Quantinuum Application Programming Interface (API) Specification

Version 2.4 May 31, 2023

TABLE OF CONTENTS

INTRODUCTION

This document provides specific instructions and controlled access commands solely for limited testing purposes. All terms and conditions for use of the Quantinuum API are contained within the terms and conditions or the Quantinuum Single Use End User License Agreement (“EULA”).

INTERFACE ENDPOINT

Quantinuum L4 API is a REST interface over HTTPS secure channel with server authentication. The interface is exposed at the following endpoint prefix:

https://qapi.quantinuum.com/v1/

Only HTTP GET and POST methods are used to issue API calls.

API VERSIONING

L4 REST API version is embedded in the endpoint query string. V1 is the currently supported version. Future versions increment version number in the following manner: v2, v3, v4, etc.

When new version is added (v2), the old one (v1) remains active and supported. This is known as v-1 support model. Older versions may remain supported for longer, but only v-1 support is guaranteed.

API ACCESS CONTROL

Access to the Quantinuum L4 API is granted to users via a "Token-Based" authentication. An "id-token" and "refresh-token" are provided to registered users via the "/login" endpoint.

The "id-token" and "refresh-token" are JSON Web Tokens (JWT) using the JSON Object Signing and Encryption (JOSE) specifications. The API implementation uses these tokens to locate the corresponding user account, to check the account balance and to perform any additional access control checks.

An API "id-token" may be revoked and re-issued if an old token is misplaced or stolen. Revoked tokens become permanently ineligible.

API DETAILS

The Quantinuum L4 API is logically divided into four areas – authentication, machine state, job submission, job status and metering.

All API requests, with the exception of "/login", need an "id-token" as part of the header with the "Authorization" key name.

HTTP header for POST requests and GET/POST responses shall contain "Content-Type" header set to "application/json".

All POST API’s send data to the server in the JSON-encoded message body.

All GET API's provide the route to the requested resource in the query string.

Responses to both GET and POST requests are JSON-encoded in the message body. The structure of the JSON is specified in the individual API documentation.

HTTP responses support compression to reduce transmission data size. Requests may include the "Accept-Encoding" header set to one of the algorithms supported by the client, e.g. "gzip, compress, deflate". Responses may be compressed using gzip (LZ77 with 32-bit CRC), compress (LZW) or deflate (ZLIB) algorithms based on matching client option. Response shall specify the encoding used via the "Content-Encoding" header when compression is in use. The use of compression for job status JSON responses is strongly recommended.

Responses to incorrectly formatted messages, non-existent job ID's, etc. result in generic HTTP error response codes.

Users are not allowed to access data belonging to other users. An API token implicitly establishes a name space for the specific user and cross-namespace access is strictly prohibited.

The API uses a small set of data types: strings, booleans and integers. All strings are expected to be UTF-8 encoded ASCII subset. No international characters are allowed at this time.

Optional and mandatory parameters are annotated as [o] and [m]. These annotations are not part of the actual JSON. They are only present in documentation for clarity,

The API documentation may be exposed online alongside with API endpoint in the future.

Authentication API Group

This API endpoint allows registered users to authenticate with username and password and to receive an "id-token" along with a "refresh-token".

The "id-token" has an expiration time of **one hour** and can be used during that time with any of the other API endpoints.

Once the "id-token" expires a new one can be requested using the "refresh-token". The "refresh-token" has an expiration time of **30 days**.

Login

This API returns an "id-token" and a "refresh-token".

Endpoint:	**/login**
Method:	POST
Headers:	Content-Type: application/json

Body:	`{`
	`  "email": <email>,`
	`  "password": <password>,`
	`}`

	<email> and **<password>** established via "User Management" console

Return:	`{`
	`  "id-token":<JWS>,`
	`  "refresh-token": <JWE>`
	`}`
	
	<JWS> JSON Web Token (JWT) in JSON Web Signature (JWS) format.
	<JWE> JSON Web Token (JWT) in JSON Web Encryption (JWE) format.

Token Refresh

This API returns a new "id-token".

