ciscosparkapi Documentation

Release 0.10.post6.dev0+g2dd57ca

Chris Lunsford

Oct 16, 2018

Contents

Simple, lightweight, scalable Python API wrapper for the Cisco Spark APIs

---

Welcome to the docs! ciscosparkapi is a *community developed* Pythonic wrapping of the Cisco Spark APIs. The package represents all of the Cisco Spark API interactions via native Python tools. Making working with the Cisco Spark APIs in Python a *native* and *natural* experience.

ciscosparkapi helps you get things done faster. We take care of the API semantics, and you can focus on writing your code.

With ciscosparkapi, you can easily:

- Interact with the Cisco Spark APIs in an interactive Python session
- Quickly create code that enables you get something done in Spark
- Leverage the API wrapper to cleanly add Spark functionality to your project without having to write the boilerplate code for working with the Spark APIs

To *dive in* and see how ciscosparkapi makes your life better, check out the *Quickstart*!

CHAPTER 1

The User Guide

1.1 Installation

1.1.1 PIP Install

ciscosparkapi is available via PIP and the Python Package Index (PyPI). To install ciscosparkapi, simply run this command from your terminal of choice:

```
$ pip install ciscosparkapi
```

The ciscosparkapi package is distributed as a *source distribution* (no binaries).

1.1.2 PIP Upgrade

To ensure that you have the latest version, check-for and install upgrades via PIP:

```
$ pip install ciscosparkapi --upgrade
```

1.1.3 Get the Source Code

ciscosparkapi is developed on GitHub. If you like and use this package, please take a few seconds to Star the package on the CiscoDevNet/ciscosparkapi GitHub page. Your feedback and contributions are always welcome.

Use the following command to download the source code (GIT repository):

```
$ git clone https://github.com/CiscoDevNet/ciscosparkapi.git
```

You can then install the package to your environment, with the following command:

```
$ python setup.py install
```

1.2 Introduction

1.2.1 Work with the Cisco Spark APIs in Native Python!

Sure, working with the Cisco Spark APIs is easy (see developer.ciscospark.com). They are RESTful, naturally structured, require only a simple Access Token for authentication, and the data is elegantly represented in intuitive JSON. What could be easier?

```
import requests

URL = 'https://api.ciscospark.com/v1/messages'
ACCESS_TOKEN = '<your_access_token>'
ROOM_ID = '<room_id>'
MESSAGE_TEXT = '<message_text>'

headers = {'Authorization': 'Bearer ' + ACCESS_TOKEN,
           'Content-type': 'application/json;charset=utf-8'}
post_data = {'roomId': ROOM_ID,
             'text': MESSAGE_TEXT}
response = requests.post(URL, json=post_data, headers=headers)
if response.status_code == 200:
    # Great your message was posted!
    message_id = response.json['id']
    message_text = response.json['text']
    print("New message created, with ID:", message_id)
    print(message_text)
else:
    # Oops something went wrong...  Better do something about it.
    print(response.status_code, response.text)
```

Like I said, EASY. However, in use, the code can become rather repetitive...

- You have to setup the environment every time
- You have to remember URLs, request parameters and JSON formats (or reference the docs)
- You have to parse the returned JSON and work with multiple layers of list and dictionary indexes
- When requesting lists of items, you have to deal with pagination

Enter **ciscosparkapi**, a simple API wrapper that wraps all of the Spark API calls and returned JSON objects within native Python objects and methods.

With ciscosparkapi, the above Python code can be consolidated to the following:

```
from ciscosparkapi import CiscoSparkAPI

api = CiscoSparkAPI()
try:
    message = api.messages.create('<room_id>', text='<message_text>')
    print("New message created, with ID:", message.id)
    print(message.text)
except SparkApiError as e:
    print(e)
```

ciscosparkapi handles all of this for you:

- Reads your Spark access token from a `SPARK_ACCESS_TOKEN` environment variable

- Wraps and represents all Spark API calls as a simple hierarchical tree of native-Python methods (with default arguments provided everywhere possible!)
- If your Python IDE supports **auto-completion** (like PyCharm), you can navigate the available methods and object attributes right within your IDE
- Represents all returned JSON objects as native Python objects - you can access all of the object's attributes using native *dot.syntax*
- **Automatic and Transparent Pagination!** When requesting 'lists of objects' from Spark, requests for additional pages of responses are efficiently and automatically requested as needed
- **Automatic Rate-Limit Handling** Sending a lot of requests to Cisco Spark? Don't worry; we have you covered. Spark will respond with a rate-limit response, which will automatically be caught and "handled" for you. Your requests and script will automatically be "paused" for the amount of time specified by Spark, while we wait for the Spark rate-limit timer to cool down. After the cool-down, your request will automatically be retried, and your script will continue to run as normal. Handling all of this requires zero (0) changes to your code - you're welcome.

  Just know that if you are are sending a lot of requests, your script might take longer to run if your requests are getting rate limited.
- Multipart encoding and uploading of local files, when creating messages with local file attachments

All of this, combined, lets you do powerful things simply:

```
from ciscosparkapi import CiscoSparkAPI

api = CiscoSparkAPI()

# Find all rooms that have 'ciscosparkapi Demo' in their title
all_rooms = api.rooms.list()
demo_rooms = [room for room in all_rooms if 'ciscosparkapi Demo' in room.title]

# Delete all of the demo rooms
for room in demo_rooms:
    api.rooms.delete(room.id)

# Create a new demo room
demo_room = api.rooms.create('ciscosparkapi Demo')

# Add people to the new demo room
email_addresses = ["test01@cmlccie.com", "test02@cmlccie.com"]
for email in email_addresses:
    api.memberships.create(demo_room.id, personEmail=email)

# Post a message to the new room, and upload a file
api.messages.create(demo_room.id, text="Welcome to the room!",
                    files=["https://developer.ciscospark.com/images/logo_spark_lg@256.
↪png"])
```

That's more than 6 Spark API calls in less than 23 lines of code (with comments and whitespace), and likely more than that since ciscosparkapi handles pagination for you automatically!

Head over to the *Quickstart* page to begin working with the **Cisco Spark APIs in native Python**!

1.2.2 MIT License

ciscosparkapi is currently licensed under the MIT Open Source License, and distributed as a source distribution (no binaries) via *PyPI*, and the complete *source code* is available on GitHub.

1.2.3 ciscosparkapi License

1.3 Quickstart

Dive in! . . . to get started using the ciscosparkapi package:

Make sure that you have:

