**Private Pre-Shared Key Sync with GSuite Groups Guide**

by Tim Smith, SA – 09/19/2022 – v1.0.0

Overview**:**

This guide covers how to set up and run the script to sync your GSuite users with the Private Pre-shared-key (PPSK) within ExtremeCloud IQ (XIQ) public cloud only. PPSK is a solution provided by Extreme Networks to fill the gap between a Wi-Fi SSID solution using a single PSK for all users and deploying a complete 802.1X solution. Extreme Networks PPSK solution allows creating a dedicated key for each user or device on the same SSID, lowering the number of SSIDs broadcasting in the air and minimizing the airtime consumption due to overhead management frames. This solution also adds the ability to assign VLANs based on user/device groups to avoid the need for separate SSIDs to segregate these groups.

This guide enables you to leverage your existing GSuite User groups to automatically create a Private Pre-shared key for every GSuite user and remove the PPSK user if a user is disabled or removed from the group in GSuite.

Each GSuite user must have a unique email address for this script to work correctly.

Target Audience**:** Technical

# 

# PPSK Use Cases:

* Identity for IoT devices
* BYOD for employees
* Staff device onboarding
* Secure Guest Onboarding (time-based keys with employee sponsorship)
* Hospitality vertical using the hyper-segmentation feature, Private Client Groups (PCGs)
* Third-party via API integration

Table of Contents

[Overview 1](#_Toc114757296)

[Target Audience 1](#_Toc114757297)

[PPSK Use Cases: 1](#_Toc114757298)

[Prerequisites: 4](#_Toc114757299)

[Scripting Environment Preparation: 4](#_Toc114757300)

[Information: 4](#_Toc114757301)

[Device Choice: 4](#_Toc114757302)

[Python Installation: 4](#_Toc114757303)

[4](#_Toc114757304)

[Mac OSX Big Sur 5](#_Toc114757305)

[Windows 10 5](#_Toc114757306)

[Required Modules: 5](#_Toc114757307)

[Checking for existing Modules 5](#_Toc114757308)

[Installing required modules 5](#_Toc114757309)

[5](#_Toc114757310)

[Script Variables: 6](#_Toc114757311)

[PCG Support (optional) 6](#_Toc114757312)

[Generating the XIQ Token 7](#_Toc114757313)

[Swagger 7](#_Toc114757314)

[Login 7](#_Toc114757315)

[Authorize in Swagger 8](#_Toc114757316)

[Generating Specific Tokens 9](#_Toc114757317)

[Creating the GSuite Token and Refresh Token 10](#_Toc114757318)

[Creating the GSuite Project 10](#_Toc114757319)

[10](#_Toc114757320)

[Creating the GSuite API OAuth Consent Screen 11](#_Toc114757321)

[Creating the GSuite API OAuth Credentials 11](#_Toc114757322)

[Generating a Token and Refresh token 13](#_Toc114757323)

[GSuite Group Name 14](#_Toc114757324)

[XIQ User Group ID 14](#_Toc114757325)

[XIQ Network Policy ID 15](#_Toc114757326)

[Running the Script: 15](#_Toc114757327)

[Log File 16](#_Toc114757328)

[Scheduling Script to Run 16](#_Toc114757329)

[Mac & Linux-based Systems 16](#_Toc114757330)

[Setting up a Cron Job 16](#_Toc114757331)

[Cron Job Time Format 17](#_Toc114757332)

[Cron Job Script and Script Location 17](#_Toc114757333)

[Cron Job Output and Job Completion 17](#_Toc114757334)

[Cron Job Command Example 18](#_Toc114757335)

Windows-based

[Setting up Windows Task Scheduler 18](#_Toc114757337)

[Start a Program 18](#_Toc114757338)

[Editing the Time 19](#_Toc114757339)

[Troubleshooting: 20](#_Toc114757340)

[Log File 20](#_Toc114757341)

[Invalid XIQ token 20](#_Toc114757342)

[Invalid XIQ token format 20](#_Toc114757343)

[Expired XIQ token – Code 401 & JWT expired 20](#_Toc114757344)

[Invalid XIQ Username/password – Code 401 20](#_Toc114757345)

[Failed Authentication for GSuite API 20](#_Toc114757346)

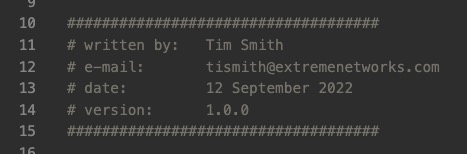
[Invalid or Wrong Group Name entered 20](#_Toc114757347)

[XIQ User Failed to Create – Code 400 21](#_Toc114757348)

[XIQ Timeout Error 21](#_Toc114757349)

[gsuite\_test.py 21](#_Toc114757350)

# Prerequisites:

* ExtremeCloud IQ Public Cloud, Private Cloud (IQVA on-prem is not supported.)  
  o The key directory can be stored in the cloud (unlimited keys) or local on all access points (10,000 key maximum limit)
* Knowledge of XIQ by adding access points, creating network policies, and SSIDs
* XIQ PPSK SSID and associated User Groups configured
* RadSec Proxy requires TCP Port 2083 to be open on your internet firewall
* One or more XIQ native access points
* Not supported on wired systems, A3 NAC, or campus-based Wi-Fi systems (WiNG or IdentiFi)
* Download the following files
  + XIQ-GSuite-PPSK-Sync.py
    - Version 1.0.0 is the current version. See lines 10-15 in the script
  + gsuite\_setup.py
    - Version 1.0.0 is the current version. See lines 10-15 in the script
    - Used to generate token and refresh token – see the [generate token and refresh token](#_Generating_a_Token) section.
  + gsuite\_test.py (optional – see [troubleshooting](#_Troubleshooting:) section)
    - Version 1.0.0 is the current version. Lines 11-16
  + requirements.txt (optional – see [modules](#_Installing_required_modules) section)

# 

# Scripting Environment Preparation:

### Information:

The XIQ-GSuite-PPSK-Sync.py script requires, at minimum, Python 3.6 and tested up to Python 3.9. This script can be executed manually but ideally would be set up as a cronjob to be run every 8, 12, or 24 hours. This script can be executed from any device with python and the needed modules installed. This device will need to be able to reach the GSuite Cloud as well as reach out to ExtremeCloud IQ.

