watson-workspace-sdk Documentation

Cathal A. Dinneen

Mar 07, 2019

Contents

CHAPTER 1

Getting Started

Installing package pip install https://github.com/cathaldi/watson-workspace-python-sdk

If you want to send messages from a client you just need import Message and create one after your app has been created .

```
from watson-workspace-sdk import Message
import watson_workspace_sdk as ww

WORKSPACE_ID = os.environ.get('WORKSPACE_ID')  # array of spaces
APP_ID = os.environ.get('APP_ID')
APP_SECRET = os.environ.get('APP_SECRET')

workspace_connection = ww.Client(APP_ID, APP_SECRET)  # Authenticates and stores jwt␣
↪token for requests

Message.create(WORKSPACE_ID, "Hello World", "Some text here", 'green')
```

Client object can get a dict[space_id, Space] of spaces. Which returns a the spaces the bot has been added to. (defaults to 10) .. code-block:: python

import watson-workspace-sdk as ww

workspace_connection = ww.Client(APP_ID, APP_SECRET spaces_list = workspace_connection.get_space_ids()

CHAPTER 2

Making an App/Bot

A bot will need to be hosted on a server that

- has a domain name
- supports HTTPS

Log into https://developer.watsonwork.ibm.com/apps and Create New App

An application Id and Secret will be generated. Take note of the secret as it is only generated once but you can regenerate a new one whenever you want.

If you just want to use the app for one way communication to workspace, for instance sending messages you should be set as this can be achieved through authorising with the APP_ID and APP_SECRET.

To react to events like new messages in a space, users being removed, reactions added etc. we'll need to set up a bot that makes use of Webhooks

CHAPTER 3

Webhooks

3.1 Adding webhooks to your application

After creating your application navigate to **Listen to events** and click **Add an outbound webhook** In the image below we have added an endpoint and we are watching for the event **message-annotation-added** which will be triggered when any annotation is added for any message. So if we were to add a focus annotation or click an annotation. It also triggers when Watson Workspace adds Watson API annotations

![alt text](docs/images/ww_add_listener.png "Image showing the result of clicking a Watson Workspace event")

![alt text](docs/images/ww_add_action.png "Image showing the result of clicking a Watson Workspace event")

3.1.1 Decorators

There are a number of decorators to help handle events from Watson Workspace.

@verify_workspace Verify incoming requests originate from Watson Workspace. Requests are verified with through a provided webhook secret taken as a parameter.

@handle_verification Handles verification messages from Watson Workspace when enabling a webhook and also every 5 minutes responding to Watson Workspace's periodic verification.

3.2 Example

When triggered

```
@app.route('/messages', methods=["POST"])
@verify_workspace_origin(os.environ.get("webhook_secret"))
@handle_verification(os.environ.get("webhook_secret"))
def message_webhook(*args, **kwargs):
    webhook_event = Webhook.from_json(request.json)
```

(continues on next page)

(continued from previous page)

```
    if webhook_event.user_id == workspace_connection.id:
        return "" # if this bot sent the message, ignore

    Message.create(space_id=webhook_event.space_id, title="", text=request.json().get(
↪"content"), actor="Echo Bot", color="blue")
```

CHAPTER 4

Watson Annotations

For each chat message sent in Watson Workspace many Annotations are generated utilising the Watson APIs to extract information from the chat message. Whether is it used or not these will be sent with each message, so if there is a webhook listening in for annotation-added these will have to be ignored,handled or filtered.

There may be some information that is useful for example message-nlp-entities picked out a location which could display an annotation displaying travel options.

4.1 Sample Sentence

We need to get this client on the call, it is very important we don't lose them after the San Francisco fiasco

To stress, this is a sample sentence.

message-nlp-entities .. code-block:: javascript

{"language":"en","entities":[{"count":1,"relevance":0.33,"text":"San Francisco","type":"Location"}]}

Language was picked up, along with the entity San Francisco which is of type Location

message-nlp-keywords

```
{"language":"en","keywords":[{"relevance":0.903347,"text":"San Francisco fiasco"},{
↪"relevance":0.263684,"text":"client"}]}
```

Keywords noted were "San Francisco fiasco" and "client"

message-nlp-docSentiment

```
{"language":"en","docSentiment":{"score":0.488068,"type":"neutral"}}
```

message-nlp-concepts

```
{"language":"en","concepts":[{"dbpedia":"http://dbpedia.org/resource/2000s_music_
↪groups","relevance":0.840367,"text":"2000s music groups"},{"dbpedia":"http://
↪dbpedia.org/resource/São_Francisco_(disambiguation)","relevance":0.757086,"text":
↪"São Francisco"}]}
```

(continues on next page)

(continued from previous page)

message-nlp-taxonomy


```
{"language":"en","taxonomy":[{"confident":false,"label":"/business and industrial/
↪advertising and marketing/public relations","score":0.246501},{"confident":false,
↪"label":"/health and fitness/weight loss","score":0.218009},{"confident":false,
↪"label":"/art and entertainment/music/music genres/hip hop","score":0.162234}]}
```

CHAPTER 5

API

5.1 Annotation

class watson_workspace_sdk.models.annotation.**Annotation**(*title: str*, *display_text: str*)

Bases: object

Annotations are ephemeral messages that are initiated by the requester. For example clicking on a focus annotation could initiate a call and display an annotation message only to the user that clicked it.