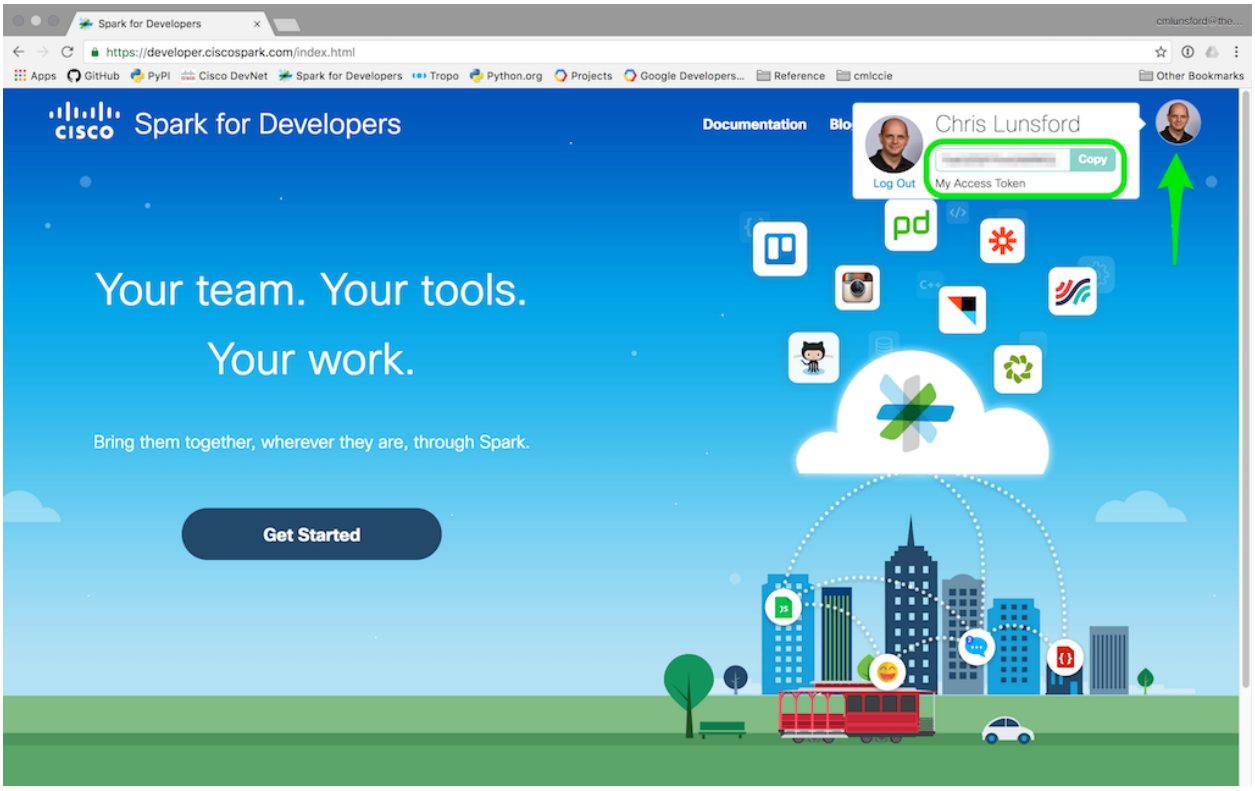
- A Cisco Spark Account *(a free account works fine,* sign-up for one here *)*
- ciscosparkapi *installed*
- ciscosparkapi *upgraded to the latest version*

1.3.1 Get your Spark Access Token

To interact with the Cisco Spark APIs, you must have a **Spark Access Token**. A Spark Access Token is how the Spark APIs validate access and identify the requesting user.

To get your personal access token:

1. Login to developer.ciscospark.com
2. Click on your avatar in the upper right hand corner
3. Click 'Copy' to copy your access token to your clipboard



1.3.2 Use your Spark Access Token

As a best practice, you can store your Spark access token 'credential' as an environment variable in your development or production environment. By default, ciscosparkapi will look for a `SPARK_ACCESS_TOKEN` environment variable when creating new connection objects.

There are many places and diverse ways that you can set an environment variable, which can include:

- A setting within your development IDE
- A setting in your container / PaaS service
- A statement in a shell script that configures and launches your app

It can be as simple as setting it in your CLI before running your script...

```
$ SPARK_ACCESS_TOKEN=your_access_token_here
$ python myscript.py
```

...or putting your credentials in a shell script that you `source` when your shell starts up or before your run a script:

```
$ cat mycredentials.sh
export SPARK_ACCESS_TOKEN=your_access_token_here
$ source mycredentials.sh
$ python myscript.py
```

However you choose to set it, if you have your access token stored in a `SPARK_ACCESS_TOKEN` environment variable when using ciscosparkapi, you are good to go. ciscosparkapi will pull and use this access token, by default, when creating new *`CiscoSparkAPI`* objects.

If you don't want to set your access token as an environment variable, or perhaps your application will acquire access tokens via some other means, you can manually provide your access token when creating a CiscoSparkAPI object.

1.3.3 Create a CiscoSparkAPI "Connection Object"

To make interacting with the Cisco Spark APIs as simple and intuitive as possible, all of the APIs have 'wrapped' underneath a single interface. To get started, import the *CiscoSparkAPI* class and create an API "connection object".

```
>>> from ciscosparkapi import CiscoSparkAPI
>>> api = CiscoSparkAPI()
```

As discussed above (*Use your Spark Access Token*), ciscosparkapi defaults to pulling your Spark access token from a SPARK_ACCESS_TOKEN environment variable. If you do not have this environment variable set and you try to create a new *CiscoSparkAPI* object without providing a Spark access token, a *ciscosparkapiException* will be raised.

```
>>> from ciscosparkapi import CiscoSparkAPI
>>> api = CiscoSparkAPI()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "ciscosparkapi/__init__.py", line 114, in __init__
    raise ciscosparkapiException(error_message)
ciscosparkapiException: You must provide an Spark access token to interact
with the Cisco Spark APIs, either via a SPARK_ACCESS_TOKEN environment
variable or via the access_token argument.
```

Use the access_token argument to manually provide your access token, when creating a new *CiscoSparkAPI* connection object.

```
>>> from ciscosparkapi import CiscoSparkAPI
>>> api = CiscoSparkAPI(access_token='lkj345w...')
```

Note that this can be very useful if you are reading in access token(s) from a file or database and/or when you want to create more than one connection object.

```
>>> from ciscosparkapi import CiscoSparkAPI
>>> chris_at = 'lkj345w...'
>>> veronica_at = 'kl45kln...'
>>> chris_api = CiscoSparkAPI(access_token=chris_at)
>>> veronica_api = CiscoSparkAPI(access_token=veronica_at)
```

1.3.4 Making API Calls

Now that you have created a *CiscoSparkAPI* "connection object," you are ready to start making API calls.

```
>>> api.people.me()
Person({"displayName": "Chris Lunsford", "firstName": "Chris", "created": "2012-06-
↪15T20:36:48.914Z", "lastName": "Lunsford", "emails": ["chrlunsf@cisco.com"], "avatar
↪": "https://1efa7a94ed216783e352-c62266528714497a17239ececf39e9e2.ssl.cf1.rackcdn.
↪com/V1~ba1ecf557a7e0b7cc3081998df965aad~7-HrvYOJSQ6eJgWJuFVbzg==~1600", "id":
↪"Y2lzY29zcGFyazovL3VzL1BFT1BMRS9mZjhlZTZmYi1hZmVmLTRhNGQtOTJiMS1kNmIyMTZiNTg5NDk"})
```

It really is that easy.

All of the calls have been wrapped and represented as native Python method calls, like `CiscoSparkAPI.people.me()` which gets the person details for the authenticated user (the user who's access token you are using) - see the https://api.ciscospark.com/v1/people/me API endpoint documentation.

As you can see, we have represented the API endpoints using simple terms that are aligned with the API docs; for example, representing the `people/me` API endpoint as a `people.me()` method available underneath the *`CiscoSparkAPI`* connection object.

A full list of the available API methods, with their descriptions and parameters, is available in the *User API Doc*, and a brief summary of the structure is provided here.

