



WE BRING **IOT** PROJECTS TO LIFE

qiot.io

API Reference

Revision 1

February 20, 2017

QUANTUM^{IOT}

qiot.io

The purpose of this document is to share with you the full functionality of the qiot.io platform. The content and reference materials cited here demonstrate how to effectually utilize and interact with qiot.io to ensure your organization is taking full advantage of the platform's capabilities.

We want you to be confident that you're collecting the type of mission-critical data points that can be turned into actionable insight—insight that enables you to realize the greatest, most lucrative possibilities of a truly interconnected and intelligent world.

THE QIOT.IO ADVANTAGE ↪

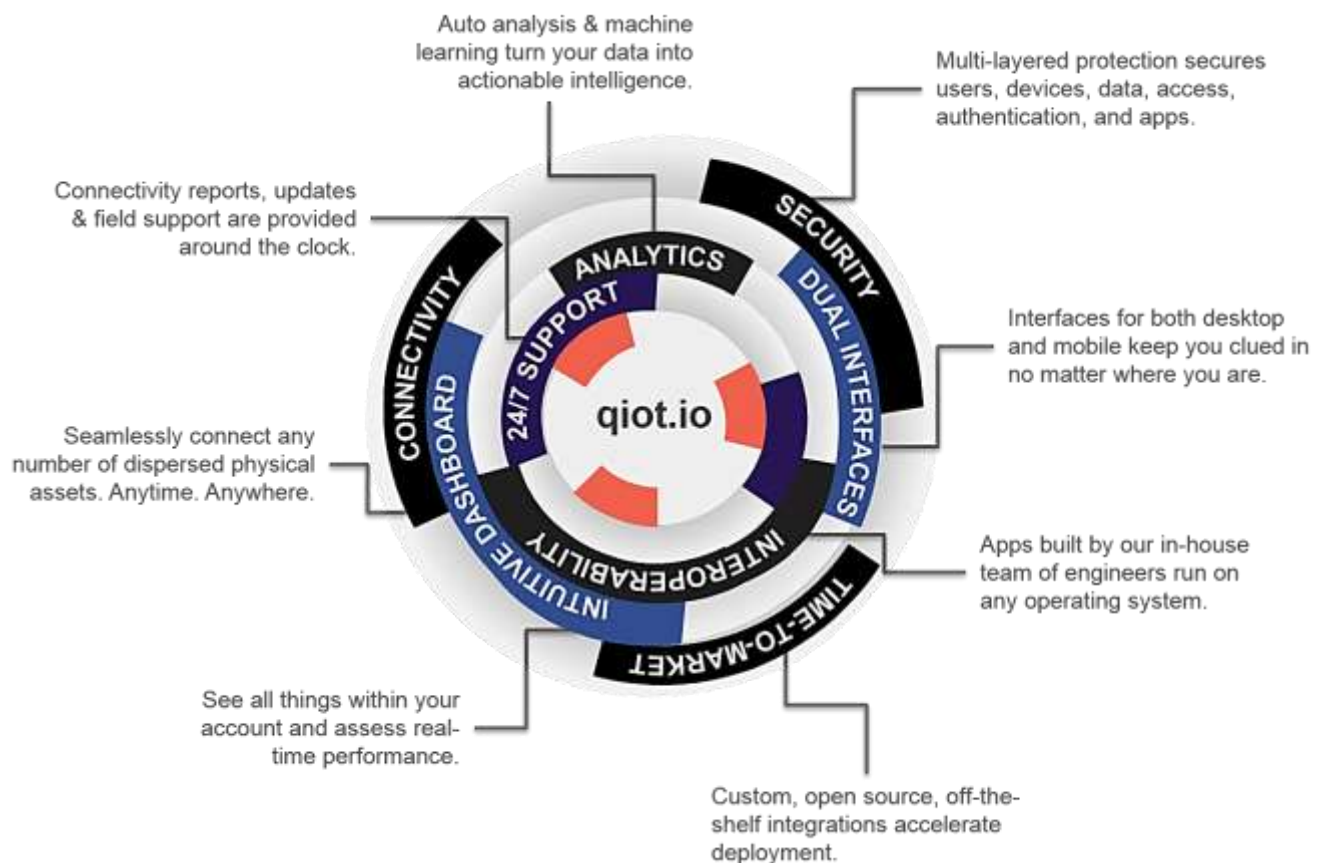


Figure 1 - Key Features of qiot.io

TABLE OF CONTENTS

OVERVIEW.....	6
ARCHITECTURE	7
TECHNOLOGIES.....	9
SECURITY.....	10
DEFINITIONS	12
ACCOUNT ADMINISTRATOR	12
TEAM	12
TEAM MEMBER	12
THING.....	12
LABEL	13
COLLECTION.....	13
IDENTIFIER.....	13
THING TYPE.....	14
ACCOUNT ID.....	14
ACCOUNT SECRET	14
ACCOUNT TOKEN.....	14
API TOKEN.....	14
AUTHENTICATION TOKEN	14
AUTHORIZATION TOKEN	14
QUICKSTART GUIDE	15
CREATING AN ACCOUNT.....	15
AUTHENTICATING WITH THE PLATFORM.....	15
NAVIGATING THE DASHBOARD	16
CONNECTING A THING.....	18
SENDING DATA	22
READING DATA	23
QIOT.IO APIs.....	24
REST API GENERAL NOTES.....	24

MQTT API GENERAL NOTES.....	25
THING REGISTRATION API.....	26
REST DEFINITION	26
REST EXAMPLE	26
REST RESPONSE.....	26
THING MAILBOX API	28
REST DEFINITION	28
REST EXAMPLE	29
REST RESPONSE.....	29
MQTT DEFINITION	30
MQTT EXAMPLE	30
MQTT RESPONSE	30
THING LOG API - PUBLISH	31
REST DEFINITION - PUBLISH	31
REST EXAMPLE - PUBLISH.....	32
REST RESPONSE - PUBLISH	32
MQTT DEFINITION - PUBLISH.....	33
MQTT EXAMPLE - PUBLISH	34
MQTT RESPONSE - PUBLISH.....	34
THING LOG API - SUBSCRIBE	34
REST DEFINITION - SUBSCRIBE	34
REST EXAMPLE - SUBSCRIBE	35
REST RESPONSE - SUBSCRIBE	35
MQTT DEFINITION - SUBSCRIBE.....	36
MQTT EXAMPLE - SUBSCRIBE	37
MQTT RESPONSE - SUBSCRIBE	37
THING COLLECTION SUBSCRIPTION API	38
REST DEFINITION	38
REST EXAMPLE	38
REST RESPONSE.....	38
MQTT DEFINITION	40
MQTT EXAMPLE	40
DEVELOPER TERMS.....	41
QUANTUMIOT SUPPORT.....	41

TABLE OF FIGURES

Figure 1 - Key Features of qiot.io	2
Figure 2 - qiot.io Microservices	6
Figure 3 - qiot.io's Structural Architecture	7
Figure 4 - Architectural Detail of Messaging Bus Microservice	8
Figure 5 - qiot.io Connectors	9
Figure 6 - qiot.io Data Flow	10
Figure 7 - qiot.io Login	15
Figure 8 - qiot.io Dashboard	16
Figure 9 - Accessing Details of Dashboard Modules	17
Figure 10 - Account Settings	18
Figure 11 - Connecting a New Thing	19
Figure 12 - "Create Thing" Pop-Up Window	19
Figure 13 - "Add Identifier" Option for Things	21
Figure 14 - Things Index	22
Figure 15 - Thing and Platform Interaction	28

OVERVIEW

The INTERNET OF THINGS (IoT) is complex, but solutions in this space share certain common needs. They must identify devices, capture data and store it, detect critical events, deliver alerts and warning notifications, and enable data access on demand. Our qiot.io solution addresses all of these needs, and much more. In doing so, it frees enterprises and developers from recreating these common components, enabling them to instead focus on their own unique solutions.

At QuantumIOT, we like to think of qiot.io as an "un-platform." It is a hand-selected and hand-crafted collection of loosely coupled microservices for your connected devices. These microservices are interwoven together through an innovative and unique framework of mashup technologies.

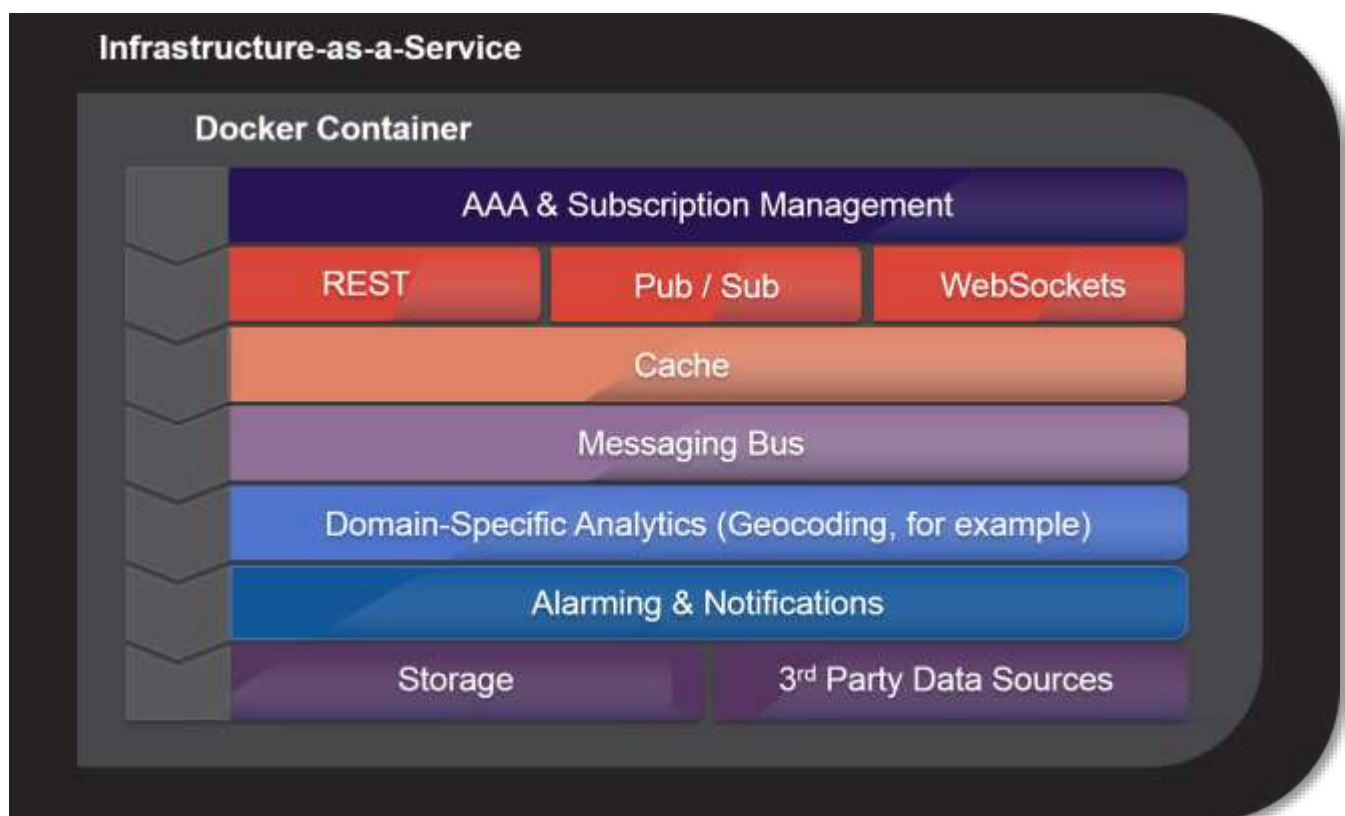


Figure 2 - qiot.io Microservices

ARCHITECTURE

Qiot.io is a cloud-deployed, stand-alone solution, customized to meet your specific business needs. Its underlying architecture and invariant interfaces are described here in this document and form qiot.io's cross-functional core. The real power of the platform, however, stems from its versatility and the custom features designed to catalyze your most ambitious IoT plans and uncover real, lucrative returns. As you can see in the architecture diagram below, qiot.io is robust and elastic in nature and is both linearly and horizontally scalable.

