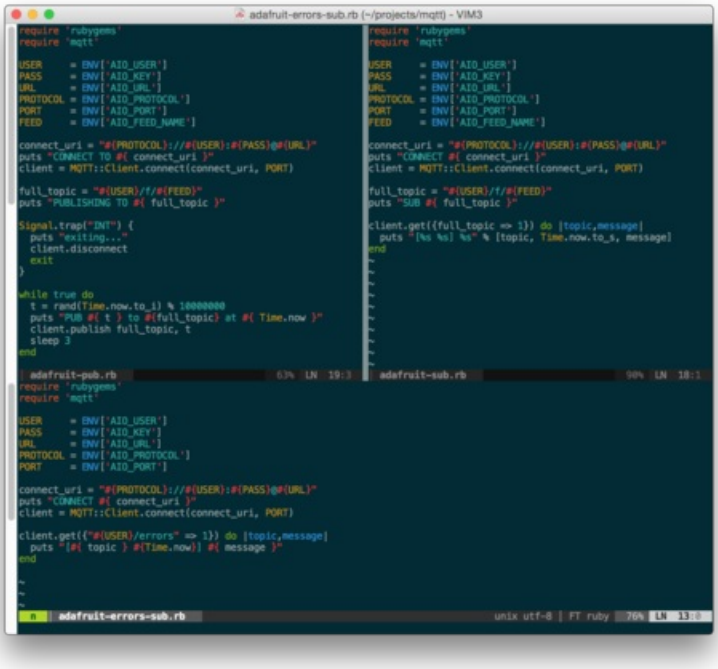
adafruit learning system

Naming Things in Adafruit IO

Created by Adam Bachman



Last updated on 2020-01-31 08:42:15 PM UTC

Introduction

There's an old joke that goes something like this:

> There are two hard things in computer science: cache invalidation, naming things, and off-by-one errors.
>
> — Jeff Atwood (@codinghorror) August 31, 2014 (https://adafru.it/oXD)

Fortunately this guide doesn't cover cache-invalidation.

It's taken us a bit of time to get a handle on how we give things inside Adafruit IO names, but we're making progress. Our primary goal in continuing work on Adafruit IO and the supporting client code is to make the MQTT / HTTP API easier to understand, anticipate, and use.

Let's take a look at how we identify Feeds in Adafruit IO and how the rules we've set up will affect your code.

The Two Feed Identifiers

Feeds have two properties that we regularly interact with when we're talking to the Adafruit IO API: **Name** and **Key**.

Name

The **Name** is the user provided name for this Feed. It should be a "human readable" descriptive term that helps you find your Feed in the web-based user interface (https://adafru.it/fsU) and keep track of which code is talking to what feed.

The rules for Feed names are:

- A Feed Name **MUST** include at least one ASCII letter.
- A Name **MAY** include any upper or lower case ASCII letters, numbers, spaces, dashes, or underscores (" ", "-", or "_", respectively).
- A new Feed Name **MAY NOT** be the same as an existing Feed's Key if the Feeds are in the same group. By default, all newly created Feeds are part of the Group whose name matches your username.
- Names are case-sensitive. This means "Blender" and "BLENDER" are treated as two different Names.

Names are helpful because they let us associate a human friendly word with our data. We *see* names when we browse io.adafruit.com on the web and when we get Feed records from the HTTP API. We *use* names when subscribing or publishing to a Feed with the MQTT API.

Some examples of valid, possibly useful names are:

- `Temperature`
- `door one`
- `99 Red Balloons`
- `books_I_would_like_to_read_before_2022`

Key

The **Key** is a system-generated, URL-safe identifier based on the given Feed Name that can be used in API requests to refer to a particular Feed. Keys are generated based on the Name given when the Feed is created and follows strict rules. The rules for Feed keys are simple:

- A Feed Key **MAY ONLY** contain lower case ASCII letters, numbers, and the dash character ("-").
- Two Feeds in the same Group may not have the same Key.

These rules in combination with the default Group all Feeds are added to means means a new Feed cannot be created if it will use a duplicate Key and a Feed's Name cannot be modified if the new Name will produce a Key that conflicts with another Feed in any of the Feed's Groups.

The rules Adafruit IO uses to generate Keys from Names are roughly:

1. Remove formatting. This step requires a lot of discrete operations, but boils down to transliterating Unicode to ASCII and replacing any non-URL safe characters with "-".
2. Collapse whitespace and replace with "-".
3. Collapse all instances of "-" into a single "-" and remove them from the beginning and end of the string.
4. Make the whole thing lowercase.

