INTRODUCTION

Our primary goal at Particle is to help product creators (engineers and entrepreneurs) bring new connected products to market.

This guide will teach you everything you need to transition from a breadboard prototype to a device manufactured at scale running on Particle's software platform. Many of the lessons within are broadly applicable to a wide range of products (whether they run on our platform or not). Other lessons are platform-specific. If you are an experienced engineer with manufacturing experience, some of these lessons will feel juvenile; if so, simply skip over the bits that you already know.

But wait, I thought you were a hardware company?

Sort of. We're a software company that serves hardware companies. In order to most effectively deliver our software platform, we sell hardware (development kits and modules) that are pre-programmed with our software stack. All of our magic is in the software.

Prerequisites

Before you start to follow this guide, you should already have:

- A functional prototype running on a Core, Photon, PØ, P1, or Electron
- A working knowledge of Particle's development tools and cloud platform
- Created a Product in the Console

If you are not yet familiar with these tools, please read the previous guides above ("Getting started" and "Tools and features").

Steps to deliver a product on the Particle platform

At a high level, you should expect to follow these steps to transition from your first functional prototype to a mass-manufactured product:

- Set up your product in Particle's Console
- Design your own printed circuit board (PCB)
- Build your own web app
- Build your own mobile apps (iOS and Android)
- Manufacture, program, and test devices

There are, of course, other issues you will likely encounter, such as injection molded plastic tooling, packaging, etc. This tutorial will focus strictly on electronics and software, as that is where the Particle platform is most relevant.

What if I need help?

It takes a village to build a hardware product. It's uncommon for any company to try to do everything themselves; you will likely engage a variety of professional services firms or freelancers as you develop your product.

We are building a network of professional services firms who can support you with the following aspects of your product:

- Mechanical engineering
- Electrical engineering
- Firmware development
- Web development
- Mobile app development
- Contract manufacturing
- Packaging
- Logistics

If you are interested in engaging us or our partners to help develop your product, please contact our sales team.

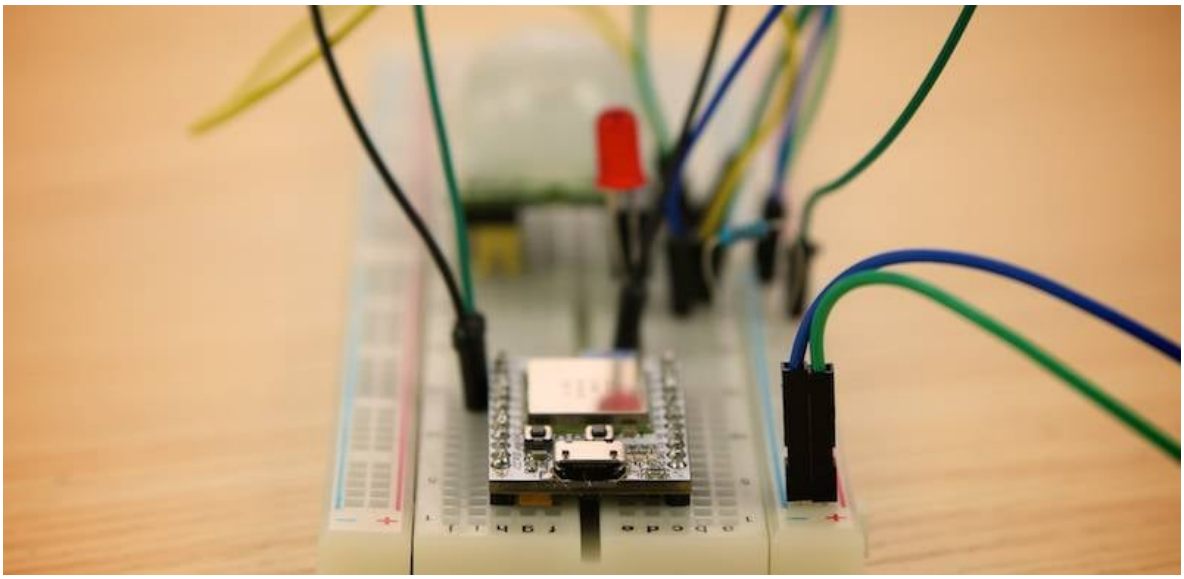
Let's get started!

Okay! First, let's start with:

Designing your own PCB >.

DESIGNING YOUR OWN PCB

So you've got a prototype in a breadboard. It's probably a mess of wires, something like this:



Your first step towards a manufacturable product is to re-design your printed circuit board (PCB) using our modules (PØ or P1) rather than our development kits (Photon/Core). Before you dive in, make sure you read the manual! Hardware datasheets for the Photon (PØ) and P1 are located at the links below:

- Photon (PØ) Datasheet
- P1 Datasheet

Great! Did you actually read them? No? Don't worry--we'll wait for you.

...

Yes? OK--let's move forward on good faith, then :-)

Hardware Design Requirements

To work with Particle, your hardware must meet a small list of design requirements. They are:

- **Your device should use a supported hardware module.** While the firmware stack can be run on a variety of connectivity modules, we provide our own modules (the PØ and P1) as an out-of-the-box solution, and we can support other Broadcom WICED modules easily. You may also choose to port our firmware libraries to another module by implementing our Hardware Abstraction Layer (HAL); this is, however, a more involved process. Please contact our sales team if you are interested in engaging us to support another hardware solution.
- **Your device must have an RGB LED and a button to enter 'setup mode'**. The RGB LED shows the user the connectivity status, while the 'setup' button lets your customer reconfigure the device. These components should be wired according to the Photon reference design.
- **Your device must have an RF circuit and an antenna.** If you use the Photon or P1, the antenna is included in the hardware. If you use the PØ, you must connect your own antenna. Please use the Photon as a reference design; if you use an antenna of equal or lesser gain, you may leverage Particle's FCC/CE/IC modular certification for the Photon.
- **Your device must expose JTAG programming pins**. All Particle development kits (Photon/Core) expose JTAG pins by default. Although the Particle platform has been optimized for over-the-air firmware updates, JTAG programming is required for advanced debugging and development, or modifications to the underlying Particle firmware libraries. All product creators working with the PØ or P1 should expose these pins on their PCB. To be specific, if you're integrating the Photon, your test fixture needs to be able to make an electrical connection to the male JTAG pins on the Photon, a female socket connected to those male headers, or to test pads that are electrically connected to those headers on your PCB. If you're integrating the PØ or P1, you'll need to expose test pads that are routed to the appropriate JTAG solder pads on the PØ/P1 module. These pins/pads are identified in the corresponding datasheets for the PØ and P1, here and here.

- **Serial Test Pads** (*Recommended*). Although it's not *absolutely* required, it's highly recommended that you expose one hardware serial peripheral (UART) via test pads or pins. As we'll discuss later in the Manufacturing section of this guide, exposing serial will make it easy to leverage Particle's open-source test firmware for capturing basic information about your device on the manufacturing line.

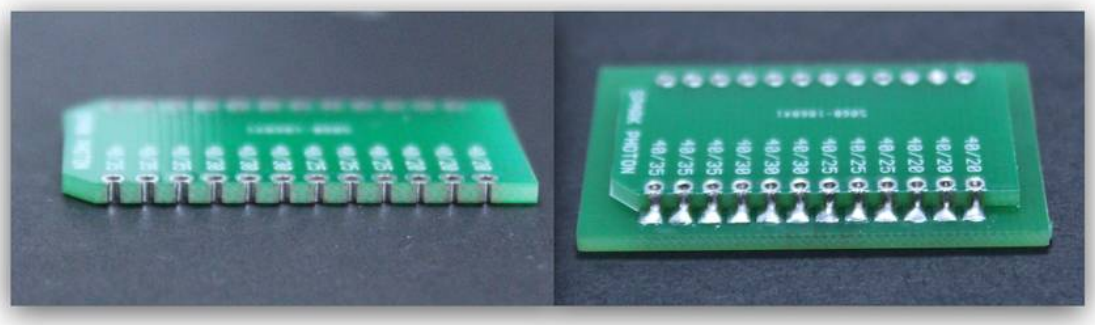
Note that, for additional security and to minimize tampering by customers, you can expose JTAG and serial test pins on a perforated section of your PCB that can be broken off after successful testing. This is a better option than deciding not to expose either JTAG or serial, which is, once again, a bad decision.

Transitioning from a dev kit to a module

Your first prototype was most likely built with a Core or a Photon in a breadboard. Your final product is likely to take one of four forms:

Photon in female headers. You may design female 0.1" headers into your product to provide a "slot" for the Photon. This is common for low-volume hand-soldered production. Please reference the Photon design files and the Photon datasheet for design guidelines.

Surface mounted Photon. The Photon comes with castellated edges and can be purchased without headers; in this form it can be surface mounted directly to your PCB like many other radio modules on the market (see photo below). Please reference the Photon design files and the Photon datasheet for design guidelines, including recommended pad sizes and spacing on the host PCB and reflow requirements.



Transition to PØ. The PØ is the module on the Photon, which includes a microcontroller and Wi-Fi chip. The PØ can be purchased in packs of 10 from our online store or in bulk (contact sales). When you transition from a Photon to the PØ, you must reimplement on your board the following subsystems:

- Voltage regulator (depending on your input voltage)
- RGB LED and 'setup' button (see hardware requirements above)
- Antenna and RF circuit (see hardware requirements above)

Specifically, the RF circuit requires specialized testing in order to verify effectiveness. Please see the section on "RF Validation" in the "Prototyping early designs" section further down the page.

Transition to P1. The P1 is a second module that is nearly identical to the PØ except that it's larger, includes 1MB of external flash, and includes an antenna and u.FL connector on-board. The P1 is slightly more expensive than the PØ ($2 more in single units), but this may be a worthwhile trade-off as it helps you avoid additional RF design work and validation. When you transition from a Photon to the P1, you must reimplement on your board the following subsystems:

- Voltage regulator (depending on your input voltage)
- RGB LED and 'setup' button (see hardware requirements above)

NOTE: RF Keepouts.
Transitioning from a dev kit to a module requires that you observe the RF keepouts documented in the datasheets for the Photon (PØ) and P1. If you do not observe these keepouts, the RF performance of the Particle hardware and your product may be negatively affected.

PCB design resources

If you have never designed your own circuit boards, there is a wealth of knowledge available online. Here are some great resources that you might find helpful:

Popular PCB design software packages:

- EAGLE from CadSoft. A popular choice for open source projects because it's powerful and cheap (free versions are available for open source

projects). We use EAGLE extensively (as do our peers at Arduino, Adafruit, SparkFun, Seeed Studio, etc.), and we also provide EAGLE libraries for our hardware designs. SparkFun has some great tutorials on EAGLE, and software licenses are available on Adafruit.

- Altium is the gold standard of PCB design software. A steeper learning curve than some of the other tools, but MASSIVELY powerful. Licenses cost many thousands of dollars, but Altium is the most powerful PCB design software out there.
- Upverter is bringing PCB design online with a browser-based design tool. Upverter adds Github-like collaboration to PCB design. If you're learning from scratch, Upverter is worth a look.
- KiCad gets a special mention for being open source. If you're all about FOSS (Free and Open Source Software), KiCad is for you.
- Fritzing is a beautiful tool for prototyping. It's not necessarily suitable for complex designs, but it's extremely easy to use.

If you don't have strong preferences, we would currently recommend using EAGLE, since you'll be able to use our Photon board design as a reference for your own PCB.

Particle Fritzing and EAGLE parts - Libraries for Fritzing and EAGLE to include Particle parts into your PCB design.

Best practices for PCB design. There are a lot of resources on the web that establish best practices for designing PCBs. This post from our blog, Proto2Prod, focuses specifically on the key fabrication specifications that have a significant impact on mass production. Designing your product with a strategy in mind for mass production is critical, and will save you development time, money, and headaches down the road.

Prototyping early designs

Before you manufacture thousands of units, it's best to start with just a handful. We've compiled a short list of resources you'll need to get going:

Purchasing Particle hardware for prototyping. The fastest way to get low-volume dev kits and modules for prototyping is to purchase them directly from our online store. Particle sells both our PØ and P1 modules in cut-tape strips of 10

modules that are perfect for low volume PCB assembly (PCBA) with a pick-and-place or hand assembly.

Popular PCB manufacturers:

- Advanced Circuits
- Seeed Studio Fusion PCB
- OSH Park
- HQPCB

If you would like introductions to high-volume overseas PCB manufacturers, please contact our sales team.

Popular low volume PCB assembly (PCBA) solutions:

- Tempo Automation
- Seeed Studio PCB Fusion

For tips and resources for prototyping PCBs by hand (our preferred prototyping method at Particle), please visit our blog, Proto2Prod.

RF validation - *Coming soon!*. You have to validate your RF design with a professional shop and equipment you probably don't have access to. Here's strategies for dealing with that.

Thinking about mass production

Reach out to us! Once you've validated your hardware design with a series of prototypes, it's time to start thinking about how to scale up for mass production. If you haven't already, let us know you're building a product on Particle! Our team has lots of experience bringing Internet-connected devices to market, and can give valuable feedback on a wide variety of topics like manufacturing overseas, Kickstarter, VC funding, and everything in between. If we don't know you exist, it's much harder for us to help :-)

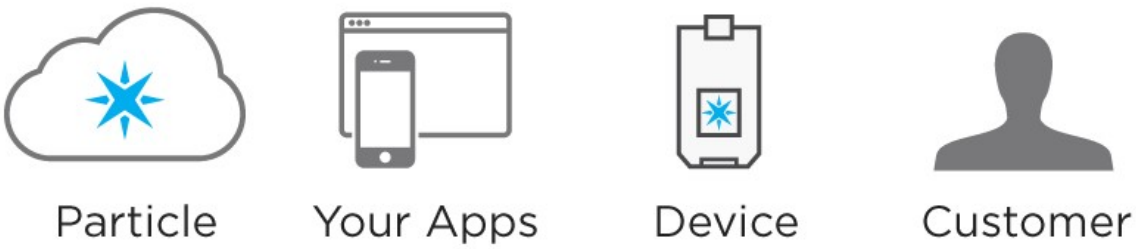
Preferred Services Partners Our customer success team has assembled a broad menu of trusted, professional services partners for every step of the product development process. Please take a look here and see if there's a partner that fits your development needs!

AUTHENTICATION

Introduction

For any Particle-powered product to function properly, there are **four involved parties**:

- Particle
- Your product's applications
- A device
- The customer



Each of the four parties plays a critical role in how your product will function

Understanding how all four interact, manage data, and secure information is critical to launching a successful product on the Particle platform. Specifically, this section of the guide will go deep into decisions you must make on authentication and security with regards to your product. The results of these decisions will impact where and how data is stored as well as the ways in which your product's applications interact with Particle to control devices and manage customers.

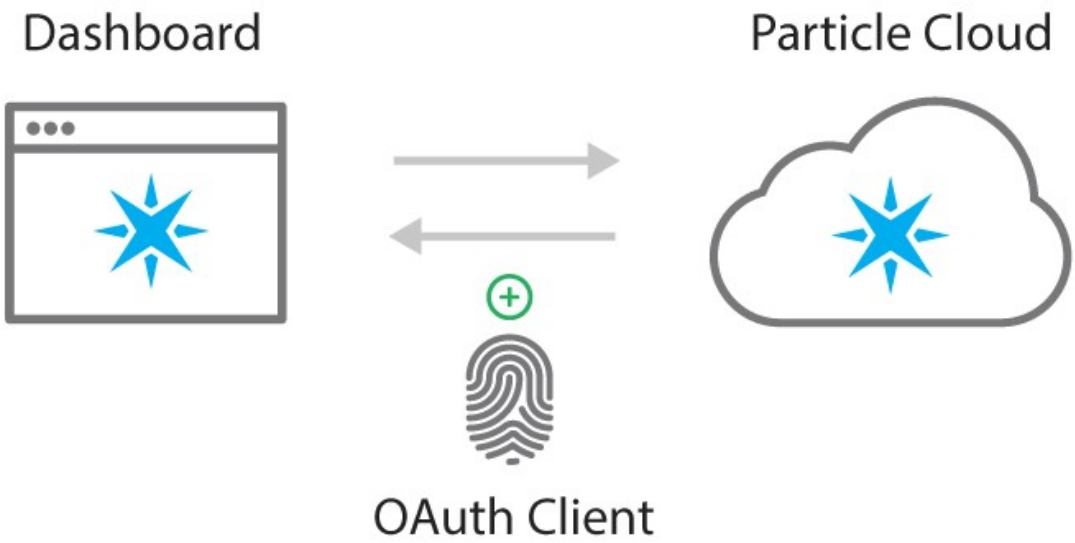
Note: *Many of the implementation details discussed in this document are still under development. We will add specific code examples and context as they become available in the mobile and JavaScript SDKs.*

OAuth

Particle tightly adheres to OAuth specifications to ensure secure and correct access to private data and devices. OAuth is an open standard for authorization, providing client applications "secure delegated access" to server resources on behalf of a resource owner (Wikipedia).

Integral to OAuth is the concept of *clients*, and *client credentials*. An OAuth client generally represents an application, like a native iOS application running on an iPhone or a web app running in a browser. As part of the setup process for your product, you will need to create one or more OAuth clients for your product.

Your product's OAuth client will be needed when hitting the Particle API to perform actions that only your product's team members or applications should be able to perform. An example of this is creating a customer that belongs to your product via the Particle API. Only those with a valid OAuth client will be able to perform this action, protecting your product from fake or erroneous new customer accounts.



Your OAuth Client will ensure secure communication between your devices and the Particle cloud

Client credentials are comprised of two pieces of information: A **client ID**, and a **secret**. Passing these two pieces of data to the Particle API will allow you to create **access tokens**, which are described in detail in the next section.