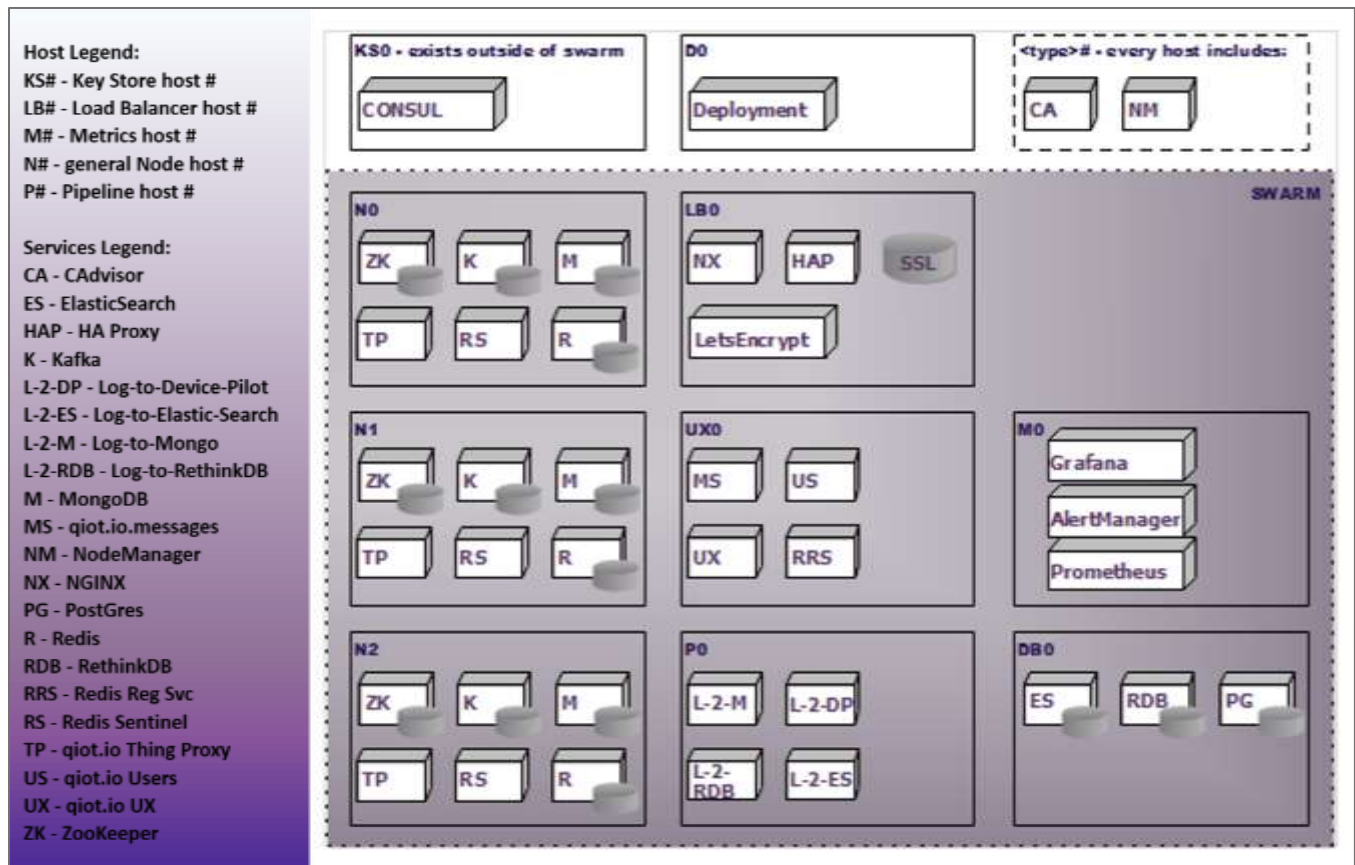


Figure 3 - qiot.io's Structural Architecture

Qiot.io's design allows docker deployments to be set up through ENV variables, which means developers can configure the system using environment config files without having to modify versioned files in the repository. The app reads configuration values from three different sources, in the order of priority outlined below:

- 1) ENV vars
- 2) .env file
- 3) .env.defaults file

Configuration values from sources with lower priority are overridden by the values present in sources with higher priority. When the NODE_ENV environment variable is set, the list of configuration sources is the following (assuming NODE_ENV = production):

- 1) ENV vars
- 2) .env.production file # '.env.' + NODE_ENV
- 3) .env.defaults file

The .env.defaults file is a versioned file and is systematically modified whenever a config key is added or removed. When configuring an environment, there's no need to change this file. In addition to providing the third level of priority for default configuration values, the purpose of the .env.defaults file is to inform developers which ENV variables can be configured.

Qiot-io's framework allows for IoT scale throughput. It offers geographic redundancy and accommodates distributed computing models. Its service-oriented architecture delivers heavy, yet simple, extensibility and facilitates rapid prototyping of new features, data sources, and integration patterns.

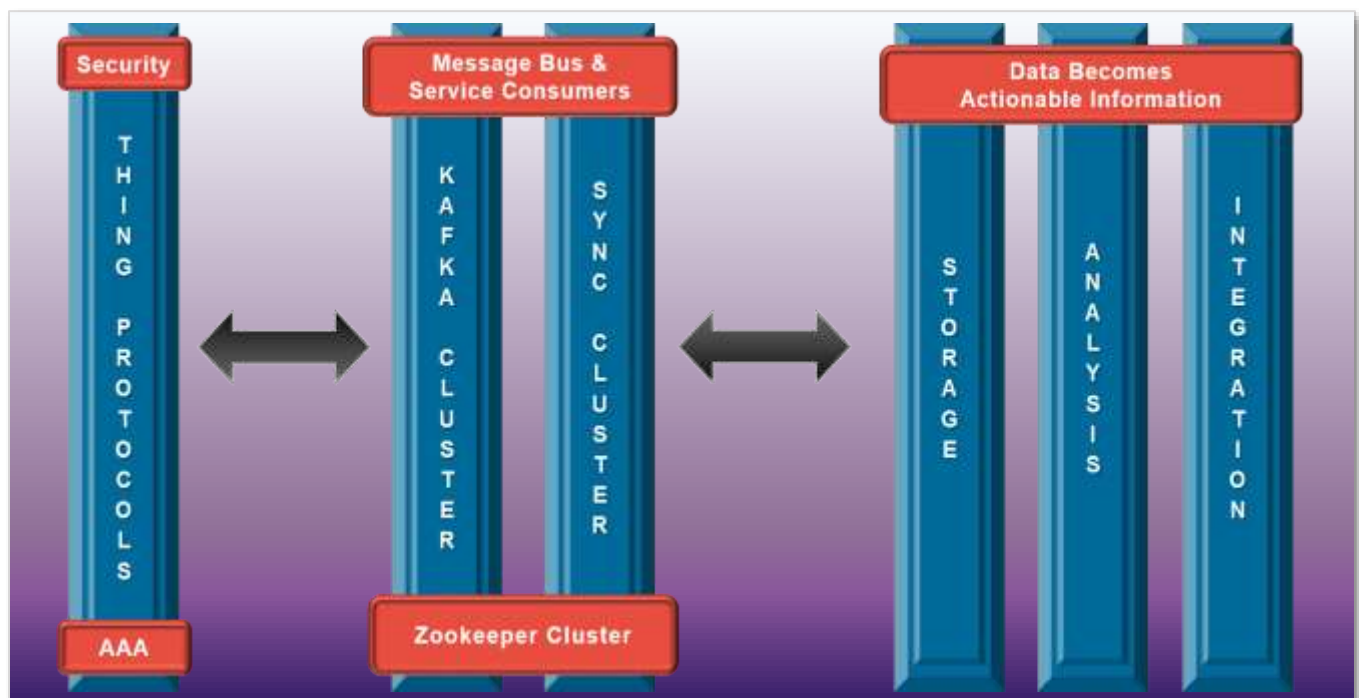


Figure 4 – Architectural Detail of Messaging Bus Microservice

TECHNOLOGIES

Systems integration is a crucial, yet often complicated and burdensome piece of a successful IoT deployment. Not with qiot.io. We've taken "complicated" and "burdensome" out of the equation, which means you can realize your IoT vision quickly and painlessly.

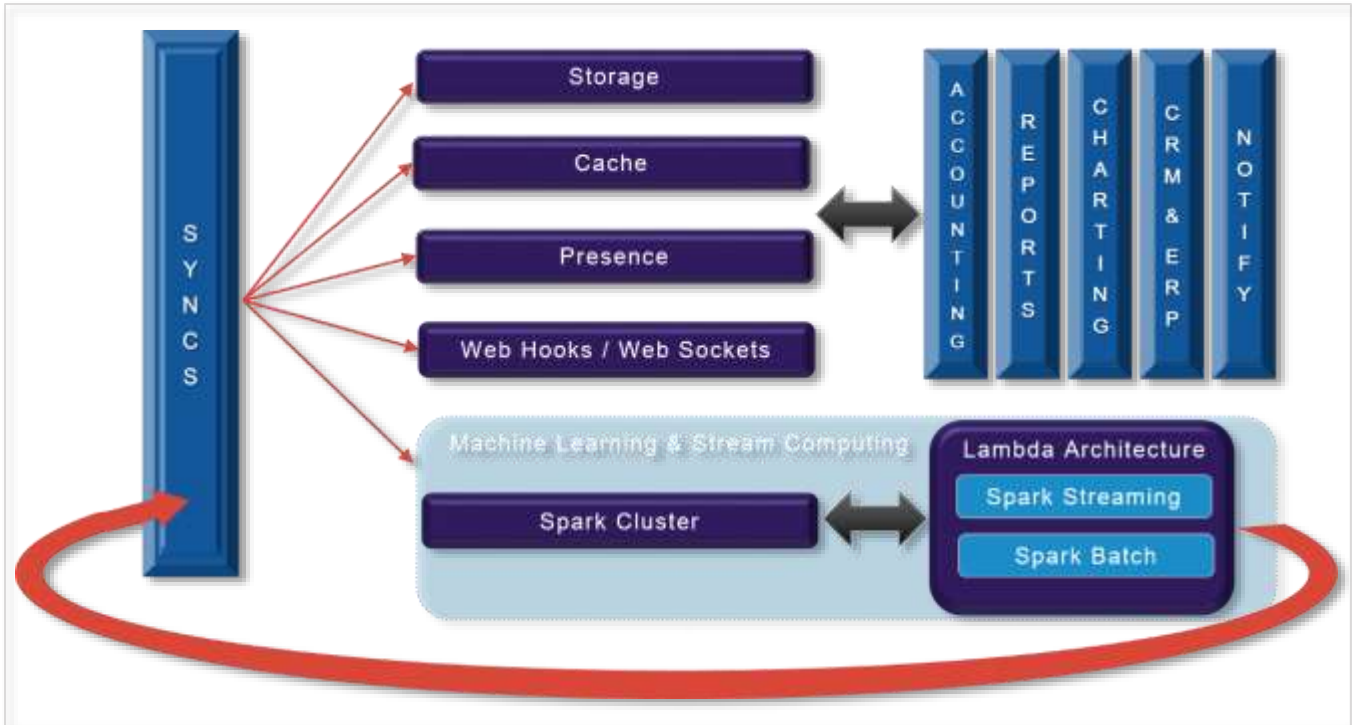


Figure 5 – qiot.io Connectors

The qiot.io platform can be deployed within any of the major cloud service providers, including but not limited to Amazon Web Services (AWS), the Google App Engine, IBM's BlueMix, Microsoft's Windows Azure, and Salesforce.