Endpoint:	**/login**
Method:	POST
Headers:	Content-Type: application/json

Body:	`{`
	`  "refresh-token": <JWE>,`
	`}`
	
	<JWE> JSON Web Token (JWT) in JSON Web Encryption (JWE) format.

Return:	`{`
	`  "id-token":<JWS>,`
	`  "refresh-token": <JWE>`
	`}`
	
	<JWS> JSON Web Token (JWT) in JSON Web Signature (JWS) format.
	<JWE> JSON Web Token (JWT) in JSON Web Encryption (JWE) format.

Federated Login

This API returns a new "id-token" by taking in an an external id-token from a supported provider.

Note: Only Microsoft Azure Active Directory supported.

Endpoint:	**/login**
Method:	POST
Headers:	Content-Type: application/json
Body:	`{`

	`"provider-token": <JWS>,`
	`}`
	
	<JWS> JSON Web Token (JWT) in JSON Web Signature (JWS) format.

Return:	`{`
	`  "id-token":<JWS>,`
	`  "refresh-token": <JWE>`
	`}`
	
	<JWS> JSON Web Token (JWT) in JSON Web Signature (JWS) format.
	<JWE> JSON Web Token (JWT) in JSON Web Encryption (JWE) format.

Multi-factor Authentication (MFA) Login

This API returns a new "id-token" after passing a valid MFA verification code.

Note: Native Logins Only

Endpoint:	**/login**
Method:	POST
Headers:	Content-Type: application/json

Body:	`{`
	`  "email": <email>,`
	`  "password": <password>,`
	`  "code": <code> (required when MFA enabled)`
	`}`
	<email> and **<password>** established via "User Management" console
	<code> is the 6-digit verification code in the user's authenticator app

Return:	`{`
	`  "id-token":<JWS>,`
	`  "refresh-token": <JWE>`
	`}`
	
	<JWS> JSON Web Token (JWT) in JSON Web Signature (JWS) format.
	<JWE> JSON Web Token (JWT) in JSON Web Encryption (JWE) format.

Password Change

This API returns a new "id-token" after passing a valid MFA verification code.

Note: Native Logins Only

Endpoint:	**/login/password**
Method:	POST
Headers:	Content-Type: application/json Authorization: <id-token>

Body:	
	```
	{
	  "current_password": <email>,
	  "new_password": <password>,
	  "code": <code> (required when MFA enabled)
	}
	```
	<current_password> the user's current password
	<new_password> the user's newly proposed password
	<code> is the 6-digit verification code in the user's authenticator app

Return:	Success: Password successfully changed. Please login with your new password Failure: Unable to change password for user: <email>

Machine State API Group

This API group provides a way to list available machines and check the state of a machine.

Machine List API

This API returns a list of available machine names at that time.

Endpoint:	**/machine** **?config=<true or false>** using this option will provide additional configuration details about the available machines
Method:	GET
Headers:	Authorization: <id-token>

Return:	
	```
	[                       [m]
	  "H1-1",
	  "H1-2",
	  "H1-1E",
	  "H1-2E",
	  "H1-1SC",
	  "H1-2SC"
	]
	```

Array of available machine names is returned. Only these name can be used in the subsequent API calls.

if **config** is set to **true**. Additional information will be in the list as a dictionary

```
[                           [m]
  {
    "name": <name>,                                                [m]
    "n_qubits": <n_qubits>,                                        [m]
    "gateset": <gateset>,                                          [m]
    "wasm": <wasm>,                                                [m]
    "n_classical_registers": <n_classical_registers>,              [m]
    "n_shots": <n_shots>,                                          [m]
    "system_family": <system_family>,                              [o]
    "system_type": <system_type>,                                  [m]
    "emulator": <emulator>,                                        [o]
    "syntax_checker": <syntax_checker>,                            [o]
    "batching": <batching>                                         [m]
  }
]
```