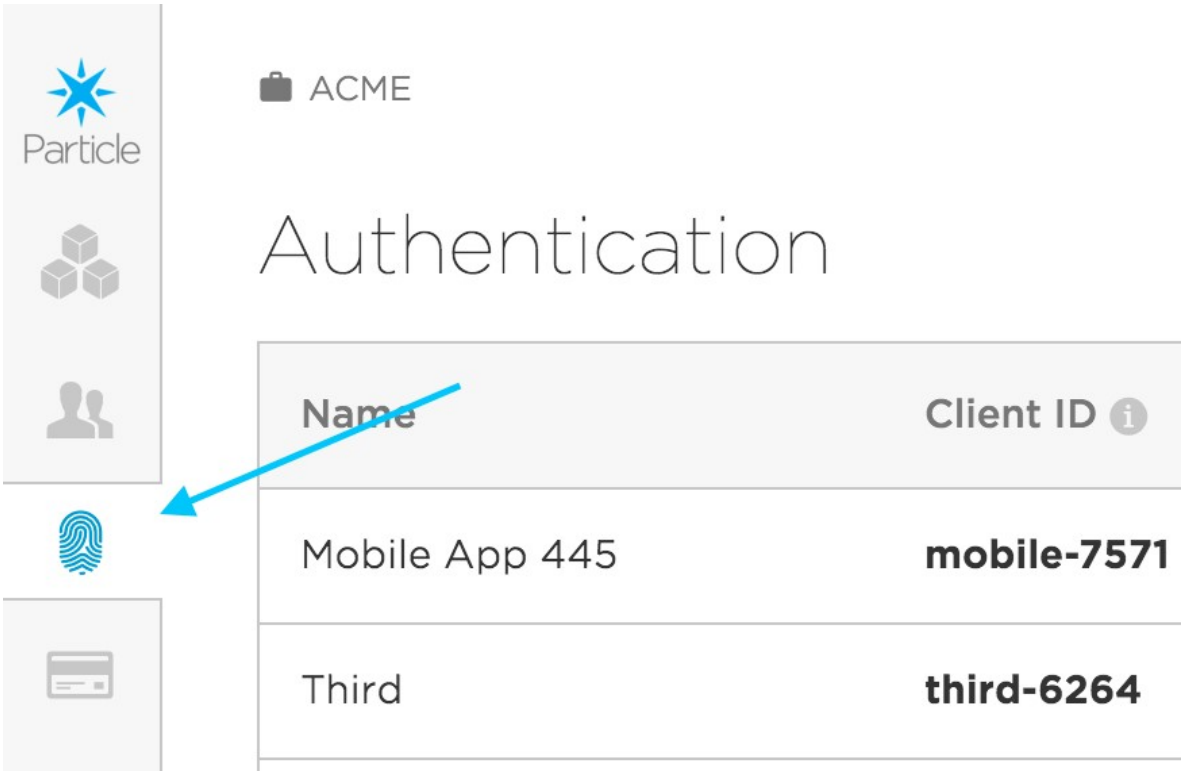
Scopes

Scopes allow you to specify exactly what type of access the client (and tokens created using the client) should have. Scopes are used for security reasons to limit the allowed applications of OAuth client credentials. Depending on which authentication method you choose, you may need to specify a scope when creating your OAuth client. More on scopes.

Creating an OAuth Client

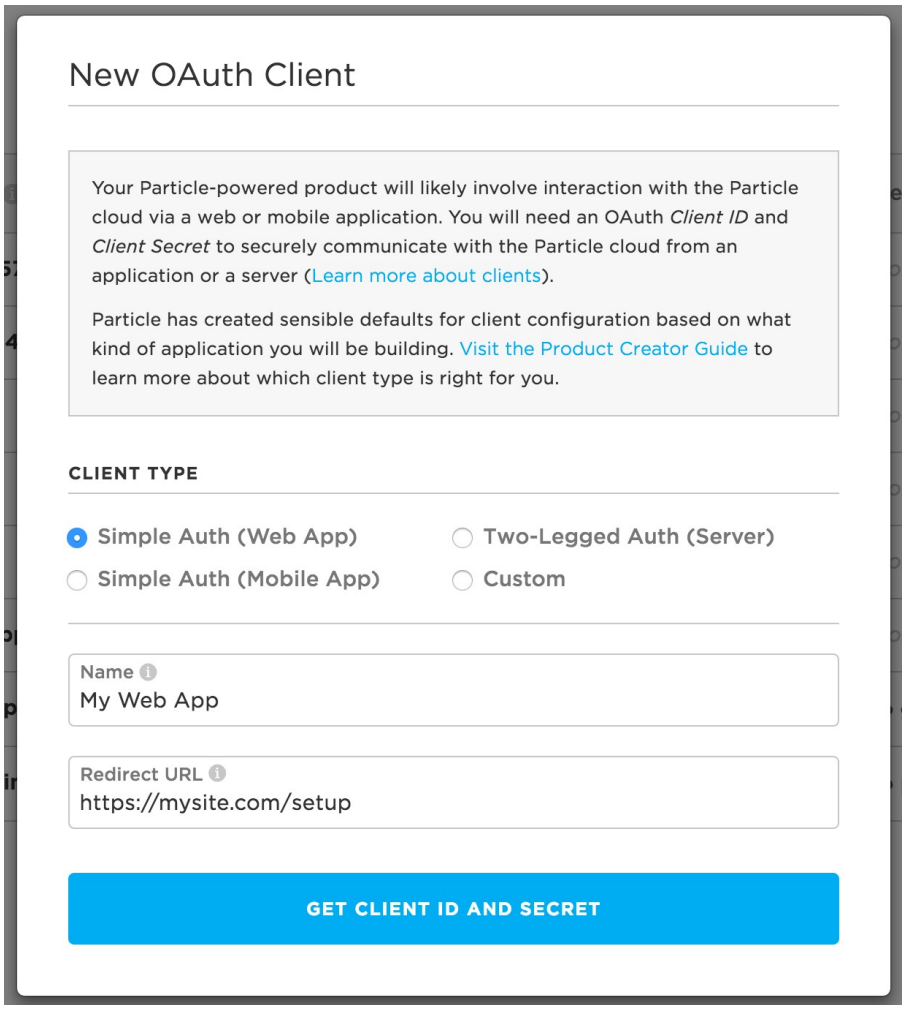
You can now use the Particle console to create and manage your OAuth clients. The new "Authentication" view is your hub for creating, and managing your product's clients.

You can find the Authentication view by clicking on Products icon on the left navigation bar in the Console, then selecting your product then clicking on the thumbprint icon. If you don't have a product yet, create one on the Console.



When you visit the Authentication page, you will see a list of any OAuth clients that already have been created for your product. To create a new client, click the "+ New Client" button in the top right corner of the screen.

This will launch a modal allowing you to configure an OAuth client that suits your needs. The configuration of your OAuth client will depend both on what *medium* your customers will use to interact with their Particle devices (i.e. mobile vs. web app) in addition to what *authentication method* you choose for your product. You can skip to choosing an authentication method if you'd like to create an OAuth client now.



The console provides an easy way to create OAuth clients

The console will provide your client ID and secret once you configure your client correctly. **Your client secret will only be shown once for security purposes**. Ensure that you copy it for your records to be used in your mobile or web app.

Never expose your client credentials, especially if they are unscoped. Credentials are sensitive pieces of information that, if exposed, could allow unauthorized people or applications to interact with your product's data and devices. More on registering clients.

Access Tokens

A related concept to understand is how Particle uses access tokens for authentication and security. If you have ever logged into the Web IDE, called a function on a Particle device, or read a variable via the API, you are already using access tokens! It is important to note that OAuth credentials are needed to create access tokens.

What's an access token?

An *access token* is a special code that is tied to a Particle customer or user, that allows reading data from and sending commands to that person's device(s). Any API endpoint that returns private information, or allows control of an individual's device requires an access token. Tokens *inherit scopes* from the OAuth clients used to create them, and can have more specific scopes of their own. More on access tokens.

Here's a specific example. Let's say that on one of your own personal devices (with device ID `0123456789abcdef01234567`), you want to call the `lightsOn` function. Here is how you would do this using the API:

```
curl -X POST -H "Authorization: Bearer 1234" \
https://api.particle.io/v1/devices/0123456789abcdef01234567/lightsO
n \
-d arg=livingRoom
```

In order for the API to call the function on your device, you *must* include a valid access token for the user that is the owner of the device. In this case, you include

your access token of `1234`. Behind the scenes, the API looks up the access token in our database, checks to make sure that the user who created the access token is the owner of the device included in the request, and will only continue if the token has the proper permissions to call the function.

For security purposes, most access tokens will expire after 90 days. Certain access tokens can be specifically set to *never expire*, like the one you can find in the settings pane of `build.particle.io`.

Customer access tokens

Similarly, *your customers* will have access tokens that provide verification of ownership of a particular device. As a product creator, you will be creating a web or mobile application for your customers to use your product. As part of your application, you will generate access tokens on behalf of your customers, and use these access tokens to make the desired calls to the Particle API to successfully read data from and control that customer's device.

The access token is linked to the customer and used to control the customer's device(s)

Luckily, the mobile SDKs and ParticleJS will expose helper methods to handle token creation and management, without much/any additional code needed from you or your engineers.

Choosing an Authentication Method

Take a deep breath. We've covered a lot so far and you're picking things up quick! This section will help you determine the best place for you to go next.

As described earlier, end-users of your product are referred to as *customers* in the Particle ecosystem. As a product creator, you will need to make a strategic decision about how you will want to handle authentication for your customers. Your choice will likely be influenced by how much you would like to hide Particle from your customers, as well as how you would like your product to function.

There are three ways to manage authentication for your customers.

- **Simple Authentication**: Customers are created and managed directly by Particle. This is the simplest and fastest way to get your connected product app working.
- **Two-legged Authentication**: You create and manage accounts for your customers yourself, and request scoped access to control customers' devices from Particle. This provides the maximum amount of flexibility and security for your application.
- **Login with Particle**: Your customers will create a Particle account and a separate account on your website, and link the two together using OAuth 2.0 *(Coming soon)*.

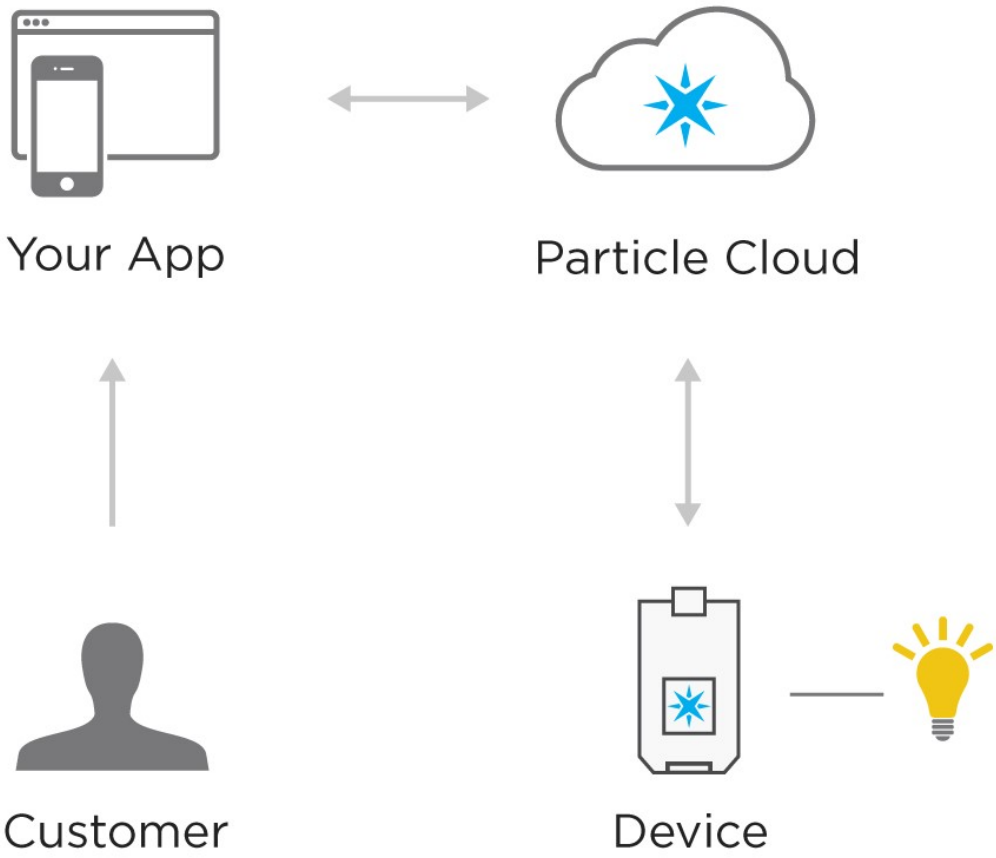
You will also need to choose what *medium* your customers will use to authenticate and setup their devices. This will likely depend on how you envision your customers interacting with their connected product.

- **Mobile**: Use an iOS or Android mobile app to allow your customers to authenticate, setup their devices, and interact with their product. More info
- **Web**: Use a web browser and HTTP to allow your customers to authenticate, setup their devices, and interact with their product (*Coming soon*).

When you're ready, click on the authentication method that makes most sense to you.

Simple Authentication

As the title suggests, Simple Authentication is the simplest and most straightforward to implement. This is because in this method, your application will not have its own server/back-end architecture. Instead, your web or mobile app will hit the Particle API directly for both session management and device interactions. Below is a diagram communicating how simple authentication works at a high level:



Your application interacts directly with the Particle API using Simple Authentication

Let's take a simple example. Imagine you are the creator of a smart light bulb that can be controlled via a smartphone app. The *customer*, or the end-user of the product, uses the mobile app to create an account. Behind the scenes, your mobile app hits the Particle API directly to create a customer. Then the customer

goes through the setup process and links their device to their customer account. Again, your mobile app uses the Particle API to successfully claim the device to the customer account. After the device is setup, the customer can toggle a light on and off with the mobile app. This works as your app is able to call functions on the customer's device using the customer's access token.

All of this is able to happen without the need to have your own server. All communication flows from the mobile client to the Particle cloud, then down to the customer's device.

Advantages of Simple Auth

Simple auth is ideal for getting a Particle product up-and-running quickly. Without needing to build your own back-end, development time to creating an app to work with a Particle device is greatly reduced. There are less moving parts and opportunities to introduce bugs. In addition, Particle's mobile SDKs and JavaScript SDK will handle much of the heavy lifting for you when it comes to session management and device interaction. In short, simple auth is...simple.

Another advantage of simple authentication is the ability to hide Particle from your customers. The SDKs allow for front-end skinning and customization that will allow you to create your own brand experience for customers of your app. All interaction with Particle will happen behind the scenes, hidden from your customers (unless they are tech savvy enough to monitor the network traffic to and from your app).

Disadvantages of Simple Auth

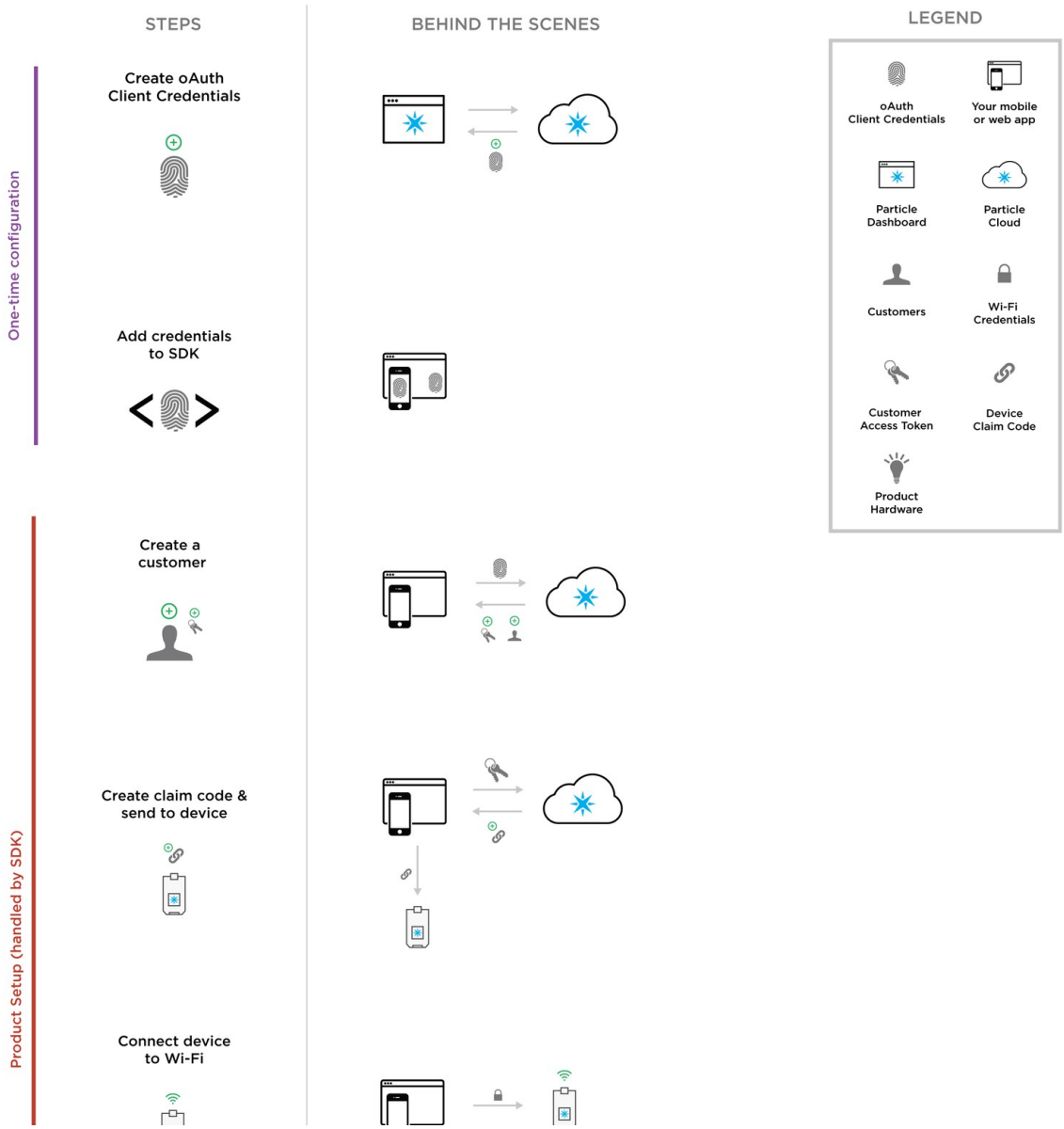
Without your own server, you lose some level of flexibility and ability to customize in your application. For instance, if you wanted to store custom information about your customer specific to your application like their name or their favorite pizza topping, this would not be currently supported with simple auth.

