pyOutlook Documentation

Release 3.3.1b0

Jens Astrup

Aug 06, 2017

Contents

CHAPTER 1

About:

pyOutlook was created after I found myself attempting to connect to the Outlook REST API in multiple projects. This provided some much needed uniformity. It's easier to deal with than the win32com package by Microsoft, but obviously has a far smaller scope.

Python Versions

pyOutlook is only tested in, and targets, Python 3.5 and 3.6.

CHAPTER 2

Recommended:

pyOutlook does not handle OAuth for the access tokens provided by Outlook. These are provided by you via the OutlookAccount class as a string. There are various OAuth packages out there: (pip install) oauth2, python-oauth2, requests_oauthlib, etc that can facilitate the process.

CHAPTER 3

Notes:

All protected & private methods and anything under pyOutlook.internal is subject to change without deprecation warnings

CHAPTER 4

Contents:

Installation

Pip

```
pip install pyOutlook
```

Source

pyOutlook's PyPI page has a tar.gz and zip distribution for each release.

Quick Start

Instantiation

pyOutlook interacts with Outlook messages and folders through the class OutlookAccount(). The OutlookAccount acts as a gatekeeper to the other methods available, and stores the access token associated with an account.

Instantiation Example:

```
from pyOutlook import OutlookAccount
account_one = OutlookAccount('token 1')
account_two = OutlookAccount('token 2')
```

From here you can access any of the methods as documented in the *pyOutlook* section. Here are two examples of accessing an inbox and sending a new email.

Examples

Retrieving Emails

Through the OutlookAccount class you can call one of many methods - `get_messages()`, `inbox()`, etc. These methods return a list of *Message* objects, allowing you to access the attributes therein.

```
inbox = account.inbox()
inbox[0].body
>>> 'A very fine body'
```

Sending Emails

As above, you can send emails through the OutlookAccount class. There are two methods for sending emails - one allows chaining of methods and the other takes all arguments upfront and immediately sends.

Message

You can create an instance of a *Message* and then send from there.

```
from pyOutlook import *
# or from pyOutlook.core.message import Message

account = OutlookAccount('token')
message = Message(account, 'A body', 'A subject', [Contact('to@email.com')])
message.attach(bytes('some bytes', 'utf-8'), 'bytes.txt')
message.send()
```

new_email()

This returns a *Message* instance.

```
body = 'I\'m sending an email through Python. <br> Best, <br>Me'

email = account.new_email(body=body, subject='Hey there', to=Contact('myemail@domain.
↪com'))
email.sender = Contact('some_other_account@email.com')
email.send()
```

Note that HTML formatting is accepted in the message body.

send_email()

This method takes all of its arguments at once and then sends.

```
account_one.send_email(
    to=[Contact('myemail@domain.com')],
    # or to=['myemail@domain.com')]
    subject='Hey there',
    body="I'm sending an email through Python. <br> Best, <br> Me",
)
```

Contacts

All recipients, and the sender attribute, in `Messages` are represented by `Contacts`. Right now, this allows you to retrieve the name of a recipient, if provided by Outlook.

```
message = account.inbox()[0]
message.sender.name
>>> 'Dude'
```

When providing recipients to `Message` you can provide them either as a list of strings, or a list of `Contacts`. I prefer the latter, as there are further options in the Outlook API for interacting with Contacts - functionality for those may be added in the future.

pyOutlook

Outlook Account

class `pyOutlook.core.main.`**`OutlookAccount`**(*access_token*)

Sets up access to Outlook account for all methods & classes.

`access_token`

A string OAuth token from Outlook allowing access to a user's account

`deleted_messages`()

last ten deleted messages.

Returns list[Message]

`draft_messages`()

last ten draft messages.

Returns list[Message]

`get_folders`()

Returns a list of all folders for this account

`get_message`(*message_id*) → pyOutlook.core.message.Message

Gets message matching provided id.

the Outlook email matching the provided message_id.

Parameters **`message_id`** – A string for the intended message, provided by Outlook

Returns Message

`get_messages`(*page=0*)

Get first 10 messages in account, across all folders.

