

Example

Calling the *api_clients.list()* method returns a list of ApiClient class objects that the user has read access to. .. code-block:: python

```
>>> client.api_clients.list()
[<ApiClient>, <ApiClient>, <ApiClient>]
```

create(*data*)

Create an API clients from a data dictionary

Arguments: data {dict} – A dictionary describing an API client

Returns: ApiClient – The instance created on the Speckle Server

Example

```
>>> api_client_dict =  {
        'role': 'Hybrid',
        'documentName': 'Test',
        'documentType': 'DataServer',
        'documentLocation': 'http://some.server.com',
        'online': True,
    }
>>> new_api_client = client.api_clients.create(data=api_client_dict)
>>> new_api_client.id
'54759eb3c090d83494e2d804'
>>> new_api_client.documentName
'Test'
```

get(*id*)

Get a specific API client from the SpeckleServer

Arguments: id {str} – The ID of the API client to retrieve

Returns: ApiClient – The API client

Example

```
>>> api_client = client.api_clients.get('54759eb3c090d83494e2d804')
>>> api_client.id
'54759eb3c090d83494e2d804'
>>> api_client.role
'Hybrid'
>>> api_client.dict()
```

(continues on next page)

(continued from previous page)

```
{
    'id'='54759eb3c090d83494e2d804',
    'private': False,
    'canRead': [],
    'canWrite': [],
    'owner': '7354308922443094682',
    'createdAt': '2019-06-09T20:23:22+00:00',
    'updatedAt': '2019-06-09T22:38:35+00:00',
    'role': 'Hybrid',
    'documentName': 'Test',
    'documentType': 'DataServer',
    'documentLocation': 'http://some.server.com',
    'online': True,
}
```

update(*id*, *data*)

Update a specific API client

Arguments: id {str} – The ID of the API client to update data {dict} – A dict of values to update

Returns: dict – a confirmation payload with the updated keys

Example

```
>>> api_client.id
'54759eb3c090d83494e2d804'
>>> client.api_clients.update(id=api_client.id, data={'documentName': 'newTest
↪', 'documentLocation': 'https://some.new.server.com/dump'})
{
    "success": True,
    "message": "Client updated following fields: ['documentName',
↪'documentLocation']"
}
```

delete(*id*)

Delete a specific API client

Arguments: id {str} – The ID of the API client to delete

Returns: dict – A confirmation payload

Example

```
>>> api_client.id
'54759eb3c090d83494e2d804'
>>> client.api_clients.delete(id=api_client.id)
{
    "success": True,
    "message": 'Client was deleted! Bye bye data.'"
}
```

Comments methods

class speckle.resources.comments.**Resource**(*session*, *basepath*, *me*)
API Access class for Comments

list()
List all comments

Returns list – A list of comment data class instances

get(*id*)
Get a specific comment from the SpeckleServer

Parameters **{str} -- The ID of the comment to retrieve**(*id*) –

Returns Comment – The comment

update(*id*, *data*)
Update a specific comment

Parameters

- **{str} -- The ID of the comment to update**(*id*) –
- **{dict} -- A dict of values to update**(*data*) –

Returns dict – a confirmation payload with the updated keys

delete(*id*)
Delete a specific comment

Parameters **{str} -- The ID of the comment to delete**(*id*) –

Returns dict – A confirmation payload

comment_get(*id*)
Retrieve comments attached to a comment

Parameters **{str} -- The ID of the comment to retrieve comments from**(*id*) –

Returns list – A list of comments

comment_create(*id*, *data*)
Add a comment to a comment

Parameters

- **{str} -- The ID of the comment to comment on**(*id*) –
- **{dict} -- A comment dictionary object**(*data*) –

Returns Comment – The comment created by the server

assigned()
Get the list of all comments where the logged in user is assigned

Returns list – A list of comments the user is assigned to

Objects Methods

class speckle.resources.objects.**Resource**(*session*, *basepath*, *me*)
API Access class for Speckle Objects

list(*query=None*)
List all Speckle objects

Returns list – A list of Speckle object data class instances

create(*data*)
Create a Speckle object from a data dictionary

Parameters **{dict} -- A dictionary describing a Speckle object** (*data*) –

Returns SpeckleObject – The instance created on the Speckle Server

get(*id*, *query=None*)
Get a specific Speckle object from the SpeckleServer

Parameters **{str} -- The ID of the Speckle object to retrieve**(*id*) –

Returns SpeckleObject – The Speckle object

update(*id*, *data*)
Update a specific Speckle object

Parameters

- **{str} -- The ID of the Speckle object to update**(*id*) –
- **{dict} -- A dict of values to update**(*data*) –

Returns dict – a confirmation payload with the updated keys

delete(*id*)
Delete a specific Speckle object

Parameters **{str} -- The ID of the Speckle object to delete**(*id*) –

Returns dict – A confirmation payload

comment_get(*id*)
Retrieve comments attached to a Speckle object

Parameters **{str} -- The ID of the Speckle object to retrieve comments from**(*id*) –

Returns list – A list of comments

comment_create(*id*, *data*)
Add a comment to a Speckle object

Parameters

- **{str} -- The ID of the Speckle object to comment on**(*id*) –
- **{dict} -- A comment dictionary object**(*data*) –

Returns CommentSchema – The comment created by the server

get_bulk(*object_ids*, *query=None*)
Retrieve and optionally update a list of Speckle objects

Parameters

- **{list} -- A list of object IDs**(*object_ids*) –
- **{dict} -- A dictionary to specifiy which fields to retrieve, filters, limits, etc**(*query*) –

Returns list – A list of SpeckleObjects

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index