The script, when run, will create a XIQ-GSuite-PPSK-sync.log file. This log file will show information about PPSK users created and deleted. It will also show how many users were parsed from XIQ and the GSuite user group when run. Any API errors experienced will also show up in the log file.

## 

## Device Choice:

This script can be executed from any device that can run python 3.6 or higher. The device could be a server running Redhat, a pc/laptop running Windows 10 or Mac OSX, or even a Raspberry Pi-type device. The device will need to be on the network and be able to reach the GSuite Cloud as well as reach ExtremeCloud IQ. This can be done through a proxy. Proxy config is beyond the scope of this guide.

## 

## Python Installation:

Depending on the device used, you may need to install python or a different version of python. The easiest way to check the version of python is to open the terminal (Power Shell on Windows) and type this command.

### 

python3 --version

Below are some examples of installing python3 for Windows and Mac OSX. Linux systems that were tested all had python3.6 or higher installed by default.

### Mac OSX Big Sur

* Open the terminal and enter python3 –version
  + This triggers the installation of Developer Tools
* Graphical user interface, text

  Description automatically generatedClick Install
* Click Agree
* pip3 is needed to install python modules
* With Big Sur, the Developer tools do not install pip3
* Mac terminal will be used to install pip3
* Running this command will check if pip is installed.

pip3 --version

* Run the following command to install

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py && python3 get-pip.py

### Windows 10

* Search Microsoft store for python 3.9 and click install
* Graphical user interface, text, application, Teams

  Description automatically generatedLog in with Microsoft credentials
* The Windows store installs pip3 with python3. Pip3 will be used to install the needed modules.

## Required Modules:

The **requests, google-api-python-client, google-auth-httplib2, google-auth-oauthlib**

modules are the only module required for the XIQ-GSuite-PPSK-Sync.py script. For Linux and Mac

### Checking for existing Modules

You can check if the required module is installed using the terminal (PowerShell for Windows). Run the following command.

python3 -c “import requests”

The module is not installed if a ‘*ModuleNotFoundError: No module named '<module name>*' error is returned.

### Installing required modules

The required module can be installed using pip3 using the downloaded requirements.txt file with the following command.

## 

pip3 install requests

pip3 install -r requirements.txt

pip3 install google-api-plython-client

Or the module can be installed individually using.

pip3 install google-auth-oauthlib

pip3 install google-auth-httplib2

# Script Variables:

The Global Variable section of the script (file name: XIQ-GSuite-PPSK-Sync.py) will need to be updated with the correct values. We will briefly cover each of these and, for some, will go into more detail below.

**Lines 19** is info for GSuite

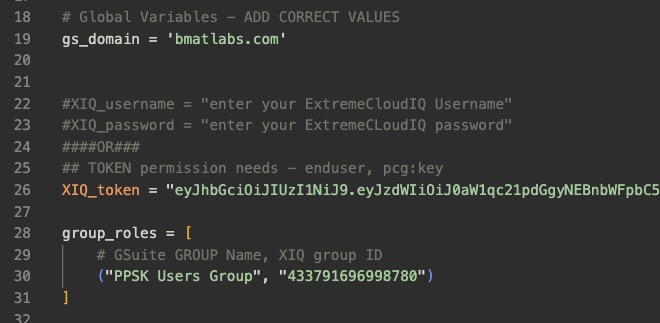
1. **gs\_domain –** The Customer’s domain name in GSuite.

Text

Description automatically generated

**Lines 22 – 26** are for XIQ authentication. 2 methods could be used, but the token method is recommended.

1. XIQ username and password –
   * **Lines 22 and 23** - uncomment by deleting the **#** at the beginning of the line. Then fill in the username and password.
   * **Line 26** - comment out the line by adding # to the beginning of the line.
2. Token method – a token can be generated to only allow access to view/create/delete PPSK users. This is the preferred method. Details on generating this token are below in the [Generating the XIQ Token](#_Generating_the_XIQ_1) and specifically the [Generating Specific Tokens](#_Generating_Specific_Tokens) sub-section.

**Lines 28 – 32** are where we will define which GSuite groups to sync with which XIQ User Groups. Note that the brackets around the groups create a list. Each set of groups should be in a set of parentheses, and a comma should separate each element. A coma should separate each set of groups. If only one set of groups is required, **line 31** can be deleted and the comma at the end of **line 30** removed.

* We will cover how to get the needed GSuite group name in the [GSuite Group Name](#_GSuite_Group_Name) section.
* We will cover how to get the XIQ User Group ID in the [XIQ User Group ID](#_XIQ_User_Group) section.
* **NOTE**: The order is very important here. If the same GSuite user is in multiple groups, the user will be put in the first XIQ User Group in the list. XIQ users can only be in one PPSK User group.

#### PCG Support (optional)

**Line 34** To enable PCG Support, change the **PCG\_Enable** Variable from **False** to **True.**

**Lines 36-42** If PCG is Enabled, **PCG\_Maping** should be updated with correct information. If PCG is not Enabled, **PCG\_Maping** will not be used and does not need to be updated.

1. **Line 37** – This should be replaced with the XIQ User Group ID number that correlates with the PCG. See [XIQ User Group ID](#_XIQ_User_Group) section.
2. **Line 38** – This is the name of the User Group associated with the ID on line 42
   * This is needed to add and remove Users from the PCG.
3. **Line 39** – This is the Network Policy ID associated with the PCG
   * We will cover how to get the Network Policy ID in the [Network Policy ID](#_XIQ_Network_Policy) section
4. **Line 40** – This is the Network Policy Name associated with the PCG

## Generating the XIQ Token

You can view our developer portal site at <https://developer.extremecloudiq.com/> There is a link to our swagger page here and other developer tools. There is also a Communities section to reach out to with any questions.