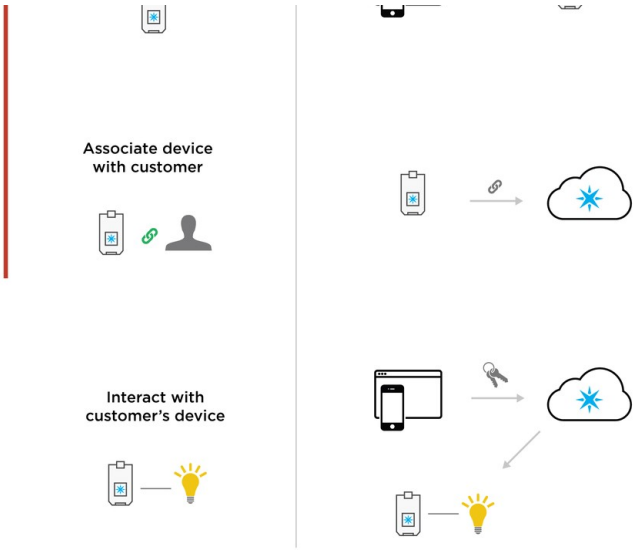
In addition, using simple auth would make it more difficult to capture and use historical data about devices and customers' behavior. With your own server and database, you could store data about what time a customer turns on their lights, for example. Using simple auth, this would not be supported.

Simple Auth Implementation

If you choose to go with simple authentication for your web or mobile application, you should get to know the diagram below very well. While a majority of the steps are wrapped by the mobile and JavaScript SDKs, it is still important to grasp how customer authentication, device setup, and device interaction work.

Each one of the steps will be covered in detail below. Note that the first two steps are a one-time configuration process, whereas product setup will occur for each new customer that sets up a device.





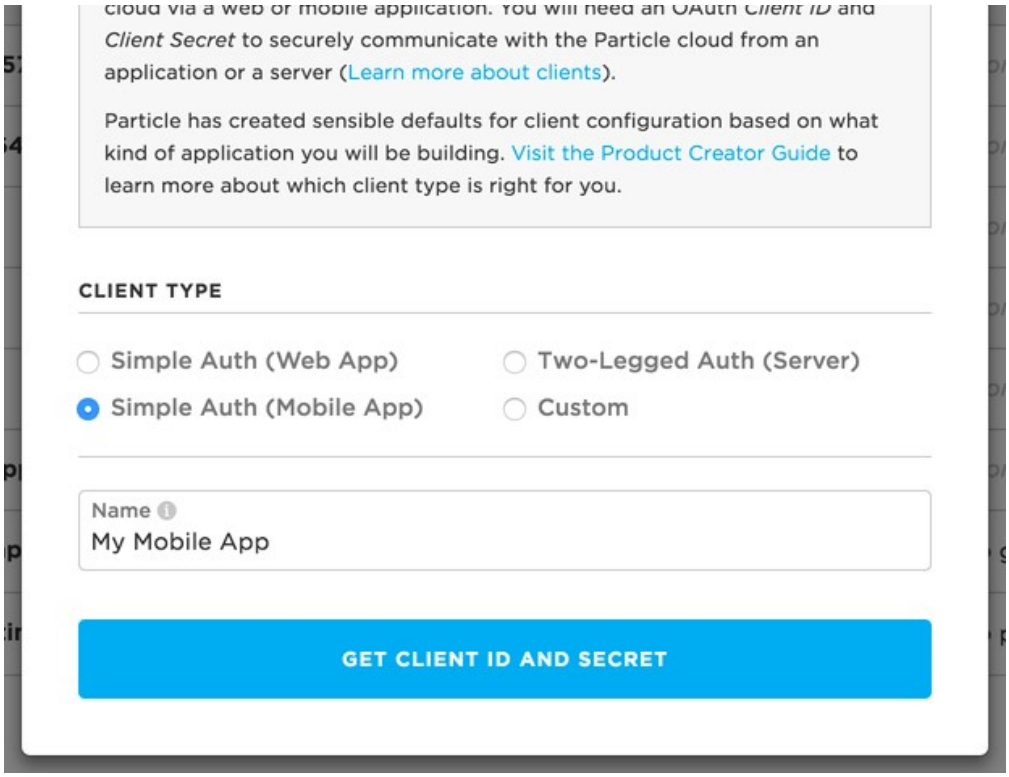
The full simple authorization flow. See here for full size diagram

1. Creating OAuth Client Credentials

The first thing you will need to do is ensure that you have created proper OAuth client credentials for your product. In simple authentication, communication will be direct from a client application (web or mobile app) to the Particle API. This is much less secure than server to server communication. As a result, you will create *scoped* client credentials that will be able to do one thing and one thing only: **create new customers for your product**.

You will create your OAuth client using the Authentication view in your product's Particle console. For info on how to find the Authentication page and create a client, check out the earlier discussion.

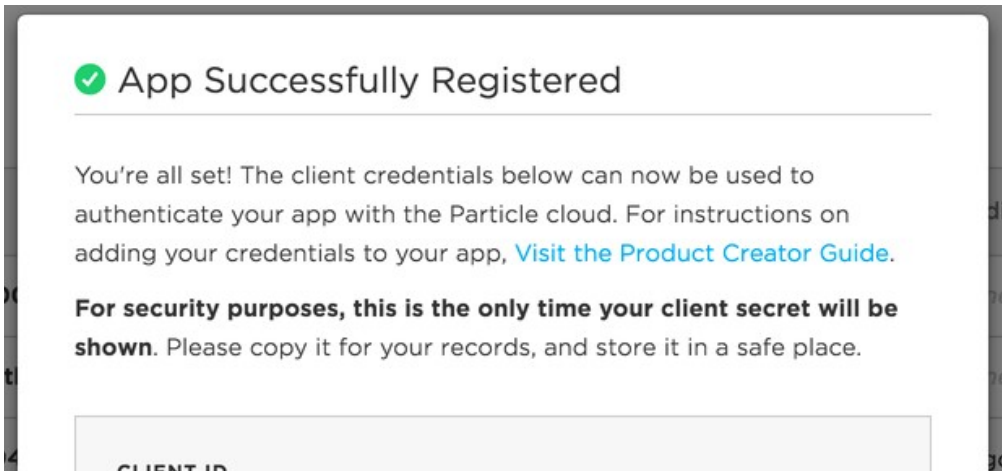
If you are building a **mobile app** for your Particle product, you should choose **Simple Auth (Mobile App)** from the client type options when creating a new OAuth client. This will provide the recommended client configuration, and only requires you to provide a name for your new client.

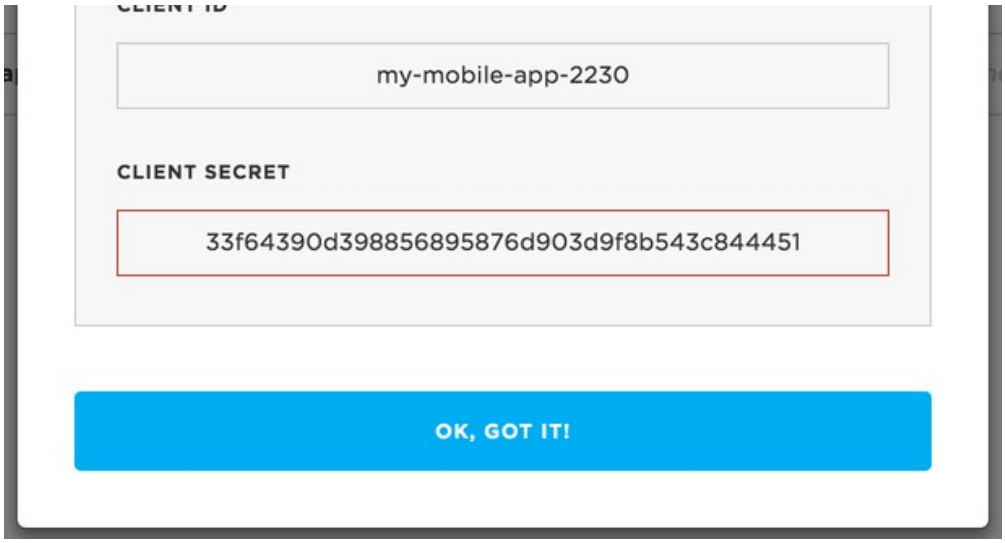


Creating an OAuth client for a mobile app using Simple Auth

If you are building a **web app** instead, select **Simple Auth (Web App)**. You'll notice that you have to provide both a name and a *redirect URI*. A redirect URI is required for any web browser-based OAuth flows, and should be set to the URL of the first page of device setup for your product. The redirect will be triggered once a customer is created successfully, and the next step in the process is setting up their device.

Regardless of your app medium, you will receive an OAuth client ID and secret upon successful creation of a client that will look like this:





Your client secret will only be shown once, so be sure to capture it for your records

This client ID and secret will be added to your application in the next step below. Note that clients created using the default Simple Auth configurations will be scoped for *customer creation only*. This is for security purposes. In Simple Auth, client credentials can be uncovered relatively easily with some basic scraping techniques. Scoping the client will prevent unintended access to your product's devices or data.

Creating OAuth client can still be done directly against the Particle API. For info on this, see reference documentation on creating OAuth clients.

2. Add OAuth Credentials to SDK

For both the mobile & JavaScript SDKs, you will need to add your client credentials to a configuration file. The client application will need the client credentials that you just generated when creating new customers. Without these credentials, calls to `POST /v1/products/:productIdOrSlug/customers` will fail.

You will need to add your OAuth credentials to your web or mobile application

If you are creating a mobile application, you will need to include **both** the client ID and secret in your configuration file. If you are creating a web application, you **only should include your client ID**.

For instructions on how to add client credentials to your iOS app, please see iOS OAuth client configuration. For Android apps, please see Android OAuth client configuration

3. Create a customer

You have now moved from the one-time configuration steps to a process that will occur for each new customer that uses your web or mobile app. As mentioned earlier in this section, much of what will be discussed in the next four steps will be magically handled by the Particle SDKs, with no custom code needed from you.

After navigating to your application, one of the first things your customer will need to do is create an account. Because you are not running your own web server, the customer will be created in the Particle system. They will provide a username and password in the form, that will serve as their login credentials to the app.

Specifically, the SDK will grab the customer's username and password, and hit the `POST /v1/products/:productIdOrSlug/customers` API endpoint, passing along the customer's credentials *as well as* the OAuth client credentials you added to the config file in the previous step.

The create customer endpoint requires your OAuth client credentials, and returns an access token for the newly created customer

For a mobile app, the SDK will require both the client ID and the secret to successfully authenticate. For a web app, the endpoint will only pass the client ID.

The `POST` customers endpoint both creates the customer as well as logs them in. As a result, an access token will be available to your application after successful customer creation. Remember that it is this access token that will allow the app to do things like claim the device, and interact with it.

Specific implementation details coming soon

4. Create Claim Code & Send to Device

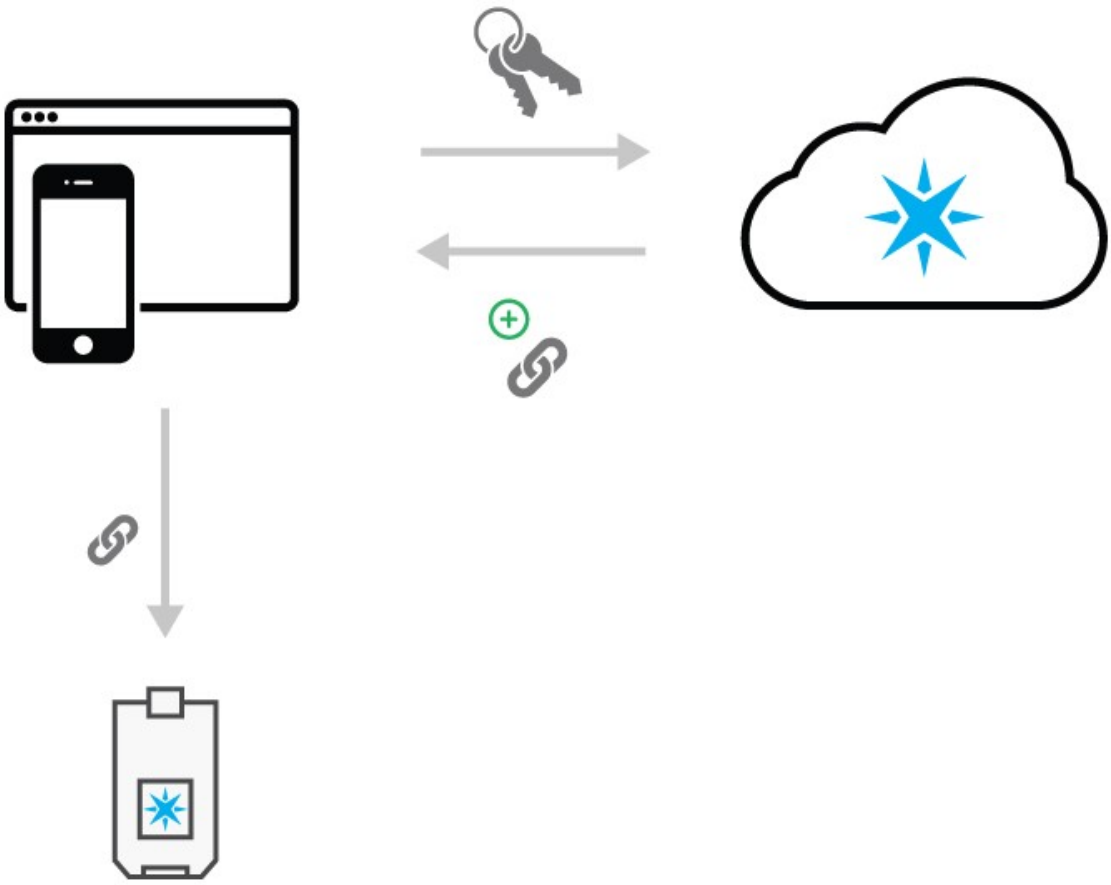
This step actually comprises a lot of things that happen behind the scenes, but has been combined for simplicity and ease of communication. A **claim code** is what is used to associate a device with a person. In your case, the claim code will associate a device with a customer.

In order for a device to be setup successfully, your application must retrieve a claim code on behalf of the customer setting up their device and send that claim code to the device. When the device receives proper Wi-Fi credentials and is able to connect to the Internet, it sends the claim code to the Particle cloud. The Particle cloud then links the device to the customer, and grants the customer access over that device.

The first thing that must happen is retrieving a claim code from the Particle cloud for the customer. A special endpoint exists for products to use to generate claim codes on behalf of their customers.

This endpoint is `POST /v1/products/:productIdOrSlug/device_claims` . The customer's access token is required, and is used to generate a claim code that will allow for the link between the device and the customer.

Once your mobile/web app has a claim code, it then must then send it to the device.



Your app will use the customer access token to generate a device claim code and send it to the device

This happens by connecting the customer's device to the *device's Wi-Fi access point*. When the photon is in listening mode, it is broadcasting a Wi-Fi network that the customer's computer or phone can connect to.

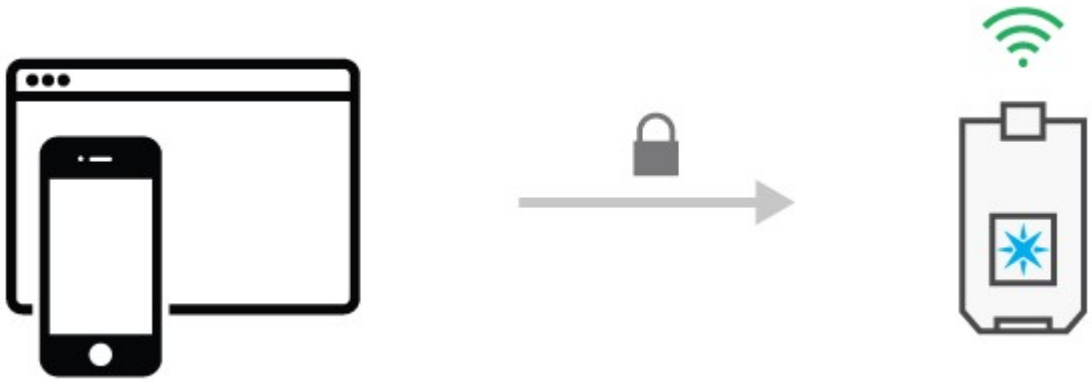
Once the customer's device is connected to the Particle device's network, your mobile app then will send the claim code generated in the last step to the Particle device.