`CiscoSparkAPI`	*people*	`list()`
		`create()`
		`get()`
		`update()`
		`me()`
	rooms	`list()`
		`create()`
		`get()`
		`update()`
		`delete()`
	memberships	`list()`
		`create()`
		`get()`
		`update()`
		`delete()`
	messages	`list()`
		`create()`
		`get()`
		`delete()`
	teams	`list()`
		`create()`
		`get()`
		`update()`
		`delete()`
	team_memberships	`list()`
		`create()`
		`get()`
		`update()`
		`delete()`
	webhooks	`list()`
		`create()`
		`get()`
		`update()`
		`delete()`
	organizations	`list()`
		`create()`
	licenses	`list()`
		`create()`
	roles	`list()`
		`create()`
	events	`list()`
		`get()`
	access_tokens	`get()`

Continued on next page

Table 1.1 – continued from previous page

		`refresh()`

You can easily access and call any of these methods directly from your *CiscoSparkAPI* connection object:

```
>>> chris_id =
↪"Y2lzY29zcGFyazovL3VzL1BFT1BMRS9mZjhlZTZmYi1hZmVmLTRhNGQtOTJiMS1kNmIyMTZiNTg5NDk"
>>> api.people.get(personId=chris_id)
Person({"displayName": "Chris Lunsford", "firstName": "Chris", "created": "2012-06-
↪15T20:36:48.914Z", "lastName": "Lunsford", "emails": ["chrlunsf@cisco.com"], "avatar
↪": "https://1efa7a94ed216783e352-c62266528714497a17239ececf39e9e2.ssl.cf1.rackcdn.
↪com/V1~ba1ecf557a7e0b7cc3081998df965aad~7-HrvYOJSQ6eJgWJuFVbzg==~1600", "id":
↪"Y2lzY29zcGFyazovL3VzL1BFT1BMRS9mZjhlZTZmYi1hZmVmLTRhNGQtOTJiMS1kNmIyMTZiNTg5NDk"})
```

1.3.5 Catching Exceptions

If something should go wrong with the API call, an exception will be raised. *SparkApiError* exceptions are raised when an error condition is returned from the Cisco Spark cloud. Details will be provided in the error message.

```
>>> from ciscosparkapi import CiscoSparkAPI, SparkApiError
>>> api = CiscoSparkAPI()
>>> room = api.rooms.create("ciscosparkapi Test Room")
>>> me = api.people.me()
>>> api.memberships.create(roomId=room.id, personId=me.id)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "ciscosparkapi/api/memberships.py", line 212, in create
    json_obj = self._session.post('memberships', json=post_data)
  File "ciscosparkapi/restsession.py", line 187, in post
    check_response_code(response, erc)
  File "ciscosparkapi/utils.py", line 104, in check_response_code
    response=response)
ciscosparkapi.exceptions.SparkApiError: Response Code [409] - The request
could not be processed because it conflicts with some established rule of
the system. For example, a person may not be added to a room more than
once.
```

You can catch any errors returned by the Cisco Spark cloud by catching *SparkApiError* exceptions in a try-except block.

```
>>> try:
...     api.memberships.create(roomId=room.id, personId=me.id)
... except SparkApiError as e:
...     memberships = api.memberships.list(roomId=room.id)
...     for membership in memberships:
...         if membership.personId == me.id:
...             print("Doh!  I forgot that I am automatically added to a"
...                   "room when I create it.")
...             break
...     else:
...         print(e)
...
Doh!  I forgot that I am automatically added to a room when I create it.
>>>
```

ciscosparkapi will also raise a number of other standard errors (`TypeError`, `ValueError`, etc.); however, these errors are usually caused by incorrect use of the package or methods and should be sorted while debugging your app.

1.3.6 Working with Returned Objects

The Cisco Spark cloud returns data objects in JSON format, like so:

```
{
  "displayName": "Chris Lunsford",
  "firstName": "Chris",
  "created": "2012-06-15T20:36:48.914Z",
  "lastName": "Lunsford",
  "emails": [
    "chrlunsf@cisco.com"
  ],
  "avatar": "https://1efa7a94ed216783e352-c62266528714497a17239ececf39e9e2.ssl.cf1.
↪rackcdn.com/V1~ba1ecf557a7e0b7cc3081998df965aad~7-HrvYOJSQ6eJgWJuFVbzg==~1600",
  "id":
↪"Y2lzY29zcGFyazovL3VzL1BFT1BMRS9mZjhlZTZmYi1hZmVmLTRhNGQtOTJiMS1kNmIyMTZiNTg5NDk"
}
```

Sure, JSON data objects can easily be parsed and represented in Python using dictionaries, but when working with an 'object' wouldn't it be nice to be able to work with it like an object - using native object syntax (like accessing attributes using '.' notation)? ciscosparkapi enables you to do just that:

```
>>> me = api.people.me()
>>> me.id
u'Y2lzY29zcGFyazovL3VzL1BFT1BMRS9mZjhlZTZmYi1hZmVmLTRhNGQtOTJiMS1kNmIyMTZiNTg5NDk'
>>> me.displayName
u'Chris Lunsford'
```

Representing and treating Spark data objects as Python data objects, can really help clean up your code and make coding easier:

1. You don't need to create variables to hold the data attributes, just use the attributes available underneath the data object.

   ```
   >>> # Do this
   >>> api.people.get(personId=me.id)
   Person({"displayName": "Chris Lunsford", "firstName": "Chris", "created": "2012-
   ↪06-15T20:36:48.914Z", "lastName": "Lunsford", "emails": ["chrlunsf@cisco.com"],
   ↪"avatar": "https://1efa7a94ed216783e352-c62266528714497a17239ececf39e9e2.ssl.
   ↪cf1.rackcdn.com/V1~ba1ecf557a7e0b7cc3081998df965aad~7-HrvYOJSQ6eJgWJuFVbzg==~
   ↪1600", "id":
   ↪"Y2lzY29zcGFyazovL3VzL1BFT1BMRS9mZjhlZTZmYi1hZmVmLTRhNGQtOTJiMS1kNmIyMTZiNTg5NDk
   ↪"})
   >>> # Instead of this
   >>> my_id = me.id
   >>> api.people.get(personId=my_id)
   Person({"displayName": "Chris Lunsford", "firstName": "Chris", "created": "2012-
   ↪06-15T20:36:48.914Z", "lastName": "Lunsford", "emails": ["chrlunsf@cisco.com"],
   ↪"avatar": "https://1efa7a94ed216783e352-c62266528714497a17239ececf39e9e2.ssl.
   ↪cf1.rackcdn.com/V1~ba1ecf557a7e0b7cc3081998df965aad~7-HrvYOJSQ6eJgWJuFVbzg==~
   ↪1600", "id":
   ↪"Y2lzY29zcGFyazovL3VzL1BFT1BMRS9mZjhlZTZmYi1hZmVmLTRhNGQtOTJiMS1kNmIyMTZiNTg5NDk
   ↪"})
   ```

2. If your IDE supports auto-completion (like PyCharm for example), then you can easily see and 'tab-out' available attributes while coding.

   For Example: When working with a `Person` object, your can type the object name followed by a dot '`me.`' and see a list of available attributes. Typing a few more letters '`me.dis`' narrows down the attribute list to

'displayName', and you can now simply hit '<tab>' to complete your attribute 'me.displayName'.

This speeds up coding and reduces typo coding errors.

3. When accessing 'optional' attributes, like the `teamId` attribute of a Spark Room object (only present when the room is part of a Spark Team), the *ciscosparkapi.Room* object will return `None` when the attribute is not present and will return the attribute's value when it is present. This avoids some boiler plate code and/or needless exception handling, when working with optional attributes.

```
>>> # Instead of doing this
>>> if hasattr(room, 'teamId'):
...     # Do something with the teamId attribute
...     pass
>>> # Or this
>>> try:
...     # Do something with the teamId attribute
...     room.teamId
... except AttributeError as e:
...     pass
>>> # You can do this, which is cleaner
>>> if room.teamId:
...     # Do something with the teamId attribute
...     pass
```

4. It just feels more *natural*. :-) When iterating through sequences, and working with objects in those sequences (see the next section), working with objects as objects is definitely more Pythonic.

   The Zen of Python (PEP 20): "Beautiful is better than ugly." "Simple is better than complex."

A full list of the currently modeled *Spark Data Objects*, with their attributes, is available *here* in the *User API Doc*.

What if Spark adds new data attributes?

Attribute access WILL WORK for the newly added attributes (yes, without a package update!), but tab-completion WILL NOT. ciscosparkapi is written to automatically take advantage of new attributes and data as they are returned; however, tab-completion (which relies on source code and introspection) will not work until we update the ciscosparkapi package (which is easy to do; raise the issue on the issues page and bug us to add it).