#### Swagger

We will use the swagger interface to generate the token [https://api.extremecloudiq.com/.](https://api.extremecloudiq.com/)  
On the swagger page, clicking on any API will expand information about the API and give you the ability to try it. Clicking the “Try it out” button, filling out any needed information, and then clicking the execute button will allow you to try that specific API call.

The 2nd generation APIs are based on access tokens generated by a XIQ account. External XIQ accounts need to be granted API to leverage the new APIs. See [A Guide to Getting Started with v2 APIs in XIQ](https://extremeportal.force.com/ExtrArticleDetail?an=000102173) for more details. Currently, these tokens can only be generated through the */login* POST API request. They cannot be generated through the XIQ GUI.

Graphical user interface

Description automatically generated with medium confidence

##### Login

*Request Body*

{  
 "username": "xiq@example.com",  
 "password": "changeme"  
}

The */login* POST request is used to generate an access token. In the request body, enter a local administrator XIQ account username and password, and the API will respond with an access token that can be used for any following calls. This token will be valid for **24 hours** after creation. This token will have the ability to be used for **any** of the API calls the user is authorized for within XIQ.

Graphical user interface, text, application

Description automatically generated

For the purpose of this script, we will use this token to generate a separate token with limited access and a specified expiration time. Copy the access token created, not including the “”s.

##### Authorize in Swagger

At the top of the Swagger page, click the authorize button. A window will pop up, allowing you to paste the access token. Clicking “Authorize” Swagger will set swagger to use the added access token for the API calls on the page.

Graphical user interface, application, Teams

Description automatically generated

##### Generating Specific Tokens



The */auth/apitoken* POST request allows you to specify an expiration time and set permissions for a token. This is a great way to create a token for a specific application or script, only allowing the token to perform the needed tasks.

The expiration time uses Epoch time, which is the number of seconds since midnight on Jan 1, 1970 (UTC). <https://www.epochconverter.com/> is a webpage that can convert a readable time to an epoch time or epoch time to a more readable time. Set a time for 1 year out and get the epoch time.

For this script, we will want to have the following permissions - enduser, pcg:key   
This will give us access to view, create, and delete ppsk users and view, create, and delete pcg-key-based users if necessary.

Adding the desired expiration time and a list of permissions, this API will return a token that will only be usable by the specified APIs.

*Request Body*

{  
 "description": "Token for XIQ-AD-PPSK-Sync.py script",  
 "expire\_time": 1628186428,  
 "permissions": [  
 "enduser”,  
 “pcg:key”  
 ]  
}

*Response Body*

**{**

**"access\_token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ0aW1qc21pdGgyNEBwcm90b25tYWlsLmNvbSIsInNjb3BlcyI6WyJhY2NvdW50OnIiXSwidXNlcklkIjoyMTc5MjMyMSwicm9sZSI6IkFkbWluaXN0cmF0b3IiLCJjdXN0b21lcklkIjoyMTc5MTk3MSwiY3VzdG9tZXJNb2RlIjowLCJoaXFFbmFibGVkIjpmYWxzZSwib3duZXJJZCI6MTc5MTYxLCJvcmdJZCI6MCwiZGF0YUNlbnRlciI6IklBX0dDUCIsImlzcyI6ImV4dHJlbWVjbG91ZGlxLmNvbSIsImlhdCI6MTYyODE4MzA4OSwiZXhwIjoxNjI4MTg2NDI4fQ.CtBGq4YVGB9FzCodr6Oi5IG8yy1-4B-77AWl5rVG3S0",**

**"create\_time": "2021-11-29T15:47:57.000+0000",**

**"expire\_time": "2021-11-29T16:10:08.000+0000",**

**"creator\_id": 21792321,**

**"customer\_id": 21791971,**

**"description": "Token for XIQ-AD-PPSK-Sync.py script",**

**"permissions": [**

**"enduser",**

**"pcg:key"**

**]**

**}**

Copy the newly created access\_token and add it to the XIQ\_token variable in the script.

## Creating the GSuite Token and Refresh Token

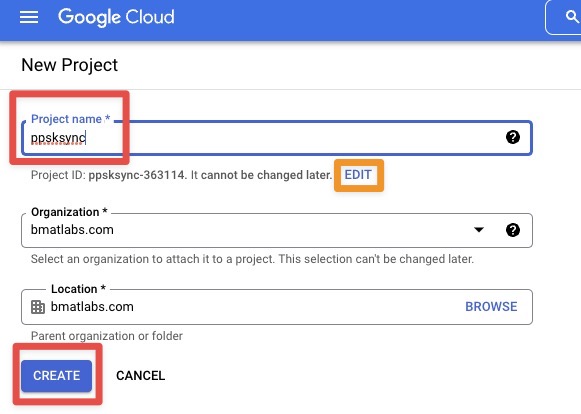
A token will need to be generated every time the script is run in order to access the GSuite user data using API. This token is only valid for 1 hour but can be easily generated within the script using a provided client\_secret.json file and a Refresh token. The next few sections below cover what is needed to allow API access to your GSuite environment, creating the client\_secret.json file, authorizing the script to use the information in the json file, and generating the refresh token.

