Quantinuum Systems Troubleshooting Guide

Version 1.10 May 9, 2023

TABLE OF CONTENTS

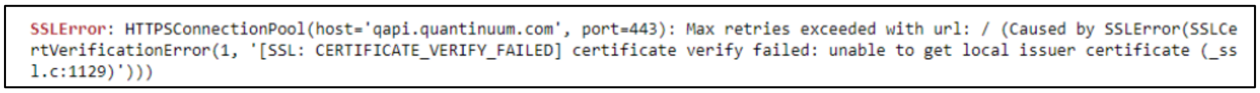
INTRODUCTION

The **Quantinuum System Model H1, powered by Honeywell**, is the best-in-class trapped ion quantum computer. We are delighted that you've decided to explore its use for your development of quantum algorithms and solutions. This troubleshooting guide covers common scenarios and issues you may encounter during development as well as steps to resolve them.

LOGGING IN

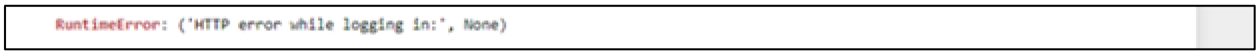
Windows Users

If you are a Windows user, it is possible when you go to use the Quantinuum API that you will receive an error related to your certificate failing. This could be an SSL Error, Runtime HTTP error, or HTTPS Connection Pool error. Examples are below. These may not represent all possible error related to this issue.



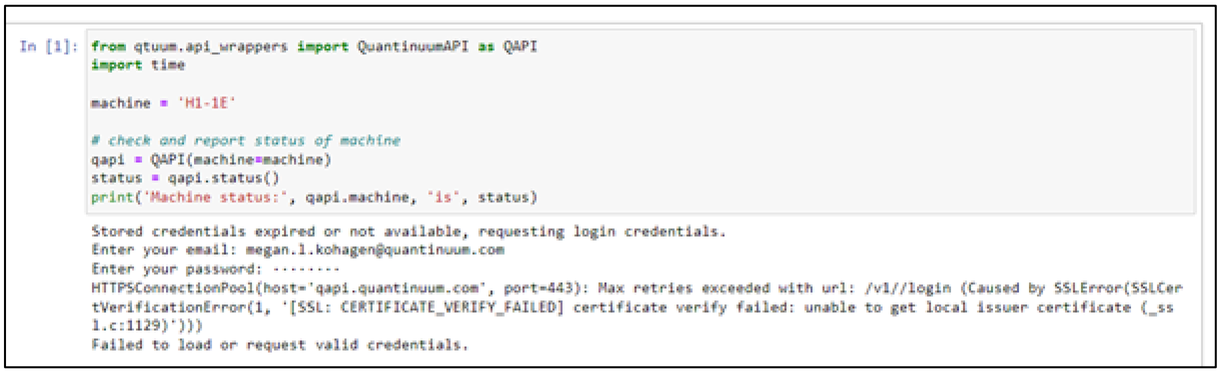
```
SSLError: HTTPSConnectionPool(host='qapi.quantinuum.com', port=443): Max retries exceeded with url: / (Caused by SSLError(SSLCe
rtVerificationError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certificate (_ss
l.c:1129)')))
```

Figure 1 SSLError



```
RuntimeError: ('HTTP error while logging in:', None)
```

Figure 2 Runtime Error



```
In [1]: from qtuum.api_wrappers import QuantinuumAPI as QAPI
        import time

        machine = 'H1-1E'

        # check and report status of machine
        qapi = QAPI(machine=machine)
        status = qapi.status()
        print('Machine status:', qapi.machine, 'is', status)

        Stored credentials expired or not available, requesting login credentials.
        Enter your email: megan.l.kohagen@quantinuum.com
        Enter your password: ········
        HTTPSConnectionPool(host='qapi.quantinuum.com', port=443): Max retries exceeded with url: /v1//login (Caused by SSLError(SSLCer
        tVerificationError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certificate (_ss
        l.c:1129)')))
        Failed to load or request valid credentials.
```

Figure 3 HTTPSConnectionPool Error

These errors occur since the quantinuum.com domain is relatively new and hasn't worked its way into the standard Windows certificate verification. In the future this won't be needed, but in the meantime, install the package `python-certifi-win32` in your python environment. This package has the latest certificate verification for Windows. Replace python3 with whatever python interpreter you have.

```
python3 -mpip install python-certifi-win32
```

Keyring

If using the OpenQASM method of connecting to the Quantinuum systems and the `api_wrappers.py` functions in the `qtuum` folder on the User Portal, you may encounter errors around logging in due to the `keyring` module. Errors may state the system is unable to retrieve an id token or login. An example of the type of error you may see is in Figure 4.