Annotations typically have a Title, Body along with an optional sender name, buttons to trigger events and change display colour.

Attributes

title [str] Title/Heading of the Annotation

display_text [str] Main body of the Annotation

buttons [Optional[List[Button]]] List of Buttons if any that the Annotation has

Methods

add_button(*display_text: str*, *action: str*, *style: str = 'PRIMARY'*) → None

Adds a new button object to the Annotation

Parameters

- **display_text** – Display text for button.
- **action** – Workspace action to be executed on click. Typically defined in Actions in Workspace Dev UI
- **style** – Visual style of the button. Primary or Secondary.

Returns None

```
>>> new_annotation.add_button("Yes","guessed_correct_event")
>>> new_annotation.add_button("No","guessed_incorrect_event")
```

5.2 Card

class watson_workspace_sdk.models.card.**Button**(*display_text: str*, *action: str*, *style: str = 'Primary'*)

Bases: object

Basic class that handles Button settings for Annotations and Attachments

Attributes

display_text [str] Button text

action [str] Watson Workspace action to execute when button is clicked

style [str] Button style : Primary/Secondary Default is Primary

class watson_workspace_sdk.models.card.**Card**(*title: str*, *subtitle: str*, *display_text: str*)

Bases: object

Cards are used as part of Action Fulfillment to present multiple options to the user. Each option is displayed as a card. Cards are similar to Annotations in that they have a title, body and optional buttons but also have a subtitle field.

Attributes

type [str] Card type

title [str] Card Title

subtitle [str] Card subtitle

display_text [str] Main body of Card

date [Long] Epoch timestamp in milliseconds

buttons [List[Buttons]] List of buttons for the Card

Methods

add_button(*display_text*, *action*, *style='PRIMARY'*) → None

Adds a button to the card :param display_text: Button label text :param action: Watson Workspace action to execute :param style: Button UI Style, default Primary :return: None

5.3 Message

class watson_workspace_sdk.models.message.**Message**(*id: str = ''*)

Bases: object

Used for creating,updating and deleting Messages

Attributes

content [str] Title/Heading of the Annotation

content_type [str] Message content type, appMessage by default

annotations [Annotation] List of annotations attributed to the message, usually created by Watson API

created [datetime] Message creation time

updated [datetime] Message modification time

created_by [Person] Message creator

id [str] Message Id

updated_by [Person] Person who last modified message

version [str] Message version

Methods

add_focus(*phrase: str = ''*, ***, *lens: str = 'Opportunity'*, *category: str = 'Inquiry'*, *actions: str = '"test"'*, *confidence: float = 0.99*, *payload: str = ''*, *start: int = 0*, *end: int = 0*, *version: int = 1*, *hidden: str = 'false'*) → None

Parameters

- **phrase** – Subset of message to mark as focus topic
- **lens** – Lens to view request, options are ActionRequest, Question, and Commitment
- **category** –
- **actions** – Watson Workspace Action to initiate when focus is triggered
- **confidence** – Confidence level
- **payload** –
- **start** – inclusive index (including markdown) at which to start as the focus topic
- **end** – inclusive index (including markdown) at which to end as the focus topic
- **version** – Focus version
- **hidden** – Is focus hidden

Returns None

```
>>> my_message.add_focus(start=55, end=67, actions='"check_weather"')
```

add_reaction(*reaction: str*) → None

Adds a supported unicode character as a reaction to a message :param reaction: Unicode reaction to append to message :return: None >>> my_message.add_reaction(“”)

count_reaction(*reaction: str*) → int

Parameters **reaction** – Unicode reaction to verify e.g.