<name> name of system

<n_qubits> maximum number of qubits supported by system

<gateset> list of gate names supported by system

<wasm> flag that specifies if system supports WASM

<n_classical_registers> maximum number of classical registers supported

<n_shots> maximum number of shots supported by system

<system_family> system family name, if system belongs to a family

<system_type> "hardware" | "emulator" | "syntax checker"

<emulator> emulator name for the system

<syntax_checker> syntax checker name for the system

<batching> flag that specifies if system supports batching

Machine State API

Machine state API has the following structure:

Endpoint:	**/machine/<name>**

	<name> is the name of the quantum computer. Must be one of the names returned by machine list API.
Method:	GET
Headers:	Authorization: <id-token>

Return:	`{`
	`  "state": "<state>"    [m]`
	`}`
	
	<state> may be one of the following:
	1. online
	2. offline
	3. reserved
	4. in maintenance
	
	Online: machine is processing user jobs.
	Offline: machine is not processing user jobs.
	In Maintenance: machine is processing user jobs but at a limited rate.
	Reserved: indicates that some user other than the one calling the API has the machine reserved for exclusive use. The user that has an active reservation will receive online response.

Quantum Job API Group

This API group is responsible for handling quantum jobs. It supports job submission, retrieval of results, and job cancellation.

Job Submission API

A quantum job may be submitted using the following API:

Endpoint:	**/job**
Method:	POST
Headers:	Content-Type: application/json Authorization: <id-token>

Body:	`{`
	`  "machine": "<machine-name>",                   [m]`
	`  "name": "<job-name>",                           [o]`
	`  "count": <count>,                               [o]`
	`  "max-cost": <max-cost>,                         [o]`
	`  "options":                                      [o]`
	`  {`
	`    "no-opt": <boolean>,                          [o]`

```
    "tket-opt-level": <tket-opt-level>,            [o]
    "simulator": <string>,                         [o]
    "error-model": <boolean>,                      [o]
    "error-params": {
        'p1': <p1>,                                [o]
        'p2': <p2>,                                [o]
        'p_meas': <p_meas>,                        [o]
        'p_init': <p_init>,                        [o]
        'p_crosstalk_meas': <p_crosstalk_meas>,    [o]
        'p_crosstalk_init': <p_crosstalk_init>,    [o]
        'p1_emission_ratio': <p1_emission_ratio>,  [o]
        'p2_emission_ratio': <p2_emission_ratio>,  [o]
        'quadratic_dephasing_rate': <quad_deph_rt>,[o]
        'linear_dephasing_rate': <lin_deph_rt>,    [o]
        'coherent_to_incoherent_factor': <c_t_i_f>,[o]
        'coherent_dephasing': <boolean>,           [o]
        'transport_dephasing': <boolean>,          [o]
        'idle_dephasing': <boolean>,               [o]
        'przz_a': <przz_a>,                        [o]
        'przz_b': <przz_b>,                        [o]
        'przz_c': <przz_c>,                        [o]
        'przz_d': <przz_d>,                        [o]
        'scale': <scale>,                          [o]
        'p1_scale': <p1_scale>,                    [o]
        'p2_scale': <p2_scale>,                    [o]
        'meas_scale': <meas_scale>,                [o]
        'init_scale': <init_scale>,                [o]
        'memory_scale': <memory_scale>,            [o]
        'emission_scale': <emission_scale>,        [o]
        'crosstalk_scale': <crosstalk_scale>,      [o]
        'leakage_scale': <leakage_scale>           [o]
    }
  },
  "language": "<lang>",                            [m]
  "program": "<program-json>",                     [m]
  "cfl": "<cfl>",                                  [o]
  "notify": <boolean>,                             [o]
  "batch-exec": <max-batch-cost>/"<batch-id>",     [o]
  "batch-end": <boolean>                           [o]
  "group": <group-name>                            [o]
}
```

<machine-name> is the name of the quantum computer to use for executing the job. Must be one of the names returned from the machine list API.

<job-name> string is a user-assigned name for the job.

<count> is an integer between 1 and 10,000 that specifies the number of times the job is requested to run.