1.3.7 Working with Returned 'Lists' of Objects

Challenge

When you ask Spark for a list of items (like all of the rooms that you are a member of or all of the messages in a room), Spark needs to return these items to you in an efficient way. Sending all of the messages in a room in one transaction or request isn't really feasible (imaging if the room had existed for years!). Additionally, what if you found what you were looking for in the first few (most recent) messages? Sending all of the items would have been a waste of time and resources.

To facilitate efficient transactions when requesting lists of items, the Spark APIs implement RFC5988 (Web Linking) to efficiently send 'pages' of responses (see Pagination on the Spark for Developers site). When you make a request to an Spark API that leverages pagination, Spark returns the first 'page' of results and a link to the 'next page' of results. If information you need isn't contained the first page, you can request the next and so forth.

Solution

Python has a similar construct as well - iterable objects. Iterable objects return their members one at a time, until they have all been returned.

ciscosparkapi marries these two concepts (pagination and iterables) to create a simple interface for working with sequences of returned objects.

```
>>> # Returns a iterable object yielding all of the rooms you are a member of
>>> rooms = api.rooms.list()

>>> # Which can easily be iterated to find what you are looking for
>>> for room in rooms:
...     if 'ciscosparkapi' in room.title:
...         demo_room = room
...         break

>>> demo_room
Room({"title": "ciscosparkapi Test Room", "created": "2016-11-12T03:24:39.278Z",
↪"isLocked": false, "lastActivity": "2016-11-12T03:24:39.308Z", "creatorId":
↪"Y2lzY29zcGFyazovL3VzL1BFT1BMRS9mZjhlZTZmYi1hZmVmLTRhNGQtOTJiMS1kNmIyMTZiNTg5NDk",
↪"type": "group", "id":
↪"Y2lzY29zcGFyazovL3VzL1JPT00vOGI1MTIwZTAtYTg4Ny0xMWU2LWFhZjUtZTlmYWEzMWQ1ZmRm"})
```

ciscosparkapi provides this functionality by returning `GeneratorContainer` objects for API calls that return lists of items.

In short, `GeneratorContainer` s are iterable objects that incrementally yield 'the next object' returned from your Spark API query request until all items have been returned, and they are reusable. If you create an `rooms` `GeneratorContainer`, like we did above with `rooms = api.rooms.list()`, you can use that object to iterate through the rooms not just once but many times.

Note: Every time you iterate a `GeneratorContainer` object, fresh API calls are made so you are always working with 'live data' from the Cisco Spark Cloud.

ciscosparkapi *automatically handles the pagination for you* so that you don't have to think about it or write the boiler plate code to handle requesting pages of responses. ciscosparkapi automatically and efficiently requests additional pages from Spark as needed to yield the items you have requested.

A `GeneratorContainer` records all of the parameters of your API call, and uses them to request data from Spark each time you iterate the container.

```
>>> # Returns a iterable object representing all of group rooms you are a member of
>>> group_rooms = api.rooms.list(type='group')

>>> # Returns a iterable object representing all of direct rooms you are a member of
>>> direct_rooms = api.rooms.list(type='direct')

>>> # Iterate through your group rooms
>>> for room in group_rooms:
...     pass

>>> # Iterate through your direct rooms
>>> for room in direct_rooms:
...     pass

>>> # You can iterate through your group rooms again;
>>> # if a new room has been created since the last time, it will show up.
>>> for room in group_rooms:
...     pass
```

These iterable objects are great, but what if I really DO want a list?

Sometimes you really DO want a `list` of items. Perhaps you want to work with the same static `list` of items to ensure you are looking at 'all of the items' and to make sure that your list doesn't change while you are working with it...

Whatever your reason for doing so, you can easily 'convert' an iterable object to a standard Python `list` with the `list()` function. This may take a little time for all of the API calls to be made, but the list will contain all of the returned objects.

```
>>> rooms_iterable = api.rooms.list()
>>> rooms_list = list(rooms_iterable)
```


1.4 User API Doc

1.4.1 CiscoSparkAPI

The *CiscoSparkAPI* class is the main interface for the package. All of the Spark APIs (people, rooms, etc.) and their API endpoints have been wrapped and hierarchically organized underneath the *CiscoSparkAPI* class.

class ciscosparkapi.**CiscoSparkAPI**

Cisco Spark API wrapper.

Creates a 'session' for all API calls through a created CiscoSparkAPI object. The 'session' handles authentication, provides the needed headers, and checks all responses for error conditions.

CiscoSparkAPI wraps all of the individual Cisco Spark APIs and represents them in a simple hierarchical structure.

CiscoSparkAPI `people`

`rooms`

`memberships`

`messages`

`teams`

`team_memberships`

`webhooks`

`organizations`

`licenses`

`roles`

`events`

`access_tokens`

__init__(*access_token=None*, *base_url='https://api.ciscospark.com/v1/'*, *timeout=None*, *single_request_timeout=60*, *wait_on_rate_limit=True*, *object_factory=<function spark_data_factory>*)

Create a new CiscoSparkAPI object.

An access token must be used when interacting with the Cisco Spark API. This package supports two methods for you to provide that access token:

1. You may manually specify the access token via the access_token argument, when creating a new CiscoSparkAPI object.
2. If an access_token argument is not supplied, the package checks for a SPARK_ACCESS_TOKEN environment variable.

A ciscosparkapiException is raised if an access token is not provided via one of these two methods.

Parameters

- **access_token** (*basestring*) – The access token to be used for API calls to the Cisco Spark service. Defaults to checking for a SPARK_ACCESS_TOKEN environment variable.
- **base_url** (*basestring*) – The base URL to be prefixed to the individual API endpoint suffixes. Defaults to ciscosparkapi.DEFAULT_BASE_URL.
- **timeout** (*int*) – [deprecated] Timeout (in seconds) for RESTful HTTP requests. Defaults to ciscosparkapi.DEFAULT_TIMEOUT.
- **single_request_timeout** (*int*) – Timeout (in seconds) for RESTful HTTP requests. Defaults to ciscosparkapi.DEFAULT_SINGLE_REQUEST_TIMEOUT.
- **wait_on_rate_limit** (*bool*) – Enables or disables automatic rate-limit handling. Defaults to ciscosparkapi.DEFAULT_WAIT_ON_RATE_LIMIT.
- **object_factory** (*callable*) – The factory function to use to create Python objects from the returned Cisco Spark JSON data objects.

Returns A new CiscoSparkAPI object.

Return type *CiscoSparkAPI*

Raises

- TypeError – If the parameter types are incorrect.
- *ciscosparkapiException* – If an access token is not provided via the access_token argument or SPARK_ACCESS_TOKEN environment variable.

single_request_timeout
Timeout (in seconds) for an single HTTP request.

wait_on_rate_limit
Automatic rate-limit handling enabled / disabled.

people

class ciscosparkapi.api.people.**PeopleAPI**
Cisco Spark People API.

Wraps the Cisco Spark People API and exposes the API as native Python methods that return native Python objects.

list(*email=None*, *displayName=None*, *id=None*, *orgId=None*, *max=None*, ***request_parameters*)
List people

This method supports Cisco Spark's implementation of RFC5988 Web Linking to provide pagination support. It returns a generator container that incrementally yields all people returned by the query. The generator will automatically request additional 'pages' of responses from Spark as needed until all responses have been returned. The container makes the generator safe for reuse. A new API call will be made, using the same parameters that were specified when the generator was created, every time a new iterator is requested from the container.

Parameters

- **email** (*basestring*) – The e-mail address of the person to be found.

- **displayName** (*basestring*) – The complete or beginning portion of the displayName to be searched.
- **id** (*basestring*) – List people by ID. Accepts up to 85 person IDs separated by commas.
- **orgId** (*basestring*) – The organization ID.
- **max** (*int*) – Limit the maximum number of items returned from the Spark service per request.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A GeneratorContainer which, when iterated, yields the people returned by the Cisco Spark query.