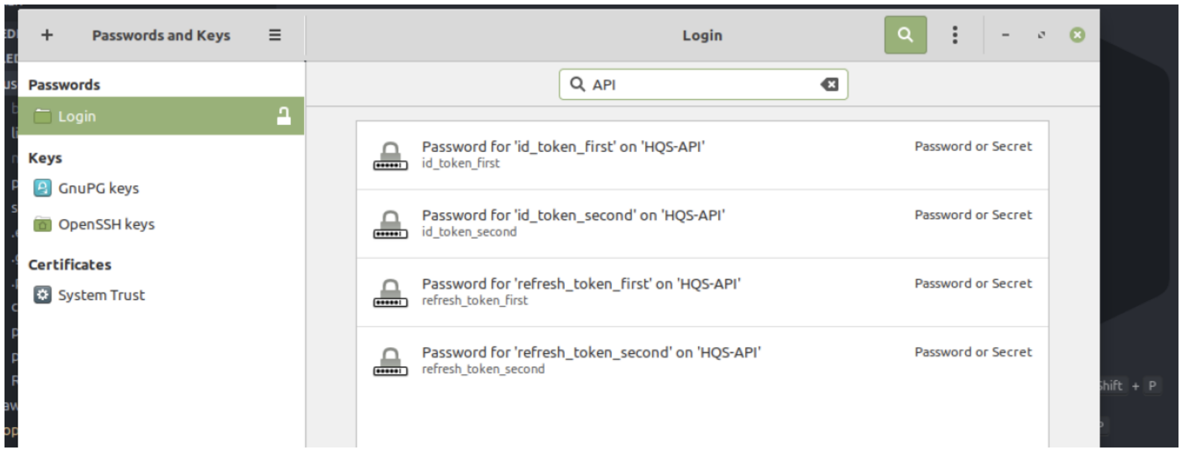
```
pytket.extensions.quantinuum.backends.api_wrappers.QuantinuumAPIError: Unable to retrieve id
token or refresh or login.
```

Figure 4 Login Error

The methods provided utilize the python package `keyring` to provide a way to store your username and password so you don't need to enter your credentials each time you access the Quantinuum API. The module is used to get and save your password information for a set period of time. If you go to login after the expiration timestamp, you will be prompted to enter your username and password again. For more information, see the keyring docs.

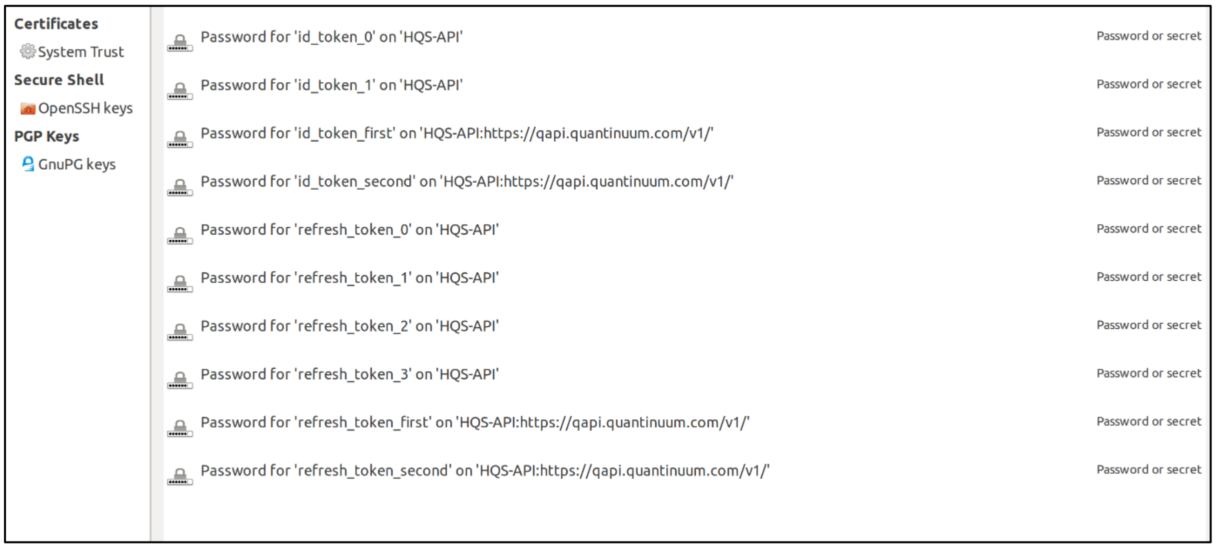
Issues with `keyring` occur more frequently on Linux/Ubuntu systems. On Ubuntu, the python modules `secretstorage` and `dbus-python` are recommended to install in your python environment. See the `keyring` Ubuntu documentation for more information.