<max-cost> is an integer that specifies the number of H-System Quantum Credits (HQC) the user is willing to spend on this job. Actual cost is returned to the user is a result of estimation. Jobs that exceed max allowed cost are not run even if action is specified as "run". <max-cost> parameter is optional. Max cost can be between 0 and account HQC balance. If omitted, <max-cost> defaults to account HQC balance.

"options" is an optional object containing a set of values used for fine-tuning program compilation.

- "no-opt" being set to True disables all compiler optimizations including all TKET compilation passes. Default: False
- "tket-opt-level" can take the value null, 0, 1, or 2. The values 0, 1, or 2 follow the definitions found in the pytket-quantinuum compiler pass definitions as found here: https://cqcl.github.io/pytket-quantinuum/api/index.html#default-compilation. The value null corresponds to basic single-qubit gate combinations, with the program is treated as having an implicit "barrier" instruction after every gate operation. Default: 2
- "simulator" specifies which simulator model to use, either "state-vector" or "stabilizer". The stabilizer simulator is faster, but does not support the full set of operations. If an operation is supplied to the stabilizer simulator that is not supported an error is returned indicating that the program failed due an incompatible operation. Default: "state-vector"
- "error-model" whether or not to apply an error model. Default: True
- "error-params" list options pertaining to specific parameters of the emulator error model. Default runs of emulator use all these parameters with values as listed in the Quantinuum System Model H1 Emulator Product Data Sheet. Updating values changes the error model to the values the user specifies.
    - p1: probability of a fault occuring during a single-qubit gate, takes values between 0 and 1.
    - p2: probability of a fault occuring during a two-qubit gate, takes values between 0 and 1.
    - p_meas: probability of a bit flip being applied to a measurement, takes either (1) a floating value between 0 and 1, or (2) a tuple of 2 floating point values. If a single float is provided, then that error rate is used to bitflip both 0 and 1 measurement results. If a tuple is supplied, then the first element is the probability a bit flip is applied if a 0 result occurs during measurement while the second error rate is for if a 1 is measured.
    - p_init: probability of a fault occuring during initialization of a qubit, takes values between 0 and 1.

	○ p_crosstalk_meas: probability of a crosstalk measurement fault occurring, takes values between 0 and 1. ○ p_crosstalk_init: probability of a cross-talk fault occurring during initialization of a qubit, takes values between 0 and 1. ○ p1_emission_ratio: fraction of p1 that is spontaneous emission for a single qubit instead of depolarizing noise, takes values between 0 and 1. ○ p2_emission_ratio: fraction of p2 that is spontaneous emission for a single qubit in a two-qubit gate instead of depolarizing noise, takes values between 0 and 1. ○ quadratic_dephasing_rate: the gate RZ (frequency x duration) is applied during transport and qubit idling, applied by default for the state-vector simulator ○ linear_dephasing_rate: Pauli Z is applied during transport and qubit idling according to the probability (sin(frequency x duration)/2)^2, applied by default for the stabilizer simulator ○ coherent_to_incoherent_factor: If coherent_dephasing is set to False then the frequency for the quadratic error model (quadratic_dephasing_rate) is multiplied by coherent_to_incoherent_factor to attempt to make up for increased noise due to coherent effects; how sensitive circuits are to coherent effects depends on the circuit. Therefore, users may want to adjust this factor appropriately. ○ coherent_dephasing: Whether or not to use a coherent or incoherent simulation Default: True for state-vector simulations (simulator: “state-vector”), False for stabilizer simulations (simulator: “stabilizer”) ○ transport_dephasing: Turn transport dephasing on or off. Default: True ○ idle_dephasing: Turn idle dephasing on or off. Default: True ○ przz_a: depolarizing fit parameter 1, any real number that scales to $[-\frac{\pi}{2},\frac{\pi}{2}]$, see the Quantinuum System Model H1 Emulator Product Data Sheet for the model and default value ○ przz_b: depolarizing fit parameter 2, any real number that scales to $[-\frac{\pi}{2},\frac{\pi}{2}]$, see the Quantinuum System Model H1 Emulator Product Data Sheet for the model and default value ○ przz_c: depolarizing fit parameter 3, any real number that scales to $[-\frac{\pi}{2},\frac{\pi}{2}]$, see the Quantinuum System Model H1 Emulator Product Data Sheet for the model and default value