Again, this will all be part of the boilerplate code of the SDKs, meaning that you will not need to worry much about the nitty-gritty details about how this works.

Specific implementation details coming soon

5. Connect device to Wi-Fi

Now that your app is connected directly to the customer's Particle-powered device, it can provide the device with Wi-Fi credentials to allow it to connect to the Internet.



Your app will send the customer's device Wi-Fi credentials

Through your mobile or web app, your customer will choose from a list of available Wi-Fi networks, and provide a password (if necessary) to be able to connect. The app sends these credentials to the device. Once received, the device resets and uses these credentials to connect to the Internet.

Specific implementation details coming soon

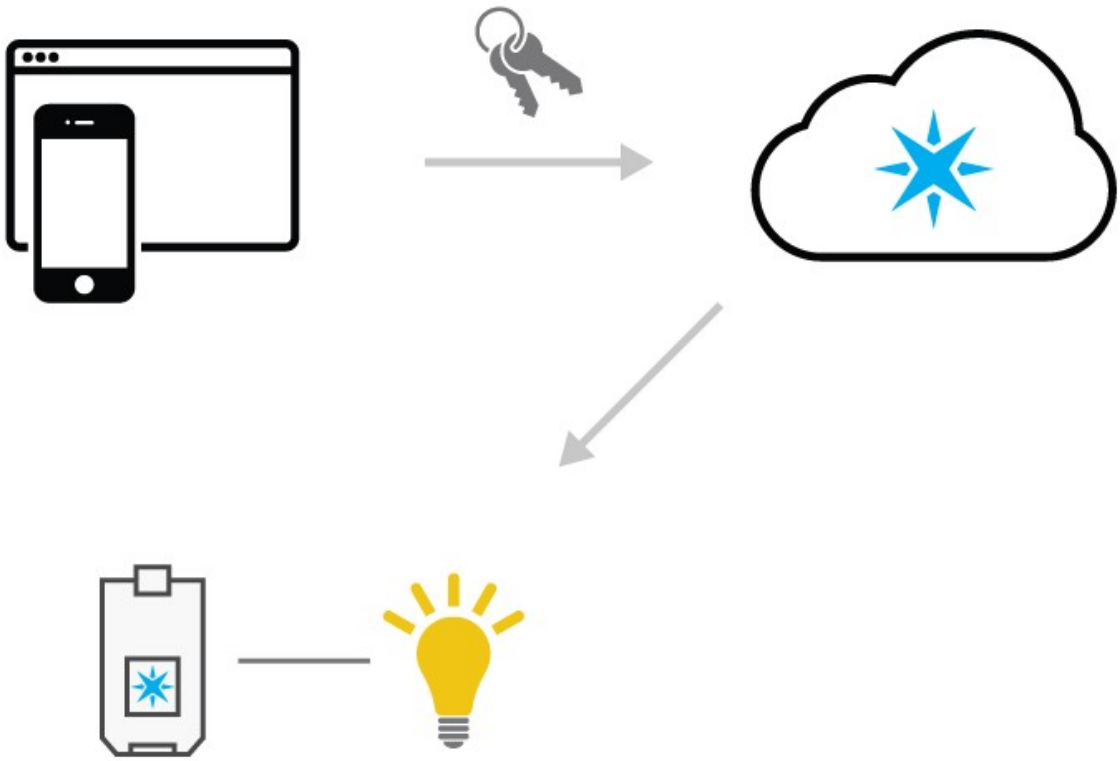
6. Associate device with customer

The device sends the claim code to the cloud, which is used to link the device to the customer

The device uses the Wi-Fi credentials to connect to the Internet, and immediately sends the claim code to the Particle cloud. At that point, the device is considered "claimed" by the customer. This means that any access token generated by that customer can be used to read data from or interact with the device.

7. Interact with Customer's Device

Congratulations! You've now successfully created a customer, gotten the device online, and tied the device to the new customer. You have everything you need to make your product's magic happen.



Use your customer's access token to call functions, check variables, and more!

Your application, armed with the customer's access token, can now successfully authenticate with any device-specific Particle API endpoint, giving it full access and control over the device. Some of the things you can do include:

- Call functions on the device
- Read variable values from the device
- See if the device is online or not
- Much more!

Note that now, your app will never communicate directly to the device. The customer will trigger a call to the Particle API, which then communicates with the device to carry out the desired action.

Further Considerations

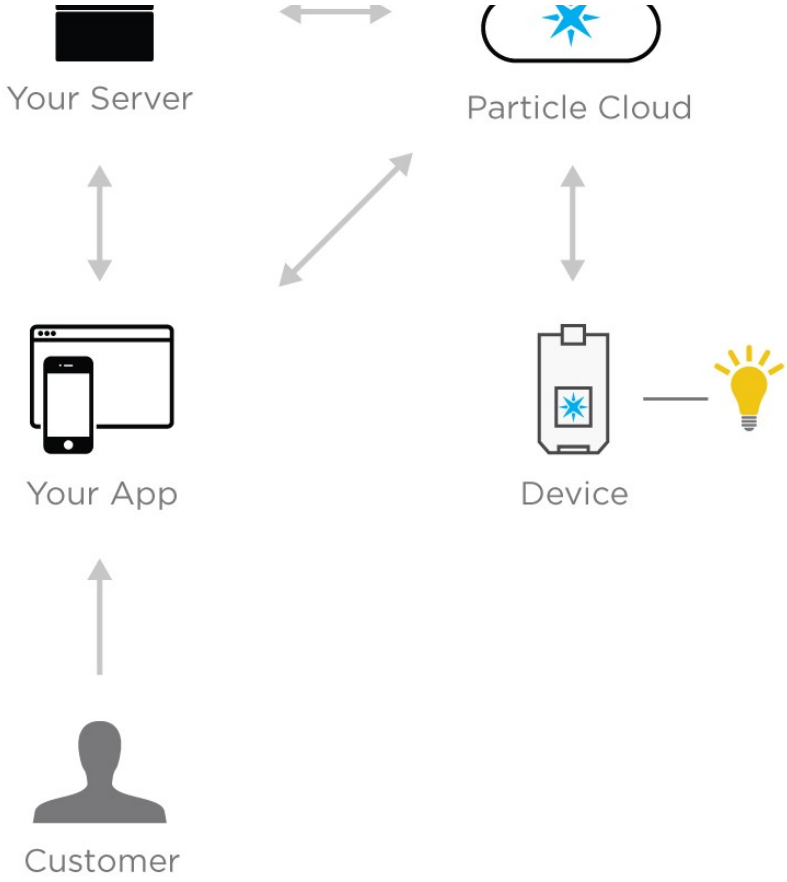
Signup and device claiming only will happen one time for each customer. After this has been completed, subsequent visits to your application will continue to use customer access tokens to interact with the device via the Particle API.

If a customer's access token expires, the customer will be asked to log in again, generating a fresh access token to interact with the device.

Two-Legged Authentication

The main difference between two-legged and simple authentication is the presence of a back-end architecture to compliment your mobile or web application. Your application would communicate with both your server as well as the Particle cloud.

The most common reason to use two-legged authentication is the desire to store & manage customer accounts yourself in your own database.



Two-legged authentication involves the presence of your own server

Advantages of Two-Legged

Two-legged authentication is the ideal choice for a product creator looking for maximum visibility, control, and flexibility of their web or mobile application. With two-legged, you gain the ability to implement custom logic, integrations with third-party services, and store application-specific data that are not currently part of the Particle platform.

For example, if you were building a connected hot tub, you could use your own web server and database to allow a customer to set their desired water temperature, then use that piece of data to set the temperature when the hot tub is turned on.

Another advantage of two-legged authentication is beefed-up security. Server-to-server communication (your server to the Particle API) is much more secure than client-to-server communication (your mobile/web application to the Particle API). For sensitive transactions like passing OAuth credentials to get customer

access tokens, using your server to talk to the Particle API over HTTPS is safe and protected.

Disadvantages of Two-Legged

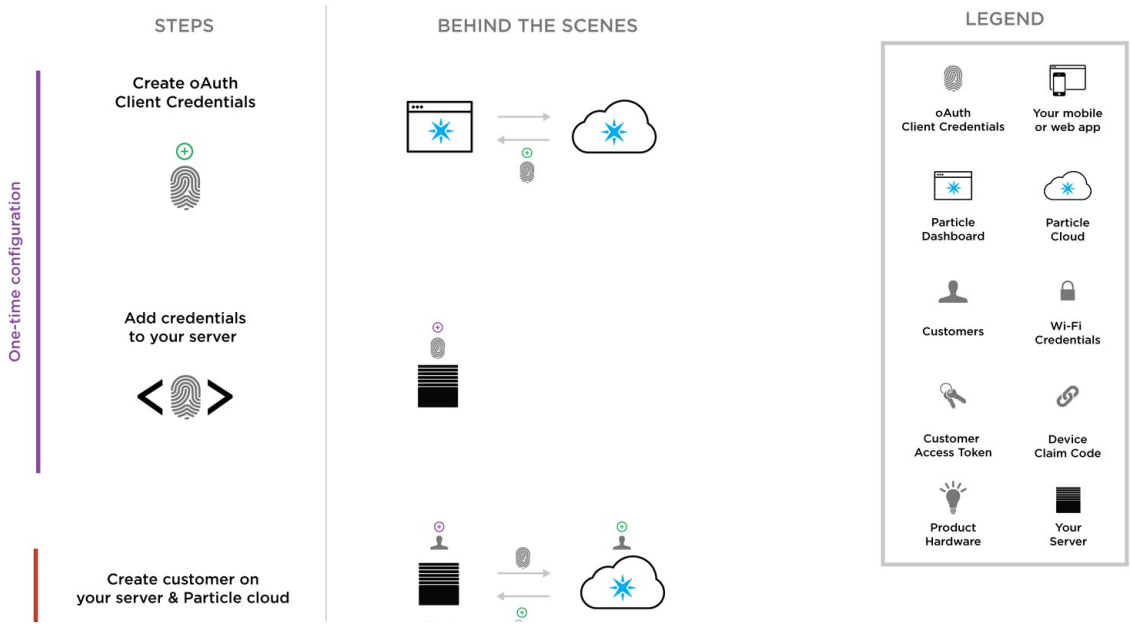
Because of the introduction of your own web server, implementing two-legged authentication adds complexity to the architecture of your application and the flow of data. There are simply more pieces of the puzzle that must all fit together.

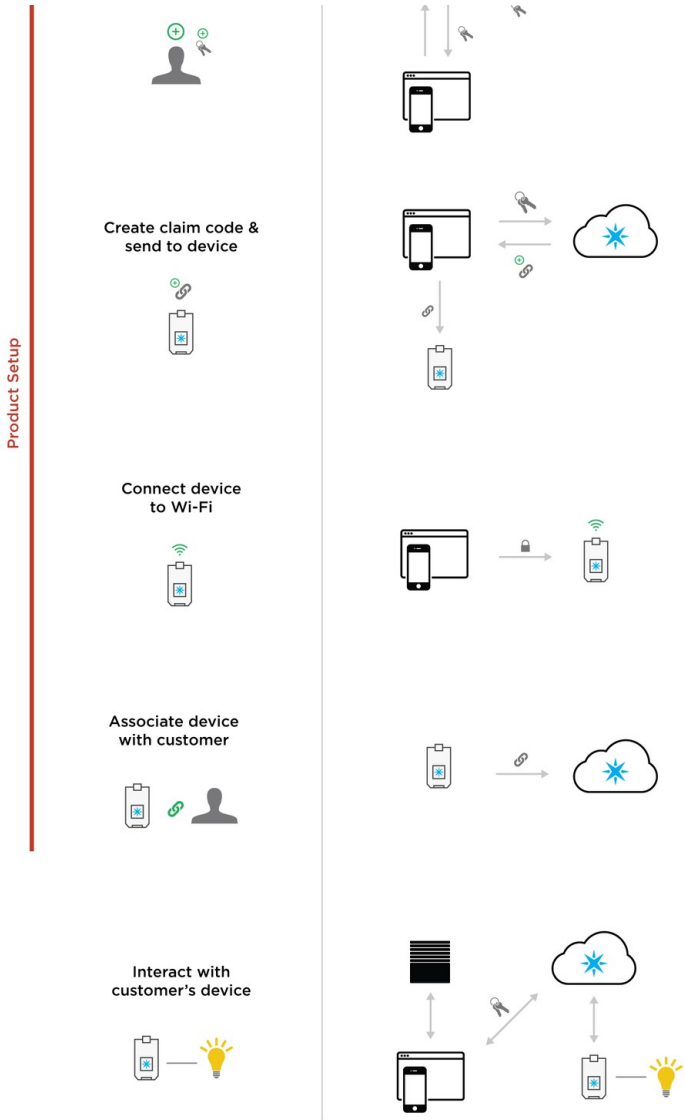
This will likely result in more development time than choosing Simple Authentication, and can introduce more points of failure for your application.

Two-Legged Implementation

Below is a diagram of the entire setup and authentication flow for the two-legged option. If you choose this authentication method, it is important that you understand the diagram very well. When comparing to the simple auth implementation, you'll notice that many of the steps are similar, with the exception of steps involving interaction with your web server.

Each one of the steps will be covered in detail below. Note that the first two steps are a one-time configuration process, whereas product setup will occur for each new customer that sets up a device.





The full two-legged authorization flow. See here for full size diagram

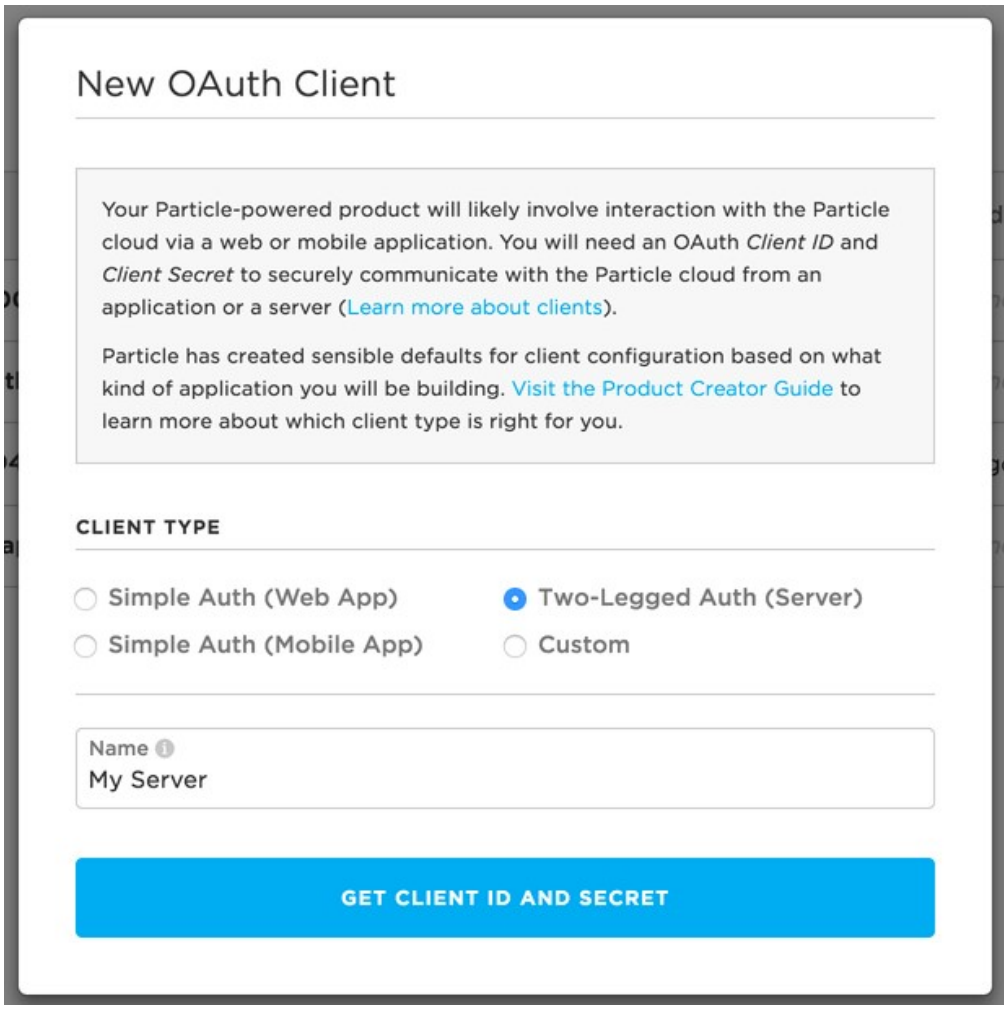
1. Creating OAuth Client Credentials

Like Simple Authentication, you will need to create valid OAuth client credentials for your product. Unlike simple authentication, your OAuth client credentials will be sent to the Particle API from your server, not directly from your mobile/web application. The client credentials will be used for two purposes:

- Creating new customers
- Creating *scoped access tokens* for customers

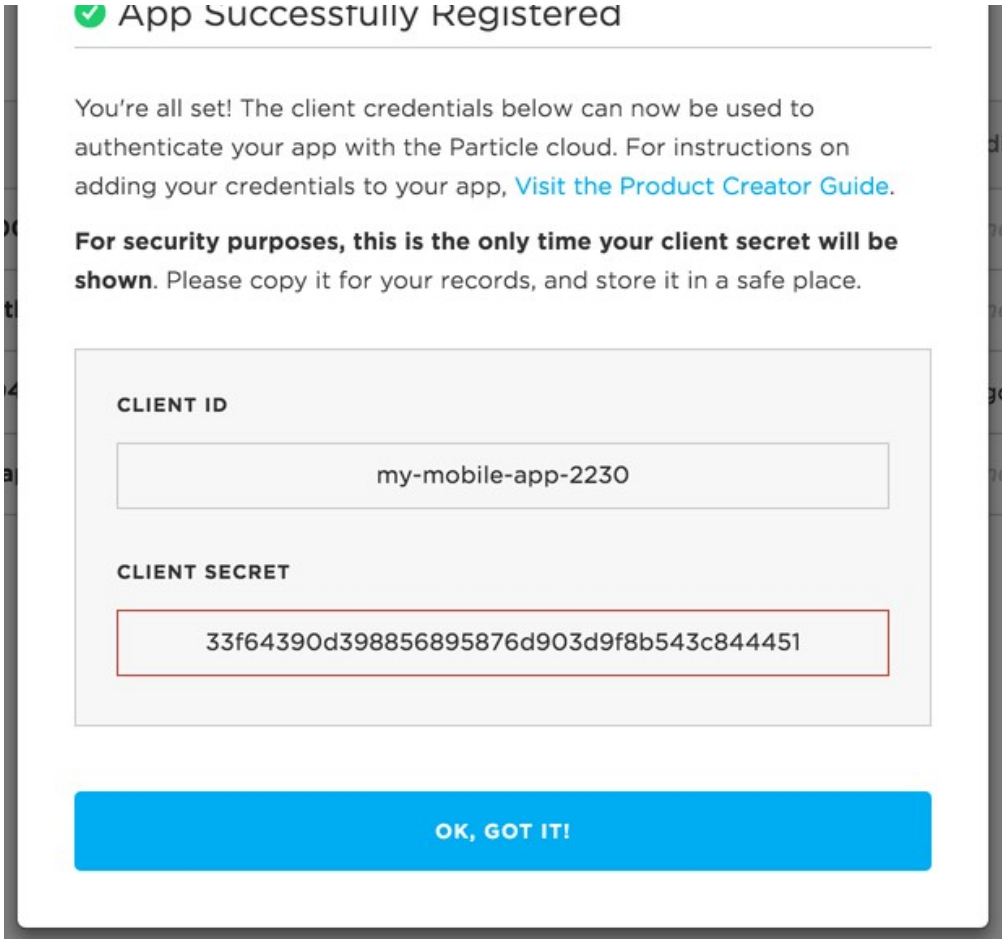
You will create your OAuth client using the Authentication view in your product's Particle console. For info on how to find the Authentication page and create a client, check out the earlier discussion.