Return type GeneratorContainer

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

create(*emails*, *displayName=None*, *firstName=None*, *lastName=None*, *avatar=None*, *orgId=None*, *roles=None*, *licenses=None*, ***request_parameters*)

Create a new user account for a given organization

Only an admin can create a new user account.

Parameters

- **emails** (*list*) – Email address(es) of the person (list of strings).
- **displayName** (*basestring*) – Full name of the person.
- **firstName** (*basestring*) – First name of the person.
- **lastName** (*basestring*) – Last name of the person.
- **avatar** (*basestring*) – URL to the person's avatar in PNG format.
- **orgId** (*basestring*) – ID of the organization to which this person belongs.
- **roles** (*list*) – Roles of the person (list of strings containing the role IDs to be assigned to the person).
- **licenses** (*list*) – Licenses allocated to the person (list of strings - containing the license IDs to be allocated to the person).
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A Person object with the details of the created person.

Return type *Person*

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

get(*personId*)

Get a person's details, by ID.

Parameters **personId** (*basestring*) – The ID of the person to be retrieved.

Returns A Person object with the details of the requested person.

Return type *Person*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

update(*personId*, *emails=None*, *displayName=None*, *firstName=None*, *lastName=None*, *avatar=None*, *orgId=None*, *roles=None*, *licenses=None*, ***request_parameters*)

Update details for a person, by ID.

Only an admin can update a person's details.

Email addresses for a person cannot be changed via the Spark API.

Include all details for the person. This action expects all user details to be present in the request. A common approach is to first GET the person's details, make changes, then PUT both the changed and unchanged values.

Parameters

- **personId** (*basestring*) – The person ID.
- **emails** (*list*) – Email address(es) of the person (list of strings).
- **displayName** (*basestring*) – Full name of the person.
- **firstName** (*basestring*) – First name of the person.
- **lastName** (*basestring*) – Last name of the person.
- **avatar** (*basestring*) – URL to the person's avatar in PNG format.
- **orgId** (*basestring*) – ID of the organization to which this person belongs.
- **roles** (*list*) – Roles of the person (list of strings containing the role IDs to be assigned to the person).
- **licenses** (*list*) – Licenses allocated to the person (list of strings - containing the license IDs to be allocated to the person).
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A Person object with the updated details.

Return type *Person*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

delete(*personId*)

Remove a person from the system.

Only an admin can remove a person.

Parameters **personId** (*basestring*) – The ID of the person to be deleted.

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

me()
Get the details of the person accessing the API.

Raises SparkApiError – If the Cisco Spark cloud returns an error.

rooms

class ciscosparkapi.api.rooms.**RoomsAPI**
Cisco Spark Rooms API.

Wraps the Cisco Spark Rooms API and exposes the API as native Python methods that return native Python objects.

list(*teamId=None*, *type=None*, *sortBy=None*, *max=None*, ***request_parameters*)
List rooms.

By default, lists rooms to which the authenticated user belongs.

This method supports Cisco Spark's implementation of RFC5988 Web Linking to provide pagination support. It returns a generator container that incrementally yields all rooms returned by the query. The generator will automatically request additional 'pages' of responses from Spark as needed until all responses have been returned. The container makes the generator safe for reuse. A new API call will be made, using the same parameters that were specified when the generator was created, every time a new iterator is requested from the container.

Parameters

- **teamId** (*basestring*) – Limit the rooms to those associated with a team, by ID.
- **type** (*basestring*) – 'direct' returns all 1-to-1 rooms. *group* returns all group rooms. If not specified or values not matched, will return all room types.
- **sortBy** (*basestring*) – Sort results by room ID (*id*), most recent activity (*lastactivity*), or most recently created (*created*).
- **max** (*int*) – Limit the maximum number of items returned from the Spark service per request.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A GeneratorContainer which, when iterated, yields the rooms returned by the Cisco Spark query.

Return type GeneratorContainer

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

create(*title*, *teamId=None*, ***request_parameters*)
Create a room.

The authenticated user is automatically added as a member of the room.

Parameters

- **title** (*basestring*) – A user-friendly name for the room.
- **teamId** (*basestring*) – The team ID with which this room is associated.

- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A Room with the details of the created room.

Return type *Room*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

get(*roomId*)
Get the details of a room, by ID.

Parameters **roomId** (*basestring*) – The ID of the room to be retrieved.

Returns A Room object with the details of the requested room.

Return type *Room*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

update(*roomId*, *title=None*, ***request_parameters*)
Update details for a room, by ID.

Parameters

- **roomId** (*basestring*) – The room ID.
- **title** (*basestring*) – A user-friendly name for the room.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A Room object with the updated Spark room details.

Return type *Room*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

delete(*roomId*)
Delete a room.

Parameters **roomId** (*basestring*) – The ID of the room to be deleted.

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

memberships

class ciscosparkapi.api.memberships.**MembershipsAPI**
Cisco Spark Memberships API.

Wraps the Cisco Spark Memberships API and exposes the API as native Python methods that return native Python objects.

list(*roomId=None*, *personId=None*, *personEmail=None*, *max=None*, ***request_parameters*)

List room memberships.

By default, lists memberships for rooms to which the authenticated user belongs.

Use query parameters to filter the response.

Use *roomId* to list memberships for a room, by ID.

Use either *personId* or *personEmail* to filter the results.

This method supports Cisco Spark's implementation of RFC5988 Web Linking to provide pagination support. It returns a generator container that incrementally yields all memberships returned by the query. The generator will automatically request additional 'pages' of responses from Spark as needed until all responses have been returned. The container makes the generator safe for reuse. A new API call will be made, using the same parameters that were specified when the generator was created, every time a new iterator is requested from the container.

Parameters

- **roomId** (*basestring*) – Limit results to a specific room, by ID.
- **personId** (*basestring*) – Limit results to a specific person, by ID.
- **personEmail** (*basestring*) – Limit results to a specific person, by email address.
- **max** (*int*) – Limit the maximum number of items returned from the Spark service per request.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A GeneratorContainer which, when iterated, yields the memberships returned by the Cisco Spark query.

Return type GeneratorContainer

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

create(*roomId*, *personId=None*, *personEmail=None*, *isModerator=False*, ***request_parameters*)

Add someone to a room by Person ID or email address.

Add someone to a room by Person ID or email address; optionally making them a moderator.

Parameters

- **roomId** (*basestring*) – The room ID.
- **personId** (*basestring*) – The ID of the person.
- **personEmail** (*basestring*) – The email address of the person.
- **isModerator** (*bool*) – Set to True to make the person a room moderator.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A Membership object with the details of the created membership.

Return type *Membership*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

get(*membershipId*)
Get details for a membership, by ID.

Parameters **membershipId** (*basestring*) – The membership ID.

Returns A Membership object with the details of the requested membership.

Return type *Membership*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

update(*membershipId*, *isModerator=None*, ***request_parameters*)
Update properties for a membership, by ID.

Parameters

- **membershipId** (*basestring*) – The membership ID.
- **isModerator** (*bool*) – Set to True to make the person a room moderator.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A Membership object with the updated Spark membership details.

Return type *Membership*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

delete(*membershipId*)
Delete a membership, by ID.

Parameters **membershipId** (*basestring*) – The membership ID.

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

messages

class ciscosparkapi.api.messages.**MessagesAPI**
Cisco Spark Messages API.

Wraps the Cisco Spark Messages API and exposes the API as native Python methods that return native Python objects.

list(*roomId*, *mentionedPeople=None*, *before=None*, *beforeMessage=None*, *max=None*, ***request_parameters*)
Lists messages in a room.

Each message will include content attachments if present.

The list API sorts the messages in descending order by creation date.