It's also important to note that **when you change a Feed's Name the Key will not automatically update**. If you would like to keep the Key in sync with the Name, you'll need to update both of the attributes.

Keys are handy because they let us use a human friendly URL when communicating with the AIO API. For example, `https://io.adafruit.com/abachman/feeds/beta-test` and `abachman/f/beta-test` are nicer and easier to remember than `https://io.adafruit.com/abachman/feeds/588995` or `abachman/f/588995`.

Aside: Naming things in MQTT

The official MQTT protocol (https://adafru.it/f29) has its own rules for naming things and in MQTT the things we're concerned with are called "Topic Names". If you read Todd's blog post on MQTT in Adafruit IO (https://adafru.it/oXE), you'll know we are *like* an MQTT broker, but we've got some extra guidelines.

You don't need to memorize the official rules, we handle it for you under the hood. But here they are for illustration:

- All Topic Names and Topic Filters MUST be at least one character long
- Topic Names and Topic Filters are case sensitive
- Topic Names and Topic Filters can include the space character
- A leading or trailing '/' creates a distinct Topic Name or Topic Filter
- A Topic Name or Topic Filter consisting only of the '/' character is valid
- Topic Names and Topic Filters MUST NOT include the null character (Unicode U+0000)
- Topic Names and Topic Filters are UTF-8 encoded strings, they MUST NOT encode to more than 65535 bytes

> Retrieved from the MQTT Version 3.1.1 OASIS Standard (https://adafru.it/oXF), July 8, 2016.

The full MQTT topic used to describe a Feed in Adafruit IO is in the form, `username/feeds/identifier` where `username` should be replaced with the username of the account that owns the Feed and `identifier` should be replaced with the **Name or Key** that uniquely identifies the Feed you're talking about.

So, MQTT considers the whole topic `test_username/feeds/identifier` when validating names but for the purposes of describing Feeds, Adafruit IO is only considering the `identifier` portion.

Naming and Accessing Feeds With the MQTT API

Naming a Feed on the fly and then referring to it reliably can be tricky. Here are the rules we're using right now to generate new Feeds and provide continuing access to them from the MQTT interface. For the purposes of demonstration, we'll be using the example code provided here (https://adafru.it/oYa), but any MQTT publisher or subscriber code should work the same.

1. Listening

Start an MQTT subscription to topic in the form `username/f/identifier`, for the purpose of the following examples I'll be using, `test_username/f/Test Mode`. A Feed with the name "Test Mode" doesn't exist yet, but that's okay with the MQTT API. The subscription will remain active and start receiving whenever you start publishing to a Feed whose Name or Key matches the given `identifier` value exactly.

NOTE: no new Feeds are created in this step.

```
$ AIO_FEED_NAME='Test Mode' ruby adafruit-errors-sub.rb
CONNECT TO mqtts://test_username:12345@io.adafruit.com
SUB test_username/f/Test Mode
```

We'll also start an MQTT subscriber listening to `test_username/errors`. This will let us see when there are problems with publishing or subscribing to Feeds.

```
$ ruby adafruit-errors-sub.rb
CONNECT TO mqtts://test_username:12345@io.adafruit.com
```

2. Initial MQTT publish / creating a new Feed

To create the Feed in Adafruit IO and to start populating it with data, we'll need to publish and MQTT message to the appropriate topic. In this case, we're subscribing to a Feed named "Test Mode", so we'll need to publish on a Feed with the same name.

Using the example script provided, we'll publish a simple MQTT message with the topic `test_username/f/Test Mode`:

```
$ AIO_FEED_NAME='Test Mode' ruby adafruit-pub.rb
CONNECT TO mqtts://test_username:12345@io.adafruit.com
PUBLISHING TO test_username/f/Test Mode
PUB 2609815 to test_username/f/Test Mode at 2016-07-11 12:53:23 -0400
```

If this is your first time publishing to the Feed, the subscriber that's listing to `test_username/f/Test Mode` should receive its first message:

```
[test_username/f/Test Mode 2016-07-11 12:53:23 -0400] 2609815
```

This first is a Feed created message and the second is the actual data received message.