For two-legged authentication, you should choose **Two-Legged Auth (Server)** from the client type options when creating a new OAuth client. This will provide the recommended client configuration, and only requires you to provide a name for your new client.



The recommended client configuration for Two-Legged Authentication

You will receive an OAuth client ID and secret upon successful creation of a client that will look like this:



Your client secret will only be shown once, so be sure to capture it for your records

This client ID and secret will be added to your server in the next step below.

Creating OAuth client can still be done directly against the Particle API. For info on this, see reference documentation on creating OAuth clients. Because the communication is server-to-server, you do not need to specify a scope. Without a scope, your client credentials can be successfully used for both of the purposes listed above.

2. Add OAuth Credentials to your server

Your *server* will need access to your newly created OAuth client ID and secret. Unlike simple authentication, **both** client ID and secret will be needed for two-legged authentication. Your server should have access to the client credentials anytime it needs to make an API call to Particle.

You must add your OAuth client credentials to your server

Because of the presence of your server, you should not need to add these credentials to your web or mobile application.

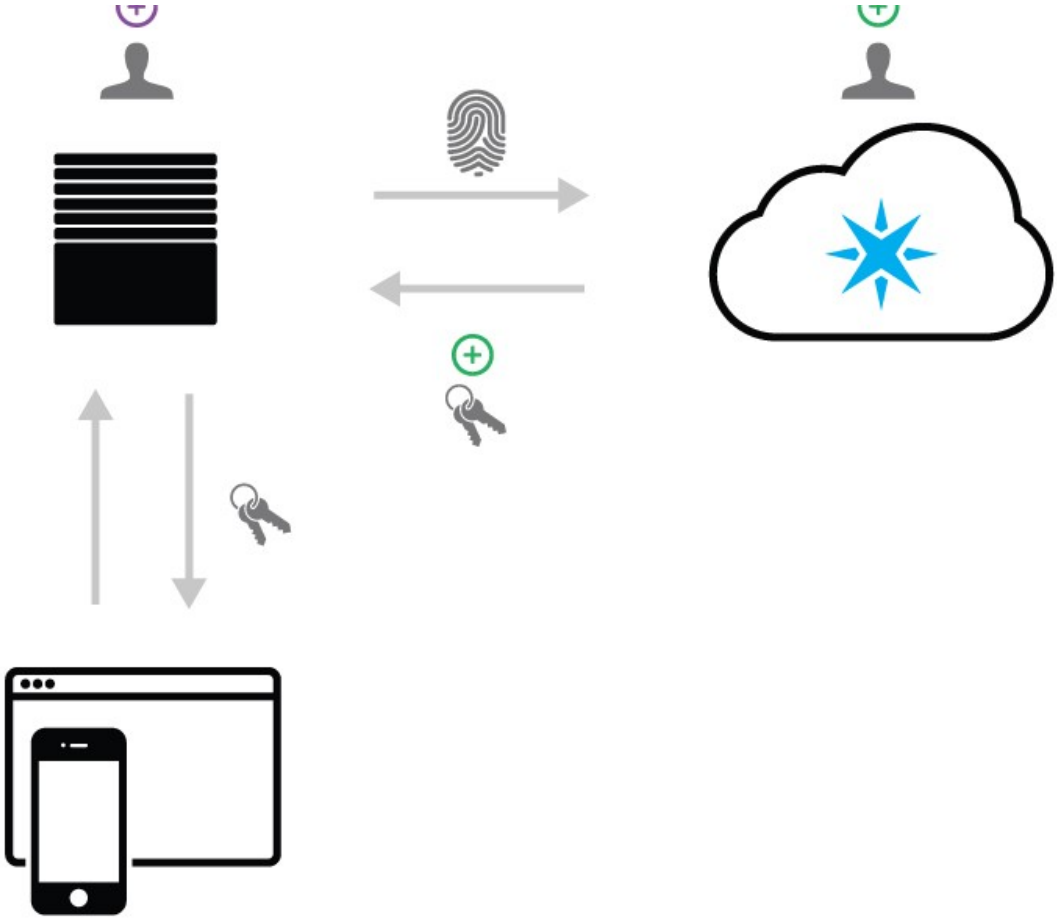
Do not share your client ID and secret publicly. These credentials provide the ability to fully control your product's devices, and access sensitive information about your product. We recommend never publishing the client ID and secret to a GitHub repository.

Coming soon: example implementation of client credentials

3. Create a customer

When using two-legged authentication, you will likely be managing customers on your own database *(Note: We realize that you may not call end-users of your product "customers" like we do. You may simply refer to them as users internally. However, we will continue calling them customers here to avoid confusion)*.

An important thing to understand is that even though you will be creating customers yourself, you will also need to create a *shadow customer* on the Particle cloud. That is, for every customer you create on your back-end, an mirroring customer record must be created using the Particle API.



You will create a Particle shadow customer in addition to creating the customer on your own back-end.

A Particle shadow customer is **required** to interact with Particle devices when using two-legged auth. Specifically, the customer must exist in the Particle system so that your server can generate *access tokens* on behalf of the customer. This allows your mobile or web application to interact with the customer's device.

The Particle shadow customer should be created at the exact time that the customer is created in your system. As you will be managing customer credentials on your own server/database, a shadow customer **should not have a password** when they are created. You will still be able to generate access tokens for the customer using your OAuth client ID and secret instead of passing a username/password for that customer.

The API endpoint to create a customer is `POST /v1/products/:productIdOrSlug/customers`. A request to create a customer could look something like:

```
curl -X POST -u "client-id-goes-here:client-secret-goes-here" -d
email=abu@agrabahmonkeys.com \
-d no_password=true
https://api.particle.io/v1/products/widget-v1/customers
```

Note that there is no password for the customer. An email address is the only piece of information required to create a customer in the Particle system, and **must be collected by your application during signup**. As a result, you must pass the `no_password=true` flag to create the customer with no password. Note that in this endpoint, you should use your client ID and secret instead of an access token.

As the diagram above suggests, you will receive an access token in the response of the `POST` to creating of the customer. You will use this access token during the device claiming process as well as to interact with the device once it's set up.

Reference docs on creating a customer

If you are using Particle's iOS SDK, there is a hook available to inject the customer's access token into the mobile client. You can learn about this hook here.

Device Setup (Steps 4, 5 & 6)

Now that you have created the customer, and received a valid access token for that customer, it is now time to start the device setup process. This process will occur in exactly the same fashion as with Simple Authentication, and will not involve your server.

It is important to note that the device setup process will not involve your server. All communication will be between your web/mobile application, the customer's device, and the Particle cloud. These steps will also be handled by the mobile & JavaScript SDKs, involving extra work from you.

Because these steps are the same as for Simple Authentication, documentation will not be duplicated here. Instead, you can check out:

Step 4. Create claim code and send to device

Step 5. Connect device to Wi-Fi

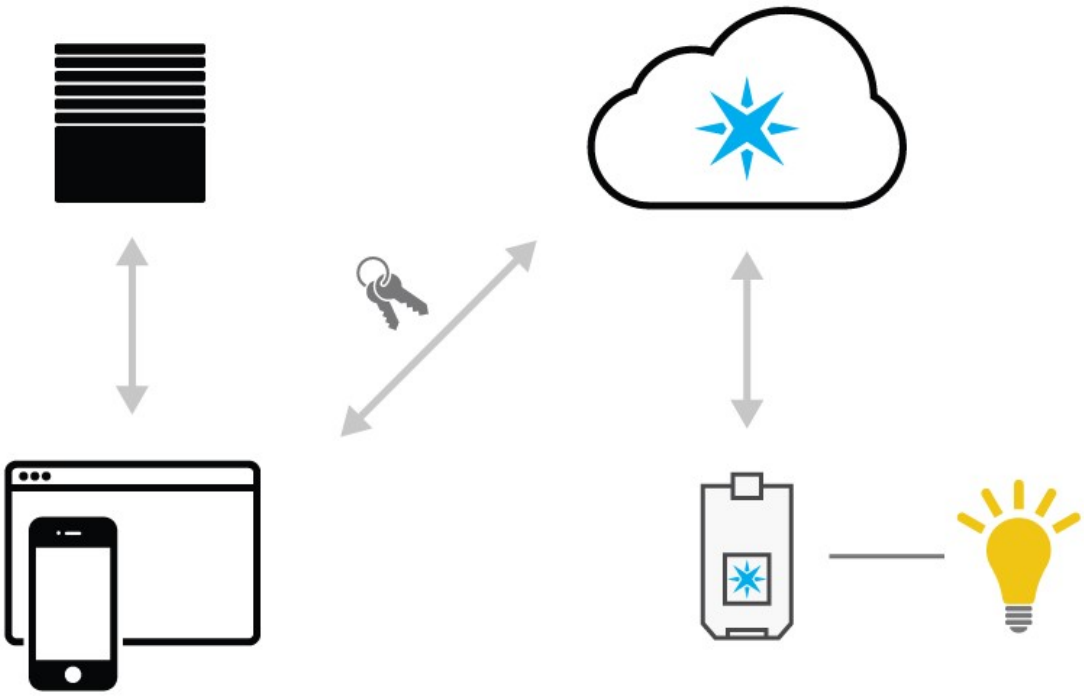
Step 6. Associate device with customer

7. Interact with Customer's Device

Once the device has been setup, and the customer created, you're now ready to interact with the device! Hooray! It's important to understand that while you do have your own server, **we recommend hitting the Particle API directly from your application client for any device-related actions**. This includes:

- Calling functions on the device
- Reading variables on the device
- Listing devices for a customer

This should be straightforward, as using the SDKs will provide helper methods for these actions.



For all direct interactions with the device, hit the Particle API from your application client

The alternative would be telling your application client hit your back-end server, which would then trigger a call to the Particle API. This introduces an extra intermediary in communication, and involves needing to unnecessarily "wrap" Particle API endpoints.

The *only* reason for your server to hit the Particle API is to generate new scoped access tokens for your customers. You should generate a fresh access token each time the customer logs into your application.

To do this, you will use the `POST /oauth/token` endpoint, but in a special way. The request will look like this:

```
curl -u my-org-client-1234:long-secret -d
grant_type=client_credentials \
-d scope=customer=jane@example.com
https://api.particle.io/oauth/token
```

Breaking this down:

- The `-u` is a HTTP Basic Auth header, where you will pass your OAuth client ID and secret. This allows you to generate access tokens for customers that belong to your product
- `grant_type` is the OAuth grant type, which in this case is `client_credentials`
- `scope` is set to customer, and what allows you to specify the customer you'd like an access token for. The value of `customer` should be set to the `email` you passed when creating the customer.

The response should look like this:

```
{
  "access_token": "254406f79c1999af65a7df4388971354f85cfee9",
  "token_type": "bearer",
  "expires_in": 7776000,
  "refresh_token": "b5b901e8760164e134199bc2c3dd1d228acf2d90"
}
```

The response includes an `access_token` for the customer, that should be included for all subsequent API calls for the session. In addition, there's a `refresh_token` that you could use to generate a new access token in the event that the token expires. Here's how to use your refresh token to get a new access token:

```
curl -X POST -u client-id-1234:secret \
-d grant_type=refresh_token -d
refresh_token=b5b901e8760164e134199bc2c3dd1d228acf2d90 \
https://api.particle.io/oauth/token
```

The response will be identical to the new access token creation endpoint above.

Login with Particle

(Coming Soon)

SECURITY

When it comes to IoT devices, there is no shortage of security concerns. This part of the guide will be a discussion of security-related topics that arise with a product at scale.

Treating Devices as Products

When you created a product, you created a new type of device in the Particle cloud. So far, development kits that you have been using for prototyping have been considered as a "Photon," "P1", "Electron," or "Core" by the Particle cloud. However, when you added devices to your product, you are signaling to the cloud that you would like them to be treated as part of your product instead of as development kits.

There are certain security implications that arise around devices behaving as products. Ideally, the following is always true:

- Only the *devices* that be given product privileges will receive them
- Only the *people* who should have the ability to interact with the product device will be allowed to do so

There can be, however, cases when the Particle cloud won't have all the information it needs to be certain that a device should in fact be treated as a part of your product. As such, you have the power as a product creator to choose the level of strictness around which devices receive product privileges and which do not.

Before making this decision, it is important to get a grasp of the role of the Product ID in the Particle cloud.

Product IDs

A Product ID is used to group devices together and treat them as a cohesive unit. A group of devices that all share a Product ID are able to do things like

download product-specific firmware, publish events that will appear in a product's event stream, and trigger product webhooks.

When you created your product, a unique numeric ID was assigned to it. This ID will be used countless times during the development and manufacturing process for your product. When a device is added to a product, the *Product ID* of the device changes from a development kit to a member of a product.

You will be able to find your product's ID at any time in your product's console navigation bar:

Your product ID is marked with a key icon

You should store your Product ID in a safe place where you won't forget it. It may be helpful to memorize your Product ID if possible.

Definitions of Product ID

There are **2** places where a Product ID is defined for a given device.

- **Cloud**: The Particle cloud keeps a record of devices and their associated Product IDs
- **Device**: The device itself will report a Product ID when coming online based on the firmware running on the device

Your goal as a product creator is to **ensure that both of these Product ID definitions match** for the devices that should behave as your product. When both of these IDs match, the Particle cloud can proceed with confidence in treating the device as part of the product. If these IDs *don't* match, the device will be limited in its privileges as a member of the product.

The table below shows the three possible combinations of Product IDs between the cloud and the device. In any case when the Product ID of the device does not

match the Product ID saved in the cloud, you should expect limited product functionality.

Cloud	Device	Status
1	1	✓ OK
1	2	⚠ Limited Privileges
2	1	⚠ Limited Privileges

The device's firmware Product ID should match the Product ID of the device in the cloud

Product Privileges

A device that has been successfully added to a product will be granted special privileges. These privileges include (but are not limited to):

- Receiving over-the-air (OTA) firmware updates from this product
- Publishing events that will appear in this product's event stream
- Claiming by a customer of your organization
- Triggering product webhooks

These privileges are only guaranteed when there are no mismatches between the Product ID of the device in the cloud and the Product ID reported by the device in firmware.

Mismatched Product IDs

There is the possibility that the Product ID saved in the Particle cloud for a given device does not match the self-identified Product ID reported by the device in firmware.

These mismatches can occur from a multitude of reasons, including:

- Your team forgot to add the device to the product before directly flashing it with product firmware either during development or during manufacturing

- Someone outside your team mistakenly flashed their device with firmware containing your Product ID
- Someone outside your team is deliberately attempting to "spoof" your product by purposefully putting your Product ID in their device's firmware

You as a product creator have the ability to choose how these devices should be handled in the case of mismatched Product IDs. There are two choices for handling these kinds of devices: *Quarantining*, or *Auto Approval*.