This method supports Cisco Spark's implementation of RFC5988 Web Linking to provide pagination support. It returns a generator container that incrementally yields all messages returned by the query. The generator will automatically request additional 'pages' of responses from Spark as needed until all responses have been returned. The container makes the generator safe for reuse. A new API call will be made, using the same parameters that were specified when the generator was created, every time a new iterator is requested from the container.

Parameters

- **roomId** (*basestring*) – List messages for a room, by ID.
- **mentionedPeople** (*basestring*) – List messages where the caller is mentioned by specifying "me" or the caller *personId*.
- **before** (*basestring*) – List messages sent before a date and time, in ISO8601 format.
- **beforeMessage** (*basestring*) – List messages sent before a message, by ID.
- **max** (*int*) – Limit the maximum number of items returned from the Spark service per request.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A GeneratorContainer which, when iterated, yields the messages returned by the Cisco Spark query.

Return type GeneratorContainer

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

create(*roomId=None*, *toPersonId=None*, *toPersonEmail=None*, *text=None*, *markdown=None*, *files=None*, ***request_parameters*)

Post a message, and optionally a attachment, to a room.

The files parameter is a list, which accepts multiple values to allow for future expansion, but currently only one file may be included with the message.

Parameters

- **roomId** (*basestring*) – The room ID.
- **toPersonId** (*basestring*) – The ID of the recipient when sending a private 1:1 message.
- **toPersonEmail** (*basestring*) – The email address of the recipient when sending a private 1:1 message.
- **text** (*basestring*) – The message, in plain text. If *markdown* is specified this parameter may be optionally used to provide alternate text for UI clients that do not support rich text.
- **markdown** (*basestring*) – The message, in markdown format.
- **files** (*list*) – A list of public URL(s) or local path(s) to files to be posted into the room. Only one file is allowed per message. Uploaded files are automatically converted into a format that all Spark clients can render.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A Message object with the details of the created message.

Return type *Message*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.
- `ValueError` – If the files parameter is a list of length > 1, or if the string in the list (the only element in the list) does not contain a valid URL or path to a local file.

get(*messageId*)
Get the details of a message, by ID.

Parameters **messageId** (*basestring*) – The ID of the message to be retrieved.

Returns A Message object with the details of the requested message.

Return type *Message*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

delete(*messageId*)
Delete a message.

Parameters **messageId** (*basestring*) – The ID of the message to be deleted.

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

teams

class ciscosparkapi.api.teams.**TeamsAPI**
Cisco Spark Teams API.

Wraps the Cisco Spark Teams API and exposes the API as native Python methods that return native Python objects.

list(*max=None*, ***request_parameters*)
List teams to which the authenticated user belongs.

This method supports Cisco Spark's implementation of RFC5988 Web Linking to provide pagination support. It returns a generator container that incrementally yields all teams returned by the query. The generator will automatically request additional 'pages' of responses from Spark as needed until all responses have been returned. The container makes the generator safe for reuse. A new API call will be made, using the same parameters that were specified when the generator was created, every time a new iterator is requested from the container.

Parameters

- **max** (*int*) – Limit the maximum number of items returned from the Spark service per request.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A GeneratorContainer which, when iterated, yields the teams returned by the Cisco Spark query.

Return type GeneratorContainer

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

create(*name*, ***request_parameters*)
Create a team.

The authenticated user is automatically added as a member of the team.

Parameters

- **name** (*basestring*) – A user-friendly name for the team.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A Team object with the details of the created team.

Return type *Team*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

get(*teamId*)
Get the details of a team, by ID.

Parameters **teamId** (*basestring*) – The ID of the team to be retrieved.

Returns A Team object with the details of the requested team.

Return type *Team*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

update(*teamId*, *name=None*, ***request_parameters*)
Update details for a team, by ID.

Parameters

- **teamId** (*basestring*) – The team ID.
- **name** (*basestring*) – A user-friendly name for the team.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A Team object with the updated Spark team details.

Return type *Team*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

delete(*teamId*)
Delete a team.

Parameters **teamId** (*basestring*) – The ID of the team to be deleted.

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

team_memberships

class ciscosparkapi.api.team_memberships.**TeamMembershipsAPI**
Cisco Spark Team-Memberships API.

Wraps the Cisco Spark Memberships API and exposes the API as native Python methods that return native Python objects.

list(*teamId*, *max=None*, ***request_parameters*)
List team memberships for a team, by ID.

This method supports Cisco Spark's implementation of RFC5988 Web Linking to provide pagination support. It returns a generator container that incrementally yields all team memberships returned by the query. The generator will automatically request additional 'pages' of responses from Spark as needed until all responses have been returned. The container makes the generator safe for reuse. A new API call will be made, using the same parameters that were specified when the generator was created, every time a new iterator is requested from the container.

Parameters

- **teamId** (*basestring*) – List team memberships for a team, by ID.
- **max** (*int*) – Limit the maximum number of items returned from the Spark service per request.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A GeneratorContainer which, when iterated, yields the team memberships returned by the Cisco Spark query.

Return type GeneratorContainer

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

create(*teamId*, *personId=None*, *personEmail=None*, *isModerator=False*, ***request_parameters*)
Add someone to a team by Person ID or email address.

Add someone to a team by Person ID or email address; optionally making them a moderator.

Parameters

- **teamId** (*basestring*) – The team ID.
- **personId** (*basestring*) – The person ID.
- **personEmail** (*basestring*) – The email address of the person.
- **isModerator** (*bool*) – Set to True to make the person a team moderator.

- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A TeamMembership object with the details of the created team membership.

Return type *TeamMembership*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

get(*membershipId*)
Get details for a team membership, by ID.

Parameters **membershipId** (*basestring*) – The team membership ID.

Returns A TeamMembership object with the details of the requested team membership.

Return type *TeamMembership*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

update(*membershipId*, *isModerator=None*, ***request_parameters*)
Update a team membership, by ID.

Parameters

- **membershipId** (*basestring*) – The team membership ID.
- **isModerator** (*bool*) – Set to True to make the person a team moderator.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A TeamMembership object with the updated Spark team membership details.

Return type *TeamMembership*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

delete(*membershipId*)
Delete a team membership, by ID.

Parameters **membershipId** (*basestring*) – The team membership ID.

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

webhooks

class ciscosparkapi.api.webhooks.**WebhooksAPI**
Cisco Spark Webhooks API.

Wraps the Cisco Spark Webhooks API and exposes the API as native Python methods that return native Python objects.

list(*max=None*, ***request_parameters*)

List all of the authenticated user's webhooks.

This method supports Cisco Spark's implementation of RFC5988 Web Linking to provide pagination support. It returns a generator container that incrementally yields all webhooks returned by the query. The generator will automatically request additional 'pages' of responses from Spark as needed until all responses have been returned. The container makes the generator safe for reuse. A new API call will be made, using the same parameters that were specified when the generator was created, every time a new iterator is requested from the container.

Parameters

- **max** (*int*) – Limit the maximum number of items returned from the Spark service per request.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A GeneratorContainer which, when iterated, yields the webhooks returned by the Cisco Spark query.

Return type GeneratorContainer

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

create(*name*, *targetUrl*, *resource*, *event*, *filter=None*, *secret=None*, ***request_parameters*)

Create a webhook.

Parameters

- **name** (*basestring*) – A user-friendly name for this webhook.
- **targetUrl** (*basestring*) – The URL that receives POST requests for each event.
- **resource** (*basestring*) – The resource type for the webhook.
- **event** (*basestring*) – The event type for the webhook.
- **filter** (*basestring*) – The filter that defines the webhook scope.
- **secret** (*basestring*) – The secret used to generate payload signature.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A Webhook object with the details of the created webhook.

Return type *Webhook*

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

get(*webhookId*)

Get the details of a webhook, by ID.

Parameters **webhookId** (*basestring*) – The ID of the webhook to be retrieved.

Returns A Webhook object with the details of the requested webhook.

Return type *Webhook*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

update(*webhookId*, *name=None*, *targetUrl=None*, ***request_parameters*)
Update a webhook, by ID.