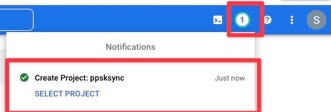
##### Creating the GSuite Project

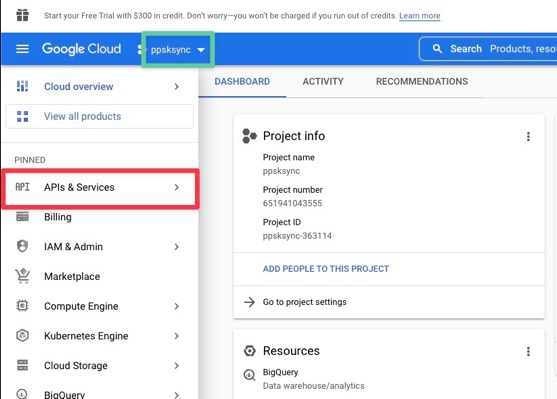
Graphical user interface, website

Description automatically generatedGraphical user interface, application

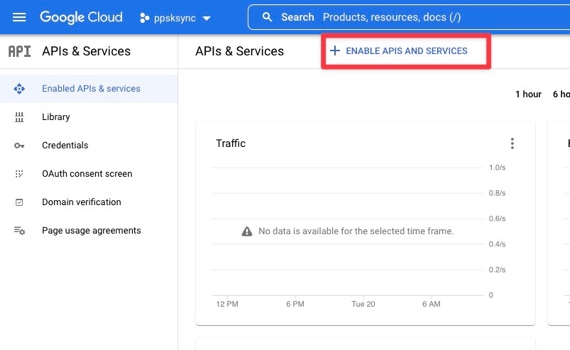
Description automatically generatedTo create the GSuite Project that is needed to access the GSuite user data using API open a browser and go to [console.cloud.google.com](https://console.cloud.google.com/). Once logged in, click the dropdown on the top left. This could have the domain name you are using, another project you’ve created, or may just say “New Project.” Clicking that dropdown will open a new window allowing you to select different projects you have. On the top right of that window is a button to create a new project.

In the newly opened window, set the name of the project you will create. You can optionally edit the Project ID as well if you want. Once completed, click the **Create button.**

The project will take a second to create. You can monitor the notification section to see once it's complete.

Once you see the green check in the notifications, you can open the project. You can do this either by clicking **Select Project** in the notification or by clicking the dropdown and selecting the new project.

With the new project selected, open the APIs & Services tab on the left.

We will first need to enable the Admin SDK API. This will allow API access to groups and users.

Graphical user interface, application

Description automatically generatedSearch for “Admin SDK API” and select the result. This will open a window where you can select to **Enable** the APIs.

## Graphical user interface, application Description automatically generated

##### Creating the GSuite API OAuth Consent Screen

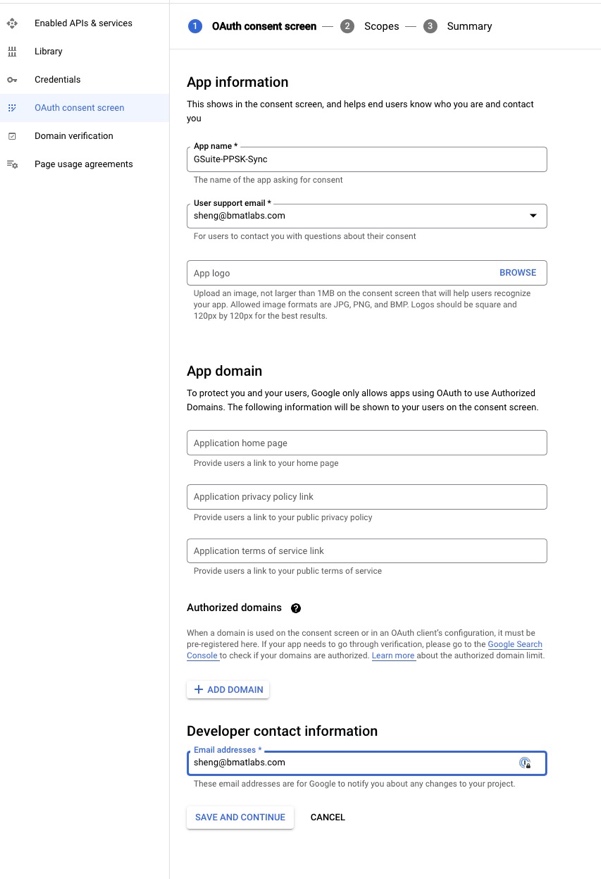
Now that your GSuite project is created, you will need to set up the Consent Screen for OAuth. This will allow you to provide approval for the script to be granted access. This page will just be used when running the gsuite\_setup.py script. See the [Generating GSuite token and refresh token](#_Generating_a_Token) section.

Graphical user interface, application, Teams

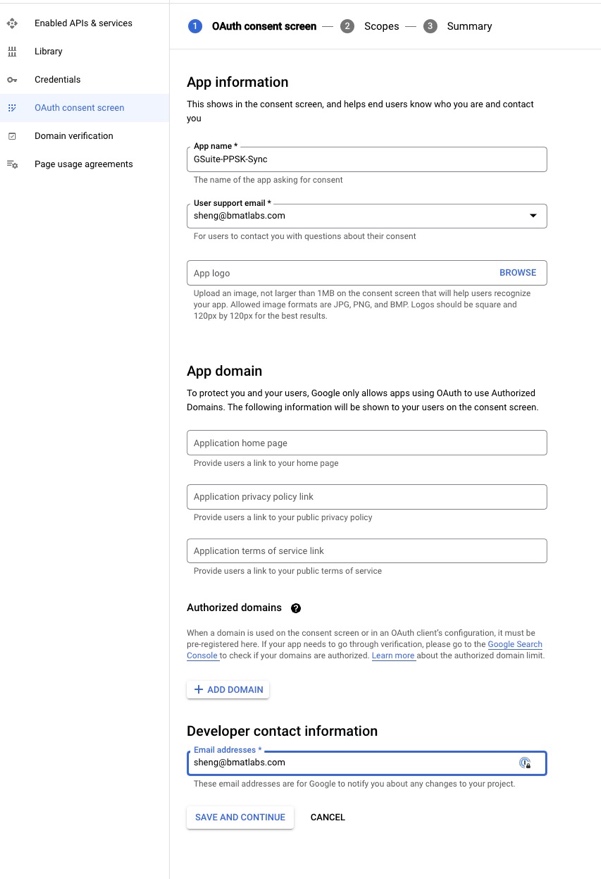
Description automatically generated

On the left of the APIs & Services page, select the **OAuth consent screen** tab, select the **Internal** User Type button, and click **Create.**