Additionally, qiot.io uses the common communication protocols Representational State Transfer (REST) and Message Queuing Telemetry Transport (MQTT) to relay information between the platform, connected devices, and applications.

All qiot.io REST APIs are TLS 1.2-encrypted and require an HTTP Header QIOT Authorization Account Token for authenticating with the platform. Do not confuse "Authorization" and "Authentication" in this context; unfortunately, the legacy HTTP Header "Authorization" really means "Authentication."

MQTT APIs give you the option of TLS 1.2 transport-level encryption and authenticate via username and password. MQTT brokers are hosted on standard ports at 1883 and 8883, respectively.

At QuantumIOT, we engineer solutions that bring meaningful, actionable intelligence to our clients' operations. With the power of qiot.io at your fingertips, the IoT can deliver seamless connectivity between any number of dispersed physical assets and yield the kind of data and insight that not only saves you time and money, but transforms the way your organization operates. Evolve and refine your business model based on actual consumption, measured utilization, and real-time market demand.

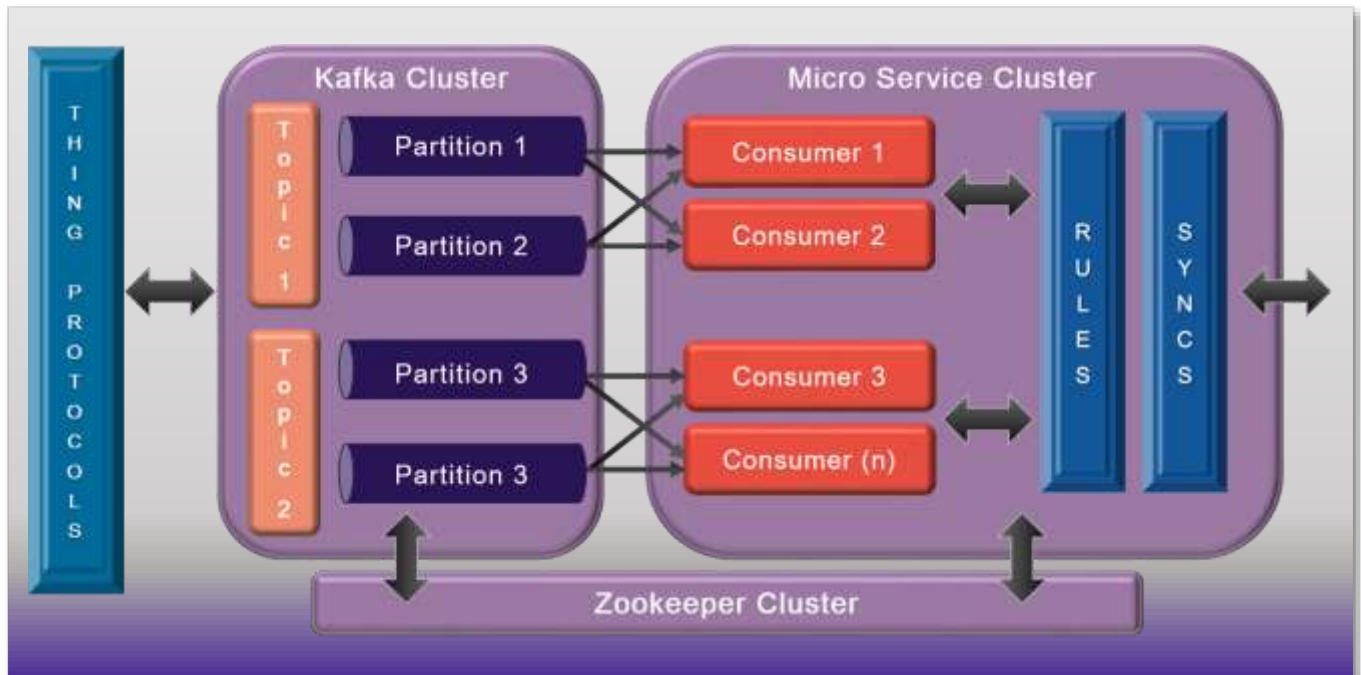


Figure 6 – qiot.io Data Flow

Ready for the next step? With qiot.io in place, we can help make your data work for you. Whether it's stream computing, data lakes, machine learning, or custom-designed algorithms your business calls for, qiot.io harnesses the power of the cloud to give you the kind of real-time analytics and compelling metrics to fully optimize your operations and elevate performance to a whole new level.

SECURITY

Everything we deploy at QuantumIOT lives up to the highest industry security standards. All of our devices and sensors are designed with secure, reliable firmware that is protected against exploitation, but performance and penetration testing are just the beginning. We lead the IoT industry in strategic, multi-layered protection for devices, data, connectivity, applications, authentication, and access.

If a system-generated token has been compromised or could potentially expose your business to a security breach, it can be deleted or suspended, denying authentication to its associated user, device, or app. If necessary, a new token can then be regenerated and reassigned to the user, device, or app in question, permitting secure communication with the platform once again.

In addition, qiot.io's assiduous monitoring capabilities mean system administrators are notified immediately of any system degradation, performance issues, or outages. The solution also retains certain automatic restart capabilities aimed at ensuring messages are not lost and performance does not slip below acceptable thresholds.

OPERATING MARGINS have **improved by as much as 19%** within the last five years for retail supply chain operations that **have adopted data and analytics, according to the McKinsey Institute.**

– *"The Age of Analytics: Competing In A Data-Driven World,"*
McKinsey Global Institute, December 2016

DEFINITIONS

Below are the principal entities, or components, you'll see throughout the platform. You'll encounter them every time you check your qiot.io Dashboard and interact with them as components of API calls and responses, device properties, and data collection parameters.

ENTITY:	DESCRIPTION:
Account Administrator	User who has access to manage—create, review, update, and delete (CRUD)—all other users, Teams , Collections , and Things within an account. An account may have more than one Account Administrator. A special use case of Account Administrator is the Account Owner. The Account Owner is the first user designated as an Account Administrator and is the one who originally creates the account itself.
Team	A group of users, identified as Team Members , granted access to a Collection of Things owned by an Account Administrator . Teams are created by Account Administrators, and individual Team Members are invited to a team by the Account Administrator owner. A Team can likewise be assigned Collections whereby the associated Team Members can then manage (CRUD) Things within that Collection.
Team Member	Any user designated as a member of a specific Team , as created by an Account Administrator . A Team Member is invited to join a Team. (Note: Account Administrators may also be Team Members.)
Thing	Any entity that can deliver data to the platform or be sent commands through the platform. A Thing (denoted by the icon ) can subscribe to its own channel as well as other channels to receive commands. Each Thing is assigned: ➤ Authentication Token - Initially generated by the system upon creation, but may be regenerated by any user privileged to do so. ➤ Label - Consists of Thing name (a 16-character alphanumeric tag automatically generated by the system) and a description of the Thing (entered by the user). ➤ Collection - A Thing is assigned to a Collection when it is created within the system but can subsequently be moved to other Collections. A Thing may only be assigned to a single Collection at a time.

	➤ Identifier - A distinguishing denomination, such as an IMEI, serial number, etc. It must only be unique within the context of the account that owns the Thing. ➤ Type - References a set of meta data which clues the rest of the system in on the Thing's features and functions as well as information about units of measure, etc.
Label	Free form text field that defaults to the Thing name plus a descriptor keyed in by the user. Any string of characters may be added to the Label as a reminder to the user of the purpose of the Thing. This could be a serial number or any other alphanumeric identifier, or even something as simple as "Bob's Taco Truck."
Collection	A parent category that can be defined as a logical grouping of Things. A Collection (denoted by the icon ) may be assigned to one or more Teams, granting those Team Members CRUD privileges to manage the Things within that Collection. Each Collection will have: ➤ Authorization Token - Generated by the system when Collection is created, but can be forcibly regenerated by anyone granted CRUD privileges for that Collection. ➤ Thing Count - A tally indicating the total number of Things currently within the Collection. ➤ Last Updated Timestamp - A digital record of the time at which the most recent update to its contents occurred.
Identifier	Each Thing created must be associated with a unique set of Unicode characters for recognizing and distinguishing that particular device. Below are the Identifiers currently adopted and substantiated by qiot.io: ➤ ESN: Electronic Serial Number ➤ ICCID: Integrated Circuit Card Identifier ➤ IMEI: International Mobile Equipment Identity ➤ IMSI: International Mobile Subscriber Identity ➤ MAC: Media Access Control address ➤ MEID: Mobile Equipment Identifier ➤ MSISDN: Mobile Station International Subscriber Directory Number ➤ SN: Serial Number <i>For additional details on each supported Identifier, see the step-by-step instructions for Connecting a Thing in the Quickstart Guide on page 18 of this document.</i>

Thing Type	A Thing Type data model is a JSON object representing tuples of information about the Thing , including its features, functions, and units of measure. Thing Types follow a freeform format and are designated with the icon  .
Account ID	Comparable to a username. Both the Account ID and Account Secret are used for communication protocols in which username/password authentication is required. Message Queuing Telemetry Transport (MQTT) is an example of such a protocol.
Account Secret	Comparable to a password. The Account Secret, along with Account ID , is used for communication protocols in which username/password authentication is required, such as MQTT.
Account Token	Equates to a username+password. The Account Token is used for communication protocols in which tokens are accepted for authentication, such as interfaces utilizing HTTP-based Representational State Transfer (REST).
API Token	
Authentication Token	A base64(AccountID:AccountSecret) that grants connected entities access to the platform.
Authorization Token	Stipulates what communication channels within the platform can be accessed by the associated entity.

Consumer IoT devices will **EXCEED** the number of [smartphones and tablets combined](#) within the **next three years**.

– IDC's Worldwide IoT Spending Guide, Nov. 2016

The analysts at IDC report a **16.1% increase in IoT adoption** among U.S. enterprises in 2016.

