**Private Pre-Shared Key Sync with Active Directory Guide**

by Tim Smith, SA – 05/16/2025 – v1.5.0

Overview**:**

This guide covers setting up and running the script to sync your local domain Active directory 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 in 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 the creation of a dedicated key for each user or device on the identical SSID, limiting the number of SSIDs broadcasting in the air and minimizing 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 Active Directory security groups to automatically create a Private Pre-shared key for every AD user and remove the PPSK user if a user is disabled or removed from the group in the AD server.

Each AD 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](#_Toc198280698)

[Target Audience 1](#_Toc198280699)

[PPSK Use Cases: 1](#_Toc198280700)

[Prerequisites: 4](#_Toc198280701)

[Scripting Environment Preparation: 4](#_Toc198280702)

[Information: 4](#_Toc198280703)

[Device Choice: 4](#_Toc198280704)

[Python Installation: 4](#_Toc198280705)

[4](#_Toc198280706)

[Mac OSX Sequoia 5](#_Toc198280707)

[Windows 11 5](#_Toc198280708)

[Required Modules: 5](#_Toc198280709)

[Checking for existing Modules 5](#_Toc198280710)

[Installing required modules 5](#_Toc198280711)

[5](#_Toc198280712)

[Script Variables: 6](#_Toc198280713)

[PCG Support (optional) 6](#_Toc198280714)

[Generating the XIQ Token 7](#_Toc198280715)

[Swagger 7](#_Toc198280716)

[Login 8](#_Toc198280717)

[Authorize in Swagger 8](#_Toc198280718)

[Generating Specific Tokens 9](#_Toc198280719)

[AD Group Distinguished Name 10](#_Toc198280720)

[XIQ User Group ID 10](#_Toc198280721)

[XIQ Network Policy ID 11](#_Toc198280722)

[AD Filter 11](#_Toc198280723)

[Active Directory Disable Codes 12](#_Toc198280724)

[Running the Script: 13](#_Toc198280725)

[Log File 13](#_Toc198280726)

[Scheduling Script to Run 13](#_Toc198280727)

[Mac & Linux-based Systems 13](#_Toc198280728)

[Setting up a Cron Job 13](#_Toc198280729)

[Cron Job Time Format 14](#_Toc198280730)

[Cron Job Script and Script Location 14](#_Toc198280731)

[Cron Job Output and Job Completion 14](#_Toc198280732)

[Cron Job Command Example 15](#_Toc198280733)

[Windows based Systems 15](#_Toc198280734)

[Setting up Windows Task Scheduler 15](#_Toc198280735)

[Start a Program 15](#_Toc198280736)

[Editing the Time 16](#_Toc198280737)

[Troubleshooting: 17](#_Toc198280738)

[Log File 17](#_Toc198280739)

[Invalid XIQ token 17](#_Toc198280740)

[Invalid XIQ token format 17](#_Toc198280741)

[Expired XIQ token – Code 401 & JWT expired 17](#_Toc198280742)

[Invalid XIQ Username/password – Code 401 17](#_Toc198280743)

[Unable to reach server (Active Directory) 17](#_Toc198280744)

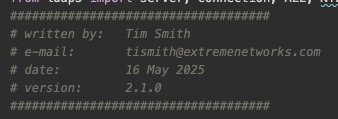
[XIQ User Failed to Create – Code 400 17](#_Toc198280745)

[XIQ Timeout Error 18](#_Toc198280746)

[Email is not set for AD User 18](#_Toc198280747)

[AD\_Test.py 18](#_Toc198280748)

# Prerequisites:

* ExtremeCloud IQ Public Cloud, Private Cloud (IQVA on-prem is not supported.)
* The key directory can be stored in the cloud (unlimited keys) or locally 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-AD-PPSK-Sync.py
    - Version 2.1.0 is the current version. See lines 8-13 in the script
  + AD\_Test.py (optional – see [troubleshooting](#_AD_Test.py) section)
    - Version 2.0.4 is the current version. Lines 5-10
  + requirements.txt (optional – see [modules](#_Installing_required_modules) section)

# 

# Scripting Environment Preparation:

### **Information**:

The XIQ-AD-PPSK-Sync.py script requires, at minimum, Python 3.6 and tested up to Python 3.12. 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 must reach the Active Directory server and access ExtremeCloud IQ.

The script, when run, will create an XIQ-AD-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 Active Directory when run. Any API errors experienced will also show up in the log file.

## 

## Device Choice:

This script can be executed from any device running 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 local Active Directory server as well as reach ExtremeCloud IQ. This can be done through a proxy. The proxy configuration is beyond the scope of this guide.