Parameters

- **webhookId** (*basestring*) – The webhook ID.
- **name** (*basestring*) – A user-friendly name for this webhook.
- **targetUrl** (*basestring*) – The URL that receives POST requests for each event.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A Webhook object with the updated Spark webhook details.

Return type *Webhook*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

delete(*webhookId*)
Delete a webhook, by ID.

Parameters **webhookId** (*basestring*) – The ID of the webhook to be deleted.

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

organizations

class ciscosparkapi.api.organizations.**OrganizationsAPI**
Cisco Spark Organizations API.

Wraps the Cisco Spark Organizations API and exposes the API as native Python methods that return native Python objects.

list(*max=None*, ***request_parameters*)
List Organizations.

This method supports Cisco Spark's implementation of RFC5988 Web Linking to provide pagination support. It returns a generator container that incrementally yields all objects returned by the query. The generator will automatically request additional 'pages' of responses from Spark as needed until all responses have been returned. The container makes the generator safe for reuse. A new API call will be made, using the same parameters that were specified when the generator was created, every time a new iterator is requested from the container.

Parameters

- **max** (*int*) – Limit the maximum number of items returned from the Spark service per request.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A GeneratorContainer which, when iterated, yields the organizations returned by the Cisco Spark query.

Return type GeneratorContainer

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

get(*orgId*)

Get the details of an Organization, by ID.

Parameters **orgId** (*basestring*) – The ID of the Organization to be retrieved.

Returns An Organization object with the details of the requested organization.

Return type *Organization*

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

licenses

class ciscosparkapi.api.licenses.**LicensesAPI**

Cisco Spark Licenses API.

Wraps the Cisco Spark Licenses API and exposes the API as native Python methods that return native Python objects.

list(*orgId=None*, *max=None*, ***request_parameters*)

List all licenses for a given organization.

If no orgId is specified, the default is the organization of the authenticated user.

This method supports Cisco Spark's implementation of RFC5988 Web Linking to provide pagination support. It returns a generator container that incrementally yields all objects returned by the query. The generator will automatically request additional 'pages' of responses from Spark as needed until all responses have been returned. The container makes the generator safe for reuse. A new API call will be made, using the same parameters that were specified when the generator was created, every time a new iterator is requested from the container.

Parameters

- **orgId** (*basestring*) – Specify the organization, by ID.
- **max** (*int*) – Limit the maximum number of items returned from the Spark service per request.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A GeneratorContainer which, when iterated, yields the licenses returned by the Cisco Spark query.

Return type GeneratorContainer

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

get(*licenseId*)
Get the details of a License, by ID.

Parameters **licenseId** (*basestring*) – The ID of the License to be retrieved.

Returns A License object with the details of the requested License.

Return type *License*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

roles

class ciscosparkapi.api.roles.**RolesAPI**
Cisco Spark Roles API.

Wraps the Cisco Spark Roles API and exposes the API as native Python methods that return native Python objects.

list(*max=None*, ***request_parameters*)
List all roles.

This method supports Cisco Spark's implementation of RFC5988 Web Linking to provide pagination support. It returns a generator container that incrementally yields all objects returned by the query. The generator will automatically request additional 'pages' of responses from Spark as needed until all responses have been returned. The container makes the generator safe for reuse. A new API call will be made, using the same parameters that were specified when the generator was created, every time a new iterator is requested from the container.

Parameters

- **max** (*int*) – Limit the maximum number of items returned from the Spark service per request.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A GeneratorContainer which, when iterated, yields the roles returned by the Cisco Spark query.

Return type GeneratorContainer

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

get(*roleId*)
Get the details of a Role, by ID.

Parameters **roleId** (*basestring*) – The ID of the Role to be retrieved.

Returns A Role object with the details of the requested Role.

Return type *Role*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

events

class ciscosparkapi.api.events.**EventsAPI**

Cisco Spark Events API.

Wraps the Cisco Spark Events API and exposes the API as native Python methods that return native Python objects.

list(*resource=None*, *type=None*, *actorId=None*, *_from=None*, *to=None*, *max=None*, ***request_parameters*)

List events.

List events in your organization. Several query parameters are available to filter the response.

Note: *from* is a keyword in Python and may not be used as a variable name, so we had to use *_from* instead.

This method supports Cisco Spark's implementation of RFC5988 Web Linking to provide pagination support. It returns a generator container that incrementally yields all events returned by the query. The generator will automatically request additional 'pages' of responses from Spark as needed until all responses have been returned. The container makes the generator safe for reuse. A new API call will be made, using the same parameters that were specified when the generator was created, every time a new iterator is requested from the container.

Parameters

- **resource** (*basestring*) – Limit results to a specific resource type. Possible values: "messages", "memberships".
- **type** (*basestring*) – Limit results to a specific event type. Possible values: "created", "updated", "deleted".
- **actorId** (*basestring*) – Limit results to events performed by this person, by ID.
- **_from** (*basestring*) – Limit results to events which occurred after a date and time, in ISO8601 format (yyyy-MM-dd'T'HH:mm:ss.SSSZ).
- **to** (*basestring*) – Limit results to events which occurred before a date and time, in ISO8601 format (yyyy-MM-dd'T'HH:mm:ss.SSSZ).
- **max** (*int*) – Limit the maximum number of items returned from the Spark service per request.
- ****request_parameters** – Additional request parameters (provides support for parameters that may be added in the future).

Returns A GeneratorContainer which, when iterated, yields the events returned by the Cisco Spark query.

Return type GeneratorContainer

Raises

- `TypeError` – If the parameter types are incorrect.

- SparkApiError – If the Cisco Spark cloud returns an error.

get(*eventId*)
Get the details for an event, by event ID.

Parameters **eventId** (*basestring*) – The ID of the event to be retrieved.

Returns A event object with the details of the requested room.

Return type *Event*

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

access_tokens

class ciscosparkapi.api.access_tokens.**AccessTokensAPI**
Cisco Spark Access-Tokens API.

Wraps the Cisco Spark Access-Tokens API and exposes the API as native Python methods that return native Python objects.

base_url
The base URL the API endpoints.

timeout
Timeout in seconds for the API requests.

get(*client_id*, *client_secret*, *code*, *redirect_uri*)
Exchange an Authorization Code for an Access Token.

Exchange an Authorization Code for an Access Token that can be used to invoke the APIs.

Parameters

- **client_id** (*basestring*) – Provided when you created your integration.
- **client_secret** (*basestring*) – Provided when you created your integration.
- **code** (*basestring*) – The Authorization Code provided by the user OAuth process.
- **redirect_uri** (*basestring*) – The redirect URI used in the user OAuth process.

Returns An AccessToken object with the access token provided by the Cisco Spark cloud.

Return type *ciscosparkapi.AccessToken*

Raises

- TypeError – If the parameter types are incorrect.
- SparkApiError – If the Cisco Spark cloud returns an error.

refresh(*client_id*, *client_secret*, *refresh_token*)
Return a refreshed Access Token from the provided refresh_token.

Parameters

- **client_id** (*basestring*) – Provided when you created your integration.
- **client_secret** (*basestring*) – Provided when you created your integration.
- **refresh_token** (*basestring*) – Provided when you requested the Access Token.

Returns With the access token provided by the Cisco Spark cloud.

Return type *AccessToken*

Raises

- `TypeError` – If the parameter types are incorrect.
- `SparkApiError` – If the Cisco Spark cloud returns an error.