– IDC's Vertical IT & Communications Survey, May 2016

QUICKSTART GUIDE

CREATING AN ACCOUNT

To gain access to the qiot.io platform, first navigate to <https://qiot.io>. The login function is administered via your existing GitHub account. When you see the landing page below, simply click "**Connect with GitHub**" and follow the prompts to enter your GitHub info.



Figure 7 – qiot.io Login

AUTHENTICATING WITH THE PLATFORM

You must first have created a qiot.io account through the general account creation process as outlined above. In doing so, you will be assigned:

- Account ID - used in communication protocols like MQTT where a username is required.
- Account Secret - used in communication protocols like MQTT where a password is required.
- Account Token - used in communication protocols such as REST where a token is required.

An Authentication Token is a base64(AccountID:AccountSecret) and allows you to access the platform. Concurrently, the system will issue you an Authorization Token, which

determines what you can access when you get into the platform. All entities must provide both Authentication and Authorization Tokens in order to interact with the platform. That includes all users, Things, Collections, and third-party apps.

Account Tokens used for qiot.io authentication do not expire. These system-generated tokens remain valid as unique identifiers for their associated entity and enable all future authentications and authorizations. Because qiot.io utilizes such a rigorous and precise security structure, strengthened even further by our multi-layered authentication and authorization scheme for every entity that connects to the platform, tokens can continue to be used for the full lifecycle of the platform.

If you suspect a token has been compromised and could potentially expose your business to a security breach, it can be deleted or suspended by any user with Account Administrator privileges. Such action denies authentication to its associated user, device, or app. If necessary, a new token can be generated and then reassigned to the user, device, or app in question, allowing it to again communicate securely with the platform.

NAVIGATING THE DASHBOARD

Qiot.io's intuitive Dashboard allows you to see all Things within your account and assess operational performance in near real-time. The Dashboard, shown below in Figure 8, is the first screen you'll see when you log in to the system and is essentially your visual representation of qiot.io and everything within your account that is connected to it. Within the platform, the Dashboard is denoted with the icon .

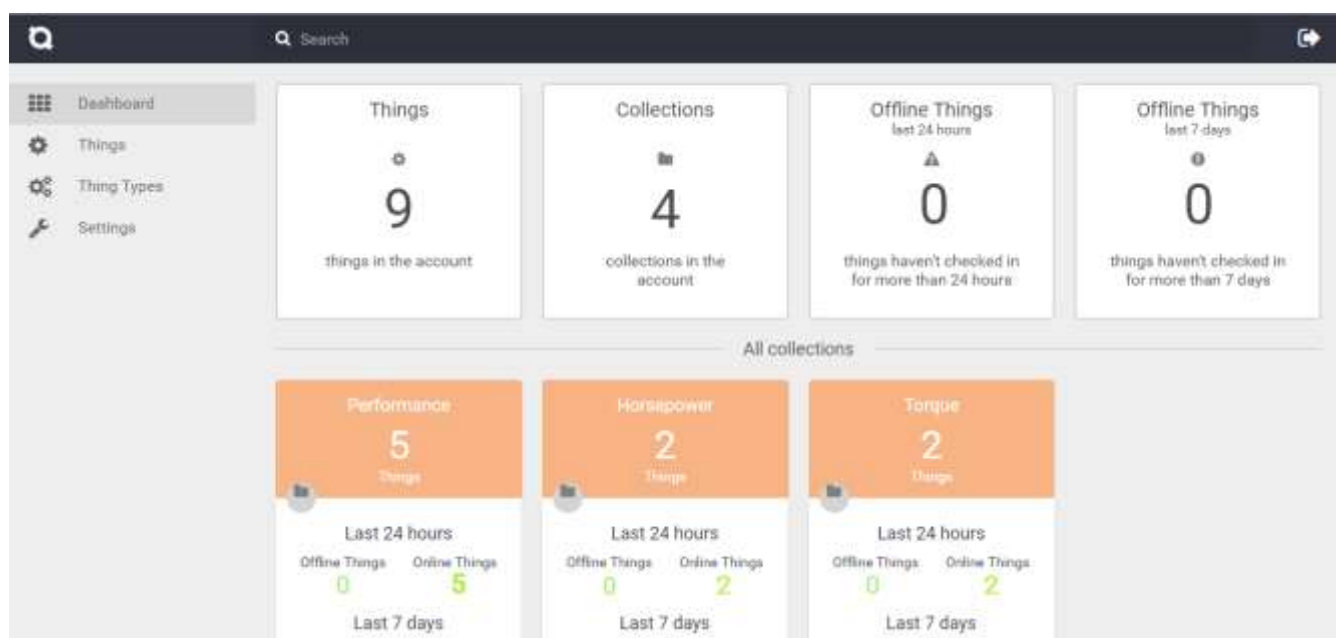


Figure 8 – qiot.io Dashboard

From the Dashboard, you can easily establish how many Things, or devices, are connected to the platform; how they are grouped; and how many Things have gone offline, or lost communication with the platform.

The two boxes at the top-right of the Dashboard with the heading "Offline Things" alert you of potential problems and are denoted with the icons ⚠️ and ❗. To preemptively detect connectivity issues, each box can be programmed to assess Thing activity within a user-specified time period, for example, the last 24 hours or the past 7 days. The system will then identify which connected Things have not successfully executed their scheduled communications during those periods.

You'll notice that by hovering over any of the boxed content on your dashboard, you are given the option of viewing it and its components in greater detail, as shown in Figure 9.

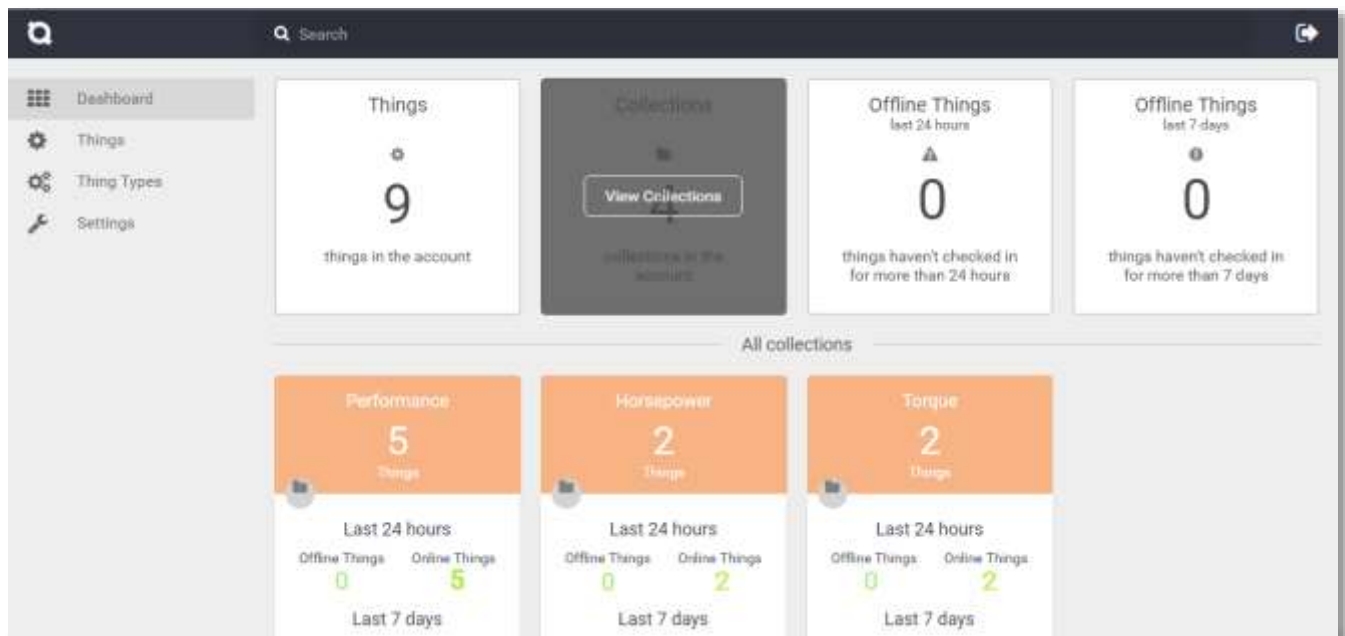
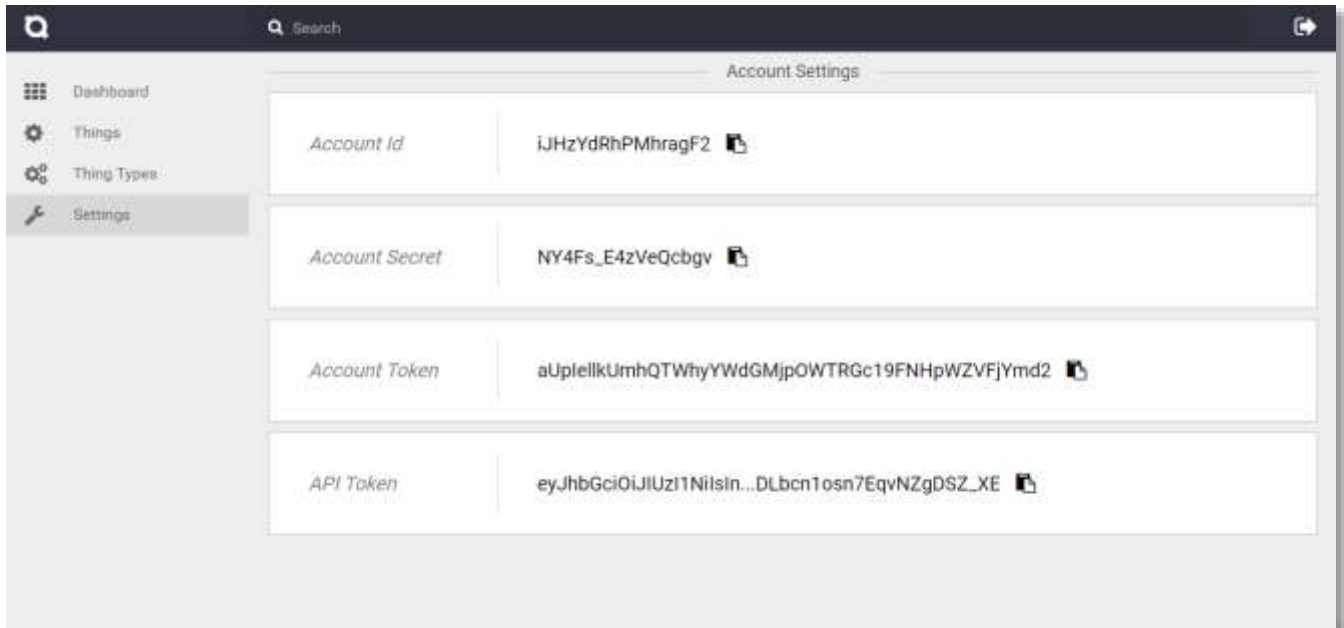


Figure 9 – Accessing Details of Dashboard Modules

Let's take a look at the other items on the left-hand Dashboard menu, starting with the most basic of these elements, your Account "Settings" (designated with the icon 🛠️). Here, you'll find all your user account credentials, including Account ID, Account Secret, Account Token, and API Token (see Figure 10 below). Click on the icon to the right of each one and you can easily copy it to your clipboard.