Returns Number of times reaction has been added to message

Return type int

classmethod create(*space_id: str*, *title='title'*, *text='text'*, *color='#36a64f'*, *actor=''*) → watson_workspace_sdk.models.message.Message

Sends a message to Watson Workspace and then creates a Message object

Parameters

- **space_id** – Target space Id
- **title** – Message Title
- **text** – Message body
- **color** – Border color for message
- **actor** – Message sender

Returns Message object of sent message

Return type *Message*

Example

```
>>> my_message = Message.create(space_id=space_id, title="Today's Weather",␣
↪text="Rain", actor="Rain App", color="blue")
```

classmethod from_json(*response_json*) → watson_workspace_sdk.models.message.Message
Creates a Message object from JSON. Useful when retrieving a list of messages. Does not send a messsage to Watson Workspace :param response_json: JSON Message Object :return: Message :rtype: Message

classmethod get(*message_id: str*) → watson_workspace_sdk.models.message.Message
Retrieves existing message from Watson Workspace

Parameters **message_id** – Message Id to retrieve

Returns Message

Return type *Message*

```
>>> my_message = Message.get(message_id)
```

static message_with_annotation(*, *conversation_id: str*, *target_user_id: str*, *target_dialog_id: str*, *annotation: watson_workspace_sdk.models.genericAnnotation.Annotation*) → None

Parameters

- **conversation_id** – Space Id
- **target_user_id** – User Id
- **target_dialog_id** – Reference to the annotation event that initiated call, provided by webhook event
- **annotation** – Annotation to send

Returns None

```
>>> test_annotation = Annotation("Test Annotation", "Here's a test annotation␣
↪with a button")
>>> test_annotation.add_button("Click here", "button_test_event")
>>> Message.message_with_annotation(conversation_id=webhook_event.space_id,␣
↪target_user_id=webhook_event.user_id,target_dialog_id=annotation.get(
↪"targetDialogId"), annotation=test_annotation)
```

static message_with_attachment(*, *conversation_id: str, target_user_id: str, target_dialog_id: str, cards: List[watson_workspace_sdk.models.card.Card]*) → None
https://developer.watsonwork.ibm.com/docs/tutorials/action-fulfillment :param conversation_id: Space Id :param target_user_id: User Id :param target_dialog_id: Reference to the annotation event that initiated call, provided by webhook event :param cards: Cards to be added :return: None

```
>>> new_card = Card("Here's test card 1 ", "A smaller title", "Body")
>>> new_card.add_button("Test Button", "test_button_event")
>>> attached_message = Message.message_with_attachment(conversation_
↪id=webhook_event.space_id, target_dialog_id=annotation.get("targetDialogId
↪"), target_user_id=annotation.get("targetDialogId"), cards=[new_card])
```

remove_reaction(*reaction: str*) → None
Removes a supported unicode character as a reaction to a message assuming user has added that character previously :param reaction: Reaction to remove :return: None

```
>>> my_message.remove_reaction("")
```

to_json() → str
Formats a message object into the format expect by watson workspace.

5.4 Person

class watson_workspace_sdk.models.person.**Person**
Bases: object

Object for Watson Workspace Users

Attributes

display_name [str] Title/Heading of the Annotation

ext_id [str] Message content type, appMessage by default

email [str] List of annotations attributed to the message, usually created by Watson API

photo_url [str] Message creation time

customer_id [str] Message modification time

ibm_unique_id [str] Message creator

created [str] Message Id

updated [datetime] Person who last modified message

created_by [datetime] Message version

presence [str] Message version

updated_by [Person] Message version

id [str] Message version

is_bot [str] Message version

Methods

classmethod from_json(*json_body: {}*) → watson_workspace_sdk.models.person.Person
Creates a Person user from a JSON object containing Person data :param json_body: JSON Person Object :return: Person :rtype: Person

classmethod get(*user_id: str*) → watson_workspace_sdk.models.person.Person
A simple method to populate Conversation object from a conversation_id

Parameters **user_id** – the person id to retrieve from Watson Workspace

Returns Person

Raises **keyError** – raises an exception

5.5 Space

class watson_workspace_sdk.models.space.**Space**(*id: str = '', created: <module 'datetime' from '/home/docs/.pyenv/versions/3.7.1/lib/python3.7/datetime.py'> = datetime.time(0, 0), created_by=None, updated: <module 'datetime' from '/home/docs/.pyenv/versions/3.7.1/lib/python3.7/datetime.py'> = datetime.time(0, 0), updated_by=None, title='', description='', members=[], members_updated=datetime.time(0, 0), conversation=None*)

Bases: `object`

Basic class that Space functionality.

Attributes

id [str] Space Id

created [datetime] Space creation date

created_by [Person] Person that created the space

updated [datetime] Space modification date

updated_by [Person] Person who last updated the space

description [str] Space description string

members [List[Person]] List of space members

members_updated [datetime] Memberlist update timestamp

conversation [str] Conversation object - may be removed

Methods

add_members(*list_of_members_to_add: [<class 'str'>]*) → None
Adds a list of members to a Space :param: list_of_members_to_add :return: None

classmethod create(*space_title*) → watson_workspace_sdk.models.space.Space
A simple method to create a space with a provided space title.

Parameters **space_title** – Title of the new workspace.