3. Tweaking Names: Publish to a Feed by name with capitalization changed

Once the Feed has been established, **publishing to any named Feed whose Key is the same as an existing Feed will add Data to the existing Feeds stream.**

```
PUB 3124870 to test_username/f/test mode at 2016-07-11 12:39:34 -0400
```

And the original Feed subscriber, which is still watching `test_username/f/Test Mode`, receives:

```
[test_username/f/Test Mode 2016-07-11 12:39:34 -0400] 3124870
```

4. Tweaking Names: Publish to a Feed by key

Once the Feed has been established, **publishing to an existing Feed's Key will add Data to the existing Feeds stream.**

```
PUB 1181702 to test_username/f/test-mode at 2016-07-11 12:42:28 -0400
```

The Feed subscriber, still watching `test_username/f/Test Mode`, receives:

```
[test_username/f/Test Mode 2016-07-11 12:42:28 -0400] 1181702
```

5. Valid name variations for publishing

When publishing, the method Adafruit IO uses internally to convert a given topic in the form `username/feeds/identifier` to a specific, existing Feed works like this:

1. Find the Feed belonging to `username` whose Key is exactly the same as `identifier`.
2. If no Feed is found, convert the given `identifier` using the Name-to-Key translation (described above) and find the Feed belonging to `username` whose Key is exactly the same as the converted value.
3. If no Feed is found, find the Feed belonging to `username` whose Name is exactly the same as `identifier`.

Thanks to the Name-to-Key conversion rules, the following topics will all **publish** to the original Feed created in step 2 and be received by the subscriber at `test_username/f/Test Mode`:

- `test_username/f/Test_Mode`
- `test_username/f/Test-Mode`
- `test_username/f/Test Mode`
- `test_username/f/ Test Mode`
- `test_username/f/Test Mode`
- `test_username/f/Test -Mode`
- `test_username/f/ Test - Mode`

And so on, including any variation of modified capitalization.

Some variations that include symbols will be converted to URL-safe Keys when looking up the requested Feed:

- `test_username/f/Test(Mode`
- `test_username/f/Test\[Mode`
- `test_username/f/Test{Mode`
- `test_username/f/test modé`

- `test_username/f/test' mode`

6. Valid name variations for subscribing

Subscriptions, on the other hand, **must use an exact Name or Key**. So, for the given examples, the only topics that will produce the Feed we care about are:

- `test_username/f/Test Mode`
- `test_username/f/test-mode`

Naming and Accessing Feeds With the HTTP API

The HTTP API follows the same Feed identifying and Name-to-Key conversion rules as the MQTT API because under the hood they're talking to the same backend.

This means if you're using the Ruby IO client library, the following will produce publications to the same feed as the MQTT examples given above.

```
require 'rubygems'
require 'adafruit/io'

client = Adafruit::IO::Client.new(key: ENV['AIO_KEY'])

[
  'Test Mode',
  'test mode',
  'test-mode',
  '44'
].each do |feed_ident|
  client.feeds(feed_ident).data.send_data(feed_ident)
end
```

Troubleshooting

It really stinks to get taken by surprise in a negative way when working with code. **Reducing surprise** of the unpleasant sort and **increasing predictability and stability** are the primary motivating factors for the subtle changes this guide introduces.

Publishing to an invalid name

While the Name-to-Key converter keeps things feeling pretty loose and improvisational in terms of referring to Feeds *once they exist*, if your initial publish is to a Feed that can't be found it will be rejected if it doesn't match the rules for valid Feed names (https://adafru.it/oYb).

In the case of our MQTT example, a publish that looks like this:

```
PUB 2948554 to test_username/f/Test Modes[ at 2016-07-11 15:42:31 -0400
```

would trigger a message on the error feed that looks like this:

```
[test_username/errors 2016-07-11 15:42:31 -0400] "Validation failed: Name may contain only letters,
digits, underscores, spaces, or dashes"
```

If the Feed named "Test Modes" already existed, then the publish would work fine, but because it doesn't Adafruit IO tries to create a Feed with the given identifier, "Test Modes[" as a Name. Since "Test Modes[" is an invalid name, Adafruit IO rejects it :(

Publishing to the wrong identifier

If you set up your MQTT subscription first, it's important to note that no feed will be created, so the Name-to-Key rules laid out above won't have the effect you may have anticipated. This happens when the Feed you eventually publish to doesn't end up with the Key or Name your subscriber has requested. The end result is a subscriber that's silent while your device is merrily publishing away. Maddening! This is a common error of subscription and publishing when trying to juggle the different identifiers that point to a given Feed.

The safest way to avoid this situation is to make sure that your subscribing topic and your publishing topic are *exactly the same*. If you want to switch to a different identifier--for example, using a Key instead of a Name--copy the value directly from Adafruit IO. When in doubt, use the Name value.

The Feed we're publishing to in the MQTT examples above has the following identifiers:

```
key:  "test-mode"
name: "Test Mode"
```

The Name-to-Key translator is how all the "valid name variations" shown above for publishing work, but they only after the Feed already exists. The *only* way to create this Feed from the MQTT interface is to publish to `test_username/f/Test Mode`.

