**Editing a List of Users from Webex Teams**

Let’s say we need to make multiple similar requests to the Webex Teams API. Maybe you want to send the same message to multiple spaces, or edit the same attribute of various users in your org. Currently, you’ll need to make those requests one at a time. This post will walk through editing and deleting multiple users in an organization to show you how to loop through a list of items to achieve a bulk action, which you can use to handle multiple types of requests with some relatively minor changes.  
  
Since we’re discussing editing and deleting multiple users from an organization, you’ll need to have admin privileges in order to run this particular demo. If you modify the code for less restricted actions, like deleting messages, admin privileges won’t be needed.

Also note that once a user is deleted, there is no way to restore that user, so a new user will need to be created if you delete any users by mistake (probably best to run this on a test org before implementing it on your real org).

This demo uses the **requests** library for python, so you may need to install that if you have not done so already:

<http://docs.python-requests.org/en/master/user/install/>

One of the examples below is written for Python 2, and the other for Python 3, so keep that in mind as you go along. Really, the only difference is how the **quote** function is imported.

In Python 2:

**from** urllib **import** quote

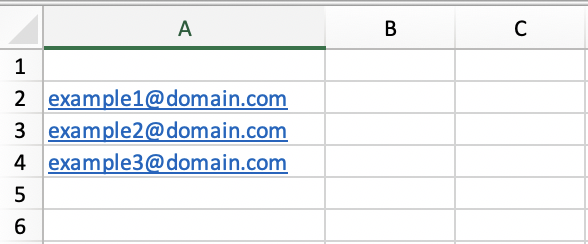
In Python 3:

**from** urllib.parse **import** quote

We’ll start with editing a list of people. Let’s say the email domain of your company is changing, so you want to update all of your user accounts to reflect this. This example assumes that you have all of the email addresses that need to be changed, in a single column CSV file, like the example CSV here:

<https://github.com/webex/Spark-API-Demos/blob/master/EditDeletePeople/people_emails.csv>

Note that the first row, where headers typically go, is intentionally left empty.



The code below requires four small changes (lines 20 through 23 if viewing from the GitHub repo). The changes required are:

bearer = "ADMIN\_TOKEN" *#your admin token*

filepath = "/Users/tahanson/Documents/people\_emails.csv"

from\_domain = "@domone.com"

to\_domain = "@domtwo.com"

* **bearer** must correspond to a valid Admin token.
  + If you are an admin, for testing purposes you can sign in to the developer portal and copy your token from the section, “Your Personal Access Token,” found here: <https://developer.webex.com/docs/api/getting-started/accounts-and-authentication>
  + Note the developer portal token currently only lasts 12 hours, so if you want this to be a more permanent solution you can run without manually updating the token each time you run it, you want to build an integration: <https://developer.webex.com/docs/integrations>
* **filepath** must be either a full/absolute filepath or a relative filepath to the CSV from the location of the python script on your machine.
* **from\_domain** is the current domain of the user email addresses. For example: john@oldcorp.com
* **to\_domain** is the desired domain of the user email addresses. For example: john@newcorp.com

Here is the full code to edit a CSV list of people (Python 3):

**import** requests

**import** json

**import** time

**from** urllib.parse **import** quote

**def** sendGET(url):

response = requests.get(url,

headers={"Accept" : "application/json",

"Content-Type":"application/json",

"Authorization": "Bearer "+bearer})

**return** response

**def** sendPUT(url, data):

response = requests.put(url, data=json.dumps(data),

headers={"Accept" : "application/json",

"Content-Type":"application/json",

"Authorization": "Bearer "+bearer})

**return** response

bearer = "ADMIN\_TOKEN" *#your admin token*

filepath = "/Users/tahanson/Documents/people\_emails.csv" *#each user on a new line.*

from\_domain = "@domone.com"

to\_domain = "@domtwo.com"

lines = []

**with** open(filepath, 'r') **as** f:

lines = f.readlines()

people\_url = 'https://api.ciscospark.com/v1/people?email={0}'

edit\_url = 'https://api.ciscospark.com/v1/people/{0}'

failure = False

**for** line **in** lines[1:]:*#basically, this is 'for every email address in the list'*

**try**:

**if** failure:

**break**

retry = True

**while** retry **and** **not** failure:

*#First, you have to get the person's personId, by looking them up via email*

person\_email = line.strip('\r\n')

**print**("Person Email to look up: <{0}>".format(line.strip('\r\n')))

result = sendGET(people\_url.format(quote(person\_email)))

**if** result.status\_code < 300:

person\_details = result.json().get('items') *#could print full person\_details if you really wanted to.*

**if** len(person\_details) > 0:

person\_id = person\_details[0]['id']

**print**("Person Id to edit is: {0}".format(person\_id)) *#we'll just print this person's Id.*

**print**("Editing person...")

new\_email = person\_email.replace(from\_domain, to\_domain)

data = person\_details[0]

data.update({"emails":[new\_email]})

result = sendPUT(edit\_url.format(person\_id), data)

**print**("Edit result (200 is success): {0}".format(result.status\_code))

**else**:

**print**("That user does not seem to exist.")

retry = False

**else**:

**print**("Failed due to lookup user because: ")

**print**(result.status\_code)

**if** result.status\_code == 429:

sleep\_time = result.headers.get('Retry-After')

**if** sleep\_time **is** **None**:

**print**("429 received without a Retry-After header... defaulting to 60.")

sleep\_time = 60

**else**:

sleep\_time = int(sleep\_time)

**print**("Sleeping for {0} seconds before trying again.".format(sleep\_time))

time.sleep(sleep\_time)

**else**:

**print**(result.json())

**print**("Stopping here because this error will likely repeat.")

failure = True

**break**

**except** Exception **as** e:

**print**(e)

**print**(e.headers)

**print**(e.read())

To summarize the above, we loop over each email in the .csv file. For each email address, we look up the person details from Webex Teams with our **sendGET** function. This also gives us the personId, which we need to make any changes to a person. Then, we edit the email address to use the new domain, and update that person detail dictionary. Then we tell Webex Teams to update the person with our **sendPUT** function. The bulk of this effort takes place between lines 39 and 54 (when viewing from the GitHub file).