As a troubleshooting step, you may want to check your password manager to see what tokens are being stored. There should be 4 tokens, 2 for an id_token, 2 for a refresh_token, as seen in Figure 5. Figure 6 shows an example with extra tokens stored due to this user having used the old HQS API.



Password for 'id_token_first' on 'HQS-API' — id_token_first — Password or Secret
Password for 'id_token_second' on 'HQS-API' — id_token_second — Password or Secret
Password for 'refresh_token_first' on 'HQS-API' — refresh_token_first — Password or Secret
Password for 'refresh_token_second' on 'HQS-API' — refresh_token_second — Password or Secret

Figure 5 Password Manager on Linux showing the correct number of tokens



Password for 'id_token_0' on 'HQS-API' Password or secret
Password for 'id_token_1' on 'HQS-API' Password or secret
Password for 'id_token_first' on 'HQS-API:https://qapi.quantinuum.com/v1/' Password or secret
Password for 'id_token_second' on 'HQS-API:https://qapi.quantinuum.com/v1/' Password or secret
Password for 'refresh_token_0' on 'HQS-API' Password or secret
Password for 'refresh_token_1' on 'HQS-API' Password or secret
Password for 'refresh_token_2' on 'HQS-API' Password or secret
Password for 'refresh_token_3' on 'HQS-API' Password or secret
Password for 'refresh_token_first' on 'HQS-API:https://qapi.quantinuum.com/v1/' Password or secret
Password for 'refresh_token_second' on 'HQS-API:https://qapi.quantinuum.com/v1/' Password or secret

Figure 6 Extra Tokens in Password Manager due to use of old HQS API

VPNs

If you have trouble logging in and your workplace utilizes a Virtual Private Network (VPN), it is possible this may interfere with your connection to the API. The VPN may attempt to pretend it is quantinuum.com and interferes with your certificate chain. You will not be able to sign in or access Quantinuum services. If you believe this is the issue, work with your VPN provider to fix this issue. This requires SSL and cert commands you may not be able to implement.

If you're a Windows user, ensure that you have first installed `python-certifi-win32` in your python environment to check this was not the issue.

RESULTS

Attempting to understand why a job failed or why results aren't what was expected can take a lot of time and effort to debug. We outline the practical steps to take when working with Quantinuum Systems. As a reminder of what is outlined in the Quantinuum Systems User Guide and code examples, the workflow for submitting on Quantinuum Systems should follow this pattern.

1. *Submit to Syntax Checker:* To avoid time submitting to a quantum computer and waiting in the queue only to find the job failed, use the Syntax Checker first. This is beneficial regardless of whether you plan to use an emulator or quantum computer. Syntax Checkers are online 24/7 and if the circuit has any syntax errors, the error is returned with an indication of what caused the circuit to fail. Once successful, the Syntax Checker also returns the cost in H-System Quantum Credits (HQCs).
2. *Submit to an Emulator:* Once a circuit has been checked with the Syntax Checker, next submit to the Emulator. Quantinuum Emulators are also online 24/7 so queue time is generally minimal. This will allow you to run your circuit and check whether results are what you expect and identify potential circuit design errors on your part before submitting to a quantum computer.

