Using Python with the Admin + Client-Server APIs

You can use Python to make consume and utilise APIs, including those available with Matrix - such as the Synapse Admin API, and the Matrix Client-Server API. See the below docs to learn more about them before progressing with this guide.

- Getting Started Using the Admin API
- Getting Started Using the Client-Server API

The key requirement before progressing is getting the Matrix Accounts' `access_token`, if your using the Synapse Admin API, you must use a Matrix Account which is a Synapse Admin.

Using `python`

You will need Python setup on your system to make use of the script. The best way to use Python is to keep individual projects / scripts in separate virtual environments (`venv`). The documentation on this can be found here, for example on Windows you'd use:

```
python -m venv .\myPythonProject\
.\myPythonProject\Scripts\Activate.ps1
```

The script uses the `requests` library in order to make the API requests, to install in in you `venv`, after activating run:

```
python -m pip install requests
```

You will then be able to run scripts you create by using:

```
python .\myPythonProject\scriptName.py
```

Writing a `python` script

At it's most basic, you will need to setup the below template within your script:

```
import requests


homeseverURL = 'example.com'
accountToken = 'accountTokenStringExample'
requestHeaders = {
    'Authorization': 'Bearer ' + accountToken
}
requestData = {
```

```
    'key': 'value'
}

getResponse = requests.post('API Endpoint URL', headers=requestHeaders).json()
postResponse = requests.post('API Endpoint URL', headers=requestHeaders,
data=requestData).json()
```

Example #1: Join Users to Rooms

Let's say you wanted to join a list of users to a list of rooms, you would adapt this template to something like the below to make use of the Edit Room Membership API:

```
import requests

# Homeserver
homeserverURL = 'example.com'

# Credentials
accountToken = 'accountTokenStringExample'

# Rooms to Auto-Join
roomAliasList = ['#room1:example.com', '#room2:example.com']

# Users to Auto-Join
userList = ['@user1:example.com', '@user2:example.com']

# API Auth Header
requestHeaders = {
    'Authorization': 'Bearer ' + accountToken,
}

# Loop through all rooms
for roomAlias in roomAliasList:
    # Construct API Endpoint URL
    editRoomMembershipURL = 'https://' + homeserverURL + '/_synapse/admin/v1/join/' +
roomAlias
    # Loop through all users
    for user in userList:
      # Construct POST contents
        editRoomMembershipData = {
```

```
            "user_id": user
        }
        # Send Request
        response = requests.post(editRoomMembershipURL, headers=requestHeaders,
data=editRoomMembershipData).json()
```

Example #2: Delete Older Media

If you are looking to remove all media older than a specific Unix Timestamp, you could adjust the template above to make use of Delete local media by date or size.

```
import requests

# Homeserver
homeserverURL = 'example.com'

# Credentials
accountToken = 'accountTokenStringExample'

# API Auth Header
requestHeaders = {
    'Authorization': 'Bearer ' + accountToken,
}

# Construct API Endpoint URL
unixTimestamp : str = '1672531200000'
deleteMediaURL = 'https://' + homeserverURL + '/_synapse/admin/v1/media/delete?before_ts=' +
unixTimestamp

# Send Request
response = requests.post(deleteMediaURL, headers=requestHeaders).json()
```

Example #3: Remove specific `external_ids`

If you are looking to remove specific External IDs from user accounts, you can use the below. Simply replace the `url`, `adminToken` and `authProvider` variables at the top of the script with the desired values. This script makes use of List Accounts to get all users, Query User Account to get each users' `external_ids` and finally Create or Modify Account to remove that specific `external_id`.

```
import requests
import json
```

```
url = 'synapse.example.com'  # Replace with Synapse FQDN
adminToken = 'exampleToken'  # Replace with Admin Token
authProvider = 'exampleAuth' # Replace with Auth Provider to Delete

headers = {
    'Authorization': 'Bearer ' + adminToken,
}
userAccountURL = 'https://' + url + '/_synapse/admin/v2/users'

# GET ALL USER ACCOUNTS
def get_all_users():
    all_users = []
    list_account_from = 0
    users = requests.get(userAccountURL + '?limit=5', headers=headers).json()
    all_users.extend(users['users'])

    while 'next_token' in users.keys():
        list_account_from = int(users['next_token'])
        users = requests.get(userAccountURL + '?from=' + str(list_account_from),
headers=headers).json()
        all_users.extend(users['users'])

    return all_users

# DELETE EXTERNAL ID WITH SPECIFIED AUTH PROVIDER
def delete_matching_external_id(auth_provider, user_accounts):
    for user in user_accounts:
        user_details = requests.get(userAccountURL + '/' + user['name'],
headers=headers).json()
        if 'external_ids' in user_details.keys():
            body = {
                'external_ids': []
            }
            for external_id in user_details['external_ids']:
                if external_id['auth_provider'] != auth_provider:
                    body['external_ids'].append(external_id)
            body_json = json.dumps(body)
            result = requests.put(userAccountURL + '/' + user['name'], headers=headers,
data=body_json).json()
```

```
delete_matching_external_id(authProvider, get_all_users())
```

Revision #11
Created 6 February 2024 14:08:58 by Kieran Mitchell Lane
Updated 15 April 2024 11:10:18 by Kieran Mitchell Lane