- przz_d: depolarizing fit parameter 4, any real number that scales to $[-\frac{\pi}{2}, \frac{\pi}{2}]$, see the Quantinuum System Model H1 Emulator Product Data Sheet for the model and default value
- scale: scale all error rates in the model linearly, takes values from 0 to 1, Default: 1.0
- p1_scale: scale the probability of single-qubit gates having a fault, takes values between 0 and 1.
- p2_scale: scale the probability of two-qubit gates having a fault, takes values between 0 and 1.
- meas_scale: scale the probability of measurement having a fault, takes values between 0 and 1.
- init_scale: scale the probability of initialization having a fault, takes values between 0 and 1.
- memory_scale: linearly scale the probability of dephasing causing a fault, takes values between 0 and 1.
- emission_scale: scale the probability that a spontaneous emission event happens during a single or two-qubit gate, takes values between 0 and 1.
- crosstalk_scale: scale the probability that measurement or intialization crosstalk events get applied to qubits, during mid-circuit measurement and reset (initialization), "crosstalk" noise can occur that effectively measures other qubits in the trap or cause them to leak, takes values between 0 and 1.
- leakage_scale: scale the probability that a leakage even happens during single or two-qubit gates as well as during initalization or crosstalk; on the device half the time, spontaneous emission leads to a leakage event, takes values between 0 and 1.

Note: To support backwards compatibility, this object may be provided as an array with a "no-opt" element.

<lang> string identifies specific language dialect. Currently supported language dialect is "OPENQASM 2.0". Other languages or dialects may be added in the future.

<program-json> is a JSON-formatted program source. Program source is converted to JSON-compliant format by escaping quotes and other special characters. No other modification to program source is performed.

"notify" field allows the customer to decide whether or not to be notified when the job has completed or failed. This assumes that the customer has their notification details in the system (either email or SMS). By default, this field is false.

<batch-exec> is either a positive number or batch-id. If batch-exec is a number a new batch is requested with the max-batch-cost as the number

	given. If batch-exec is a batch-id the job will be submitted to the batch. The batch-id for any batch is the job id of the first job submitted. **<batch-end>** is a boolean that marks the end of a batch when set as true. Once the job is completed the batch will terminate. **<group-name>** is a string that identifies the 'user group' to assign on the incoming job submission. A user may belong to one or more user groups as defined by their organization administrator. If this field is omitted at the time of job submission, a default user group will automatically be used instead. The submitter may choose to override their default user group by specifying an alternative group name. A user group may or may not have a 'quota' assigned which dictates how much credits are available to submit against a particular machine or machines available in their organization's access plan(s). As costs are incurred, they will be deducted against whatever user group was used at the time of submission.
Return:	Success response: `{ "job": "<job-id>", [m] "status": "<status>" [m] }` Error response: `{ "job": "<job-id>", [m] "status": "<status>", [m] "error": [m] { "code": <err_code>, "text": "<err_string>" } }` **<status>** string can be one of: • queued • failed When status is "failed", an error response is returned containing error details. The specific error is identified by a numeric code and user-readable string. **<job-id>** returned by the API is an opaque string generated by the system. The string does not carry any information about the job other than identifying it uniquely.

Job Status API

Users monitor job status using the job status API. In addition to this API, users may set notification preferences via the user portal to allow email or text notifications related to job status. These portal-configured notifications are out of scope for this API specification.

Different results formatters may be added in the future via additional query string parameters. Current API specification supports raw results only.

