pushjet (Python API)

Release 1.0.0

Jul 25, 2017

Contents

```
import pushjet
service = pushjet.Service.create("Python Pushjet API News")
service.send("This is a pretty sweet module!")
```

pushjet is the name of a Python module for interfacing with Pushjet. Simple and snazzy name, eh? Take a look at the examples and get a running start.

Contents:

CHAPTER 1

Getting started

First of all, install the Pushjet Python client. This can be done by simply running `pip install pushjet`. Then `import pushjet`, and you're all set to start. Something you should know is that "message" is Pushjet's term for a notification, since they're generic and could just as well be presented as... well... not notifications.

Creating and using a service

All communication with devices is done through services. These can be created frivolously without problem. It's fairly simple, check it out:

```
service = pushjet.Service("Cool name", "http://example.com/icon.png")
# At this point, service.public_key can be used to subscribe to it.
service.send(
    "Hey, check it out, there's a sale on!", # Message
    "SALE!",                                 # Title
    "http://example.com/item_on_sale.html"   # Link
)

# To later retrieve the service, store the service.secret_key somewhere
# and initialize the service as following.
service = pushjet.Service(secret_key)
```

More methods and properties can be found in the reference for the *`Service`* class.

Subscribing to a service and getting messages

Devices can subscribe to services using the *`Device`* class. These don't need to be registered - you can simply start using a random ID at any time. Common usage of the class looks like this:

```
import uuid
device = pushjet.Device(uuid.uuid4())
```

```
# Both of these work just fine.
device.subscribe(service)
device.subscribe("43c2-9317cc-eb3c7a3c4854-3e9b6-59e9e5a10")

for message in device.get_messages():
    print("Message from {service}:".format(service=message.service.name))
    if message.title:
        print(message.title)
    print(message.message)
    if message.link:
        print(message.link)
```

For complete information (for example on getting the services a device is subscribed to), see the reference for the *Device*, *Subscription*, and *Message* classes.

Notes on using a custom API instance

If you want to use another Pushjet server, the module lets you do that too. Simply use the *Api* class and use all the other classes as members of it, as following:

```
api = pushjet.Api("https://mypushjetapi.example.com/")
# You can then use:
api.Service("e1d669b8963b2f52a3e23581126365d2")
api.Device("036a256d-1fb7-453a-aa56-f6720ef7fd6c")
```

CHAPTER 2

Reference

Interface

Note that all methods can raise *RequestError* if the server somehow becomes unavailable. They can also raise *ServerError* if there's a bug in the server, but that's fairly unpredictable and in a perfect world would never be able to happen.

class pushjet.**Service**(*secret_key=None*, *public_key=None*)

A Pushjet service to send messages through. To receive messages, devices subscribe to these.

Parameters

- **secret_key** – The service's API key for write access. If provided, *send()*, *edit()*, and *delete()* become available. Either this or the public key parameter must be present.
- **public_key** – The service's public API key for read access only. Either this or the secret key parameter must be present.

Variables

- **name** – The name of the service.
- **icon_url** – The URL to the service's icon. May be None.
- **created** – When the service was created, as seconds from epoch.
- **secret_key** – The service's secret API key, or None if the service is read-only.
- **public_key** – The service's public API key, to be used when subscribing to the service.

classmethod create(*name*, *icon_url=None*)

Create a new service.

Parameters

- **name** – The name of the new service.
- **icon_url** – (optional) An URL to an image to be used as the service's icon.

Returns The newly-created *Service*.

delete()
Delete the service. Irreversible.

edit(*name=None*, *icon_url=None*)
Edit the service's attributes.

Parameters

- **name** – (optional) A new name to give the service.
- **icon_url** – (optional) A new URL to use as the service's icon URL. Set to an empty string to remove the service's icon entirely.

refresh()
Refresh the server's information, in case it could be edited from elsewhere.

Raises *NonexistentError* if the service was deleted before refreshing.

send(*message*, *title=None*, *link=None*, *importance=None*)
Send a message to the service's subscribers.

Parameters

- **message** – The message body to be sent.
- **title** – (optional) The message's title. Messages can be without title.
- **link** – (optional) An URL to be sent with the message.
- **importance** – (optional) The priority level of the message. May be a number between 1 and 5, where 1 is least important and 5 is most.

class pushjet.**Device**(*uuid*)
The "receiver" for messages. Subscribes to services and receives any messages they send.

Parameters **uuid** – The device's unique ID as a UUID. Does not need to be registered before using it. A UUID can be generated with `uuid.uuid4()`, for example.

Variables **uuid** – The UUID the device was initialized with.

get_messages()
Get all new (that is, as of yet unretrieved) messages.

Returns A list of *Message*s.

get_subscriptions()
Get all the subscriptions the device has.

Returns A list of *Subscription*s.

subscribe(*service*)
Subscribe the device to a service.

Parameters **service** – The service to subscribe to. May be a public key or a *Service*.

Returns The *Service* subscribed to.

Raises *NonexistentError* if the provided service does not exist.

Raises *SubscriptionError* if the provided service is already subscribed to.

unsubscribe(*service*)
Unsubscribe the device from a service.

Parameters **service** – The service to unsubscribe from. May be a public key or a *Service*.

Raises *NonexistentError* if the provided service does not exist.

Raises *SubscriptionError* if the provided service isn't subscribed to.

class pushjet.**Subscription**
A subscription to a service, with the metadata that entails.

Variables

- **service** – The service the subscription is to, as a *Service*.
- **time_subscribed** – When the subscription was made, as seconds from epoch.
- **last_checked** – When the device last retrieved messages from the subscription, as seconds from epoch.
- **device_uuid** – The UUID of the device that owns the subscription.

class pushjet.**Message**
A message received from a service.

Variables

- **message** – The message body.
- **title** – The message title. May be None.
- **link** – The URL the message links to. May be None.
- **time_sent** – When the message was sent, as seconds from epoch.
- **importance** – The message's priority level between 1 and 5, where 1 is least important and 5 is most.
- **service** – The *Service* that sent the message.

Custom API instances

class pushjet.**Api**(*url*)
An API with a custom URL. Use this if you're connecting to a self-hosted Pushjet API instance, or a non-standard one in general.

Parameters **url** – The URL to the API instance.

Variables **url** – The URL to the API instance, as supplied.

class **Device**(*uuid*)
Create a *Device* bound to the API. See *pushjet.Device* for documentation.

class Api.**Service**(*secret_key=None*, *public_key=None*)
Create a *Service* bound to the API. See *pushjet.Service* for documentation.

Exceptions

exception pushjet.**PushjetError**
All the errors inherit from this. Therefore, `except PushjetError` catches all errors.

exception pushjet.**AccessError**
Raised when a secret key is missing for a service method that needs one.

exception pushjet.**NonexistentError**
Raised when an attempt to access a nonexistent service is made.

exception `pushjet.`**`SubscriptionError`**
Raised when an attempt to subscribe to a service that's already subscribed to, or to unsubscribe from a service that isn't subscribed to, is made.

exception `pushjet.`**`RequestError`**
Raised if something goes wrong in the connection to the API server. Inherits from `ConnectionError` on Python 3, and can therefore be caught with `except ConnectionError` there.

Variables **`requests_exception`** – The underlying requests exception. Access this if you want to handle different HTTP request errors in different ways.

exception `pushjet.`**`ServerError`**
Raised if the API server has an error while processing your request. This getting raised means there's a bug in the server! If you manage to track down what caused it, you can open an issue on Pushjet's GitHub page.

Index