Keyword Arguments **`page`** (`int`) – Integer representing the 'page' of results to fetch

Returns List[Message]

`inbox`()

first ten messages in account's inbox.

Returns List[Message]

`new_email`(*body='', subject='', to: typing.List[pyOutlook.core.contact.Contact] = <class 'list'>*)

Creates a NewMessage object.

Returns Message

send_email(*body=None*, *subject=None*, *to: typing.List[pyOutlook.core.contact.Contact] = <class 'list'>*, *cc=None*, *bcc=None*, *send_as=None*, *attachments=None*)
Sends an email in one method using variables to set the various pieces of the email.

Parameters

- **body** (*str*) – The body of the email
- **subject** (*str*) – The subject of the email
- **to** (*list*) – A list of email addresses
- **cc** (*list*) – A list of email addresses which will be added to the 'Carbon Copy' line
- **bcc** (*list*) – A list of email addresses while be blindly added to the email
- **Contact** (*send_as*) – A string email address which the OutlookAccount has access to
- **attachments** (*list*) – A list of dictionaries with two parts [1] 'name' - a string which will become the file's name [2] 'bytes' - the bytes of the file.

sent_messages()
last ten sent messages.

Returns list[Message]

Message

class pyOutlook.core.message.**Message**(*account*, *body: str*, *subject: str*, *to_recipients: typing.Union[typing.List[pyOutlook.core.contact.Contact], typing.List[str]]*, *sender: pyOutlook.core.contact.Contact = None*, *cc: typing.List[pyOutlook.core.contact.Contact] = None*, *bcc: typing.List[pyOutlook.core.contact.Contact] = None*, *message_id: str = None*, ***kwargs*)
An object representing an email inside of the OutlookAccount.

message_id
A string provided by Outlook identifying this specific email

body
The body content of the email, including HTML formatting

body_preview
"The first 255 characters of the body"

subject
The subject of the email

sender
The `Contact` who sent this email. You can set this before sending an email to change which account the email comes from (so long as the `OutlookAccount` specified has access to the email.

to
A list of `Contacts`. You can also provide a list of strings, however these will be turned into `Contacts` after sending the email.

cc
A list of `Contacts` in the CC field. You can also provide a list of strings, however these will be turned into `Contacts` after sending the email.

bcc
A list of `Contacts` in the BCC field. You can also provide a list of strings, however these will be turned into `Contacts` after sending the email.

is_draft
Whether or not the email is a draft.

importance
The importance level of the email; with 0 indicating low, 1 indicating normal, and 2 indicating high. `Message.IMPORTANCE_LOW`, `Message.IMPORTANCE_NORMAL`, & `Message.IMPORTANCE_HIGH` can be used to reference the levels.

categories
A list of strings, where each string is the name of a category.

time_created
A datetime representing the time the email was created

time_sent
A datetime representing the time the email was sent

IMPORTANCE_HIGH = 2

IMPORTANCE_LOW = 0

IMPORTANCE_NORMAL = 1

add_category (*category_name: str*)

attach (*file_bytes*, *file_name*)
Adds an attachment to the email. The filename is passed through Django's get_valid_filename which removes invalid characters. From the documentation for that function:

```
>>> get_valid_filename("john's portrait in 2004.jpg")
'johns_portrait_in_2004.jpg'
```

Parameters

- **file_bytes** – The bytes of the file to send (if you send a string, ex for CSV, pyOutlook will attempt to convert that into bytes before base64ing the content).
- **file_name** – The name of the file, as a string and leaving out the extension, that should be sent

copy_to (*folder_id*)
Copies the email to the folder specified by the folder_id.

The folder id must match the id provided by Outlook.

Parameters **folder_id** – A string containing the folder ID the message should be copied to

copy_to_deleted ()
Copies Message to account's Deleted Items folder

copy_to_drafts ()
Copies Message to account's Drafts folder

copy_to_inbox ()
Copies Message to account's Inbox

delete ()
Deletes the email

forward(*to_recipients: typing.Union[typing.List[pyOutlook.core.contact.Contact], typing.List[str]], forward_comment=None*)
Forward Message to recipients with an optional comment.

Parameters

- **to_recipients** – A list of *Contacts* to send the email to.
- **forward_comment** – String comment to append to forwarded email.

