Contents

1 Installation 3
2 Authorization 5
3 Resources 7
4 API reference 15
5 Using in Django 23
6 Additional documentation 25
7 Contribute 27
Python Module Index 29
The super-mega-official Python package for using the SparkPost API.
CHAPTER 1

Installation

Install from PyPI using pip:

$ pip install sparkpost
Go to API & SMTP in the SparkPost app and create an API key. We recommend using the `SPARKPOST_API_KEY` environment variable:

```python
from sparkpost import SparkPost
sp = SparkPost()  # uses environment variable
```

Alternatively, you can pass the API key to the SparkPost class:

```python
from sparkpost import SparkPost
sp = SparkPost('YOUR API KEY')
```
The following resources are available in python-sparkpost:

### 3.1 Metrics

#### 3.1.1 Retrieve a list of campaigns

```python
from sparkpost import SparkPost
sp = SparkPost()
sp.metrics.campaigns.list()
```

#### 3.1.2 Retrieve a list of domains

```python
from sparkpost import SparkPost
sp = SparkPost()
sp.metrics.domains.list()
```

#### 3.1.3 Additional documentation

See the [SparkPost Metrics API Reference](#).
3.2 Recipient Lists

Let’s use the underlying recipient_lists API to create a recipient list:

```python
from sparkpost import SparkPost

sp = SparkPost()

response = sp.recipient_lists.create(
    id='UNIQUE_TEST_ID',
    name='Test Recipient list',
    recipients=[
        {'address': {'email': 'test1@test.com'}}
    ]
)

print(response)
# outputs {u'total_accepted_recipients': 3, u'id': u'UNIQUE_TEST_ID', u'total_rejected_recipients': 0, u'name':'Test Recipient list'}
```

3.2.1 Retrieve a recipient list

```python
from sparkpost import SparkPost

sp = SparkPost()

sp.recipient_lists.get('my-list-id')
```

3.2.2 List all recipient lists

```python
from sparkpost import SparkPost

sp = SparkPost()

sp.recipient_lists.list()
```
### 3.2.3 API reference

`sparkpost.recipient_lists`

### 3.2.4 Further examples

See the `python-sparkpost recipient_lists` examples.

### 3.2.5 Additional documentation

See the SparkPost Recipient Lists API Reference.

### 3.3 Suppression List

Let’s use the underlying `suppression_list` API to create a suppression entry:

```python
from sparkpost import SparkPost

sp = SparkPost()

response = sp.suppression_list.create({
  "email": "test@test.com",
  "transactional": False,
  "non_transactional": True,
  "description": "User requested to not receive any non-transactional emails."
})

print(response)
# outputs {u'message': u'Recipient successfully created'}
```

#### 3.3.1 Get a suppression entry

```python
from sparkpost import SparkPost

sp = SparkPost()

sp.suppression_list.get('test@test.com')
```

#### 3.3.2 List suppression entries

```python
from sparkpost import SparkPost

sp = SparkPost()

sp.suppression_list.list()
```
3.3.3 API reference

`sparkpost.suppression_list`

3.3.4 Further examples

See the `python-sparkpost suppression_list` examples.

3.3.5 Additional documentation

See the SparkPost Suppression List API Reference.

3.4 Templates

Let’s use the underlying templates API to create a template:

```python
from sparkpost import SparkPost
sp = SparkPost()
response = sp.templates.create(
    id='TEST_ID',
    name='Test Template',
    from_email='test@test.com',
    subject='Test email template!',
    html='<b>This is a test email template!</b>
)
print(response)
# outputs {u'id': u'TEST_ID'}
```

3.4.1 Retrieve a template

```python
from sparkpost import SparkPost
sp = SparkPost()
sp.templates.get('my-template-id')
```

3.4.2 List all templates

```python
from sparkpost import SparkPost
sp = SparkPost()
sp.templates.list()
```
3.4.3 API reference

`sparkpost.templates`

3.4.4 Further examples

See the python-sparkpost templates examples.

3.4.5 Additional documentation

See the SparkPost Templates API Reference.