CONNECTING A THING

Things can be any physical device from which you want to draw data, whether that's a thermostat inside a commercial refrigerator or a new form of bio-reading wearable technology. A Thing can also embody a localized network consisting of many strategically positioned sensors, to monitor an entire city's air quality or traffic congestion, for example.

Qiot.io puts no limitations on the number of Things, or devices, you can connect to the platform. Likewise, there are no constraints on the number of Collections, or groupings of Things, that can be created, nor on the number of applications or services that can utilize the technology and the data it amasses.

To connect a new Thing, simply navigate to "Things" on the left-hand menu bar. Click on the submenu button in the bottom right corner and click the plus (+) symbol to choose "New Thing," as shown in Figure 11.

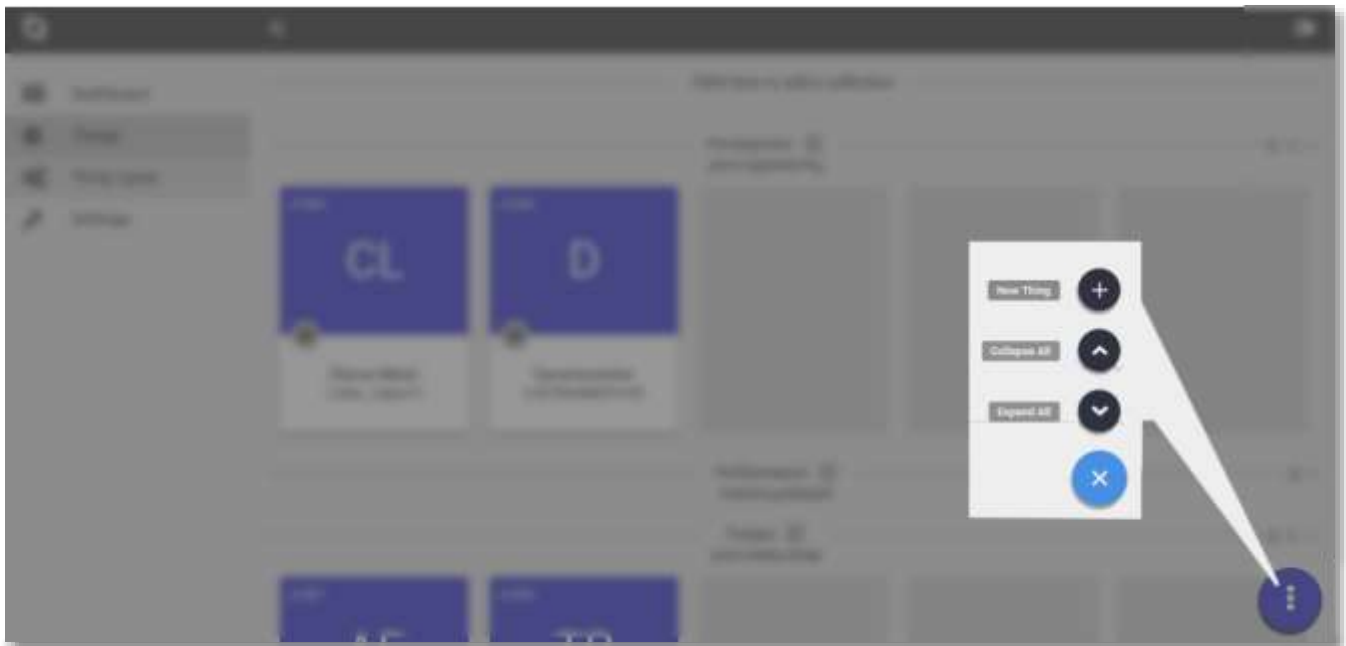


Figure 11 – Connecting a New Thing

This will open the "Create Thing" pop-up window, as depicted in Figure 12 below, where you'll input the Thing details needed to bring your device online.

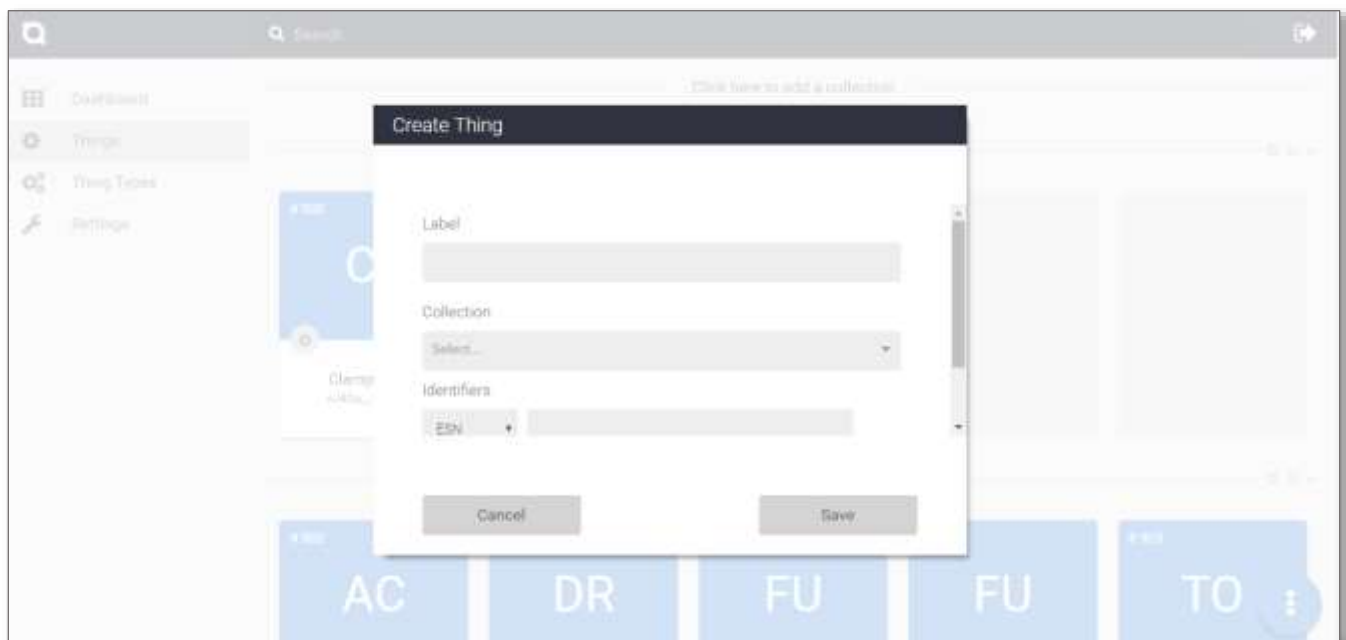


Figure 12 – "Create Thing" Pop-Up Window

Any string of characters may be entered in the Label field as a reminder to the user of the purpose of the Thing. This could be a serial number or any other alphanumeric identifier, or even something as simple as "Bob's Taco Truck." The description you enter here is added to the Thing Name to construct your device's Label.

With the "Collection" drop-down menu, you can simultaneously add your new device to an existing Collection in order to categorize it and group it with similar devices from which you are pulling the same data measurements. Each connected Thing must be associated with a Collection. In order to group any number of Things in a single Collection, you'll need to use the [Thing Collection Subscription API](#). (Note: Things not explicitly associated with a Collection will be automatically assigned to the "Default Collection.")

The Identifier field is key as it provides the platform with the sequence of Unicode characters for recognizing and distinguishing that particular device. The wide range of Identifiers currently supported are defined below and cover all types of mobile handsets; smartphones; satellite phones; subscriber-linked SIM cards; mobile stations, entailing a mobile phone or computer and a mobile broadband adapter; individual Ethernet- and WiFi-connected computers; software keys; and general consumer electronic devices.

- **ESN**: Electronic Serial Number, which is now used primarily to identify CDMA cell phones (previously used for AMPS and TDMA phones). ESNs are often represented as either 11-digit decimal numbers or 8 digit hexadecimal numbers. There have not been any new ESN assignments made since December 31, 2010.
- **ICCID**: Integrated Circuit Card Identifier, the unique serial number stored on and inscribed on the back of individual SIM cards to identify them internationally. A full ICCID is 19 or 20 characters. It's possible to extract the ICCID from the system information under Settings or by using the 'AT+ICCID?' modem command.
- **IMEI**: International Mobile Equipment Identity, a unique number used to identify 3GPP and iDEN mobile phones, as well as some satellite phones. It can usually be located by checking the inside of a phone's battery compartment, but it can also be found in the Settings menu along with other system info and can be pulled up on most phones by entering *#06# on the keypad.
- **IMSI**: International Mobile Subscriber Identity, a unique set of digits (usually 15) associated with Global System for Mobile Communications (GSM) and Universal Mobile Telecommunications System (UMTS) mobile network users. The IMSI is stored on a SIM card that can, in theory, be transferred to any handset and therefore is an identifier associated with a GSM or UMTS subscriber rather than a particular phone.
- **MAC**: Media Access Control address found on individual computers and assigned to network interfaces for communications at the data link layer of a network segment.

MAC addresses are used as a network address for most IEEE 802 network technologies, including Ethernet and Wi-Fi.

- **MEID**: Mobile Equipment Identifier is a globally unique number associated with a mobile device. More specifically, it is used to identify a physical piece of CDMA2000 mobile station equipment. MEID is a form of ESN (Electronic Serial Number) and can have multiple formats, including MEID HEX and MEID DEC.
- **MSISDN**: Mobile Station International Subscriber Directory Number, a set of numerical characters used to identify a mobile phone number internationally. A mobile station comprises all user equipment and software needed for communication with a mobile network, i.e. a mobile phone or mobile computer connected using a mobile broadband adapter. Defined by the E.164 numbering plan, MSISDN includes a country code and a National Destination Code, which identifies the subscriber's mobile operator.
- **SN**: Serial Number, assigned incrementally or sequentially to an item. It is also called a serial number, although it may be a character string that includes letters and other typographical symbols, in addition to numerical digits. Serial numbers may be used to identify individual physical or intangible objects, such as computer software or the right to play an online multiplayer game. A software serial number is often called a product key. Most general consumer electronics can be identified with an SN, found around the base of the item or within its original packaging and documentation.

Some items may have more than one Identifier. For example, an iPhone has both an IMEI and an SN. Within the "Create a Thing" menu, you have the option of associating a second, third, or even more Identifiers with a connected device by selecting "Add Identifier" located directly below the Identifier drop-down menu module (see Figure 13).