To retrieve job status and results, the following API is used:

Endpoint:	**/job/<job-id><?websocket=true or false><?results_format=simple>** **<job-id>** corresponds to the ID returned by the job submission API. **?websocket=<true or false>** using this option will return a websocket dictionary with a task token and execution ARN to be used when polling is not desired. You can open the WebSocket with the OpenConnection action and supply the task token and execution ARN to wait for results. When websocket query parameter is not specified it defaults to false. **?results_format=histogram-flat** using this option will consolidate the **<results>** for reporting purposes.
Method:	GET
Headers:	Authorization: <id-token>

Return:	Success response:
	`{`
	`  "job": "<job-id>",                  [m]`
	`  "name": "<job-name">,               [o]`
	`  "status": "<status>",               [m]`
	`  "submit-date": "<submit-date>",     [m]`
	`  "result-date": "<result-date>",     [o]`
	`  "cost": <cost>,                     [o]`
	`  "start-date": "<start-date>",       [o]`
	`  "end-date": "<end-date>",           [o]`
	`  "results":                          [o]`
	`  {`
	`    "c0": ["00", "01"],`
	`    "c1": ["0110", "1110" ],`
	`    …`
	`  },`
	`  "websocket":                        [o]`
	`  {`
	`    "task_token": "AAA...",`
	`    "executionArn" "arn:aws:state:..."`
	`  }`
	`}`
	
	Error response (failed to compile, not enough credit):
	`{`
	`  "job": "<job-id>",                  [m]`
	`  "name": "<job-name">,               [o]`
	`  "status": "<status>",               [m]`
	`  "submit-date": "<submit-date>",     [m]`
	`  "cost": <cost>,                     [o]`

```
  "error":                           [m]
  {
    "code": <err_code>,
    "text": "<err_string>"
  }
}
```

<job-id> corresponds to the ID returned by the job submission API.

<job-name> is the user assigned job name.

<job-status> may be one of the following:

- queued
- running
- failed
- completed
- canceling
- canceled

When status is "failed", the error parameter is present. The specific error is identified by a numeric code and user-readable string. The error parameter is not present when status is not failed.

<submit-date>, **<result-date>, <start-date>** and **<end-date>** are the timestamps corresponding to job submission, final job result receipt, execution start, and execution end. The timestamps are expressed in UTC with millisecond precision. Only <submit-date> is present unconditionally. <start-date> and <end-date> are present if the job reached the corresponding state. Unix timestamp format is used for all timestamps, e.g. 2018-08-25T16:18:11.719Z.

<cost> is the number of HQCs the job is estimated to consume or has actually consumed if the job has already finished execution.

<results> is a JSON-formatted array consisting of all classical measurements identified by the names of classical registers specified in the QASM source. Each classical register value is represented as a bit-string consisting of zeroes and ones. This field is only present if <job-status> is equal to "completed" or "canceled" with partial result set available. An example results parameter for a circuit with two classical registers c0 and c1 executed 8 times is shown below:

```
"results":
{
  "c0": ["00", "01", "01", "00", "10", "00", "00", "10"],
  "c1": ["0010", "0110", "0011, "0011", "1011", "0011",
"0011", "0110"]
}
```

	The results are returned in the order of execution. First row corresponds to the first shot, second to the second shot, etc.The result is somewhat verbose since the measurement results for all shots are included as is. This approach enables any form of aggregation or statistical analysis on the raw measurement set since no information is lost between actual execution and the results delivered to end user. HTTP compression is recommended for larger results sets.
	
	The order of bits in the multi-bit results is from the left to the right. The least significant bit is on the right, for example b110=6.
	
	If the "**<?results_format>**" query parameter is specified this will format the above results into one of our supported formatting types:
	
	histogram-flat:
	
	A simple histogram depicting the frequencies for each bit string. The bits for each row are ordered from smallest to largest.
	
	"results":
	
	{
	
	"c0": {"00":3, "01":"2", "10":2],
	
	"c1": {"0010":1, "0011":4, "0110":2, "1011":1}
	
	}
	
	<websocket> are necessary instructions to open a WebSocket to wait for job results. This allows users to avoid polling the system constantly for results. It is best practice to close the WebSocket after completion. This field only occurs if the query string parameter is included

Job Cancel API

Previously submitted quantum job may be canceled using this API:

Endpoint:	**/job/<job-id>/cancel** **<job-id>** corresponds to the ID returned by the job submission API.
Method:	POST
Headers:	Content-Type: application/json Authorization: <id-token>
Body:	Empty

Return:	Success response is indicated by HTTP status 200 and has no body.
	