## 

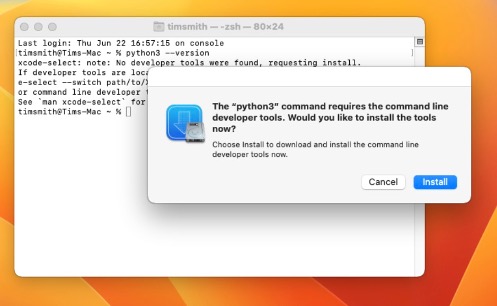
## Python Installation:

Depending on the device that is used, you may need to install Python or a different version of Python. The easiest way to check the Python version 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 Sequoia

* Open the terminal and enter python3 –version
  + This triggers the installation of Developer Tools
* Click Install
* Click Agree
* The Developer Tools that installs python3 will also install pip3
* Mac terminal will be used to install python modules



### Windows 11

* Search Microsoft Store for Python 3.11 and click install
* Log in with Microsoft credentials
* The Windows store installs pip3 with python3. Pip3 will be used to install the needed modules
* Windows power shell or command prompt can be used to install Python modules

## Required Modules:

The **requests**, **ldap3,** and **pycryptodome** modules are the only modules required for the XIQ-AD-PPSK-Sync.py script.

### Checking for existing Modules

You can check if the required modules are installed using the terminal (PowerShell for Windows). For each module, run the following command.

python3 -c “import requests”

python3 -c “import ldap3”

python3 -c “import pycryptodome”

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

### Installing required modules

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

## 

pip3 install -r requirements.txt

Or the modules can be installed individually using

pip3 install requests

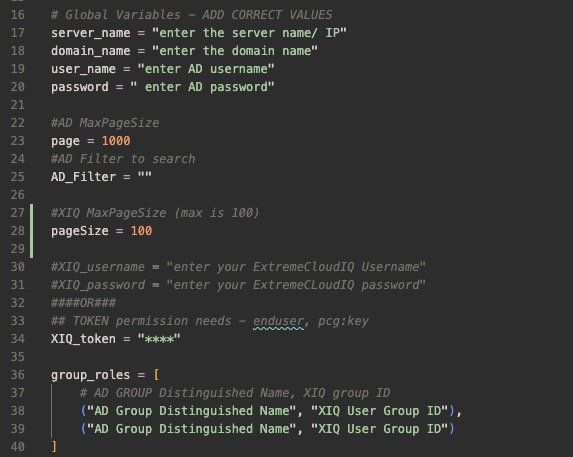
pip3 install pycryptodome

pip3 install ldap3

# Script Variables:

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

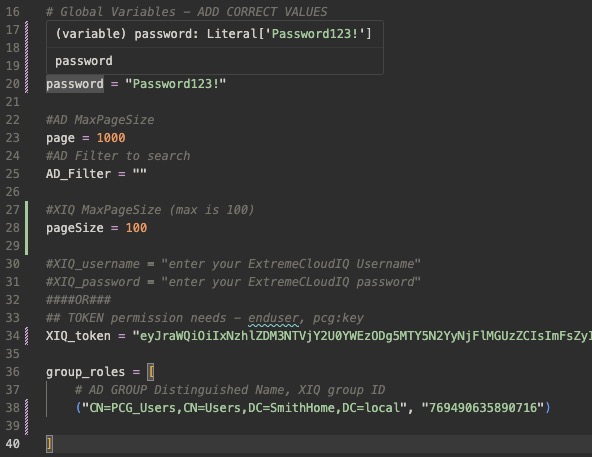
**Lines 17 – 20** are info for the Active Directory Server

1. **server\_name** – This can be the FQDN or IP address of the Active Directory Server
2. **domain\_name** – The configured Domain on the AD server – The domain portion of the FQDN

**Lines 23** can change the Max page size with the AD call. This should match your AD configuration. 1000 is the default

**Line 25** can be used to add a filter to your AD search. See the [AD Filter](#_AD_Filter_1) section for more details.

**Line 28** is the number of PPSK and PCG users to be returned per API call.

**Lines 30 – 34** are for XIQ authentication. 2 methods could be used, but the token method is recommended.

1. XIQ username and password –
   * **Lines 30 and 31** - uncomment by deleting the **#** at the beginning of the line. Then, fill in the username and password
   * **Line 34** - comment out the line by adding # to the beginning of the line.
2. Token method – A token can be generated to allow access only 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_1) sub-section.

**Lines 36 – 40** are where we will define which AD 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 be separated by a comma. A coma should separate each set of groups. If only one set of groups is required, **line 39** can be deleted, and the comma at the end of line 35 can beremoved.