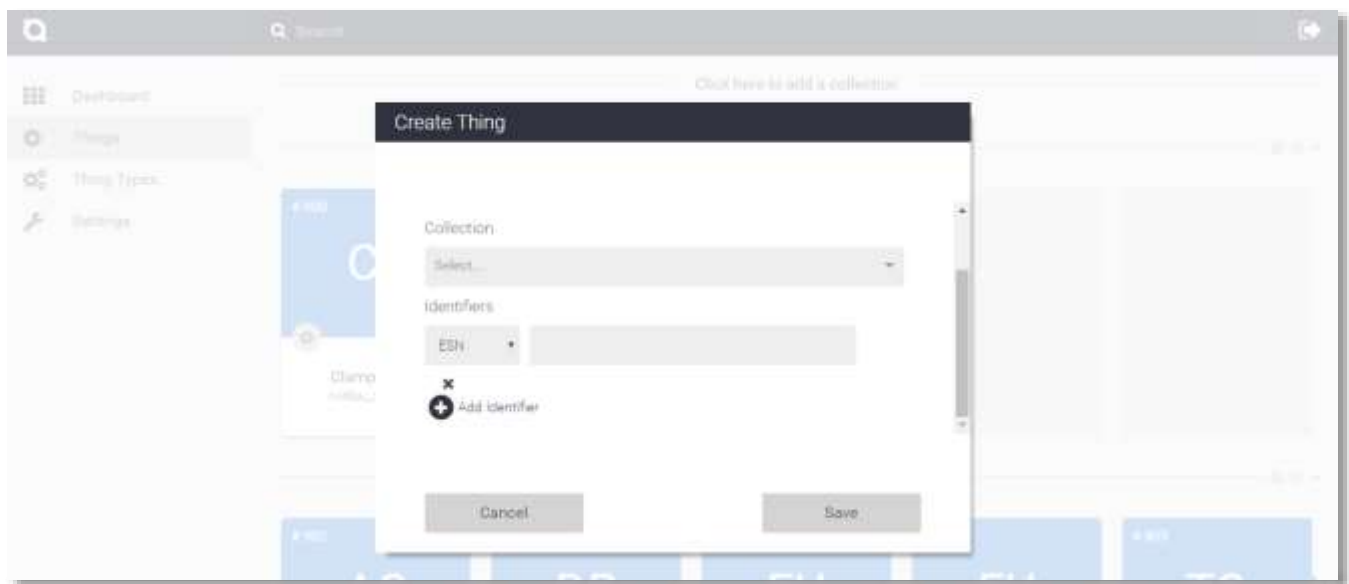


Figure 13 - "Add Identifier" Option for Things

Upon entering all the information necessary to create a Thing, click "Save" and the device will be added to the "Things" index within your account (see Figure 14 below).

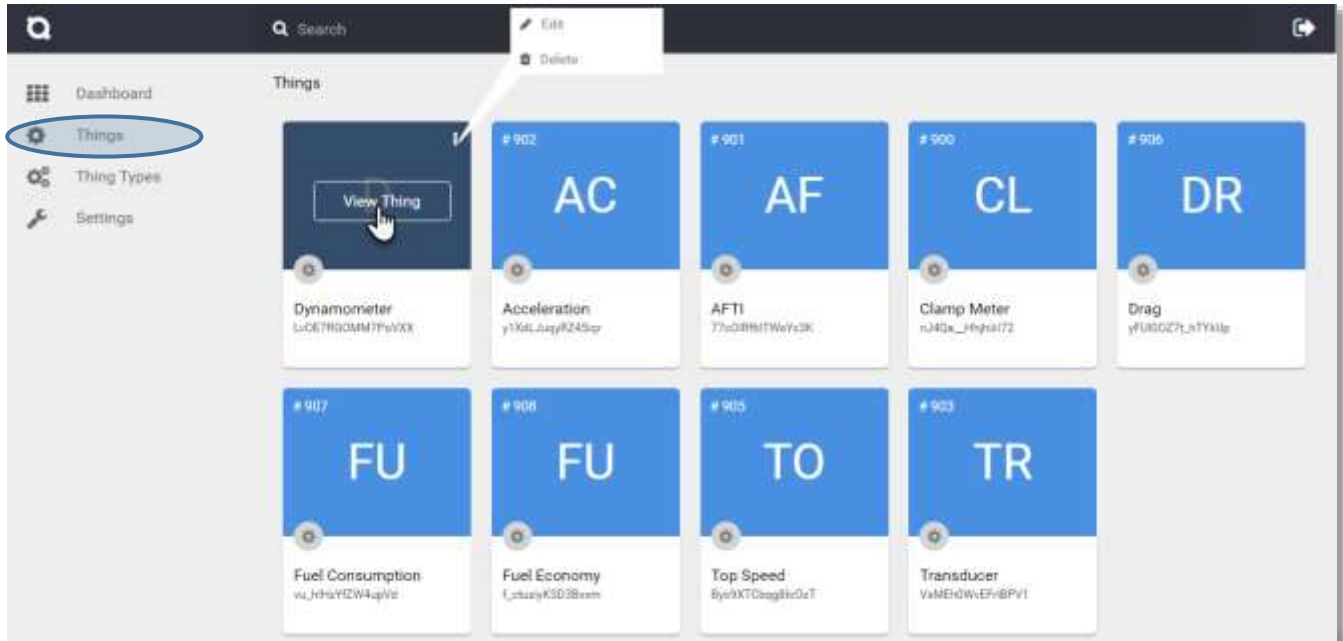


Figure 14 – Things Index

Now that your Thing has been created, the next step is for it to register with the platform. This is accomplished with the [Thing Registration API](#). (Note: At this time, Thing registration is only possible through the analog REST API.)

In response to your Thing's request to register, the platform will issue it three tokens: Thing Authentication; Collection Authentication; and Thing Authorization, which allows the device to communicate back to the platform.

SENDING DATA

All entities—including users, Things, Collections, and third-party apps—must provide both Authentication and Authorization Tokens in order to interact and communicate with the platform. Each unique Authorization Token is what qiot.io uses to identify and correlate communications to and from a specific device.

Currently, two common communication protocols are utilized to relay information between the platform, connected devices, and applications: [Representational State Transfer \(REST\)](#) and [Message Queuing Telemetry Transport \(MQTT\)](#).

All system responses follow the standards-based formats for REST and MQTT and rely on each protocol's universal status codes to provide confirmation that your communication was sent successfully.

READING DATA

With qiot.io, you can combine device and sensor data with metadata, geo-location, environmental information, or any other measureable performance metric. The right data analysis can identify bottlenecks, uncover trends, and predict failure before it's too late.

The platform uses Authorization Tokens to associate a communication with a specific Thing, or connected device. Using the appropriate Thing Token, Team Members can query the system for responses or subscribe to a Thing's specific channel to receive active notifications of performance changes.

Likewise, Collections have Collection Tokens. This allows a user to subscribe to an entire Collection and listen in to what all of the Things in that Collection have to say.

The granularity of your subscription is only as fast as you make the REST call. If a Thing publishes two times per second, but you subscribe one time per second, you'll only be hearing half of the conversation. If you're interested in active listening, see the [MQTT Thing Log API - Subscribe](#) to tap into an individual Thing's communications or the [MQTT Thing Collection Subscription API](#) for the complete Collection.

Any application that interfaces with REST or MQTT and supports Authentication and Authorization tokens can be used for data retrieval and formatting.

By **2020**, the number of

INTERCONNECTED AND OPERATIONAL

IoT devices will reach 23 billion.

– *Frost & Sullivan*

QIOT.IO APIs

The qiot.io platform uses the common communication protocols [Representational State Transfer \(REST\)](#) and [Message Queuing Telemetry Transport \(MQTT\)](#) to relay information between the platform, connected devices, and applications.

REST API GENERAL NOTES

All REST API's are TLS1.2 encrypted and require an HTTP Header QIOT Authorization Account Token for authenticating with the platform. Do not confuse "Authorization" and "Authentication" in this context; unfortunately, the legacy HTTP Header "Authorization" really means "Authentication."

All Things connected to qiot.io are accessed through a channel namespace. For REST this is part of the Uniform Resource Identifier (URI). For example, the qiot.io Thing Registration API may be accessed at:

```
https://api.qiot.io/1/r
```

All channel namespaces mirror the following format:

```
/ {api_version} / {action} /
```

Some channels include a designator for a separate namespace identity, such as a Thing Token or a Collection Token. For example, to publish to a Thing's channel namespace using the REST API, a Thing or user would post to:

```
https://api.qiot.io/1/l/{thing_token}
```

REST APIs follow standard [HTTP response codes](#), which indicate the result of the HTTP request:

- ↪ 1XX - informational
- ↪ 2XX - success
- ↪ 3XX - redirection
- ↪ 4XX - client error

→ 5XX - server error

For example, if your upload was completely and successfully sent, you'll see:

```
93 out of 93 bytes HTTP/1.1 200 OK -or- HTTP/1.1 204 OK
```

MQTT API GENERAL NOTES

The MQTT implementation of qiot.io takes advantage of three forms of security:

- 1) Authentication via username and password
- 2) Thing and Collection Tokens
- 3) Optional TLS 1.2 transport-level encryption

All connected Things are accessed through a channel namespace. When utilizing MQTT, this is reflected in the MQTT topic. For example, the qiot.io Thing Log API may be accessed at:

```
https://api.qiot.io/1/l/{thing_token}
```

All channel namespaces take the following form:

```
/ {api_version} / {action} /
```

Some channels include a designator for a separate namespace identity, such as a Thing Token or a Collection Token. For example, to publish to a Thing's channel namespace using the MQTT API, a Thing or user posts to:

```
https://api.qiot.io/1/l/{thing_token}
```

MQTT brokers are hosted on standard ports at 1883 and 8883, respectively. All publish/subscribe examples provided in this document make use of the [Mosquitto MQTT broker](#).



THING REGISTRATION API

Registration is the process by which a Thing is created within qiot.io and associated with a user account. The Thing Registration API must be authenticated using a preexisting Authentication Token.

> REST DEFINITION:

```
curl -H "Content-Type: application/json" -H "Authorization: QIOT ${account_token}" -X POST "https://api.qiot.io/1/r" -d '{
"label": "rest-log-1persec", "identity": [ {"type": "IMEI",
"value": "123456789012345"} ] }'
```

- ↪ {account_token} = token assigned to your account through the account creation process.
- ↪ "label" = Thing name plus the free form description that was added by the user when Thing was created.
- ↪ "identity" = sequence of Unicode characters used to recognize and distinguish a device. Valid identifier types are currently ESN, ICCID, IMEI, IMSI, MAC, MEID, MSISDN, and SN. Any combination of meaningful identifiers may be presented in the registration request, but they will be treated as unique within the system. No two Things may have the same set of identifiers.

> REST EXAMPLE:

```
curl -vvv -H "Content-Type: application/json" -H
"Authorization: QIOT ZHFtUmRvakdyYUppUXlxdDpYRllxRU5aYWFrUll6TEV5"
-X POST "https://api.qiot.io/1/r" -d '{ "label": "rest-log-
1persec", "identity": [ {"type": "IMEI", "value":
"123456789012345"} ] }'
```

> REST RESPONSE:

```
* Trying 54.83.79.16...
* TCP_NODELAY set
* Connected to api.qiot.io (54.83.79.16) port 443 (#0)
* TLS 1.2 connection using
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: qiot.io
* Server certificate: Let's Encrypt Authority X3
```

```

* Server certificate: DST Root CA X3
> POST /1/r HTTP/1.1
> Host: api.qiot.io
> User-Agent: curl/7.51.0
> Accept: */*
> Content-Type: application/json
>                                     Authorization: QIOT
ZHfUmRvakdyYUppUXlxdDpYRllxRU5aYWFrUll6TEV5
> Content-Length: 93
>
* upload completely sent off: 93 out of 93 bytes
< HTTP/1.1 200 OK
< Server: nginx/1.11.8
< Date: Tue, 03 Jan 2017 20:36:39 GMT
< Transfer-Encoding: chunked
< Connection: keep-alive
< Strict-Transport-Security: max-age=15768000
<
* Curl_http_done: called premature == 0
* Connection #0 to host api.qiot.io left intact

{
  "status": "success",
  "thing": {
    "label": "rest-log-1persec",
    "updated_at": "2017-01-03T20:36:39.852Z",
    "created_at": "2017-01-03T20:36:39.852Z",

"account_token": "ZHfUmRvakdyYUppUXlxdDpYRllxRU5aYWFrUll6TEV5",
    "collection_token": "1TV2Qp3nfpZw_2Ts",
    "thing_token": "0WdyHxn17jx-PxZ"
  }
}

- "collection_token" = token of the Collection a Thing was assigned to during the
  registration process.
- "thing_token" = token assigned to a particular Thing as part of the registration
  process.