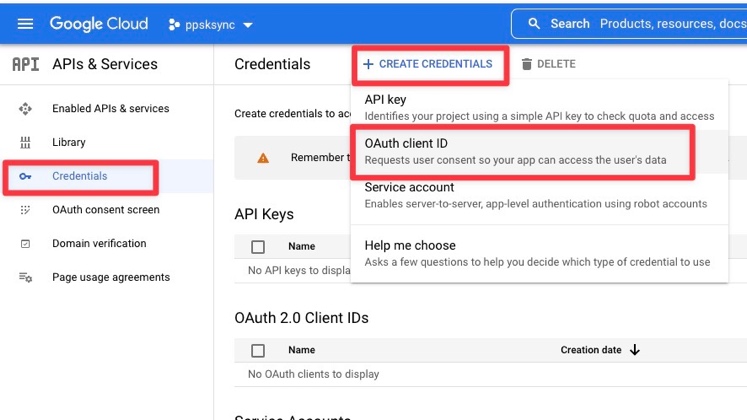
For the App information, fill out the App name, select the user support email and enter an email for the Developer contact information. Everything else on the form can be left blank.



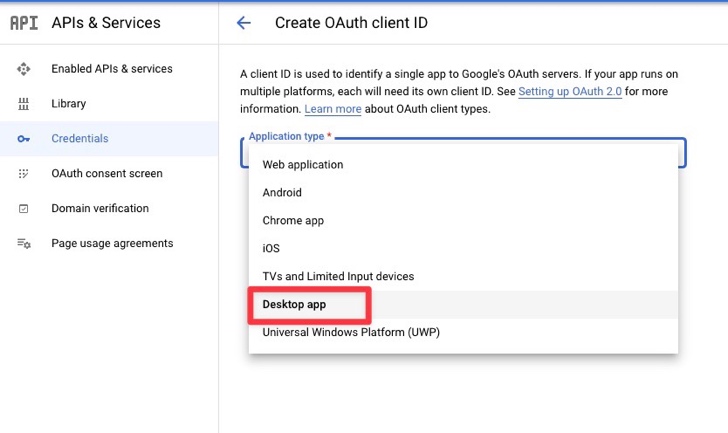
Click the **save and continue** button. On the scopes Page leave everything blank and click **save and continue** button at the bottom. We will be setting the scope for these using the gsuite\_setup.py script.



##### Creating the GSuite API OAuth Credentials

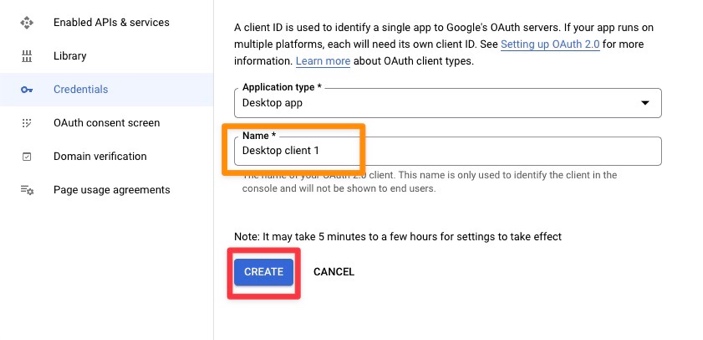
Next, we will need to create the OAuth credentials for the GSuite API.

On the left of the APIs & Services page, select **Credentials** tab. At the top of that page, select **Create Credentials,** and select **OAuth client ID** from the drop down.



When the OAuth client ID opens select the Desktop app for the Application Type.

Then on the next window enter the name you would like for the client and click **Create**.



When the OAuth client is created, select the ‘**Download JSON’** button.

Graphical user interface, text, application, email

Description automatically generated

The JSON file will download to your browsers default location. The file will be named something like “**client\_secret\_{the\_id\_created}pc.apps.googleusercontent.com.json**”. Rename this file to “**client\_secret.json**” and move it to the folder the scripts are located.

##### Generating a Token and Refresh token

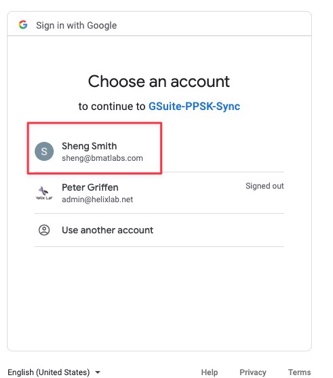
Using the client\_secret.json file, the gsuite\_setup.py script will allow you to authorize the scripts to access your GSuite user data. Ensure the client\_secret.json file has been renamed and added to the same folder as the scripts. You can run the gsuite\_setup.py script in a terminal by going to the location of the scripts and running:

Python3 gsuite\_setup.py

When this script runs, a url will be provided for you to authorize access. A web browser may open on this page as well.

Text

Description automatically generated



With the linked browser page open, select the GSuite user.