Examples

```
>>> john = Contact('john.doe@domain.com')
>>> betsy = Contact('betsy.donalds@domain.com')
>>> email = Message()
>>> email.forward([john, betsy])
>>> email.forward([john], 'Hey John')
```

is_read
Set and retrieve the 'Read' status of an email

```
>>> message = Message()
>>> message.is_read
>>> False
>>> message.is_read = True
```

move_to(*folder*)
Moves the email to the folder specified by the folder parameter.

Parameters **folder** – A string containing the folder ID the message should be moved to, or a Folder instance

move_to_deleted()
Moves the email to the account's Deleted Items folder

move_to_drafts()
Moves the email to the account's Drafts folder

move_to_inbox()
Moves the email to the account's Inbox

parent_folder
Returns the *Folder* this message is in

```
>>> account = OutlookAccount('')
>>> message = account.get_messages()[0]
>>> message.parent_folder
Inbox
>>> message.parent_folder.unread_count
19
```

Returns: *Folder*

reply(*reply_comment*)
Reply to the Message.

Notes

HTML can be inserted in the string and will be interpreted properly by Outlook.

Parameters **`reply_comment`** – String message to send with email.

`reply_all`(*reply_comment*)
Replies to everyone on the email, including those on the CC line.

With great power, comes great responsibility.

Parameters **`reply_comment`** – The string comment to send to everyone on the email.

`send`(*content_type='HTML'*)
Takes the recipients, body, and attachments of the Message and sends.

Parameters **`content_type`** – Can either be 'HTML' or 'Text', defaults to HTML.

Folder

class `pyOutlook.core.folder.`**`Folder`**(*account*, *folder_id*, *folder_name*, *parent_id*, *child_folder_count*, *unread_count*, *total_items*)
An object representing a Folder in the OutlookAccount provided.

`account`
The *`OutlookAccount`* this folder should be associated with

`id`
The static id generated by Outlook to identify this folder.

`folder_name`
The name of this folder as displayed in the account

`parent_id`
The id of the folder which houses this Folder object

`child_folder_count`
The number of child folders inside this Folder

`unread_count`
The number of unread messages inside this Folder

`total_items`
A sum of all items inside Folder

`copy_into`(*destination_folder: pyOutlook.core.folder.Folder*)
Copies the Folder into the provided destination folder.

Raises `AuthError` – Raised if Outlook returns a 401, generally caused by an invalid or expired access token.

Parameters **`destination_folder`** – The Folder that this Folder should be copied to.

Returns A new *`Folder`* representing the newly created folder.

`create_child_folder`(*folder_name*)
Creates a child folder within the Folder it is called from and returns the new Folder object.

Parameters **`folder_name`** – The name of the folder to create

Returns: *`Folder`*

delete()
Deletes this Folder.

Raises `AuthError` – Raised if Outlook returns a 401, generally caused by an invalid or expired access token.

get_subfolders()
Retrieve all child Folders inside of this Folder.

Raises `AuthError` – Raised if Outlook returns a 401, generally caused by an invalid or expired access token.

Returns List[*Folder*]

messages()
Retrieves the messages in this Folder, returning a list of *Messages*.

move_into(*destination_folder: pyOutlook.core.folder.Folder*)
Move the Folder into a different folder.

This makes the Folder provided a child folder of the destination_folder.

Raises `AuthError` – Raised if Outlook returns a 401, generally caused by an invalid or expired access token.

Parameters **destination_folder** – A *Folder* that should become the parent

Returns A new *Folder* that is now inside of the destination_folder.

rename(*new_folder_name*)
Renames the Folder to the provided name.

Parameters **new_folder_name** – A string of the replacement name.

Raises `AuthError` – Raised if Outlook returns a 401, generally caused by an invalid or expired access token.

Returns A new Folder representing the folder with the new name on Outlook.

Contact

class pyOutlook.core.contact.**Contact**(*email: str*, *name: str = None*)
Represents someone sending or receiving an email. Cuts down on the amount of dictionaries floating around that each hold the API's syntax and allows for functionality to be added in the future.

CHAPTER 5

Indices and tables

- genindex
- search

Index