	Error response is indicated by HTTP status 400 or 500 and has the following message body:
	
	`{`
	`  "error":       [m]`

	`{`
	`  "code": <err_code>,`
	`  "text": "<err_string>"`
	`}`
	`}`
	
	Job status API may be used to retrieve partial results if any.

Metering API Group

The Metering API group assists with billing and provides end users historical information about the jobs they have executed within a specific time frame.

Metering API

The Metering API is used to retrieve information about the jobs run and the number of HQCs consumed within specific period of time. Metering requests can use date range, last X number of days or last Y number of jobs.

Endpoint:	**/metering?start=<start>&end=<end>/metering?days=<days>/metering?jobs=<jobs>** **<start>** and **<end>** are start and end dates for the metering report. The dates are specified in the YYYY-MM-DD format. Start date cannot be further than 40 days in the past. **<days>** is the number of days in the past starting from today. Days can be between 1 and 40. **<jobs>** is the number of last jobs submitted. Jobs can be between 1 and 100. Note that all three methods of querying are mutually exclusive. In other words, you cannot specify both **<days>** and **<jobs>**.
Method:	GET
Headers:	Authorization: <id-token>

Return:	Success response:
	`{`
	`  "total-cost": <total-cost>,           [m]`
	`  "jobs":                                [m]`
	`  [`
	`    {`
	`      "job": "<job-id>",                 [m]`
	`      "name": "<job-name>",              [o]`
	`      "machine": "<machine>",            [m]`
	`      "submit-date": "<submit-date>",    [m]`
	`      "result-date": "<result-date>",    [o]`
	`      "cost": <cost>                     [o]`
	`    },`
	`    …`

	```
	    ]
	}
	```
	Jobs array may be empty is no jobs where submitted within specified timeframe. Jobs are returned based on <submit-date> timestamp.
	<total-cost> is the number of HQCs used on this query
	Jobs array contains all jobs within specified time or job window.
	<job-id> is the unique ID assigned to the job.
	<job-name> is the user-assigned name for the job, same as in job submission API.
	<machine> is the name of the machine used to execute the job.
	<submit-date>, **<result-date>** and **<end-date>** are the timestamps corresponding to job submission and final job results. The timestamps are expressed in UTC with millisecond precision.
	<cost> is the number of HQCs consumed by the job.

ERROR CODES

The following documents all the error codes across all the lambdas and possible resolutions.

Error Code	Description	Possible Resolution	Endpoints Affected
1	Internal server error. This is a generic unhandled error		All
2	Machine name is invalid. The requested machine does not exist or is not accessible	Use one of the available machines, use machine list endpoint	Job add
4	Count must be a number	Use a number for shot count	Job add
5	Max cost must be a number	Use a number for the max cost	Job add
6	Missing required machine parameter	Include `machine` in job request	Job add
7	Missing required language parameter	Include `language` in job request	Job add
8	Language is not supported	Use one of the supported languages, currently only `OPENQASM 2.0`	Job add
9	Missing required program parameter	Include `program` in job request (should be QASM code)	Job add