3.5 Transmissions

Here at SparkPost, our messages are known as transmissions. Let’s use the underlying transmissions API to send a friendly test message:

```python
from sparkpost import SparkPost
sp = SparkPost()
response = sp.transmissions.send(
    recipients=['someone@somedomain.com'],
    html='<p>Hello world</p>',
    from_email='test@sparkpostbox.com',
    subject='Hello from python-sparkpost',
    track_opens=True,
    track_clicks=True
)
print(response)
# outputs {u'total_accepted_recipients': 1, u'id': u'47960765679942446', u'total_rejected_recipients': 0}
```

3.5.1 Send a transmission

Using inline templates and/or recipients

```python
from sparkpost import SparkPost
sp = SparkPost()
sp.transmissions.send(
    recipients=['someone@somedomain.com'],
    text="Hello world",
    html='Hello world',
    from_email='test@sparkpostbox.com',
    subject='Hello from python-sparkpost',
    track_opens=True,
    track_clicks=True
)
```
Including cc, bcc

```python
from sparkpost import SparkPost
sp = SparkPost()
sp.transmissions.send(
    recipients=['someone@somedomain.com'],
    cc=['carboncopy@somedomain.com'],
    bcc=['blindcarboncopy@somedomain.com'],
    text="Hello world",
    html='<p>Hello world</p>',
    from_email='test@sparkpostbox.com',
    subject='Hello from python-sparkpost',
    track_opens=True,
    track_clicks=True
)
```

Sending an attachment

```python
from sparkpost import SparkPost
sp = SparkPost()
sp.transmissions.send(
    recipients=['someone@somedomain.com'],
    text="Hello world",
    html='<p>Hello world</p>',
    from_email='test@sparkpostbox.com',
    subject='Hello from python-sparkpost',
    track_opens=True,
    track_clicks=True,
    attachments=[
        {
            "name": "test.txt",
            "type": "text/plain",
            "filename": "/home/sparkpost/a-file.txt"
        }
    ]
)
```

Using substitution data

**Note:** Substitution data can be specified at the template, transmission and recipient levels. The order of precedence is as follows: recipient overrides transmission overrides template.

```python
from sparkpost import SparkPost
sp = SparkPost()
sp.transmissions.send(
    recipients=['someone@somedomain.com'],
```
Using a stored template

```python
from sparkpost import SparkPost

sp = SparkPost()

sp.transmissions.send(
    recipients=['someone@somedomain.com'],
    template='my-template-id'
)
```

Using a stored recipient list

```python
from sparkpost import SparkPost

sp = SparkPost()

sp.transmissions.send(
    recipient_list='my-recipient-list',
    template='my-template-id'
)
```

3.5.2 Retrieve a transmission

```python
from sparkpost import SparkPost

sp = SparkPost()

sp.transmissions.get('my-transmission-id')
```

3.5.3 List all transmissions

```python
from sparkpost import SparkPost

sp = SparkPost()

sp.transmissions.list()
```
3.5.4 API reference

`sparkpost.transmissions`

3.5.5 Further examples

See the `python-sparkpost transmissions examples`.

3.5.6 Additional documentation

See the SparkPost Transmissions API Reference.
Auto-generated API reference for python-sparkpost:

## 4.1 API Documentation

A complete API reference to the `sparkpost` module.

### 4.1.1 `sparkpost.recipient_lists`

```python
class sparkpost.recipient_lists.RecipientLists(base_uri, api_key, transport_class=class 'sparkpost.base.RequestsTransport')
```

RecipientLists class used to create, update, delete, list and get recipient lists. For detailed request and response formats, see the Recipient Lists API documentation.

#### create (**kwargs)
Create a recipient list based on the supplied parameters

**Parameters**

- **id (str)** – ID used to reference the recipient list
- **name (str)** – Editable display name
- **description (str)** – Detailed description of the recipient list
- **attributes (dict)** – Arbitrary metadata related to the list
- **recipients (list)** – Array of recipient dicts

**Returns** a dict with the ID, name, and number of accepted and rejected recipients

**Raises** `SparkPostAPIException` if API call fails

#### delete (list_id)
Delete a recipient list by ID
Parameters list_id (str) – ID of the recipient list you want to delete

Returns empty dict

Raises SparkPostAPIException if recipient list is not found or if recipient list is in use

get(list_id, show_recipients=None)
Get a recipient list by ID

Parameters
  • list_id (str) – ID of the recipient list you want to retrieve
  • show_recipients (bool) – If True, returns attributes for all recipients

Returns the requested recipient list if found

Raises SparkPostAPIException if recipient list is not found

get()
Get a list of your recipient lists

Returns list of recipient lists

Raises SparkPostAPIException if API call fails

update(list_id, **kwargs)
Update a recipient list by ID based on the supplied parameters

Parameters
  • list_id (str) – ID of the recipient list you want to update
  • name (str) – Editable display name
  • description (str) – Detailed description of the recipient list
  • attributes (dict) – Arbitrary metadata related to the list
  • recipients (list) – Array of recipient dicts

Returns a dict with the ID, name, and number of accepted and rejected recipients

Raises SparkPostAPIException if API call fails

4.1.2 sparkpost.suppression_list

class sparkpost.suppression_list.SuppressionList(base_uri, api_key, transport_class=<class 'sparkpost.base.RequestsTransport'>)

SuppressionList class used to search, get and modify suppression status. For detailed request and response formats, see the Suppression List API documentation.

create(entry)
Create a suppression list entry.