* We will cover how to get the needed AD Group distinguished Name in the [AD Group Distinguished Name](#_AD_Group_Distinguished_1) 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 AD 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 42** To enable PCG Support, change the **PCG\_Enable** Variable from **False** to **True**

**Lines 44-50** If PCG is Enabled, **PCG\_Mapping** should be updated with the correct information. If PCG is not Enabled, **PCG\_Mapping** will not be used and does not need to be updated.

1. **Line 45** – 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 46** – 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 47** – 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 48** – 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 and other developer tools. There is also a Communities section to reach out with any questions.

#### Swagger

We will use the swagger interface to generate the token <https://api.extremecloudiq.com/>  
On the swagger page, clicking on any API will expand information about the API and allow you 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 an XIQ account. 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 of the 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 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” 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, 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 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.

By adding the desired expiration time and a list of permissions, this API will return a token that is only 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.

## 

## AD Group Distinguished Name

The distinguished name includes more details than just the name of the group. It consists of any OUs or folders under which the AD group and the domain controllers are located. These are all needed by the script to identify and query the group details.Graphical user interface, application, Word

Description automatically generated

An easy way to get the Distinguished name is to use the find objects and search for the group name. The full distinguished name will be shown in the Search results.

Some special characters must be escaped if included in the Distinguished name. For example, If you had a CN like *CN=Users (global),* the ()’s would need to be escaped out and converted to hex-like *CN=Users \\28global\\29*. More information can be found here - [characters to escape](https://social.technet.microsoft.com/wiki/contents/articles/5312.active-directory-characters-to-escape.aspx)

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

group\_roles = [  
 # AD GROUP Distinguished Name, XIQ group ID  
 ("CN=Staff\_User,CN=Users,DC=SmithHome,DC=local", "XIQ User Group ID"),  
 ("AD Group Distinguished Name", "XIQ User Group ID")  
]

## XIQ User Group ID

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

Return to the swagger page, scroll 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 want 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 = [  
 # AD GROUP Distinguished Name, XIQ group ID  
 ("CN=Staff\_User,CN=Users,DC=SmithHome,DC=local", "**769490635824436**"),  
 ("AD Group Distinguished Name", "XIQ User Group ID")  
]

If needed, continue to gather other AD Group Distinguished 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 = [  
 # AD GROUP Distinguished Name, XIQ group ID  
 ("CN=Staff\_User,CN=Users,DC=SmithHome,DC=local", "769490635824438"),  
 ("CN=Testing,OU=Sub1,OU=Special,DC=Smithhome,DC=local", "769490635824436"),  
 ("CN=WIFI\_admins,CN=Users,DC=SmithHome,DC=local", "769490635824395")  
]

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

Each XIQ Network Policy will be assigned a unique ID when created. The easiest way to get the ID is to select the Network Policy in the XIQ GUI. When you choose 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.

Return to the swagger page, scroll 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": [**

**{**

## AD Filter

Adding an AD Filter will allow the search filter to be performed in AD. This can be beneficial if there are users in the security group that do not have a corporate email address or if you want to filter out particular email addresses.

For example, this would search only @example.org and @stu.example.org email addresses. All others in the security group would be filtered out.

AD\_Filter = "(|(mail=\*@example.org)(mail=\*@stu.example.org))"

More information about filtering can be found on the LDAP Filtering website.

## Active Directory Disable Codes

Each user in Active Directory has a userAccountControl number assigned to it, providing the user's status. Information on this number can be seen here: <https://docs.microsoft.com/en-us/troubleshoot/windows-server/identity/useraccountcontrol-manipulate-account-properties>

The script has 4 codes for Users who are disabled. Depending on the use case, other numbers may need to be added.

* 514 – NORMAL\_ACCOUNT (512) + ACCOUNTDISABLED (2)
* 642 - NORMAL\_ACCOUNT (512) + ACCOUNTDISABLED (2) + ENCRYPTED\_TEXT\_PWD\_ALLOWED (128)
* 66050 - NORMAL\_ACCOUNT (512) + ACCOUNTDISABLED (2) + DONT\_EXPIRE\_PASSWORD (65536)
* 66178 - NORMAL\_ACCOUNT (512) + ACCOUNTDISABLED (2) + DONT\_EXPIRE\_PASSWORD (65536) + ENCRYPTED\_TEXT\_PWD\_ALLOWED (128)

If any other disabled codes are needed, they can be added inside the bracket to **line 61**.

ldap\_disable\_codes = ['514','642','66050','66178',’2562’]

If disabled users are not being removed, you can use the AD\_Test.py script and see details about the users in the AD group. You can look at the ‘*userAccountControl*’ element to know the number associated with the disabled user. This number may need to be added to the ‘*ldap\_disable\_codes*’ list.