1.4.2 Exceptions

exception ciscosparkapi.**ciscosparkapiException**
Bases: `Exception`

Base class for all ciscosparkapi package exceptions.

exception ciscosparkapi.**SparkApiError**
Bases: `ciscosparkapi.exceptions.ciscosparkapiException`

Errors returned by requests to the Cisco Spark cloud APIs.

request = None
The `requests.PreparedRequest` of the API call.

response = None
The `requests.Response` object returned from the API call.

exception ciscosparkapi.**SparkRateLimitError**
Bases: `ciscosparkapi.exceptions.SparkApiError`

Cisco Spark Rate-Limit exceeded Error.

retry_after = None
The *Retry-After* time period (in seconds) provided by Cisco Spark.

Defaults to 15 seconds if the response *Retry-After* header isn't present in the response headers, and defaults to a minimum wait time of 1 second if Spark returns a *Retry-After* header of 0 seconds.

1.4.3 Spark Data Objects

Person

class ciscosparkapi.**Person**
Cisco Spark Person data model.

avatar
URL to the person's avatar in PNG format.

created
The date and time the person was created.

displayName
Full name of the person.

emails
Email address(es) of the person.

firstName
First name of the person.

id
The person's unique ID.

invitePending
Person has been sent an invite, but hasn't responded.

json_data
A copy of the Spark data object's JSON data (OrderedDict).

lastActivity
The date and time of the person's last activity.

lastName
Last name of the person.

licenses
Licenses allocated to the person.

loginEnabled
Person is allowed to login.

nickName
'Nick name' or preferred short name of the person.

orgId
ID of the organization to which this person belongs.

roles
Roles of the person.

status
The person's current status.

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

type
The type of object returned by Cisco Spark (should be *person*).

Room

class ciscosparkapi.**Room**
Cisco Spark Room data model.

created
The date and time when the room was created.

creatorId
The ID of the person who created the room.

id
The rooms's unique ID.

isLocked
Whether or not the room is locked and controlled by moderator(s).

json_data
A copy of the Spark data object's JSON data (OrderedDict).

lastActivity
The date and time when the room was last active.

teamId
The ID for the team with which this room is associated.

title
A user-friendly name for the room.

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

type
The type of room (i.e. 'group', 'direct' etc.).

Membership

class ciscosparkapi.**Membership**
Cisco Spark Membership data model.

created
The date and time the membership was created.

id
The membership's unique ID.

isModerator
Person is a moderator for the room.

isMonitor
Person is a monitor for the room.

json_data
A copy of the Spark data object's JSON data (OrderedDict).

personDisplayName
The display name of the person.

personEmail
The email address of the person.

personId
The ID of the person.

personOrgId
The ID of the organization that the person is associated with.

roomId
The ID of the room.

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

Message

class ciscosparkapi.**Message**
Cisco Spark Message data model.

created
The date and time the message was created.

files
Files attached to the the message (list of URLs).

html
The message, in HTML format.

id
The message's unique ID.

json_data
A copy of the Spark data object's JSON data (OrderedDict).

markdown
The message, in markdown format.

mentionedPeople
The list of IDs of people mentioned in the message.

personEmail
The email address of the sender.

personId
The person ID of the sender.

roomId
The ID of the room.

roomType
The type of room (i.e. 'group', 'direct' etc.).

text
The message, in plain text.

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

Team

class ciscosparkapi.**Team**
Cisco Spark Team data model.

created
The date and time the team was created.

creatorId
The ID of the person who created the team.

id
The team's unique ID.

json_data
A copy of the Spark data object's JSON data (OrderedDict).

name
A user-friendly name for the team.

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

Team Membership

class ciscosparkapi.**TeamMembership**
Cisco Spark Team-Membership data model.

created
The date and time the team membership was created.

id
The team membership's unique ID.

isModerator
Person is a moderator for the team.

json_data
A copy of the Spark data object's JSON data (OrderedDict).

personDisplayName
The display name of the person.

personEmail
The email address of the person.

personId
The ID of the person.

personOrgId
The ID of the organization that the person is associated with.

teamId
The ID of the team.

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

Webhook

class ciscosparkapi.**Webhook**
Cisco Spark Webhook data model.

appId
Identifies the application that added the webhook.

created
Creation date and time in ISO8601 format.

createdBy
The ID of the person that added the webhook.

event
The event type for the webhook.

filter
The filter that defines the webhook scope.

id
Webhook ID.

json_data
A copy of the Spark data object's JSON data (OrderedDict).

name
A user-friendly name for this webhook.

orgId
The ID of the organization that owns the webhook.

ownedBy
Indicates if the webhook is owned by the *org* or the *creator*.

Webhooks owned by the creator can only receive events that are accessible to the creator of the webhook. Those owned by the organization will receive events that are visible to anyone in the organization.

resource
The resource type for the webhook.

secret
Secret used to generate payload signature.

status
Indicates if the webhook is active.

A webhook that cannot reach your URL is disabled.

targetUrl
The URL that receives POST requests for each event.

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

Webhook Event

class ciscosparkapi.**WebhookEvent**
Cisco Spark Webhook-Events data model.

actorId
The ID of the person that caused the webhook to be sent.

appId
Identifies the application that added the webhook.

createdBy
The ID of the person that added the webhook.

data
The data for the resource that triggered the webhook.

event
The event type for the webhook.

filter
The filter that defines the webhook scope.

id
Webhook ID.

json_data
A copy of the Spark data object's JSON data (OrderedDict).

name
A user-friendly name for this webhook.

orgId
The ID of the organization that owns the webhook.

ownedBy
Indicates if the webhook is owned by the *org* or the *creator*.

Webhooks owned by the creator can only receive events that are accessible to the creator of the webhook. Those owned by the organization will receive events that are visible to anyone in the organization.

resource
The resource type for the webhook.

status
Indicates if the webhook is active.

A webhook that cannot reach your URL is disabled.

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

Organization

class ciscosparkapi.**Organization**
Cisco Spark Organization data model.

created
Creation date and time in ISO8601 format.

displayName
The human-friendly display name of the Organization.

id
The unique ID for the Organization.

json_data
A copy of the Spark data object's JSON data (OrderedDict).

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

License

class ciscosparkapi.**License**
Cisco Spark License data model.

consumedUnits
The total number of license units consumed.

id
The unique ID for the License.

json_data
A copy of the Spark data object's JSON data (OrderedDict).

name
The name of the License.

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

totalUnits
The total number of license units.

Role

class ciscosparkapi.**Role**
Cisco Spark Role data model.

id
The unique ID for the Role.

json_data
A copy of the Spark data object's JSON data (OrderedDict).

name
The name of the Role.

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

Event

class ciscosparkapi.**Event**
Cisco Spark Event data model.

actorId
The ID of the person that performed this event.

created
The date and time the event was performed.

data
The event resource data.

id
Event ID.

json_data
A copy of the Spark data object's JSON data (OrderedDict).

resource
The event resource type (*messagess*, *memberships*).

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

type
The event type (*created*, *updated*, *deleted*).

Access Token

class ciscosparkapi.**AccessToken**
Cisco Spark Access-Token data model.

access_token
Cisco Spark access token.

expires_in
Access token expiry time (in seconds).

json_data
A copy of the Spark data object's JSON data (OrderedDict).

refresh_token
Refresh token used to request a new/refreshed access token.

refresh_token_expires_in
Refresh token expiry time (in seconds).

to_dict()
Convert the Spark data to a dictionary.

to_json(***kwargs*)
Convert the Spark data to JSON.

Any keyword arguments provided are passed through the Python JSON encoder.

CHAPTER 2

The Community Guide

Community developer docs are *coming soon*. For now, please see the contribution instructions on the ciscosparkapi GitHub page to get started.

CHAPTER 3

General Information about the Cisco Spark Service

3.1 What is Cisco Spark?

> "Cisco Spark is where all your work lives. Bring your teams together in a place that makes it easy to keep people and work connected."

Check out the official Cisco Spark website for more information and to create a free account!

3.2 Spark for Developers

Leveraging the Cisco Spark APIs and developing on top of the Cisco Spark cloud is easy. Signup for a free account and then head over to the Spark for Developers website to learn more.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

W