Parameters status (dict/list) – If dict it is a single entry to create
  
  { 'email': 'test@test.com', 'transactional': True, 'non_transactional': True, 'description': 'Test description' }, if list it is multiple entries to create

Returns a dict with a message

Raises SparkPostAPIException if API call fails
**delete** *(email)*
Delete the suppression status for a specific recipient by email

**Parameters**
- **email** *(str)* – Email of the recipient whose status you want to remove

**Returns**
- TODO

**Raises**
- SparkPostAPIException if API call fails

**get** *(email)*
Retrieve a suppression list entry for a specific recipient by email

**Parameters**
- **email** *(str)* – Email of the recipient whose status you want to check_status

**Returns**
- a suppression list entry

**Raises**
- SparkPostAPIException if API call fails

**list**(**kwargs**)
List supression list entries based on the supplied parameters

**Parameters**
- **from_date** *(datetime)* – DateTime to start listing
- **to_date** *(datetime)* – DateTime to end listing
- **types** *(list)* – Types of entries to return
- **limit** *(int)* – Maximum number of entries to return

**Returns**
- a list of entries

**Raises**
- SparkPostAPIException if API call fails

**update** *(entry)*
Update a suppression list entry.

**Parameters**
- **status** *(dict/list)* – If dict it is a single entry to update { 'email': 'test@test.com', 'transactional': True, 'non_transactional': True, 'description': 'Test description' }, if list it is multiple entries to update

**Returns**
- a dict with a message

**Raises**
- SparkPostAPIException if API call fails

### 4.1.3 sparkpost.templates

**class** sparkpost.templates.Templates*(base_uri, api_key, transport_class=<class 'sparkpost.base.RequestsTransport'>)*
Templates class used to create, update, delete, list and get templates. For detailed request and response formats, see the Templates API documentation.

**create**(**kwargs**)
Create a template based on the supplied parameters

**Parameters**
- **id** *(str)* – ID used to reference the template
- **name** *(str)* – Editable display name
- **description** *(str)* – Detailed description of the template
• **published** (*bool*) – Defaults to False. Whether the template is a published or draft version

• **track_opens** (*bool*) – Defaults to transmission level setting. Used to track opens of transmission

• **track_clicks** (*bool*) – Defaults to transmission level setting. Used to track clicks of transmission

• **is_transactional** (*bool*) – Defaults to transmission level setting. Distinguishes between transactional and non-transactional messages for unsubscribe and suppression purposes

• **html** (*str*) – HTML part of template

• **text** (*str*) – Text part of template

• **subject** (*str*) – Subject of template

• **from_email** (*str*) – Friendly from of template, domain must be a verified sending domain to your account or template create will fail

• **reply_to** (*str*) – Reply to of template

• **custom_headers** (*dict*) – Used to set any headers associated with template

Returns a dict with the ID

Raises SparkPostAPIException if template uses an unverified sending domain or there’s a syntax error in the content

delete(*template_id*)
Delete a template by ID

Parameters **template_id** (*str*) – ID of the template you want to delete

Returns TODO

Raises SparkPostAPIException if template is not found or if template is in use

get(*template_id*, draft=None)
Get a template by ID

Parameters

• **template_id** (*str*) – ID of the template you want to retrieve

• **draft** (*bool*) – Defaults to None. If True, returns the most recent draft template. If False, returns the most recent published template. If None, returns the most recent template version regardless of draft or published.

Returns the requested template if found

Raises SparkPostAPIException if template is not found

list()
Get a list of your templates

Returns list of templates

Raises SparkPostAPIException if API call fails

preview(*template_id*, substitution_data, draft=None)
Get a preview of your template by ID with the provided substitution_data

Parameters
• **template_id**(str) – ID of the template you want to retrieve

• **substitution_data**(dict) – data to be substituted in the template content

• **draft**(bool) – Defaults to None. If True, previews the most recent draft template. If False, previews the most recent published template. If None, previews the most recent template version regardless of draft or published.