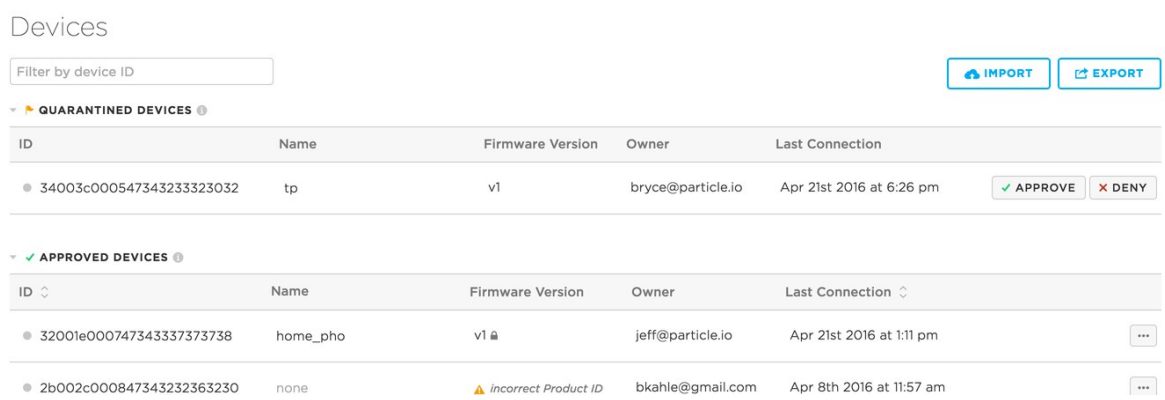
Ideally mismatches in Product IDs never happen, because you or a member of your team has taken the steps to add the device to your product before the device comes online with your product's firmware binary.

Quarantining Devices

This is the most secure and recommended option for handling unrecognized devices. By default, unrecognized devices will be quarantined unless you override the setting in your product's config page. This is done to protect against potentially malicious actions from unrecognized devices.

A device will be put into a quarantined state when it comes online and reports a Product ID in its firmware that is different from the Product ID saved in the cloud for that device. This happens when a device was flashed firmware containing a given Product ID, but that device hasn't been properly added to the product via the console.

When this happens, the device will temporarily lose all product privileges . In your product's devices hub on the Particle Console, you will see a list of your quarantined devices.



Quarantined devices will be called out in your product's device list

Note that each quarantined device has two potential actions you can take: *Approve* or *Deny*.

Approving a Quarantined Device

Approving a quarantined device effectively adds it to your product and gives it full product privileges. You should approve a quarantined device if you recognize it as belonging to a member of your team or a customer of your organization.

It is important to note that if your product has a released firmware version for this product, the device may immediately update if necessary.

Denying a Quarantined Device

Denying a quarantined device will ensure that the Particle cloud will continue blocking it from product privileges. You should deny a quarantined device if you do not recognize the device or its owner.

Denied devices will be visible in your product's device hub, but will appear collapsed. You can re-approve a denied device at any time, in case you do in fact want to add the device to your product.

Auto Approving Devices

The other option for handing unrecognized devices is to automatically approve them into your product. This is a *less secure* option, but may be necessary if you don't have access to the list of Device IDs that you are expecting to come online and self-identify as your product.

Auto approving devices will automatically add the unrecognized device to your product (with some restrictions) if it self-identifies as your product without the cloud knowing about it.

For security reasons, a device will *only* be automatically approved if either of the following conditions are met:

- The device is unowned when it comes online
- The device is owned by a user that is a team member of your organization

This safeguard has been put in place to prevent a blatant attempt to spoof your product. However, because devices can connect to the cloud without an owner, it is possible that an undesired device could be automatically added to the product. This is why we recommend that you use quarantining as the mechanism for handling unrecognized devices, unless you have a compelling reason to use auto-approval.


BUILDING YOUR OWN WEB APP

For most connected products, the heart of the customer experience lies with the hardware device and the firmware that runs it. But to unlock the true potential of the internet of things, to create a valuable, maybe even magical experience for your customers, there need to be capabilities that rely on the power of the internet.

How do you surprise and delight your customers? You want them to think, “Wow! How is that possible?” How do you take deeply complex data from around the internet and make a product that “just works” even though it shouldn't at that price point?

You do the hard work on a server, keeping the device simple.

(Back when we used to be named Spark, we made a fun video to help explain this.)

Our REST API allows you easily to build web services that interact with your products in real time. You can build these web services in any programming language (JavaScript/Node.js, Ruby, Python, C#, PHP, etc.), although we provide the most support for JavaScript/Node.js through our ParticleJS JavaScript SDK. If you are interested in libraries for interacting with our API in other programming languages, please search our forum for community-generated libraries.

You have already learned a lot about what you'll need to do to build your web application in the last section on Authentication & Security. This section will focus more specifically on building web applications.

Collecting data through webhooks or an event stream

Imagine you're making a social game that creates a live feed of pictures as customers use your product. Each time a customer hits a physical button on the hardware product, a picture is added to a Twitter-like timeline. To accomplish this, you will listen to the events published by all your products in customers' hands all over the world.

There are two ways to do this.

Webhooks

Webhooks are well-understood and easy to set up. You write your web app with a URL that Particle servers will call whenever one of your products publishes an event.

You make a single Particle API call to create the webhook, and from that moment on, events matching your webhook will trigger our servers to call your web app.

Coming soon: the ability to create a webhook for an organization or product.

Check out our Webhooks Guide for more information.

While easy to understand and set up, webhooks at scale might require some considerations. Each triggering of a webhook causes a request to be sent, so make sure your app can handle the traffic. For example, a small beta run of 1000 devices, each publishing once a minute might cause anywhere from 17 to 1000 requests in a given second, depending on how the publishes line up. Webhooks will support some basic queuing and smoothing to help address bursts like this, and a number of cloud hosting providers offer data ingestion services that can handle continuous heavy streams and bursts of requests (Azure, AWS).

Event streams

If you've created an app that subscribes to Twitter's streaming API, you'll feel right at home with the other solution. Your server can subscribe to a single stream of server-sent events that will efficiently transport all the data your products can publish without all the HTTP overhead for every event.

Check out our API reference on subscribing to event streams for more information.

When deploying on Heroku, you'll need to create worker process to listen to the event stream. Here are some links to help out.

- https://github.com/heroku-examples/node-articles-nlp

- https://devcenter.heroku.com/articles/asynchronous-web-worker-model-using-rabbitmq-in-node

Coming soon: a easy-to-deploy Heroku example. (Ping @zachary in the forum about this.)

Initiating device behavior by calling functions

Now imagine you want a customer's device to dance rainbows when someone likes a picture they've posted. The web app will need to hit the Particle API to call a function on one device to cause it to dance rainbows.

What's next?

We can't wait to show you just how easy it is to build a web app for your product on Particle!

If your product is better suited for a mobile app, you will want to build your iOS and Android mobile apps using our mobile SDKs.


BUILDING YOUR OWN MOBILE APP

While using your Photon, you've probably come across our Particle mobile apps (iOS and Android). You may have used these apps to set up your device and "tinker" with the GPIO pins.

Particle provides native mobile SDKs for both iOS (Objective-C and Swift) and Android (Java) that allow you to build your own branded app for setting up and interacting with your product. If all you need is the set-up process, you can build your own branded app by editing a 15-line configuration file in the **Device setup library** (no software development necessary). If you want to build a complete mobile interface, our iOS and Android **Cloud SDKs** provide native access to our API for interfacing with the product.

You have already learned a lot about what you'll need to do to build your web application in the last section on Authentication & Security. This section will focus more specifically on building mobile applications.

Prerequisites for Building & Deploying Mobile Apps

iOS

- Mac computer/laptop running latest macOS
- Apple developer account
- iOS device & USB lightning cable (Particle device setup process cannot run on simulator)
- XCode 6 and up
- CocoaPods installed
- Particle iOS SDKs: Cloud SDK and Device setup library
- Skills in object oriented programming. Knowledge in Objective-C / Swift and Cocoa Touch APIs. Here are few recommended free learning resources:
    - Official Apple tutorials
    - Developing iOS 8 Apps with Swift Stanford University CS193p course on iTunes U

    - Ray Wenderlich - a great iOS-centric tutorials website
    - Try iOS - Free online iOS course from Codeschool
    - and always: Stack Overflow - best Q&A website for programmers. You can probably find an answer to ALL your how-do-I-do-that iOS questions there.

Android

- PC or Mac computer running OS X, Windows, or Linux (i.e.: anything that can run Android Studio)
- Google Play developer account
- An Android device running Android v4.0 and up (the Particle device setup process isn't supported via emulators), and a USB cable to connect the device to your computer
- Android Studio v1.4 and up
- Particle Android SDKs: Cloud SDK and the Device Setup library
- Basic familiarity with Android development using Java and the Gradle build system. Here are few recommended free resources to get you started:
    - Official Google tutorials
    - Udemy Learn Android Programming From Scratch free online video course
    - Udacity Developing Android apps course
    - and as always: Stack Overflow - best Q&A website for programmers. You can find good answers to almost *any* how-do-I-do-that Android questions there.

Two-tier SDK

There are two parts to the Particle Mobile SDK: the Cloud SDK and the Device Setup library. In a nutshell, the **Cloud SDK** is a library that enables your mobile app to interact with internet-connected hardware through the Particle Cloud. It serves the same purpose as ParticleJS – it's an easy-to-use wrapper for our REST API, accessible from Objective-C and Swift. The **Device Setup library** allows you to create a setup wizard within your app for connecting your device to the internet with two lines of code.

How To Get Started?

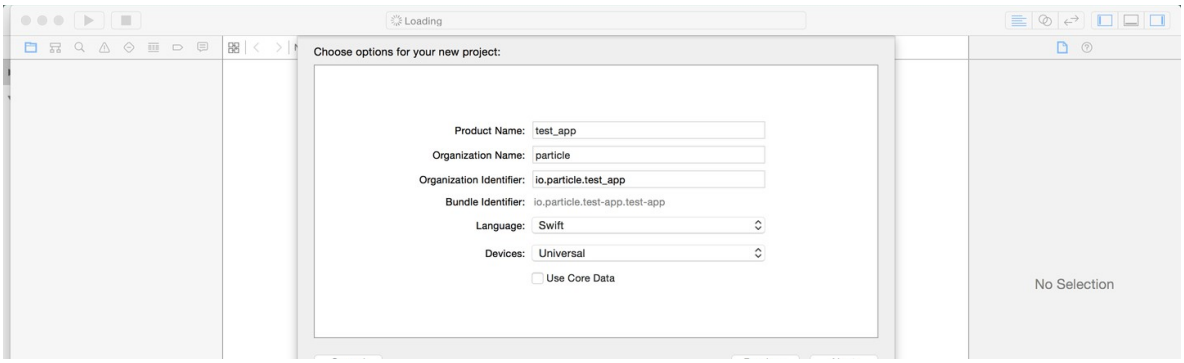
Let's go through a basic step by step example on how to integrate and use the mobile SDKs into a basic app, for both platforms:

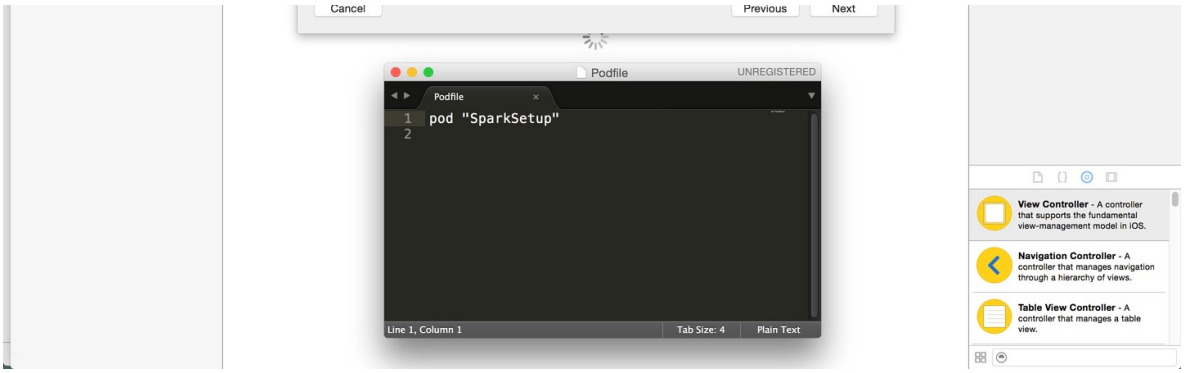
iOS

Both the Cloud SDK and Device Setup library are available through CocoaPods, the most widely used iOS dependency manager. If you don't have it, you'll need to start by installing the CocoaPods `ruby gem`; check out the CocoaPods site for more info.

Starting from scratch

Go ahead and create a new app in XCode by going to File menu and then: New -> Project -> iOS Application -> Single View Application -> Next then name your app and identifier, choose if you prefer to code in Obj-C or Swift, decide if the app is an iPhone/iPad or Universal app and click Next to place your new project in a folder. Project will open up.





Open Finder or Terminal and go to your project folder - create a new plain text file named `Podfile` in the same directory then install the Device Setup library for iOS (which has the Cloud SDK as a dependency). Simply add the following line to the `Podfile` in your iOS project root folder:

```
pod "SparkSetup"
```

save & exit. Now, from command line, while still in the project root directory type:

```
pod install
```

Go back to XCode, close the project and open the newly created `.xcworkspace` file - your project will now contain the Particle iOS SDKs ready to use.

Go to the project storyboard, drag a UIButton to your main ViewController. Double click it and type "Setup device", press the "Assistant editor" in XCode toolbar to show your viewcontroller code side by side to the storyboard. Ctrl-drag the button to your code to create a new IBAction, name the function "startDeviceSetup". In the function body add:

```
- (IBAction)startDeviceSetup:(id)sender {
    SparkSetupMainController *setupController =
[[SparkSetupMainController alloc] init];
    [self presentViewController:setupController animated:YES
completion:nil];
}
```

Or the Swift version:

```
@IBAction func startDeviceSetup(sender: AnyObject) {
    var setupController = SparkSetupMainController()
    self.presentViewController(setupController, animated: true,
completion: nil)
}
```

If you're using Objective-C, don't forget to import the file `SparkSetup.h` in your view controller implementation file. If you're using Swift, be sure to complete all the required steps to integrate the Objective-C CocoaPods libraries in your project, mainly adding bridging header file to the project settings, as described here. We've included a bridging header file in both the SDKs.

That's it. Build and run your project on a device or a simulator, tap the "Start Setup" button you created and you should see the device setup wizard pop up ready for authenticating with Particle Cloud and then setting up a new Particle Device. Make sure you set up your new Photon, and name the device `myDevice` at the last screen, you'll see why in a moment. If you already setup your device and just need to rename it you can do it from Particle Build -> Devices. You can also rename the device from the Tinker app.

Now, let's try to list your devices and read a variable from the device you just set up (by using the Cloud SDK). Stop the app and go back to the split view of your view controller and code. Drag another button and name it "Read Variable", Ctrl-Drag it to your code and create another IBAction function. Call the function "readVariableButtonTapped" and fill in its body like so:

```
- (IBAction)readVariableButtonTapped:(id)sender {
    // 1
    [[SparkCloud sharedInstance] getDevices:^(NSArray
*sparkDevices, NSError *error) {
            NSLog(@"%@",sparkDevices.description);
            // 2
            if (!error)
            {
                // 3
                for (SparkDevice *device in sparkDevices)
                {
                    if ([device.name isEqualToString:@"myDevice"])
                    {
                        SparkDevice *myPhoton = device;
                        // 4
```

```
                        [myPhoton callFunction:@"digitalwrite"
withArguments:@[@"D7",@"HIGH"] completion:^(NSNumber *resultCode,
NSError *error) {
                            // 5
                            if (!error)
                            {
                                NSLog(@"Called a function on
myDevice");
                            }
                        }];
                    }
                }
            }
        }];
}
```

or the Swift version:

```
@IBAction func readVariableButtonTapped(sender: AnyObject) {
        // 1
        SparkCloud.sharedInstance().getDevices {
(sparkDevicesList : [AnyObject]!, error :NSError!) -> Void in
            // 2
            if let sparkDevices = sparkDevicesList as?
[SparkDevice]
            {
                println(sparkDevices)
                // 3
                for device in sparkDevices
                {
                    if device.name == "myDevice"
                    {
                        // 4
                        device.callFunction("digitalwrite",
withArguments: ["D7","HIGH"], completion: {
(resultCode :NSNumber!, error : NSError!) -> Void in
                            // 5
                            println("Called a function on
myDevice")
                        })
                    }
                }
            }
```

```
        }
}
```

Step by step explanation See `// 1,2,3..` comments in code and follow:

1. here we are calling the SparkCloud to get a list of all the device the user owns (the device you just set up should appear here), we also print the list onto the console so you can check everything is in place
2. If the cloud did not return an error (in Swift we are optionally downcasting the result list to an Array of `SparkDevice` s)
3. Then lets iterate on the returned array searching for a `SparkDevice` with a `name` field which is `myDevice` (as we set it up)
4. Once its found, call the function `digitalwrite` on *this* device with two arguments, which mean `D7=HIGH` which should cause the onboard LED (connected to pin D7) to light up
5. And if there wasn't any error calling this function on the device (default Tinker firmware exposes this function always) then print to console that call was successful

Go ahead and run the app, see everything works alright. Well Done! You've just created a mobile basic app that can:

1. Set up a Particle device interactively
2. List the devices on the user's account
3. Call a function on a device and report to the user

Modifying existing app

If you prefer to have a point of reference you're more than welcome to modify an existing app. Ideas for modifiable apps include:

- Example app
- Particle Tinker open-source app
- Other users/3rd party apps, ideas?

There's also an example app written in Swift that demonstrates the basic usage of invoking the setup wizard, customizing its UI and using the returned SparkDevice class instance once Device Setup wizard completes.