10	Only one option is accepted at this time	Only use one compiler option at a time. Currently, only \`tket\` or \`no-opt\` are supported	Job add
11	Only \`tket\` or \`no-opt\` are supported	Only use one of the compiler options	Job add
12	Count must be between 1 and 10,000	Use a shot count in the expected range	Job add
13	Program is too large for this release	Reduce QASM length to below 256k chars	Job add
14	You do not have access to this machine	Request permission to use this machine or use a different machine	Job add, Machine state
15	Batch cannot be negative	Use a positive number for batch-exec	Job add
16	Batch exceeds max batch size	Use a smaller number for batch-exec	Job add
17	Batch exceeds available hqc	Request a batch under your available hqc	Job add
18	Batch does not exist	Use valid batch ID for batch-exec	Job add
19	Batching not available for system families.	Batching can't be used while targeting a system family.	Job add
20	Batch machine names do not match.	Machine names must match in all jobs submitted as part of a batch.	Job add
21	Job does not exist	The referred job does not exist (typo, system does not have a record, or you do not have access)	Job status, Job Cancel
22	Job has completed already	N/A, job cancel attempt was too late	Job cancel
23	Notify must be a boolean	Use a boolean for notify	Job add
24	Organization already exists	Specify a different organization	Organization Create
25	User already exists	Specify a different user	User Create, User Update
28	Not a Quantinuum Admin	User must be a Quantinuum Admin to perform this action	User Create
29	User doesn't exist	Specify a user that exists	Login

34	User Not Authorized	Supply correct username/password combination	Login
35	New Password Required	Log in with temporary password and change to new password	Login
36	Token or credentials missing	User must provide a valid refresh token or the missing login credentials	Login
37	Access forbidden	User's permissions need to be elevated to Quantinuum_Admin or Org_Admin	Login
38	Terms and conditions must be accepted on UI before logging in.	The user must login to the UI and accept any our hardware compliance agreement	Login
39	Provider token is invalid	The user was able to use federated sign since the provider token is invalid	Login
40	task_token must be provided	Provide the task_token from job_status	open_connection
41	executionArn must be provided	Provide the executionArn from job status	open_connection
42	executionArn is invalid	The provided execution ARN is invalid	open_connection
43	Software agreements must be accepted on UI before logging in	The user must login to the UI or open their welcome email to accept any pending software compliance agreements	Login
50	Must include query parameters	Add start/end xor days xor jobs	metering
51	Start/end parameter does not conform to YYYY-MM-DD format	Use the expected format	metering
52	End parameter must be paired with a start parameter	Add start parameter	metering
53	End date must be after start date	Use a later end date	metering
54	Only one query mode can be used at a time: start/<end> xor days or jobs	Only use one of the expected modes	metering

55	Days query parameter must be between 1 and 365 inclusive	Use a number that is between 1 and 365 inclusive	metering
56	Jobs query parameter must be a positive number	Use a number over zero	metering
60	No org access	Either organization doesn't have access or the user does not have an associated organization	list_machines
61	User's organization does not have any user groups	User's organization does not have any available user groups. User must submit without a group name	Job add
62	User does not belong to any user groups	User's organization has user groups, but the user does not belong to any of them. User must submit without group name.	Job add
63	Group does not belong to organization	User must provide a valid group name or submit without a group name.	Job add
64	User does not belong to group	User provided a valid group name, but they do not belong to it. User must provide a different group name or submit without it.	Job add
70	Unable to change password. One or more required credentials are missing	User must provide missing credentials to change password	Login Password Change
71	New password cannot be the same as the current password	User must provide a different value for the new password	Login Password Change
72	Password complexity does not meet our requirements	User must provide a more complex password to satisfy the requirements	Login Password Change
73	Multi-factor authentication (MFA) is enabled. Please try again with the required verification "code" parameter	Include 'code' in request body	Login Login Password Change
74	The verification code is invalid or has already expired. Please try again	Include valid and unexpired verification 'code' in request body	Login Login Password Change

75	Password change is not supported for account	Password change is not supported for federated accounts.	Login Password Change
100	Invalid value for parameter 'results_format'	Pass in a supported formatting type ("histogram-flat")	Job status
1000	Compilation Error	The QASM program provided could not be compiled, see error details for additional information.	Job status
1001	Job Processing Error	An error in job processing was detected.	Job status
1002	Job cost exceeds allowed cost	Verify that you have sufficient funds to execute the job and have not restricted the max cost allowed.	Job status
2000	Execution Error	The job was unable to be executed.	Job Status
3000	Run-time Error	The job encountered an error while running.	Job status
3001	Program size limit exceeded	The job could not be run as it exceeded the target system's current capabilities.	Job status