Returns Space

Return type *Space*

delete() → None
A simple method to delete the space.

Returns None

classmethod get(*space_id*) → watson_workspace_sdk.models.space.Space
A simple method to populate Conversation object from a space_id

Parameters **space_id** – the id of the space to retrieve

Returns Space

Return type *Space*

classmethod get_from_json(*json_body*) → watson_workspace_sdk.models.space.Space
Gets a Space object from JSON :param json_body: :return: Space :rtype: Space

remove_members(*list_of_members_to_delete: [<class 'str'>]*) → None
Removes a list of members to a Space :param: list_of_members_to_remove :return: None

send_message(*message_to_send: watson_workspace_sdk.models.message.Message*) → None
Sends a message to a Space :param message_to_send: Message object to send :return: None

watson_workspace_sdk.models.space.**parse_members**(*members_json*) → List[watson_workspace_sdk.models.person.Person]
Takes in a JSON Array of members and converts it to a list of Person objects :param members_json: JSON Object :return: A list of Members of a workspace :rtype: List[Person]

5.6 Decorators

watson_workspace_sdk.decorators.**filter_workspace_annotations**(*whitelist=[]*, *blacklist=[]*)
Drops and/or allows through specific annotations. Useful to cut out the Watson annotation calls that are not too informative.

blacklist = [toscana-words] # Drop these annotations.

whitelist = ["generic"] # These are standard chat messages from users

whitelist = ["actionSelected", "generic" # Allows user chat messages and action events such as clicking a button, slash command or clicking on a focus annotation.

Parameters

- **whitelist** – List of annotations to allow
- **blacklist** – List of annotations to drop

Returns Incoming request

Use: After a user creates a message, Workspace will annotate the message extracting Workspace Moment data. This is a message-annotation-edited event, and the annotation type will be a conversation-moment. My app has no use for this information so will be dropped.

watson_workspace_sdk.decorators.**handle_verification**(*webhook_secret: str*)
Checks if incoming request is a periodic Watson Workspace Verification message and replies to the challenge.

Parameters **webhook_secret** – Webhook secret - Found in Watson Workspace application page.

Returns Incoming request

watson_workspace_sdk.decorators.**ignore_bots**()
Ignore all messages from Watson Workspace bots/Apps

watson_workspace_sdk.decorators.**timer**()

watson_workspace_sdk.decorators.**user_blacklist**(*blacklist*)
Ignores messages from a blacklist of user ids

watson_workspace_sdk.decorators.**user_whitelist**(*whitelist*)
Ignore all messages unless a user is on a whitelist of user ids

watson_workspace_sdk.decorators.**verify_workspace_origin**(*webhook_secret: str*)
Checks incoming flask requests and ensures message came from Watson Workspace. If request does not originate from workspace it is logged and dropped.

Parameters **webhook_secret** – Webhook secret - Found in Watson Workspace application page.

Returns Incoming request

CHAPTER 6

Examples

6.1 Authenticate with Watson Workspace

```
import watson_workspace_sdk as ww

WORKSPACE_ID = os.environ.get('WORKSPACE_ID')  # array of spaces
APP_ID = os.environ.get('APP_ID')
APP_SECRET = os.environ.get('APP_SECRET')

workspace_connection = ww.Client(APP_ID, APP_SECRET)  # Authenticates and stores jwt␣
↪token for requests
```

6.2 Sending Messages

```
from watson_workspace_sdk import Message

my_first_message = ww.Message.create(WORKSPACE_ID, "Hello World", "Some text here",
↪'green')
```

6.3 Reacting to Messages

Adding a reaction

```
my_first_message.add_reaction("")
```

Removing a reaction

```
my_first_message.remove_reaction("")
```

6.4 Sending messages with annotations/attachments

Create message with attachment .. code-block:: python

from watson_workspace_sdk import card new_card = Card("Here's test card 1 ", "A smaller title", "Body") new_card.add_button("Test Button", "test_button_event") attached_message = Message.message_with_attachment(conversation_id=webhook_event.space_id, target_dialog_id=annotation.get("targetDialogId"), target_user_id=annotation.get("targetDialogId"), cards=[new_card])

Create message with annotation .. code-block:: python

test_annotation = Annotation("Test Annotation", "Here's a test annotation with a button") test_annotation.add_button("Click here", "button_test_event") Message.message_with_annotation(conversation_id=webhook_event.space_id, target_user_id=webhook_event.user_id,target_dialog_id=annotation.get("targetDialogId"), annotation=test_annotation)

6.5 Playing with Spaces

Get a space .. code-block:: python

my_space = ww.Space.get(space_id)

Add members .. code-block:: python

my_space.add_members()

Remove members .. code-block:: python

my_space.remove_members()

Python Module Index

W

Index

S

T

U

V

W