You can find the source code for the Cloud SDK under our GitHub account:

Repository of iOS Cloud SDK

Repository of iOS Device Setup library

Android

The overall process for building an Android app is very similar to that of building an iOS app, but with a native Java experience so Android developers will feel right at home. One major difference in the user experience of setting up a Photon from an Android app vs. an iOS app is that on Android an app can control which Wi-Fi network the phone connects to, whereas on iOS the user has to leave the app, go to settings, and change the Wi-Fi network. We've made both flows as easy as possible, but it's definitely smoother on Android since the user doesn't have to do as much.

The Cloud SDK and Device Setup library for Android are available on our GitHub and through JCenter as Apache Maven packages for easy integration as dependencies in an Android Studio project.

The guide for how to create an Android app for Particle devices in Android studio is coming soon. Meanwhile you can find the source code for the Android Cloud SDK and Device Setup under our GitHub account:

Repository of Android Cloud SDK

Repository of Android Device Setup library

What's next?

Well now you've written all of your software, so it's probably time to start:

Manufacturing >.

MANUFACTURING, PROGRAMMING, AND TESTING

You've finished your hardware and software development. Your PCB is designed and tested, you've built your own web and mobile apps, and you've had a few beta customers successfully use products that you've built for them by hand. You've lined up a contract manufacturer (CM), and now you're ready to start manufacturing.

In most cases, manufacturing with Particle hardware is the same as manufacturing with any other electrical component. However there are some things to keep in mind when you program and test your products, and some opportunities to consider since your product is connected to the Internet.

Purchasing Particle hardware at volume

You have two options when it comes to purchasing Particle hardware--purchasing from Particle, and purchasing directly from our module manufacturer. Let's run through the pros and cons of each:

- **Purchasing from Particle** means that you're buying either Photons or PØ/P1 modules through our online store, or purchasing larger volumes by contacting sales. They'll be shipped to you from our warehouses in cut tape strips of ten (10) units, or as complete reels. There are several advantages to purchasing directly from Particle. In general, we try to keep hardware in stock so that we can provide quick lead times and prompt delivery timelines. We've already established relationships with our module manufacturer, so you can leverage our order volumes to get better pricing. Working with large silicon manufacturers can be difficult and slow, and we strive to be friendly, communicative, and easy to deal with. Additionally, when you buy hardware from Particle, it includes all the necessary firmware libraries and system firmware for connecting to the Particle Cloud by default, simplifying your manufacturing process. This option is best for low to medium volume product deployments (<50K units).

- **Purchasing from our module manufacturer.** If your order volumes for the PØ or P1 are very large (50-100K), and you prefer to manage your own

supply chain relationships, we encourage you to place an order directly to our module manufacturer. We are happy to provide all necessary introductions to get your purchase order submitted; please contact sales.

Lead times. Please keep in mind that, although we do our best to keep our hardware in ready supply, you should always plan on 8-12 weeks of lead time for delivery, especially if your order is very large. This is one of the most common mistakes that our customers make; if you're planning to use Particle hardware in production, reach out to our sales team early in your development process so we can ensure your hardware is on order and in production well before you need it.

Duties/Taxes. As the end customer, you are responsible for any VAT charges or taxes associated with importing the modules to their final destination. If you are shipping into mainland China, our prices are reflective of FOB Hong Kong, and do not include import. Although we do not provide direct delivery to mainland China, we have plenty of experience that you can leverage; contact sales if you have questions about importing raw materials, freight forwarders, import licenses, VAT rebates, or anything in between.

Material import (China only)

If you are purchasing P0s, P1s, Photons, or Electrons for manufacturing in mainland China, you will need to consider China's VAT (value added tax) on electrical components. This tax (typically 17-23%) applies to *all* electrical goods purchased from international distributors, and not just to Particle hardware.

NOTE: The industry standard for the fulfillment of electrical components is to support delivery to Hong Kong, but not to locations in mainland China. This is due to the wide variety of options available for importing materials, and the varying complexity and costs associated with those options. Particle abides by this industry standard and does not currently accept orders with final delivery to mainland China.

There are a number of ways to import goods into China for manufacturing--here's a quick overview of the two most popular options. Others exist, and it is highly recommended to ask your contract manufacturer which import method they prefer and support. Each of these methods will likely require the services of a freight forwarder to physically move the goods between Hong Kong and

mainland China--your contract manufacturer probably has a preferred forwarder that you can leverage for this purpose.

1. Pay the VAT

- **Cost**: ~20% value of goods
- **Complexity**: Low
- **Reliability**: High
- **Flexibility**: High

The simplest method is to pay the VAT in full. This will add roughly 20% to the cost of your raw materials, but will ensure reliable and expedient transport of your components. There is paperwork that has to be completed, but it is relatively simple, and your contract manufacturer or freight forwarder should be familiar with this process. This is the most expensive option, but allows for the most flexibility with respect to your BOM and manufacturing timelines, as the paperwork is resubmitted each time you bring goods into the country, and no prior arrangements have to be made.

2. File for an import license

- **Cost**: ~$5,000 single time fee, small fixed fee per import/export event
- **Complexity**: High
- **Reliability**: High
- **Flexibility**: Low

The best way to eliminate the additional burden of the VAT is to file for an import license for your product. Here's how it works--you (more likely your contract manufacturer) submit an application to the government that includes a final version of your BOM and samples of your product. You can then be granted a license to import goods for the manufacturing of the product without the necessity of paying the 20% VAT. Here's the key--*all of the materials used in the manufacturing of that product must first be imported into China under the license, and then exported out of China under the same license*. It's all an exercise in accounting--the government wants to see that all of the goods that were brought into the country were converted into finished goods using the "recipe" (BOM) you provided to them in advance, which guarantees that excess materials aren't being sold into local markets. Even the parts that are manufactured domestically within mainland China have to first be *exported* to

Hong Kong so they can be *reimported*, used in manufacturing, and then *re-exported* out of the country. Import licenses take about 3-6 weeks to receive, and can be very slightly modified (adding cross parts) but not drastically changed (adding new packaging elements or electrical components) without resubmission. They have a ~$5,000 price tag, so they have a negative ROI for small batch runs or product designs that are likely to change.

3. Reimbursement

- **Cost**: 1-5% value of goods
- **Complexity**: High
- **Reliability**: Medium
- **Flexibility**: High

The third and final option, which we have found to be less common than the other two, is to work with a manufacturer that is able to participate in VAT reimbursement. Reimbursement works a lot like an import license, but the accounting is done in retrospect instead of in advance. In this scenario, your contract manufacturer would pay the full VAT, and file for reimbursement of the tax once they can demonstrate that the goods they imported were used in manufacturing and then exported as finished goods. Reimbursement of the VAT can take anywhere from 3-6 months, though, so it would be typical for a manufacturer to charge interest on the capital that was leveraged to pay the VAT up front. This might be anything from an additional 1-5% on top of the value of the goods being imported, depending on the manufacturer. Not all manufacturers are able or will be interested in this kind of importing because it adds significant burden and risk to their sourcing and purchasing processes.

Testing

Before you can begin manufacturing, you need to consider how your product is going to be tested on the manufacturing line. The firmware that runs the functional evaluation of your PCB is called "test firmware". If you're using a Photon, PØ, or P1, there is a very basic set of test routines built into the default firmware that you are free to use, however they may not provide sufficient test coverage for your product.

A good functional evaluation has many parts, but includes robust, targeted testing of the power supply, microcontroller, critical ICs, and user-exposed

peripherals. Specifically, if you are building a product with Particle hardware, you should carefully consider testing the following:

1. Serial communication. Although exposing the serial communication lines to your Photon/PØ/P1 is not a strict requirement of products build with Particle hardware, it's highly encouraged and makes gathering test results from your product during testing much simpler.

Exposing serial for debugging allows you to interact with Particle's basic test firmware to do useful things like:

- Ask your device to reboot
- Ask your device to perform a factory reset
- Ask your device to connect to a known Wi-Fi network
- Ask your device to scan for available networks
- Ask your device to report its connection strength
- Ask your device to clear all stored Wi-Fi credentials
- Ask your device to report its unique hardware identifier

These are all useful functions during manufacturing, so we highly encourage exposing serial debugging for your product.

2. RGB LED. In general, RGB LEDs are extremely moisture-sensitive components that are prone to failure and must undergo 100% inspection. You can accomplish a functional test of your LED in a few different ways:

- *Visual test*. Create a step in your test firmware that sets your product's RGB LED to white. This requires full use of the red, green, and blue parts of the RGB LED. If any of these individual LEDs are non-functional, the LED will display an incorrect color, and can be separated for rework. This is a very easy check to implement into test firmware, but relies on the operators' discretion, which is less reliable than a quantitative optical characterization.
- *Optical characterization*. Use an optical sensor positioned over the RGB on your PCB to detect the color characteristics of your product's LED. The advantages of this technique are that it allows you to quantify the brightness of your LED by wavelength, and identify partially defective units (dim, flickering) more easily than if you were relying on the test fixture operator's eyesight and discretion alone.

3. Wi-Fi Connectivity. If you're building a product with Particle hardware inside, you're device probably connects to the Internet. It's important to make sure that the Wi-Fi module on your PCB is working correctly and that your antenna and RF system is functional. There are several different approaches you can take to testing your device's wireless capabilities:

- *Connecting to the Particle Cloud*. The most foolproof test is to configure your device to connect to the Particle Cloud on the manufacturing line. This requires that you have a wired Internet connection and router to which the device can connect. Depending on the contract manufacturer that you are working with, a live Internet connection may or may not be available on the line. This will require that you clear the device credentials on your product after the test.
- *Connecting to a Local Network*. If Wi-Fi isn't available on the manufacturing line, electricity probably still is. Power up a router with a known SSID and password and configure your Photon to connect to it. If your product can request and receive an IP address, you can be confident that your radio and antenna configuration are functionally operational.
- *Scan for Wi-Fi Networks*. A more quantitative test of your product's RF performance is to scan for networks and measure their signal strength. This requires that you set up a router with a known SSID at a specified distance and orientation from your testing station in order to ensure accurate benchmarking and consistent test results. Scanning for networks is a lightweight, quick test that still requires that your product both send and receive wireless packets, so it tests both key wireless functions of your product. Additionally, measuring signal strength allows you to create a quantifiable criteria for evaluating the wireless performance of your device.

4. Happy Path Batch Testing. Regardless of the checks you build into the test firmware, you should make sure that you are doing functional batch testing of your product before final shipment. It's a good idea to take a handful of units from the "passed" bin every day and use the same tools available to the customer to configure your product onto the Internet and Particle Cloud. Test the basic function and use cases of the product. This will help you identify issues and potential pitfalls that might otherwise fall through the cracks of your functional tests and prevent a user from successfully configuring your product.

Programming the device

Now that your product has successfully completed functional testing, you need to think about how the default firmware for your product is going to be programmed onto the device.

All Particle hardware comes off the manufacturing line loaded with Tinker, application firmware designed for use with Particle's mobile app of the same name. The Tinker app makes it easy to start playing and experimenting with your Particle device without writing any code. It's great for early development, but isn't suitable for a standalone product.

If you're building a product on the Particle platform, you'll need to flash application firmware to your device that makes your Internet connected coffee-maker a coffee-maker and not a Photon. There are two basic strategies for flashing firmware to your product--both are outlined below.

Option 1: Flash code on the manufacturing line. The traditional solution to getting firmware onto your device is to flash new code to the microcontroller on the manufacturing line. In a typical testing scenario, you might insert your PCB into a test fixture, flash test code to the device which runs a suite of functional evaluations on your product (described above), and then, when the product has passed functional testing, program it with the default factory firmware that you want to greet your customer with upon startup. If your product is designed to work both with and without an Internet connection, it's imperative that you flash your firmware application on the manufacturing so it can still function in offline mode. If you have a significant number of peripherals or sensors that require specialized testing outside of the default test suite on Particle hardware, you should similarly make accommodations to program your PCB on the line. This can be accomplished using a STLink v2 Programmer or a Particle Programmer Shield connected to the exposed JTAG pins/pads on your PCB.

- *Advantages*: Flexible, robust, product is functional out-of-the-box, required for products with sensors and additional peripherals
- *Disadvantages*: Slower testing process (adds ~20s per flash event per test), requires more complicated testing hardware and software

Option 2: Flash code over-the-air during initial setup. The second option is to leave the default Tinker firmware on the device until the end-customer

configures your product onto Wi-Fi for the first time. If your device is relatively simple but requires an Internet connection to function, you can flag your devices in the Console so that when they connect to the Particle Cloud for the first time, they automatically download the most recent version of your product's application firmware. In the previous coffee-maker example, your product would think it was a Photon up until the moment it was configured by a customer for the first time, and would only be capable of making coffee after it was setup onto Wi-Fi and was re-programmed to do so. Now, this clearly isn't the right strategy for a coffee-maker that still has to function if Wi-Fi is down, but might make sense for a button that your customer can press to re-order garbage bags from Amazon. A garbage bag orderer-er doesn't need to "work" until after it's connected to the Internet, and doesn't have complicated sensors and peripherals that require specialized functional that tests. For these kinds of simple, specialized products, programming over-the-air is a powerful and effective strategy for determining the end behavior of your product.

- *Advantages*: Fast and simple; minimizes resources spent on test fixture development and optimization
- *Disadvantages*: Product is non-functional until configured onto Wi-Fi; not suitable for products that require more comprehensive testing

Have questions about which programming strategy is right for your product? Ask us! We're always happy to provide our two cents.

COMING SOON!

- Detailed instruction for flashing over JTAG and DFU using Particle's bootloader

Provisioning your device

If you've decided that you want to purchase your wireless hardware direct from our module manufacturer, there are some additional steps you'll need to complete in order for your product to successfully connect to the Particle Cloud.

- **Licensing fee.** If you buy Particle hardware directly from us, the licensing fee for accessing the Cloud is built into the cost of the hardware. If you're buying hardware directly from our module manufacturer, their quoted cost will *not* include this licensing fee, and your devices will be unable to access

the Cloud until you pay it. Generally, it's a one-time cost of ~$2 per device. To get detailed pricing, please contact sales.

- **Device ID.** In order for a device to connect to the Particle Cloud, its unique hardware identifier must be registered with our database--this helps us keep the Cloud safe and protect your devices from users and devices that don't belong there. If you purchase hardware directly from Particle, we've already captured this data and can automate the device ID insertion and tagging process behind the scenes. If you purchase from our module manufacturer directly, though, you'll have to be very diligent about collecting and reporting these identifiers to us during manufacturing so that we recognize your product when it attempts to open up a socket to our Cloud. If an end-user or customer attempts to setup a device without a recognized device ID, the Cloud will reject the connection. Note that this is a significant risk that requires significant attention during the manufacturing process.

- **Particle System Firmware.** Buying hardware directly from Particle means it will come pre-programmed with a bootloader, our system firmware (an embedded OS for your device), and Tinker, our default user app. If you purchase directly from our module manufacturer, the hardware will be blank and require that you flash it using your product's exposed JTAG pins/pads in order to function properly. At a minimum, you'll need to flash our system firmware to the device so that it has the information it needs to communicate with our Cloud during initial configuration and setup.

COMING SOON!

- Instructions and resources for flashing Particle system firmware to your device on the manufacturing line

Device tracking and serialization - *Coming soon!*

Connected devices provide the opportunity to compare customer behavior across distribution channels *if* you track your hardware properly. Consider adding a unique serial number for each device and tracking the mapping between these serial numbers and the device IDs.

- Recording device IDs for customer support and record-keeping
- Creating setup identifiers for your device
- Creating trackable bar codes
- UUIDs on the Particle platform

What's next?

Now that you've finished manufacturing, you're ready to ship! Let's talk about:

Certification >.

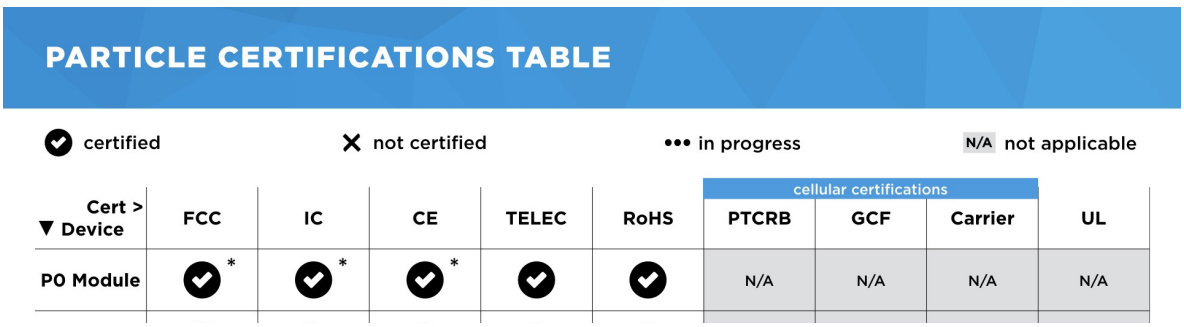
CERTIFICATION

Overview

Connected devices nearly always require certifications that ensure they operate within the wireless transmission specifications determined by international regulatory bodies like the FCC and CE. This part of the guide will provide you with more information on how to leverage Particle's existing certifications to reduce the cost, complexity, and time associated with demonstrating regulatory compliance with your end product.

Table of Contents

Here is a graphical representation of the current state of certification across Particle's hardware portfolio:



Photon	✓	✓	✓	✓	✓	N/A	N/A	N/A	N/A
P1 Module	✓	✓	✓	✗	✓	N/A	N/A	N/A	N/A
Electron G350	✓	✓	✓	N/A	✓	Contact us	Contact us	Contact us	N/A
Electron U260	✓	✓	N/A	N/A	✓	✓	N/A	Contact us	N/A
Electron U270	N/A	N/A	✓	✗	✓	N/A	Contact us	Contact us	N/A

* the P0 has no antenna, and is FCC/IC/CE certified under the reference design for the Photon

It is important to understand which certifications apply to your end product. Click here for a full size image. Updated 4/19/17.