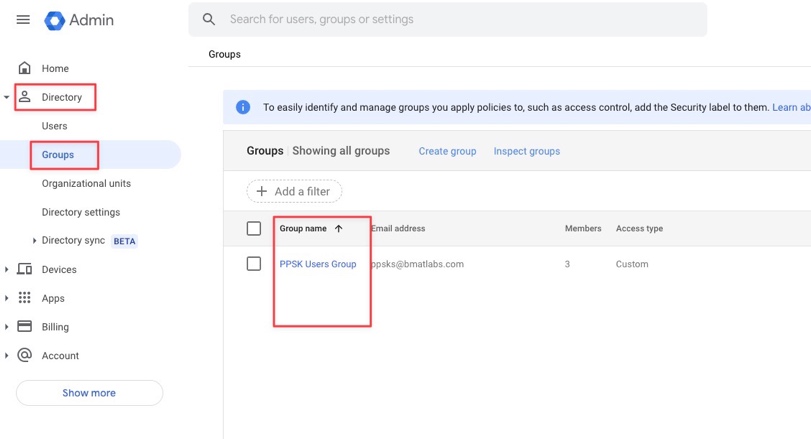
On the next page presented confirm the Project name. You can also see what permissions are requested from the script. As you can see, they are all view only.

Graphical user interface, application

Description automatically generatedOnce validated select Allow.

You can then close the browser history and the script will complete. A new file will be created called gsuite\_token.json. This file has the needed token and refresh token info as well as the client id and client secret. This file will be used in the gsuite\_test.py and the XIQ-GSuite-PPSK-Sync.py scripts to generate a new token to be used in those scripts.

## GSuite Group Name

Each group in GSuite will have a unique name assigned to it. An easy way to get this name is to log into your GSuite Console, go into GSuite Directory on the left, and select Groups. You can can then see all the groups with the Group Name listed in the column

Once the group name is obtained, add it to the group\_roles object in the script.

group\_roles = [

# GSuite GROUP Name, XIQ group ID

("PPSK Users Group", "XIQ group ID"),

("GSuite GROUP Name", "XIQ group ID")

]

## XIQ User Group ID

Each XIQ User Group will be assigned a unique ID when it gets created. This is something that the backend systems use and is not seen in the GUI. The easiest way to get the ID is from the swagger page.

Go back to the swagger page, scroll down to the Configuration – User Management section, and find the */usergroups* GET request.   
Click the “Try it out” button, then the “execute” button. When you find the Name of the XIQ User Group you would like to use, it will be inside a pair of {curly brackets}. Inside the same pair of curly brackets will be an element called **id**. This is the ID that is needed.

*Response Body*

**{**

**"page": 1,**

**"count": 10,**

**"data": [**

**{**

**"id": 769490635824436,**

**"name": "Home\_Hive",**

**"description": "",**

**"predefined": false,**

**"create\_time": "2021-10-11T18:24:33.000+0000",**

**"update\_time": "2021-10-11T18:24:33.000+0000",**

**…**

Once that is obtained, add it to the group\_roles object in the script.

group\_roles = [  
 # GSuite GROUP Name, XIQ group ID  
 ("PPSK Users Group”, "**769490635824436**"),  
 ("GSuite GROUP Name", "XIQ User Group ID")  
]

If needed, continue to gather other GSuite Group Names and XIQ User Group IDs. Enter them in the same format with each set enclosed in parentheses. All but the last one should be followed by a coma.

group\_roles = [  
 # GSuite GROUP Name, XIQ group ID  
 ("d1ba206a-0213-4ae1-95de-0778b366b778", "769490635824438"),  
 ("e7274e17-d57f-4cb3-b3e5-fe7231f952e5", "769490635824436"),  
 ("27a8d84c-f4c4-41e2-a84e-9abdac7c0c13", "769490635824395")  
]

## Graphical user interface, text, application, email Description automatically generatedXIQ Network Policy ID

Each XIQ Network Policy will be assigned a unique ID when it gets created. The easiest way to get the ID is to select the Network Policy in the XIQ GUI. When you select the Network Policy, the ID is the long number listed in the URL. The Policy Name is directly under the Policy Details.

You can also get the Network Policy ID from Swagger for all Network Policies with PCG configured.

Go back to the swagger page, scroll down to the Configuration – User Management section, and find the */pcg/key-based* GET request. If the Network Policy is configured to use PCG, it will be listed in the response. The policy id and policy name are included in this response.



*Response Body*

**[**

**{**

**"id": 1059916324374423,**

**"create\_time": "2021-12-13T15:37:55.000+0000",**

**"update\_time": "2022-01-06T20:29:15.000+0000",**

**"org\_id": 0,**

**"policy\_id": 1059916324374274,**

**"policy\_name": "JB\_Lab",**

**"ssid\_name": "PCG\_Test",**

**"enabled": true,**

**"users": [**

**{**

# Running the Script:

To run the script, open the terminal (PowerShell for Windows) to the location of the script and run:

Python3 XIQ-GSuite-PPSK-Sync.py

chmod +x XIQ-GSuite-PPSK-Sync.py

You can also make the script executable by running

Then you can run the script by typing

./XIQ-GSuite-PPSK-Sync.py

The script will print to the screen how many PPSK users were parsed and how many GSuite users were parsed. If there are any users in the list of GSuite users and not in the list of PPSK users, an API call will be made to create the PPSK user. The script will print on the screen for each user it successfully creates.

successfully created PPSK user Tim Smith

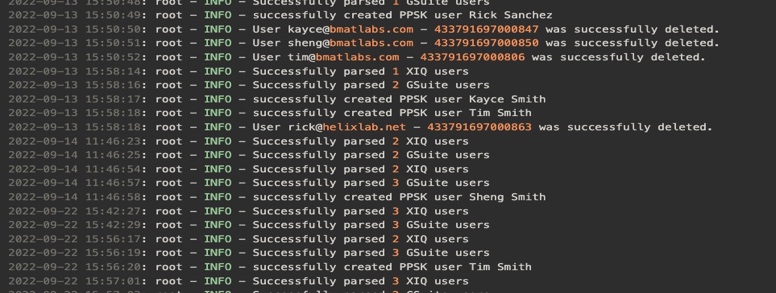
If there are any users in the list of PPSK users that are not in the list of GSuite users or in the list of disabled GSuite users, an API call will be made to delete the PPSK user. The script will print on the screen for each user it successfully deletes. – This will be the email address of the user.

User user0200@example.com - 769490635839948 was successfully deleted.