```

The following diagram illustrates how Things interact with qiot.io during the registration process:

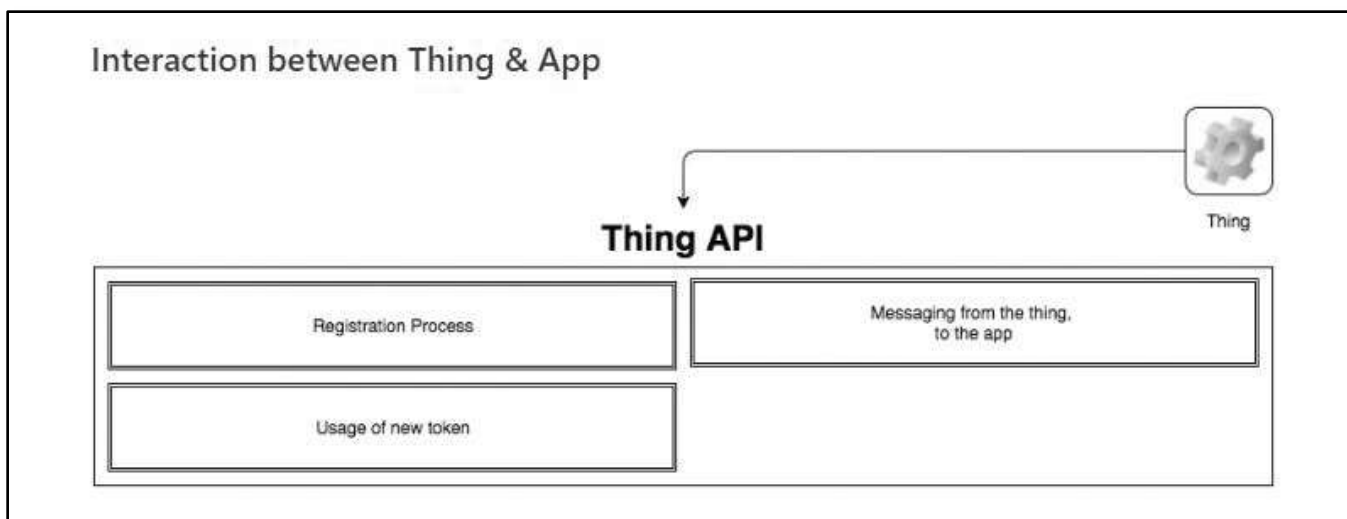


Figure 15 - Thing and Platform Interaction

NOTE: At this time, Thing registration is only possible through the analog REST API.



THING MAILBOX API

Mailbox is a special namespace channel designated for a Thing where messages destined for the Thing can be placed. A Thing can be designed to either periodically or at its own discretion listen to the mailbox channel for commands, controls, and other messages to be acted on locally.

> REST DEFINITION:

```
curl -v -H "Content-Type: application/json" -H "Authorization:
QIOT          ${account_token}"          -X          GET
"https://api.qiot.io/1/m/${thing_token}"
```

- ↪ {account_token} = token assigned to your account through the account creation process.
- ↪ {thing_token} = token assigned to the Thing when it registered.

> REST EXAMPLE:

```
curl -v -H "Content-Type: application/json" -H "Authorization:
QIOT ZHFtUmRvakdyYUppUXlxdDpYRllxRU5aYWFrUll6TEV5" -X GET
"https://api.qiot.io/1/m/0WdyyHxnl7jx-PxZ"
```

> REST RESPONSE:

```
* Trying 54.83.79.16...
* TCP_NODELAY set
* Connected to api.qiot.io (54.83.79.16) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: qiot.io
* Server certificate: Let's Encrypt Authority X3
* Server certificate: DST Root CA X3
> GET /1/l/0WdyyHxnl7jx-PxZ HTTP/1.1
> Host: api.qiot.io
> User-Agent: curl/7.51.0
> Accept: */*
> Content-Type: application/json
> Authorization: QIOT ZHFtUmRvakdyYUppUXlxdDpYRllxRU5aYWFrUll6TEV5
>
< HTTP/1.1 200 OK
< Server: nginx/1.11.8
< Date: Tue, 03 Jan 2017 21:39:54 GMT
< Transfer-Encoding: chunked
< Connection: keep-alive
< Strict-Transport-Security: max-age=15768000
<
* Curl_http_done: called premature == 0
* Connection #0 to host api.qiot.io left intact

{
  "messages":[
    {
      "time":"2017-01-03T14:05:40-0600",
      "command": "fota"
    }
  ],
  "content":{
    "encoding":"json"
  }
}
```

> MQTT DEFINITION:

```
mosquitto_sub -d -h api.qiot.io -p 1883 -u "${account_id}" -P  
"${account_secret}" -t '1/m/${thing_token}'
```

- {account_id} = username assigned to your account through the account creation process.
- {account_secret} = password assigned to your account through the account creation process.
- {thing_token} = token assigned to the Thing when it registered with qiot.io.

> MQTT EXAMPLE:

```
mosquitto_sub -d -h api.qiot.io -p 1883 -u "nYga05yWURq9dgew" -P  
"mX4DCBG-Wjc2Auye" -t '1/m/FsHGCmtyu2aycUkK'
```

> MQTT RESPONSE:

```
Client mosqsub/84550-five-tons sending CONNECT  
Client mosqsub/84550-five-tons received CONNACK  
Client mosqsub/84550-five-tons sending SUBSCRIBE (Mid: 1,  
Topic: 1/l/FsHGCmtyu2aycUkK, QoS: 0)  
Client mosqsub/84550-five-tons received SUBACK  
Subscribed (mid: 1): 0  
  
Client mosqsub/84550-five-tons received PUBLISH (d0, q1, r0,  
m1, '1/l/FsHGCmtyu2aycUkK', ... (302 bytes))  
Client mosqsub/84550-five-tons sending PUBACK (Mid: 1)  
{ "messages": [ {"cpu_percent": 12.16, "total_processes": 365,  
"running_processes": 364, "stuck_processes": 6,  
"sleeping_processes": 359, "thread_count": 1881, "load_average":  
1.98, "processor_type": "x86_64h (Intel x86-64h Haswell)",  
"active_processors": "0 1 2 3", "memory_available": "8.00 GB" } ]  
}  
  
Client mosqsub/84550-five-tons received PUBLISH (d0, q0, r0,  
m0, '1/l/FsHGCmtyu2aycUkK', ... (302 bytes))  
{ "messages": [ {"cpu_percent": 15.67, "total_processes": 365,  
"running_processes": 371, "stuck_processes": 3,  
"sleeping_processes": 359, "thread_count": 1890, "load_average":
```

```
2.08, "processor_type": "x86_64h (Intel x86-64h Haswell)",
"active_processors": "0 1 2 3", "memory_available": "8.00 GB" } ]
}
```

[... MORE STREAMING MESSAGES ...]

```
Client mosqsub/84550-five-tons received PUBLISH (d0, q0, r0,
m0, '1/l/FsHGcmtyu2aycUkK', ... (302 bytes))
```

```
{ "messages": [ {"cpu_percent": 48.55, "total_processes": 354,
"running_processes": 354, "stuck_processes": 2,
"sleeping_processes": 352, "thread_count": 1900, "load_average":
2.15, "processor_type": "x86_64h (Intel x86-64h Haswell)",
"active_processors": "0 1 2 3", "memory_available": "8.00 GB" } ]
}
```



THING LOG API - PUBLISH

Log is the process by which a Thing can publish data to the qiot.io platform. A Thing must be registered before it can log.

> REST DEFINITION - PUBLISH:

```
curl -v -H "Content-Type: application/json" -H "Authorization:
QIOT ${account_token}" -X POST
"https://api.qiot.io/1/l/${thing_token}" -d '${messages}'
```

- {account_token} = token assigned to your account through the account creation process.
- {thing_token} = token assigned to the Thing when it registered.
- {messages} = json-formatted communication(s) the Thing wants to post to the platform, given in the following form:

```
{
  "messages": [
    {
      "time": "2017-01-03T14:05:40-0600",
      "foo": "bar",
      "num": 1
    },
    {
```

```

        "time": "2017-01-03T14:45:40-0600",
        "foo": "bar",
        "num": 1
    }
}

```

A message array may contain one or more messages. It is good form to include a timestamp for each message within the array.

> REST EXAMPLE - PUBLISH:

```

curl -v -H "Content-Type: application/json" -H "Authorization:
QIOT ZHFtUmRvakdyYUppUXlxdDpYRllxRU5aYWFrUll6TEV5" -X POST
"https://api.qiot.io/1/l/0WdyyHxnl7jx-PxZ" -d '{ "messages": [
{"time": "2017-01-03T14:05:40-0600", "cpu_percent": 21.39,
"total_processes": 327, "running_processes": 327,
"stuck_processes": 6, "sleeping_processes": 326, "thread_count":
1962, "load_average": 21.25, "processor_type": "x86_64h (Intel x86-
64h Haswell)", "active_processors": "0 1 2 3", "memory_available":
"8.00 GB" } ] }'

```

> REST RESPONSE - PUBLISH:

```

* Trying 54.83.79.16...
* TCP_NODELAY set
* Connected to api.qiot.io (54.83.79.16) port 443 (#0)
* TLS 1.2 connection using
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: qiot.io
* Server certificate: Let's Encrypt Authority X3
* Server certificate: DST Root CA X3
> POST /1/l/0WdyyHxnl7jx-PxZ HTTP/1.1
> Host: api.qiot.io
> User-Agent: curl/7.51.0
> Accept: */*
> Content-Type: application/json
> Authorization: QIOT
ZHFtUmRvakdyYUppUXlxdDpYRllxRU5aYWFrUll6TEV5
> Content-Length: 339
>

```

```

* upload completely sent off: 339 out of 339 bytes
< HTTP/1.1 204 No Content
< Server: nginx/1.11.8
< Date: Tue, 03 Jan 2017 21:22:35 GMT
< Connection: keep-alive
< Location: /1/l/0WdyyHxnl7jx-PxZ
< Strict-Transport-Security: max-age=15768000
<
* Curl_http_done: called premature == 0
* Connection #0 to host api.qiot.io left intact

```

Note that there is no body in the above response. A success message that doesn't warrant a response will only contain the HTTP headers necessary to indicate success. For example:

```
HTTP/1.1 204 No Content
```

> MQTT DEFINITION - PUBLISH:

```

mosquitto_pub -d -r -u "${account_id}" -P "${account_secret}" -h
api.qiot.io -p 1883 -q 1 -i '${thing_token}' -t
'1/l/${thing_token}' -m "$messages"

```

- {account_id} = username assigned to your account through the account creation process.
- {account_secret} = password assigned to your account through the account creation process.
- {thing_token} = token assigned to the Thing when it registered with qiot.io
- "messages" = json-formatted communication(s) your thing would like to post into the platform, presented in the following form:

```

{
  "messages": [
    {
      "time": "2017-01-03T14:05:40-0600",
      "foo": "bar",
      "num": 1
    },
    {
      "time": "2017-01-03T14:45:40-0600",
      "foo": "bar",

```

```

        "num": 1
    }
}

```

A message array may contain one or more messages. It is good form to include a timestamp with each message within the array.