See the [Troubleshooting](#_AD_Test.py) Section for more info on the AD\_Test.py script.

The script variables should now be completed.

# Running the Script:

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

Python3 XIQ-AD-PPSK-Sync.py

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

You can also make the script executable by running

Then, you can run the script by typing

./XIQ-AD-PPSK-Sync.py

The script will print to the screen how many PPSK users and AD users were parsed. If there are any users in the list of AD 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 AD users or disabled AD 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 AD user must have an email address assigned, or the PPSK user will not be created. A message will print on the screen.

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

## Text Description automatically generatedLog File

Upon running the script, a log file will be created named XIQ-AD-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 a good 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 automatically at a specified interval. Ideally, this could be set for every 8, 12, or 24 hours. This would ensure the AD 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 the arrangement for the command you want to run. From the terminal window, enter the following command.

crontab -e

There are 3 parts to a cron job configuration.

#### Cron Job Time Format

Part 1 of the cron job

The first 5 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.)

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

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-AD-PPSK-Sync.py

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

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

./home/admin/documents/scripts/XIQ-AD-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 specified, 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-AD-PPSK-Sync-Output.txt

Or it can be set to turn off the email output.

> /dev/null 2>&1

#### Cron Job Command Example

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

*Every 8 hours, turning off the output*

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

*Every 12 hours with saved output*

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

## Windows based Systems

A Windows task schedule can be set up to run the script automatically at a specified interval. Ideally, this should be set for every 8, 12, or 24 hours to ensure the AD and PPSK user groups 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 ‘AD-PPSK-Sync’ and click ‘Next.’

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

Click ‘Next’ leaving recur every 1 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.

Graphical user interface, text, application

Description automatically generated

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

XIQ-AD-PPSK-Sync.py

Enter the script's location in the Start in (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 AD-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-AD-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’s 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 the XIQ Username and Password but have yet to comment out line 34. Add a # in front of **line 34**. 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. – it is recommended to use an XIQ Token

### Unable to reach server (Active Directory)

2021-11-08 13:46:07: root - ERROR - Unable to reach server DADOH-D.SmithHome.local

Check the IP address or server name entered on **Line 17.** This may need to be the fully qualified name. Try pinging the server from the device where the script is hosted.

### 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

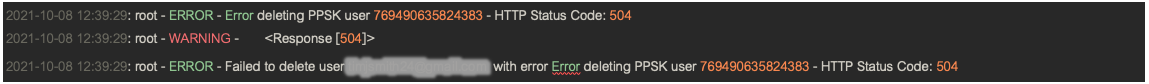
2021-11-08 14:19:36: root - WARNING - {'error\_code': 'UNKNOWN', 'error\_id': None, 'error\_message': 'UNKNOWN'}

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, 0 XIQ users were parsed, and the user failed to create. In this instance, the XIQ User Group ID needed to be corrected. If the 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 36-40**

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 seconds, it will send these **504** errors. The script will need to be rerun with no changes.

### Email is not set for AD User

2021-11-08 14:42:39: root - WARNING - User Tim Smith doesn't have an email set and will not be created in xiq

Check the AD User and see if the email is set. Email is required to create a PPSK.

## AD\_Test.py

This script was written to help troubleshoot issues with collecting data from AD. The variables are the same in this script. The AD Group Distinguished name should be added to the **Distinguished\_name** variable instead of in the group\_roles list.

AD\_Filter can be assigned here as well. This can help to see info on a specific email or user.

This script will test resolving the IP address for the server\_name. If an IP address is entered, it will try and resolve the DNS name. There should be no issue with the DNS name not resolving if the server\_name is set to the IP address. It's just more informational.

The connection to the AD Server will be performed.

If there are any errors, they will print to the screen. Otherwise, the collected data from the AD server will print on the screen. If empty [] brackets are printed, check the [distinguished name](#_AD_Group_Distinguished_1) and validate that it is correct.

Using this output, you can check the [User Account Control Number](#_Active_Directory_Disable_1) if disabled users are not being removed from XIQ. You can validate there is an email set for the user. And overall, check that information is being returned.

The IP address for DADOH-DC.SmithHome.local is 192.168.10.5

completed page of AD Users. Total Users collected is 2

User0001 Last {'userAccountControl': '512', 'email': 'user0001@example.com', 'username': 'user0001'}

User0002 Last {'userAccountControl': '512', 'email': '[]', 'username': 'user0002'}