The GSuite user must have an email address assigned, or the PPSK user will not be created. A message will print on the screen.

User Ruby Smith doesn't have an email set and will not be created in xiq

## Log File

Upon running the script, a log file will be created named XIQ-GSuit-PPSK-Sync.log. Additional runs of the script will append to this log file. This file will contain the same information that prints to the screen and any error received when making the API calls. This is an excellent place to look if issues are seen.

# Scheduling Script to Run

## Mac & Linux-based Systems

A Cron job can be set up to run the script at a specified interval automatically. Ideally, this could be set for every 8, 12, or 24 hours. This would ensure the GSuite and PPSK user groups would stay in sync. The script can also be run manually between those times if a user needs to be added or removed immediately.

### Setting up a Cron Job

Open and edit the crontab and configure the job with arrangement for the command you want to run. From the terminal window, enter the following command.

crontab -e

There are three parts to a cron job configuration.

#### Cron Job Time Format

Part 1 of the cron job

The first five characters, a b c d e, represent the job's time, date, and repetition.

a – Minute (0-59)   
b – Hour (0-23)

c – Day (0-31)

d – Month (0-12) – 0=None and 12 = December

e – Day of the Week (0-7) – 0=Sunday and 7=Sunday

* **An asterisk (\*)** stands for all values. Use this operator to keep tasks running during all months, or all days of the week.
* **A forward-slash (/)** is used to divide a value into steps. (\*/2 would be every other value, \*/3 would be every third, \*/10 would be every tenth, etc.)

The time format would look like this to set the Cron job to run every 8 hours.

0 \*/8 \* \* \*

The time format would look like this to set the Cron job to run every night at midnight.

0 0 \* \* \*

#### Cron Job Script and Script Location

Part 2 of the cron job

The next part is where you enter the script you want to run and its location.

python3 /home/admin/documents/scripts/XIQ-GSuite-PPSK-Sync.py

You can make the script executable, so you don’t have to type python3 before entering the script. Instead, you will just enter a period before the location.

chmod +x /home/admin/documents/scripts/XIQ-GSuite-PPSK-Sync.py

./home/admin/documents/scripts/XIQ-GSuite-PPSK-Sync.py

#### Cron Job Output and Job Completion

Optional Part 3 of the cron job

The last part is an optional part that specifies where the output and completion of the script should go. If not set, the cron will send an email to the owner of the crontab file.

To avoid filling up the inbox on the server, it is recommended to have something set for the output. This can be set to append to a file.

>> /home/admin/documents/scripts/XIQ-GSuite-PPSK-Sync-Output.txt

Or can it be set just turn off the email output

> /dev/null 2>&1

#### 

#### Cron Job Command Example

The command should be entered in a single line and then save the crontab file.

*Every 8 hours turning off the output*

0 \*/8 \* \* \* python3 /home/admin/documents/scripts/XIQ-GSuite-PPSK-Sync.py > /dev/null 2>&1

*Every 12 hours with saved output*

0 \*/12 \* \* \* ./home/admin/documents/scripts/XIQ-GSuite-PPSK-Sync.py >> /home/admin/documents/scripts/XIQ-GSuite-PPSK-Sync-Output.txt

## Windows based Systems

A Windows task schedule can be set up to run the script at a specified interval automatically. Ideally, this could be set for every 8, 12, or 24 hours, ensuring the GSuite group and PPSK user group would stay in sync. The script can also be executed manually between those times if a user needs to be added or removed immediately.

### Setting up Windows Task Scheduler

Open Control Panel > System and Security > Administrative Tools > Task Scheduler

Selected ‘Create basic task…’

Give your task a name like ‘GSuite-PPSK-Sync’ and click ‘Next’

Leave the Trigger set to daily and click ‘Next’ – We will come back and adjust this

Click ‘Next’ leaving recur every one day

Select ‘Start a program and click ‘Next’

#### Start a Program

For the Program/script: section, enter the path of your python.exe file.

Graphical user interface, text, application, email

Description automatically generatedThe location of your python.exe file depends on how it was installed. An easy way to find this location is to open Windows PowerShell and enter python3. This will open the python interpreter. In the interpreter, enter then.

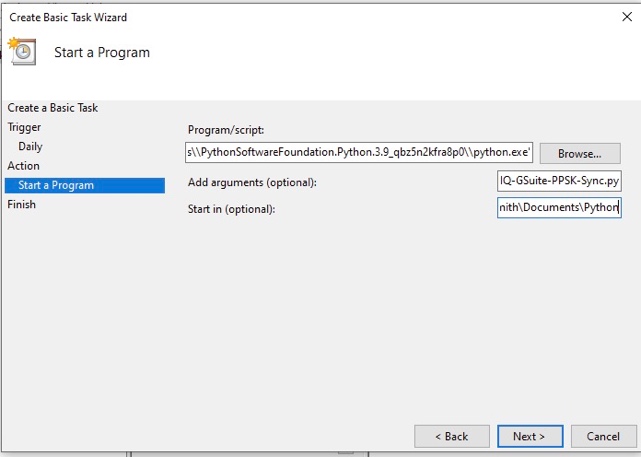
sys.executable

Import sys

exit()

This will output the location of the python.exe file. Enter to exit the interpreter.

Enter the full path of the python.exe file in the Program/Script: field.



Enter the script's name in the Add arguments (optional) field.

XIQ-GSuite-PPSK-Sync.py

Enter the script's location in the Start in the (optional) field.

C:\user\your\_python\_project\_path

#### Editing the Time

Once the task is saved, open the Task Scheduler Library folder and find the newly created GSuite-PPSK-Sync task. Click on it to open, select the Trigger tab, and edit the Daily trigger. Here you can set what time you want it to run.