> MQTT EXAMPLE - PUBLISH:

```

mosquitto_pub -d -r -u "nYga05yWURq9dgew" -P "mX4DCBG-Wjc2Auye" -h
api.qiot.io -p 1883 -q 1 -i 'FsHGCmtyu2aycUkK' -t
'1/l/FsHGCmtyu2aycUkK' -m "{ \"messages\": [ {\"time\": \"2017-01-
03T14:05:40-0600\", \"cpu_percent\": 21.39, \"total_processes\": 327,
\"running_processes\": 327, \"stuck_processes\": 6,
\"sleeping_processes\": 326, \"thread_count\": 1962, \"load_average\":
21.25, \"processor_type\": \"x86_64h (Intel x86-64h Haswell)\",
\"active_processors\": \"0 1 2 3\", \"memory_available\": \"8.00 GB\" } ]
}"

```

> MQTT RESPONSE - PUBLISH:

```

Client FsHGCmtyu2aycUkK sending CONNECT
Client FsHGCmtyu2aycUkK received CONNACK
Client FsHGCmtyu2aycUkK sending PUBLISH (d0, q1, r1, m1,
'1/l/FsHGCmtyu2aycUkK', ... (339 bytes))
Client FsHGCmtyu2aycUkK received PUBACK (Mid: 1)
Client FsHGCmtyu2aycUkK sending DISCONNECT

```



THING LOG API - SUBSCRIBE

A Log namespace channel can also be subscribed to, allowing authorized listeners to monitor what a Thing is saying.

> REST DEFINITION - SUBSCRIBE:

```

curl -v -H "Content-Type: application/json" -H "Authorization:
QIOT ${account_token}" -X GET
"https://api.qiot.io/1/l/${thing_token}"

```

- {account_token} = token assigned to your account through the account creation process.
- {thing_token} = token assigned to the Thing when it registered.

NOTE: The granularity of your subscription is only as fast as you make the REST call. If a Thing publishes 2 times per second, but you subscribe 1 time per second, you'll only be hearing half of the conversation. If you're interested in active listening, see the [MQTT Thing Log API - Subscribe](#).

> REST EXAMPLE - SUBSCRIBE:

```
curl -v -H "Content-Type: application/json" -H "Authorization:
QIOT ZHFtUmRvakdyYUppUXlxdDpYRllxRU5aYWFrUll6TEV5" -X GET
"https://api.qiot.io/1/l/0WdyyHxnl7jx-PxZ"
```

> REST RESPONSE - SUBSCRIBE:

```
* Trying 54.83.79.16...
* TCP_NODELAY set
* Connected to api.qiot.io (54.83.79.16) port 443 (#0)
* TLS 1.2 connection using
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: qiot.io
* Server certificate: Let's Encrypt Authority X3
* Server certificate: DST Root CA X3
> GET /1/l/0WdyyHxnl7jx-PxZ HTTP/1.1
> Host: api.qiot.io
> User-Agent: curl/7.51.0
> Accept: */*
> Content-Type: application/json
> Authorization: QIOT
ZHFtUmRvakdyYUppUXlxdDpYRllxRU5aYWFrUll6TEV5
>
< HTTP/1.1 200 OK
< Server: nginx/1.11.8
< Date: Tue, 03 Jan 2017 21:39:54 GMT
< Transfer-Encoding: chunked
< Connection: keep-alive
< Strict-Transport-Security: max-age=15768000
```

```

<
* Curl_http_done: called premature == 0
* Connection #0 to host api.qiot.io left intact
{
  "messages":[
    {
      "time":"2017-01-03T14:05:40-0600",
      "cpu_percent":21.39,
      "total_processes":327,
      "running_processes":327,
      "stuck_processes":6,
      "sleeping_processes":326,
      "thread_count":1962,
      "load_average":21.25,
      "processor_type":"x86_64h (Intel x86-64h Haswell)",
      "active_processors":"0 1 2 3",
      "memory_available":"8.00 GB"
    }
  ],
  "content":{
    "encoding":"json"
  }
}

```

> MQTT DEFINITION - SUBSCRIBE:

```

mosquitto_sub -d -h api.qiot.io -p 1883 -u "${account_id}" -P
"${account_secret}" -t '1/1/${thing_token}'

```

- {account_id} = username assigned to your account through the account creation process.
- {account_secret} = password assigned to your account through the account creation process.
- {thing_token} = token assigned to the Thing when it registered with qiot.io.

NOTE: For as long as you are attached to the subscription channel of the MQTT broker, you'll get streaming notifications of all data delivered to that channel.

> MQTT EXAMPLE - SUBSCRIBE:

```
mosquitto_sub -d -h api.qiot.io -p 1883 -u "nYga05yWURq9dgew" -P  
"mX4DCBG-Wjc2Auye" -t '1/l/FsHGCmtyu2aycUkK'
```

> MQTT RESPONSE - SUBSCRIBE:

```
Client mosqsub/84550-five-tons sending CONNECT  
  Client mosqsub/84550-five-tons received CONNACK  
  Client mosqsub/84550-five-tons sending SUBSCRIBE (Mid: 1,  
Topic: 1/l/FsHGCmtyu2aycUkK, QoS: 0)  
  Client mosqsub/84550-five-tons received SUBACK  
  Subscribed (mid: 1): 0  
  
  Client mosqsub/84550-five-tons received PUBLISH (d0, q1, r0,  
m1, '1/l/FsHGCmtyu2aycUkK', ... (302 bytes))  
  Client mosqsub/84550-five-tons sending PUBACK (Mid: 1)  
  { "messages": [ {"cpu_percent": 12.16, "total_processes": 365,  
"running_processes": 364, "stuck_processes": 6,  
"sleeping_processes": 359, "thread_count": 1881, "load_average":  
1.98, "processor_type": "x86_64h (Intel x86-64h Haswell)",  
"active_processors": "0 1 2 3", "memory_available": "8.00 GB" } ]  
}  
  
  Client mosqsub/84550-five-tons received PUBLISH (d0, q0, r0,  
m0, '1/l/FsHGCmtyu2aycUkK', ... (302 bytes))  
  { "messages": [ {"cpu_percent": 15.67, "total_processes": 365,  
"running_processes": 371, "stuck_processes": 3,  
"sleeping_processes": 359, "thread_count": 1890, "load_average":  
2.08, "processor_type": "x86_64h (Intel x86-64h Haswell)",  
"active_processors": "0 1 2 3", "memory_available": "8.00 GB" } ]  
}  
  
  [ ... MORE STREAMING MESSAGES ... ]  
  
  Client mosqsub/84550-five-tons received PUBLISH (d0, q0, r0,  
m0, '1/l/FsHGCmtyu2aycUkK', ... (302 bytes))  
  { "messages": [ {"cpu_percent": 48.55, "total_processes": 354,  
"running_processes": 354, "stuck_processes": 2,  
"sleeping_processes": 352, "thread_count": 1900, "load_average":  
2.15, "processor_type": "x86_64h (Intel x86-64h Haswell)",
```

```
"active_processors": "0 1 2 3", "memory_available": "8.00 GB" } ]
}
```



THING COLLECTION SUBSCRIPTION API

Collections have Collection Tokens. This allows a user to subscribe to an entire Collection and listen in to what all of the Things in that Collection have to say. Other than the namespace channel designation, this API behaves identically to the Thing Log API - Subscribe.

> REST DEFINITION:

```
curl -v -H "Content-Type: application/json" -H "Authorization:
QIOT      ${account_token}"      -X      GET
"https://api.qiot.io/1/c/${collection_token}"
```

- {account_token} = token assigned to your account through the account creation process.
- {collection_token} = token assigned to the Collection when it was created.

> REST EXAMPLE:

```
curl -v -H "Content-Type: application/json" -H "Authorization:
QIOT      ZHFtUmRvakdyYUppUXlxdDpYRllxRU5aYWFrUll6TEV5"      -X      GET
"https://api.qiot.io/1/l/1TV2Qp3nfpZw_2Ts"
```

> REST RESPONSE:

```
*    Trying 54.83.79.16...
* TCP_NODELAY set
* Connected to api.qiot.io (54.83.79.16) port 443 (#0)
*      TLS      1.2      connection      using
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: qiot.io
* Server certificate: Let's Encrypt Authority X3
* Server certificate: DST Root CA X3
> GET /1/l/1TV2Qp3nfpZw_2Ts HTTP/1.1
> Host: api.qiot.io
```

```

> User-Agent: curl/7.51.0
> Accept: */*
> Content-Type: application/json
>                               Authorization: QIOT
ZHFtUmRvakdyYUppUXlxdDpYRllxRU5aYWFrUll6TEV5
>
< HTTP/1.1 200 OK
< Server: nginx/1.11.8
< Date: Tue, 03 Jan 2017 21:39:54 GMT
< Transfer-Encoding: chunked
< Connection: keep-alive
< Strict-Transport-Security: max-age=15768000
<
* Curl_http_done: called premature == 0
* Connection #0 to host api.qiot.io left intact

{
  "messages":[
    {
      "time":"2017-01-03T14:05:40-0600",
      "cpu_percent":21.39,
      "total_processes":327,
      "running_processes":327,
      "stuck_processes":6,
      "sleeping_processes":326,
      "thread_count":1962,
      "load_average":21.25,
      "processor_type":"x86_64h (Intel x86-64h Haswell)",
      "active_processors":"0 1 2 3",
      "memory_available":"8.00 GB"
    }
  ],
  "content":{
    "encoding":"json"
  }
}

```

NOTE: The granularity of your subscription is only as fast as you make the REST call. If you are interested in active listening, see the [MQTT Thing Collection Subscription API](#) below.

> MQTT DEFINITION:

```
mosquitto_sub -d -h api.qiot.io -p 1883 -u "${account_id}" -P  
"${account_secret}" -t '1/c/${thing_token}'
```

- {account_id} = username assigned to your account through the account creation process.
- {account_secret} = password assigned to your account through the account creation process.
- {thing_token} = token assigned to the Thing when it registered with qiot.io.

NOTE: For as long as you are attached to the subscription channel of the MQTT broker, you'll get streaming notifications of all data delivered to that channel.

> MQTT EXAMPLE:

```
mosquitto_sub -d -h api.qiot.io -p 1883 -u "nYga05yWURq9dgew" -P  
"mX4DCBG-Wjc2Auye" -t '1/c/FsHGGcmtyu2aycUkK'
```



DEVELOPER TERMS

Any use or application of the guidelines, examples, and other content provided here by QuantumIOT is subject to and governed by the company's Developer Policy and Client Service-Level Agreement.

QUANTUMIOT SUPPORT

 contact@quantumiot.com

 **555.555.5555**

 github.com/quantumiot

 hub.docker.com/r/qiot/