Keeping a browser open to your Feeds page while setting up or programming your Adafruit IO devices is recommended.

Modifying a name or key

Remember, **changing a Feed's name will automatically update its Key.** This is a change to existing behavior and will require modifications to any systems you're running that refer to Feeds by Key, but it prevents more confusing situations from occurring.

Here's a non-hypothetical scenario that illustrates the trouble when we don't keep Keys and Names in sync:

- I make a new Feed and call it, "Light Switch" (IoT light switch, *low risk*). In JSON, the Feed looks like:

```
{
  "name": "Light Switch",
  "key": "light-switch"
}
```

- I have a Feather Huzzah controlling a relay, acting as a subscriber listening to `username/f/light-switch` and a publisher sending data to `username/f/Light Switch`. Things talk, everything is great with the world.
- I move the hardware over to a new spot and rename the Feed, "Blender Toggle" (IoT blender, *high risk*). In a non-sync world, the Feed's new Name is "Blender Toggle" and it's Key is still "light-switch. The Feed is now:

```
{
  "name": "Blender Toggle",
  "key": "light-switch"
}
```

- My subscriber is still listening to `username/f/light-switch`, so it still gets all the Feed's messages.
- I build a new remote control and have it publish to `username/f/Blender Toggle` and it works, because there is a Feed with that exact name.*Everything just keeps working*, which is okay for now.
- Later on, I decide I'd like to build another remote control light switch, so I put together an Arduino MKR1000 publishing to `username/f/Light Switch`.
- My new light switch controlling, MKR1000-powered, motion sensor publishes a message to `username/f/Light Switch` and my blender turns on! What the heck! Hope you're wearing Kevlar gloves!

The MQTT Feed routing rules described earlier (https://adafru.it/oYc) mean that a message received on the topic `username/f/Light Switch` gets routed to the Feed whose Key matches `light-switch`, which already exists. And controls the blender. Which should not turn on unexpectedly when the room gets dark :P

This is why we keep all Feed Keys updated to match to their respective Feed Names. In Adafruit IO, renaming the Feed to "Blender Toggle" changes the Key to "blender-toggle".

Old Feed:

```
{
    "name": "Light Switch",
    "key": "light-switch"
}
```

Change Name to "Blender Toggle" and the Feed now looks like:

```
{
    "name": "Blender Toggle",
    "key": "blender-toggle"
}
```

My existing subscriber would immediately stop working because none of the messages sent to `username/f/Blender Toggle` (new Name) will get routed to the subscriber listening to `username/f/light-switch` (old Key). *This doesn't mean the subscriber breaks or shuts down*, only that it stops getting messages for now. This is my chance to realize something is wrong and debug my system.

Here's the tricky bit, if I go in and make a new Feed and name it "Light Switch", it'll get the Key "light-switch".*If I didn't update my subscriber* when it stopped working, that means it's still listening to `username/f/light-switch`. When I start posting to my **new** Feed at `username/f/Light Switch`, the **old** subscriber at `username/f/light-switch` will start getting messages again.

The best defense against confusion is to refer to the Adafruit IO web interface to double check what the Feed you're working with has for Name and Key values. And, when you make changes in Adafruit IO always make sure the cooperating systems are updated, *especially* when dealing with control systems that interact with the world.

Fixing Existing Feeds

We've updated our Feed identification system, but we didn't go back and change any Feed names or keys for you. That means you might have Feeds you used to be able to reach that don't show up anymore. If this is the case, you should see a warning on https://io.adafruit.com (https://adafru.it/eZ8) and an indicator on your Feeds page (https://adafru.it/mxC).

The two things you'll need to check if you think there might be a Feed name/key conflict problem are that:

1. Every Feed has a unique name.
2. No Feed's *name* matches another Feed's *key.*

In general, if you're no longer using a Feed, it's a good idea to remove it to avoid conflicts in the future. And, if you're using a lot of similarly named feeds, we strongly recommended that you run through each device or service that is publishing to the feed and do an explicit check to make sure everything is posting to the right place.

And, as in any situation where you're having trouble, please reach out to us on the Adafruit IO forums (https://adafru.it/pIC) if you need help. We're here to make things easier!

Final Words

If you got this far, I hope it's clear that this is an area of Adafruit IO we've put a particular amount of thought into. Our intention continues to be building a clear, simple, powerful platform for connecting the things you build and we think this refinement supports that intention.

Please join us at the Adafruit IO forum and share your thoughts, projects, questions, or requests. We'd love to talk to you about what we're building!