This code also handles 429 errors and Retry-After headers. More details about that can be found here: <https://github.com/ciscospark/SparkRetryAfterDemo>

**Deleting a List of Users from Webex Teams**

Now let’s say we have people we want to delete from Webex Teams. Assuming we have a CSV file setup in the same way as above, our code will look very similar, and even a little simpler. We’ll trade out our **sendPUT** function for **sendDELETE.**

This time, you’ll only need to edit **bearer** and **filepath.**

Note the differing import of the **quote** function, so that this code will work for Python 2 instead:

**import** requests

**import** json

**import** time

**from** urllib **import** quote

**def** sendGET(url):

response = requests.get(url,

headers={"Accept" : "application/json",

"Content-Type":"application/json",

"Authorization": "Bearer "+bearer})

**return** response

**def** sendDELETE(url):

response = requests.delete(url,

headers={"Accept" : "application/json",

"Content-Type":"application/json",

"Authorization": "Bearer "+bearer})

**return** response

bearer = "ADMIN\_TOKEN" *#your admin token*

filepath = '/Users/tahanson/Documents/people\_emails.csv' *#each user on a new line.*

lines = []

**with** open(filepath, 'r') **as** f:

lines = f.readlines()

people\_url = 'https://api.ciscospark.com/v1/people?email={0}'

delete\_url = 'https://api.ciscospark.com/v1/people/{0}'

failure = False

**for** line **in** lines[1:]:*#basically, this is 'for every email address in the list'*

**try**:

**if** failure:

**break**

retry = True

**while** retry **and** **not** failure:

*#First, you have to get the person's personId, by looking them up via email*

person\_email = line.strip('\r\n')

**print**("Person Email to look up: <{0}>".format(person\_email))

result = sendGET(people\_url.format(quote(person\_email)))

**if** result.status\_code < 300:

person\_details = result.json().get('items') *#could print full person\_details if you really wanted to.*

**if** len(person\_details) > 0:

person\_id = person\_details[0]['id']

**print**("Person Id to delete is: {0}".format(person\_id)) *#we'll just print this person's Id.*

**print**("Deleting person...")

result = sendDELETE(delete\_url.format(person\_id))

**print**("Delete result (204 is success): {0}".format(result.status\_code))

**else**:

**print**("That user does not seem to exist.")

retry = False

**else**:

**print**("Failed due to lookup user because: ")

**print**(result.status\_code)

**if** result.status\_code == 429:

sleep\_time = result.headers.get('Retry-After')

**if** sleep\_time **is** **None**:

**print**("429 received without a Retry-After header... defaulting to 60.")

sleep\_time = 60

**else**:

sleep\_time = int(sleep\_time)

**print**("Sleeping for {0} seconds before trying again.".format(sleep\_time))

time.sleep(sleep\_time)

**else**:

**print**(result.json())

**print**("Stopping here because this error will likely repeat.")

failure = True

**break**

**except** Exception **as** e:

**print**(e)

**print**(e.headers)

**print**(e.read())

In summary, this code is almost the same as the code to edit people. We still have to look up the personId using the email address and our **sendGET** function. Then, we can simply use that ID to delete the person with our **sendDELETE** function.

**Deleting a List of RoomIds from Webex Teams**

We can simplify this even further, and show how you could use the same process to delete a list of roomIds. Assuming your roomIds are in a CSV in the same format as the emails from the previous example, we can run the following:

**import** requests

**def** sendDELETE(url):

response = requests.delete(url,

headers={"Accept" : "application/json",

"Content-Type":"application/json",

"Authorization": "Bearer "+bearer})

**return** response

bearer = "BEARER\_TOKEN" *#your bearer token*

filepath = "rooms.csv" *#each roomId on a new line.*

lines = []

**with** open(filepath, 'r') **as** f:

lines = f.readlines()

delete\_url = 'https://api.ciscospark.com/v1/rooms/{0}'

failure = False

**for** line **in** lines[1:]:*#basically, this is 'for every ID in the list'*

room\_id = line.strip('\r\n')

**print**("RoomId to delete is: {0}".format(room\_id)) *#we'll just print this roomId.*

**print**("Deleting room...")

result = sendDELETE(delete\_url.format(room\_id))

**print**("Delete result (204 is success): {0}".format(result.status\_code))

Once again the **bearer** and **filepath** will need to be changed, but otherwise this code is even simpler. It doesn’t require a **sendGET** at all, because we’re starting with the IDs. That means we can go right into the delete action.

Since we’re using a list of IDs, this could be easily translated to pretty much any Webex Teams API endpoint, whether it be deleting rooms, memberships, or people. All that would really need to be changed is the **delete\_url.**

The full code for everything discussed in this post can be found here:

<https://github.com/webex/Spark-API-Demos/tree/master/EditDeletePeople>

As always, if you have any questions please contact [devsupport@webex.com](mailto:devsupport@webex.com) - we’re happy to help!

Taylor Hanson, Customer Support Engineer II