Returns the requested template if found with content expanded using substitution data provided

Raises SparkPostAPIException if API call fails

**update**(template_id, **kwargs)

Update a template by ID based on the supplied parameters

Parameters

• **template_id**(str) – ID of the template you want to retrieve

• **name**(str) – Editable display name

• **description**(str) – Detailed description of the template

• **published**(bool) – Defaults to False. Whether the template is a published or draft version

• **track_opens**(bool) – Defaults to transmission level setting. Used to track opens of transmission

• **track_clicks**(bool) – Defaults to transmission level setting. Used to track clicks of transmission

• **is_transactional**(bool) – Defaults to transmission level setting. Distinguishes between transactional and non-transactional messages for unsubscribe and suppression purposes

• **html**(str) – HTML part of template

• **text**(str) – Text part of template

• **subject**(str) – Subject of template

• **from_email**(str) – Friendly from of template, domain must be a verified sending domain to your account or template create will fail

• **reply_to**(str) – Reply to of template

• **custom_headers**(dict) – Used to set any headers associated with template

Returns TODO

Raises SparkPostAPIException if template is not found

### 4.1.4 sparkpost.transmissions

class sparkpost.transmissions.Transmissions(base_uri, api_key, transport_class=class 'sparkpost.base.RequestsTransport'>)

Transmission class used to send, list and get transmissions. For detailed request and response formats, see the Transmissions API documentation.

**delete**(transmission_id)

Delete a transmission by ID

Parameters **transmission_id**(str) – ID of the transmission you want to delete
Returns {} if transmission is deleted

Raises SparkPostAPIException if transmission is not found or Canceled

get (transmission_id)
Get a transmission by ID

Parameters transmission_id (str) – ID of the transmission you want to retrieve

Returns the requested transmission if found

Raises SparkPostAPIException if transmission is not found

list (**kwargs)
Get a list of your transmissions

Parameters

• campaign_id – ID of the campaign used by the transmissions
• template_id – ID of the template used by the transmissions

Returns list of transmissions

Raises SparkPostAPIException if API call fails

send (**kwargs)
Send a transmission based on the supplied parameters

Parameters

• recipients (list|dict) – If list it is an list of email addresses, if dict
  {'address': {'name': 'Name', 'email': 'me' }}
• recipient_list (str) – ID of recipient list, if set recipients above will be ignored
• cc – List of email addresses to send carbon copy to
• bcc – List of email addresses to send blind carbon copy to
• template (str) – ID of template. If set HTML or text will not be used
• use_draft_template (bool) – Default to False. Set to true if you want to send a
  template that is a draft
• html (str) – HTML part of transmission
• text (str) – Text part of transmission
• subject (str) – Subject of transmission
• from_email (str) – Email that the transmission comes from. The domain must be
  a verified sending domain to your account or the transmission will fail. You can pass
  a from email or both from name and from email - testing@example.com or Test Email
  <testing@example.com> will both work.
• reply_to (str) – Reply to of transmission
• description (str) – Description of transmission
• campaign (str) – Campaign of transmission
• metadata (dict) – Any data you want to send along with transmission, used in Web-
  Hooks
• substitution_data (dict) – Corresponds to substitutions in html/text content. See
  substitutions reference.
• **attachments** – List of dicts. For example:

```python
dict(
    type='application/pdf',
    name='document.pdf',
    data='base64 encoded string'
)
```

Replace `data` with `filename` if you want the library to perform the base64 conversion. For example:

```python
dict(
    type='application/pdf',
    name='document.pdf',
    filename='/full/path/to/document.pdf'
)
```

• **inline_images** – List of dicts. For example:

```python
dict(
    type='image/png',
    name='imageCID',
    data='base64 encoded string'
)
```

Replace `data` with `filename` if you want the library to perform the base64 conversion. For example:

```python
dict(
    type='image/png',
    name='imageCID',
    filename='/full/path/to/image.png'
)
```

• **start_time** (*str*) – Delay generation of messages until this datetime. Format YYYY-MM-DDTHH:MM:SS+-HH:MM. Example: ‘2015-02-11T08:00:00-04:00’.

• **track_opens** (*bool*) – Defaults to True. Used to track opens of transmission

• **track_clicks** (*bool*) – Defaults to True. Used to track clicks of transmission

• **use_sandbox** (*bool*) – Flag must be set to use sandbox domain instead of verified sending domain. Limited to a lifetime of 50 transmissions with this domain

• **transactional** (*bool*) – Whether message is transactional or non-transactional for unsubscribe and suppression purposes

• **skip_suppression** (*bool*) – Whether or not to ignore customer suppression rules, for this transmission only. Only applicable if your configuration supports this parameter. (SparkPost Elite only)

• **ip_pool** (*str*) – The name of a dedicated IP pool associated with your account

• **inline_css** (*bool*) – Whether or not to perform CSS inlining

• **custom_headers** (*dict*) – Used to set any headers associated with transmission

Returns a dict with the ID and number of accepted and rejected recipients

Raises SparkPostAPIException if transmission cannot be sent
Configure Django to use SparkPost email backend

## 5.1 Django Email Backend

The SparkPost python library comes with an email backend for Django.