If you want the script to run every 8 or 12 hours, check the box next to ‘Repeat task every:’ and enter ‘*8 hours’* or *’12 hours’*

Graphical user interface, application

Description automatically generated

# Troubleshooting:

## Log File

The XIQ-GSuite-PPSK-sync.log file is a good place to look for potential issues. This log file will update whenever the script is run, manually or on a schedule.

### Invalid XIQ token

2021-11-05 16:58:27: root - ERROR - Error retrieving PPSK users from XIQ - HTTP Status Code: 401

2021-11-05 16:58:27: root - WARNING - {'error\_code': 'AuthInvalidToken', 'error\_id': 'cda656a5157d4c87a5143252aad71bff', 'error\_message': 'Unable to read JSON value: ?[\x19???\x14?M??'}

Check token using [Swagger](#_Swagger_1) - remember that if you generate a specific token, it may only have access to the user APIs.

### Invalid XIQ token format

2021-11-08 13:56:36: root - ERROR - Error retrieving PPSK users from XIQ - HTTP Status Code: 401

2021-11-08 13:56:36: root - WARNING - {'error\_code': 'AuthInvalidToken', 'error\_id': '555d1ce9f67b40ef83caf4a89ca92b04', 'error\_message': 'JWT strings must contain exactly 2 period characters. Found: 0'}

This may mean that you are trying to use XIQ Username and Password but did not comment out line 27. Add a # in front of **line 27**. Or the token wasn’t entered correctly

### Expired XIQ token – Code 401 & JWT expired

2021-11-08 14:14:16: root - ERROR - Error retrieving PPSK users from XIQ - HTTP Status Code: 401

2021-11-08 14:14:16: root - WARNING - {'error\_code': 'AuthTokenExpired', 'error\_id': '78d8b818a03940dd8d4accfc1b3ffb7e', 'error\_message': 'JWT expired at 2021-11-08T19:14:11Z.

The XIQ token has expired. Generate a new token using [Swagger](#_Swagger_1)

### Invalid XIQ Username/password – Code 401

2021-11-08 13:55:15: root - ERROR - Error getting access token - HTTP Status Code: 401

2021-11-08 13:55:15: root - WARNING - <Response [401]>

Check username and password for XIQ. – I recommend using XIQ Token

### Failed Authentication for GSuite API

2022-09-22 16:23:24: root - ERROR - gsuite\_token.json was not found. Please run the 'gsuite\_setup.py' script to authorize the gsuite API and receive an API token.

Check that the gsuite\_token.json file is in the correct location.

2022-09-22 16:25:44: root - ERROR - Failed to refresh GSuite Token with - ('deleted\_client: The OAuth client was deleted.', {'error': 'deleted\_client', 'error\_description': 'The OAuth client was deleted.'})

The OAuth Credentials have been deleted. Please recreate the [OAuth Credentials](#_Creating_the_GSuite)

2022-09-22 16:33:06: root - ERROR - Request had invalid authentication credentials. Expected OAuth 2 access token, login cookie or other valid authentication credential.

Token may be corrupt. Delete gsuite\_token.json and run the gsuite\_setup.py script again.

### Invalid or Wrong Group Name entered

2022-09-22 16:35:18: root - ERROR - Group 'PPSK Users' was not found in domain bmatlabs.com

Check the group name for the GSuite user group

### XIQ User Failed to Create – Code 400

2021-11-08 14:19:35: root - INFO - Successfully parsed 0 XIQ users

2021-11-08 14:19:36: root - INFO - Successfully parsed 4 LDAP users

2021-11-08 14:19:36: root - ERROR - Error adding PPSK user Tim Smith - HTTP Status Code: 400

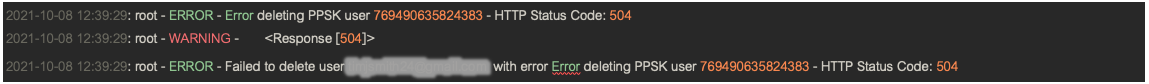
2021-11-08 14:19:36: root - WARNING - {'error\_code': 'UNKNOWN', 'error\_id': None, 'error\_message': 'UNKNOWN'}

2021-11-08 14:19:36: root - ERROR - failed to create Bauer Smith: Error adding PPSK user Tim Smith - HTTP Status Code: 400

There are a couple of possibilities when you see this error. As you can see above, there were 0 XIQ users parsed, and the user failed to create. In this instance, the XIQ User Group ID was incorrect. If XIQ users parsed was 0 and you have configured users in the user group, check the user group ID in the group\_roles list on **Lines 33-37**

The other thing that could cause this Error/Warning is if the username already exists in XIQ PPSK users within a different user group.

### XIQ Timeout Error



2021-10-08 12:43:05: root - ERROR - Failed retrieving PPSK users from XIQ - HTTP Status Code: 504

2021-10-08 12:43:05: root - WARNING - <Response [504]>

An HTTP Status Code: **504** is a timeout from XIQ. If XIQ cannot respond to the API call within 60 secs, it will send these **504** errors. The script will need to be rerun with no changes.

## gsuite\_test.py

This script was written to help troubleshoot issues with collecting data from GSuite. The variables are the same in this script, except that the Gsuite group name should be added to the **gs\_group\_name** variable instead of in the group\_roles list.

This script will test the GSuite API. If there are any errors, they will print to the screen. Otherwise, the collected data from the GSuite cloud will print on the screen. If empty [] brackets print, check the [Group Name](#_AD_Group_Object) and validate that it is correct.

Using this output, you can check whether the user is disabled in GSuite. You can validate that there is an email set for the user. And overall, check that information is being returned.

Kayce Smith {'accountEnabled': False, 'email': 'kayce@bmatlabs.com', 'username': 'kayce@bmatlabs.com'}

Sheng Smith {'accountEnabled': True, 'email': 'sheng@bmatlabs.com', 'username': 'sheng@bmatlabs.com'}

Tim Smith {'accountEnabled': True, 'email': 'tim@bmatlabs.com', 'username': 'tim@bmatlabs.com'}