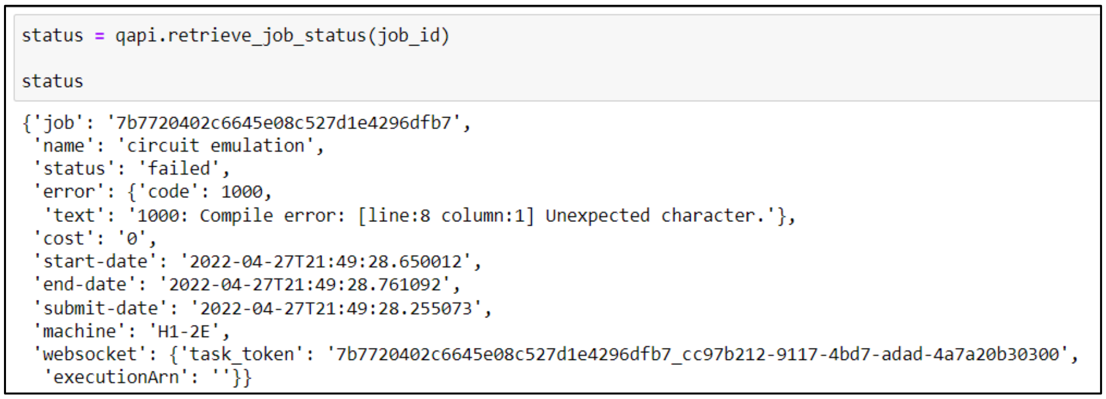
3. *Submit to a Quantum Computer:* This is the fun part. Once you're sure your circuit will run and results are looking like what you expect to get back, submit to a quantum computer!

Check Available Credits

At any point you can also use Syntax Checker results to double check against the number of credits you have remaining. To look at how many credits you have remaining for the month, navigate to the Quantinuum User Portal and click the **Account** tab. Under *Machines*, the number of credits remaining is listed for each device type (quantum computer, emulator, syntax checker).

My Circuit Failed

When you run a circuit, it is possible that you receive a failed status upon completion. The error you receive may look something like Figure 7. The 1000 error returned is a compilation error related to circuit syntax being incorrect. These types of errors can be quickly discovered and fixed by using the System Model H1 Syntax Checkers, `H1-1SC` or `H1-2SC`. Details on the functionality of the Syntax Checkers are found in the *Quantinuum Systems User Guide*. Details on the error codes are found in the *Quantinuum Systems API Specification*. These documents as well as example code for using the Syntax Checkers are found on the Quantinuum User Portal under the **Examples** tab.



```
status = qapi.retrieve_job_status(job_id)

status
```

```
{'job': '7b7720402c6645e08c527d1e4296dfb7',
 'name': 'circuit emulation',
 'status': 'failed',
 'error': {'code': 1000,
  'text': '1000: Compile error: [line:8 column:1] Unexpected character.'},
 'cost': '0',
 'start-date': '2022-04-27T21:49:28.650012',
 'end-date': '2022-04-27T21:49:28.761092',
 'submit-date': '2022-04-27T21:49:28.255073',
 'machine': 'H1-2E',
 'websocket': {'task_token': '7b7720402c6645e08c527d1e4296dfb7_cc97b212-9117-4bd7-adad-4a7a20b30300',
  'executionArn': ''}}
```

Figure 7 Failed Status after Circuit Submission

Results Aren't What I Expected

After submitting your circuit to a Quantinuum quantum computer it is possible that the end results you obtain are not what you expect. As you improve your ability to write and submit complex quantum circuits, it is not surprising that results can be puzzling. At first glance it is easy to think the quantum computer is wrong. The quantum computer only works as well as what it is given. Software code may run and be submitted, but this doesn't catch user errors in circuit design. After submitting the circuit to the Syntax Checker to ensure the circuit passes, we recommend submitting to a Quantinuum emulator. Quantinuum emulators are online 24/7 so queue time is generally minimal. This will allow you to retrieve results in minimal time, without needing to submit and wait for results from a quantum computer.

In addition, remember that results are given in little-endian format. As an example of little-endian format, a 3-bit classical register `creg` with value `'abc'` has `creg[0]=c`, `creg[1]=b`, and `creg[2]=a`.

Run Emulator with Noise Model Turned Off

As a first step, run the emulator with the noise model turned off. An example of how to do this is provided in the code examples on the Quantinuum User Portal. Check that the results returned here are what you would expect without any noise and make sense. If they don't make sense, the error is likely in your circuit design.

Run Emulator with Noise Model

Second, turn the noise model back on. Check if the results here are what you expect and make sense in the presence of noise.

Run on Quantum Hardware

Once the circuit has been debugged using the emulator, submit to a quantum computer.

SUPPORT

Quantinuum values a deep relationship with users and welcomes any questions or feedback. For initial inquiries please use the email address QCsupport@quantinuum.com. Additional personal email addresses may be made available for rapid Q&A or for more detailed discussion.