### 5.1.1 Configure Django

To configure Django to use SparkPost, put the following configuration in `settings.py` file.

```python
SPARKPOST_API_KEY = 'API_KEY'
EMAIL_BACKEND = 'sparkpost.django.email_backend.SparkPostEmailBackend'
```

Replace `API_KEY` with an actual API key.

You can also use `SPARKPOST_OPTIONS` to set options that will apply to every transmission. For example:

```python
SPARKPOST_OPTIONS = {  
    'track_opens': False,  
    'track_clicks': False,  
    'transactional': True,  
}
```

### 5.1.2 Sending an email

Django is now configured to use the SparkPost email backend. You can now send mail using Django’s `send_mail` method:
```python
from django.core.mail import send_mail

send_mail(
    subject='Hello from SparkPost',
    message='Woo hoo! Sent from Django!',
    from_email='from@yourdomain.com',
    recipient_list=['to@example.com'],
    html_message='<p>Hello Rock stars!</p>',
)
```

You can also use `EmailMessage` or `EmailMultiAlternatives` class directly. That will give you access to more specific fields like `template`:

```python
e-mail = EmailMessage(
    to=[
        {
            "address": "to@example.com",
            "substitution_data": {
                "key": "value"
            }
        }
    ],
    from_email='test@from.com'
)
e-mail.template = 'template-id'
e-mail.send()
```

Or cc, bcc, reply to, or attachments fields:

```python
from django.core.mail import EmailMultiAlternatives

e-mail = EmailMultiAlternatives(
    subject='hello from sparkpost',
    body='Woo hoo! Sent from Django!',
    from_email='from@yourdomain.com',
    to=['to@example.com'],
    cc=['ccone@example.com'],
    bcc=['bccone@example.com'],
    reply_to=['replyone@example.com']
)

e-mail.attach_alternative('<p>Woo hoo! Sent from Django!</p>', 'text/html')
e-mail.attach('image.png', img_data, 'image/png')
e-mail.send()
```

### 5.1.3 Supported version

SparkPost will support all versions of Django that are within extended support period. Refer to Django Supported Versions.

### 5.1.4 Additional documentation

See our Using SparkPost with Django in support article.
Additional documentation

The underlying SparkPost API is documented at the official SparkPost API Reference.
1. Check for open issues or open a fresh issue to start a discussion around a feature idea or a bug.
2. Fork the repository on GitHub and make your changes in a branch on your fork.
3. Write a test which shows that the bug was fixed or that the feature works as expected.
4. Send a pull request. Make sure to add yourself to AUTHORS.
S

sparkpost.recipient_lists, 15
sparkpost.suppression_list, 16
sparkpost.templates, 17
sparkpost.transmissions, 19
Index

C
create() (sparkpost.recipient_lists.RecipientLists method), 15
create() (sparkpost.suppression_list.SuppressionList method), 16
create() (sparkpost.templates.Templates method), 17

D
delete() (sparkpost.recipient_lists.RecipientLists method), 15
delete() (sparkpost.suppression_list.SuppressionList method), 16
delete() (sparkpost.templates.Templates method), 18
delete() (sparkpost.transmissions.Transmissions method), 19

G
get() (sparkpost.recipient_lists.RecipientLists method), 16
get() (sparkpost.suppression_list.SuppressionList method), 17
get() (sparkpost.templates.Templates method), 18
get() (sparkpost.transmissions.Transmissions method), 20

L
list() (sparkpost.recipient_lists.RecipientLists method), 16
list() (sparkpost.suppression_list.SuppressionList method), 17
list() (sparkpost.templates.Templates method), 18
list() (sparkpost.transmissions.Transmissions method), 20

P
preview() (sparkpost.templates.Templates method), 18

R
RecipientLists (class in sparkpost.recipient_lists), 15

S
send() (sparkpost.transmissions.Transmissions method), 20
sparkpost.recipient_lists (module), 15
sparkpost.suppression_list (module), 16
sparkpost.templates (module), 17
sparkpost.transmissions (module), 19
SuppressionList (class in sparkpost.suppression_list), 16

T
Templates (class in sparkpost.templates), 17
Transmissions (class in sparkpost.transmissions), 19

U
update() (sparkpost.recipient_lists.RecipientLists method), 16
update() (sparkpost.suppression_list.SuppressionList method), 17
update() (sparkpost.templates.Templates method), 19