FCC

Description

- **Website**: http://www.fcc.gov
- **Wikipedia**: https://en.wikipedia.org/wiki/FCC_Declaration_of_Conformity
- **Domain**: United States

The FCC (Federal Communications Commission) is an independent agency of the U.S. government that is in charge of regulating interstate and international communications. The FCC marking is required on electronic products manufactured or sold in the United States that certifies that the electromagnetic interference from the device is under limits approved by the FCC.

Integration

The Photon as well as the PØ/P1 Modules are covered under certifications from the FCC. Certified radio modules comply with the "Intentional Radiator" portion (Part 15c) of FCC certification.

- The **Photon** is certified as a single-modular transmitter that carries a modular grant. Modular certified radio modules are allowed for integration into multiple host end products by the FCC.

- The **PØ Module**, which is on the Photon, does not itself have an antenna and thus does not have a discrete certification from the FCC. However, it is certified for integration into a host product under the RF reference of the Photon as a design guideline.
- The **P1 Module** is also certified as a single-modular transmitter that carries a modular grant, and is certified for integration into multiple host end products by the FCC.

Any host product incorporating the Photon/PØ or P1 modules does not require additional testing or authorization for the Wi-Fi transmitter as long as:

- An antenna of the same type **and** equal or lesser gain to the antenna used for certification is used on the product.
- Any restrictions found in the grants are followed in the OEM's end product integration of Particle hardware.

The Photon/PØ and P1 modules were certified with antennas of the following gain:

- **Photon/PØ Module**: Chip antenna (1.3dBi), external antenna (2.15dBi)
- **P1 Module**: External antenna (2.5dBi)

Customer Responsibilities

Even though we've certified the Photon and PØ/P1 Modules, as a product creator you're still responsible for meeting test requirements determined by the class of your product as described by the FCC. In general, there are two types of products--**Class A** and **Class B**:

- **Class A**: Digital device for use in commercial, industrial, or business environments.
- **Class B**: Digital device for use in residential environment not withstanding use in commercial, business, and industrial environments. Examples of such devices include, but are not limited to, personal computers, calculators, and similar electronic devices that are marketed for use by the general public.

Particle's certifications will help you decrease the time and cost associated with certification by allowing you to reuse our FCC ID as the FCC ID for your end

product for Part 15c of your product's wireless certification. The remaining sections, for which all integrators and product creators are still responsible, are described below:

FCC Part 15 Certification

- **PART A: General Provisions and Definitions**
  In this section, you will classify your device according to the definitions set out by the FCC. Included in this section are definitions for things like "intentional radiators", "kits", "test equipment", and "digital devices".
- **PART B: Unintentional Radiators**
  This section covers devices whose purpose is not to produce radio waves, but which do anyway, including computers, voltage regulators, and oscillators/crystals. It's likely that your end product contains unintentional radiators. Part B allows for self-classication, which means that you don't have to get expensive test reports to demonstrate unintentional radiators in your product.
- **PART C: Intentional Radiators**
  This section covers devices whose purpose is to produce coherent radio waves. The Photon/PØ/P1 modules are intentional radiators. This is the most difficult part of FCC certification, and is where you can reuse Particle's hardware certifications to significantly simplify the application process. For instance, a from-scratch certification might cost $10-30K and take 4-8 weeks, while a verification of conformity might only cost $1-5K and take 2 weeks.
- **ADDITIONAL TESTING: Determined by device classification**
  You'll also be responsible for any additional testing requirements defined by the FCC for your product. You can learn more about additional required testing for your product by visiting the FCC website, linked below.

MORE INFO - Electronic Code of Federal Regulation for Telecommunications, Part 15: Radio Frequencies Devices

Documentation

Photon/PØ - (updated 3/20/17)

- FCC ID: 2AEMI-PHOTON
- Certificate of Conformity
- Test Reports
- Test Firmware and Instructions

P1 Module

- FCC ID: COFWMNBM11
- Test Reports
- Test Firmware and Instructions

Electron U260 - (Updated 4/19/17)

- FCC ID: XPYSARAU260
- Certificate of Conformity - U260 v1.0
- Test Reports - U260 v1.0
- Certificate of Conformity - U260 v1.1
- Test Reports - U260 v1.1

Electron G350

- FCC ID: XPYSARAG350
- Certificate of Conformity - G350
- Test Reports - G350

IC

Description

- **Website**: http://www.ic.gc.ca/Intro.html
- **Wikipedia**: http://en.wikipedia.org/wiki/Industry_Canada
- **Domain**: Canada

  IC (Industry Canada) is a department of the Government of Canada that, among many other things, issues Technical Acceptance Certificates (TACs) for Category I radio and broadcasting equipment.

Integration

The Photon/PØ and P1 Modules are IC certified as single-modular transmitters. Just like with FCC Certification, any host product incorporating the Photon/PØ or P1 modules does not require additional testing or authorization for the Wi-Fi transmitter by IC as long as:

- An antenna of the same type **and** equal or lesser gain to the antenna used for certification is used on the product.
- Any restrictions found in the grants are followed in the OEM's end product integration of Particle hardware.

Customer Responsibilities

Industry Canada follows the same testing and rules as the FCC in regards to certified modules in authorized equipment.

Documentation

For certificates of conformity, search here using the IC ID for each Particle device, respectively.

Photon/PØ - (updated 3/20/17)

- IC ID: **20127-PHOTON**.
- Certificate of Conformity
- Test Report

P1 Module

- IC ID: **10293A-WMNB11**.
- Certificate of Conformity
- Test Report

Electron U260 - (Updated 4/19/17)

- IC ID: **8595A-SARAU260**.
- Test Report - U260 v1.0
- Test Report - U260 v1.1

Electron G350

- IC ID: **8595A-SARAG350**.
- Test Report

CE

Description

- **Website**: https://ec.europa.eu/growth/single-market/ce-marking/
- **Wikipedia**: http://en.wikipedia.org/wiki/CE_marking
- **Domain**: Europe

The CE mark is a mandatory conformity marking for certain products sold within the European Economic Area (EEA). It is analogous in that sense to the FCC marking used on devices sold within in the United States.

Integration

Both the Photon/PØ and P1 are certified and carry the CE marking. Section 1.3.1 of the Guide to the R&TTE Directive 1999/5/EC states the following regarding the attachment of antennas to a module that already has documentation of conformance:

"Manufacturers who place on the market products without an antenna or with an antenna that is intended to allow replacement have a responsibility to provide information on the general types and/or characteristics of antennas that may be used with their equipment in order that the overall radio equipment remains compliant. The guidance of the transmitter manufacturer has to be followed when they are installed."

For the Photon/PØ and P1 Modules, these integration instructions are the same as for the FCC/IC--so long as you're integrating the modules without modifying the RF design or implementing an antenna with gain greater than that used in certification, your equipment will likely remain compliant.

The Photon/PØ and P1 Modules are certified with ETSI radio tests which can be accepted by a number of countries for radio compliance.

Customer Responsibilities

If a product has adhered to the integration guidelines and has minimal risk, it can be self-certified where manufacturers complete a Declaration of Conformity and affix the CE marking to their own product.

Please note that Particle is not responsible in any way for issues arising from inappropriately self-certified products and devices using Particle hardware. EMC testing obligations may still be required as determined by the specific end product requirements.

- The end product will still need to be filed in each country for certification using the FCC/ETSI radio reports.
- 40+ countries recognize and accept radio test reports compliant to ETSI standards as part of the filing process, but note that some countries do not recognize modular approval.
- In most cases, it is possible to leverage Particle's ETSI reports, and thus testing does not need to be repeated.

Documentation

Photon/PØ - (updated 8/6/17)

- Certificate of Conformity
- Test Reports

P1 Module - (updated 5/8/17)

- Certificate of Conformity

Electron U270

- Certificate of Conformity
- Test Reports

Electron G350

- Certificate of Conformity
- Test Reports

TELEC 

Description

- **Website**: http://www.telec.co.jp/
- **Documentation**: http://www.tele.soumu.go.jp/e/index.htm
- **Domain**: Japan

Particle has completed TELEC certification to achieve compliance with Japanese Radio Law. If you are seeking to distribute your product in Japan, please contact Particle for more information about compiling TELEC-compliant firmware.

Documentation

Photon/PØ

- Certificate of Conformity
- Test Report

RoHS 

Description

- **Website**: http://ec.europa.eu/environment/waste/rohs_eee/index_en.htm
- **Wikipedia**: https://en.wikipedia.org/wiki/Restriction_of_Hazardous_Substances_Directive
- **Domain**: Europe

RoHS stands for the "Restriction of Hazardous Substances Directive" adopted by the European Union in 2003 and effective as of 2006. It restricts the import and

distribution of electronic and electrical equipment with six hazardous materials within the EU. Those hazardous materials are listed below:

Substance Name	Allowable Limit
Lead (Pb)	less than 1000ppm
Mercury (Hg)	less than 100ppm
Cadmium (Cd)	less than 100ppm
Hexavalent chromium (Cr6+)	less than 1000ppm
Polybrominated biphenyls (PBB)	less than 1000ppm
Polybrominated diphenyl ether (PBDE)	less than 1000ppm

There are also a short list of exemptions to RoHS regulation that are worthy of note.

RoHS compliance is self-declared and there is no certification body that governs compliance, unlike the FCC. Particle has submitted its hardware to RoHS compliance testing and the test reports are available below.

Documentation

Photon/PØ

- Test Reports

Electron U260/U270/G350

- Test Reports

UL

Description

- **Wikipedia**: https://en.wikipedia.org/wiki/UL_(safety_organization)#UL_Standards
- **Domain**: Worldwide

UL (Underwriters Laboratories) is an American worldwide safety consulting and certification company. UL provides safety-related certification, validation, testing, inspection, auditing, advising, and training services to manufactures, retailers, policymakers, regulators, service companies, and consumers.

Integration

The large majority of UL certifications, which can be found here, are standards for electrical and electronic products that utilize high voltage AC electricity for power. UL certification is typically not required for low voltage or battery powered products.

Customer Responsibilities

UL certification and safety standards are not applicable to Particle hardware, but may be applicable to the host end product in which they are integrated. It is the product creator's responsibility to ensure compliance with all UL safety standards and to obtain end product certification if required.

PTCRB

Description

- **Website**: https://www.ptcrb.com/
- **Wikipedia**: https://en.wikipedia.org/wiki/PTCRB

- **Domain**: North America

The PTCRB is a certification body selected by North American cellular operators to create and manage a common cellular certification framework. PTCRB certification is a certification prerequisite for nearly all devices connecting to North American cellular networks.

Certification variants

There are two common types of PTCRB certification – one that requires subsequent certifications after integration and one that does not. The Electron is certified with **"End Product"** certification, which does not require subsequent certification after integration, which decreases the certification burden on product creators building with the Particle platform.

Module certification This is the PTCRB certification level for cellular modules. The PTCRB defines a module in the following way:

> *Modules are finished WWAN radio devices that do not directly connect to a host via a standardized external interface such as USB, PCMCIA, Compact Flash, MMC, RS-232, or IEEE-1394. A module may or may not include an integral antenna system or SIM/USIM interface.*

Most cellular module vendors will seek PTCRB approval for their module in order to make subsequent End Product certifications simpler. Particle leverages the u-blox SARA-G350, SARA-U260, and SARA-U270 modules, all of which have module certifications from the PTCRB.

End Product certification This is the PTCRB certification level for finished cellular products. End Products are defined as devices that meet all of the following attributes:

- *Physical Interface* - If a physical control interface is required for the End Product, it shall utilize one of the following: USB, PCMCIA, Compact Flash, MMC, RS-232 (DE9), or IEEE-1394. No other physical control interfaces are acceptable.
- *Power* - Obtains power from the standardized physical control interface or have a provisioned power source (i.e. dedicated battery, or a dedicated power source).

- *UICC Interface* - Includes a fully self-contained USIM/SIM socket or embedded USIM/SIM
- *Antenna* - Utilizes a self-contained antenna or provide an external antenna connector.
- *Radio Access Technologies* - Covers at least one (1) comprehensive radio technology as specified by 3GPP for GERAN, UTRA, or E-UTRA devices

The Electron meets all of these specifications, and is thus certified by the PTCRB as an End Product.

Customer Responsibilities

When an End Product like the Electron is connected to a host device (PC, PDA, etc.), no certification of the host device is required. By its design and intended user application, an End Product must:

1. Be consistent with the terms of its FCC / Industry Canada type acceptance (e.g. type of antenna, distance from user, etc.). In all cases, the End Product shall not alter its antenna system in any way from that allowed by the associated Type Acceptance or other regulatory approval.
2. Serve its final intended use without any further hardware and software modifications. If a control interface connection to a host is required for operation of the device, that connection can only be made through one of the above defined standardized physical interfaces.

Documentation

Electron 3G (U260)

- Certified Device Detail

Electron 2G (G350)

- Product has completed and passed all PTCRB test requirements. Please contact Particle if you are building an end product with our 2G cellular solutions.

GCF

Description

- **Website**: http://www.globalcertificationforum.org/
- **Wikipedia**: https://en.wikipedia.org/wiki/Global_Certification_Forum
- **Domain**: Europe

The GCF is a certification partnership between European network operators, mobile devices manufacturers, and certification/test labs. It serves a very similar function to the PTCRB for European operators.

Documentation

Electron 3G (U270)

- Product has completed and passed all GCF test requirements. Please contact Particle if you are building an end product with our 3G-U270 cellular solutions.

Battery Certifications

The 2,000mAh lithium-polymer (Li-Po) battery included with the Electron and other Particle accessories is compliant with all international safety and transportation standards. Test reports below.

Model Number: LP103450
Ratings: 3.7V DC, 2000mAh, 7.4Wh
Manufacturer: PKCELL (Shenzhen, China)
Test Reports: Updated 3/20/17

- IEC62133
- UN Section 38.3

- MSDS Test Report
- 1.2m Drop Test
- Air Freight Transport
- Sea Freight Transport

What's next?

Now that we've covered product certifications, click through to our last page for some closing comments on building products with the Particle platform:

Further Considerations >

SIM MANAGEMENT BETA

SIM Management equips product creators with the functionality needed to operate at scale with cellular-connected devices. This suite of tools is focused on reducing friction of managing a fleet of Particle SIM cards at all stages of the SIM life cycle.

Some of the most notable features of SIM management for products include:

- **Import and pre-activate SIM cards en-masse** to ensure a smooth unboxing experience for end users of your products
- **Gain meaningful insights into data usage patterns** by viewing fleet-wide usage information
- **Set sensible cellular defaults** to save you time and effort without sacrificing control of individual SIM cards
- **Take advantage of volume-based pricing** for cellular service available exclusively to product creators

Seamlessly manage a large fleet of SIM cards by associating them with your Particle product

This set of features is currently released as a beta and is subject to change. Please report any issues you experience via our support form.

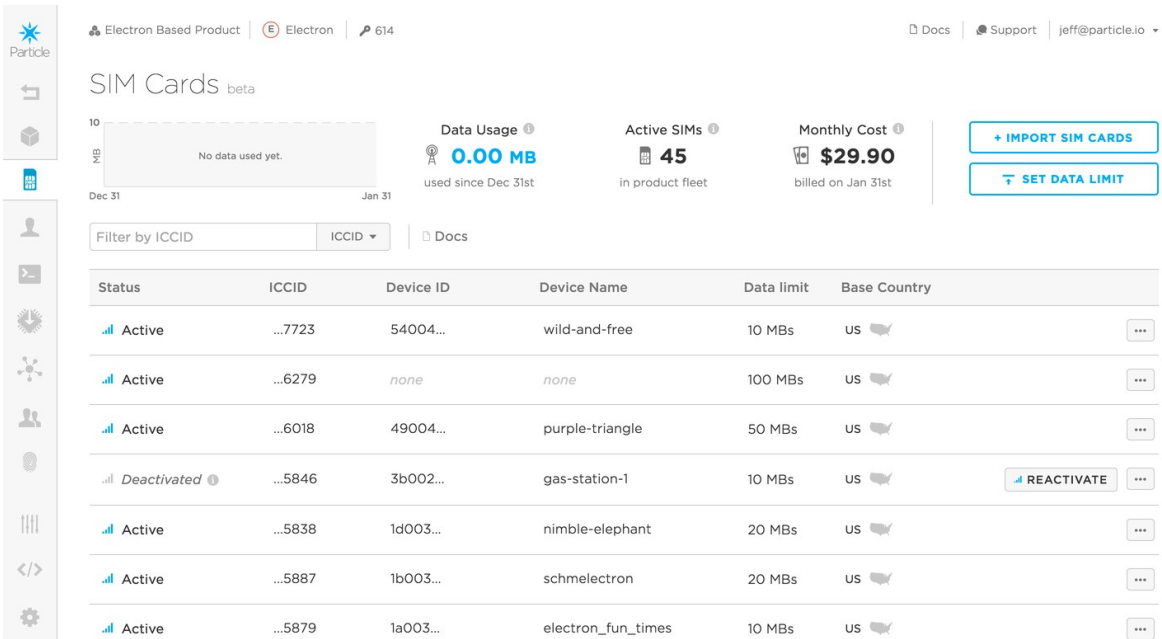
Introduction

The SIMs view is a new page on the Console that will be visible to those products that are cellular-based (i.e. you chose the Electron as the device type during product creation). It will appear as a SIM card icon in the Console side bar when looking at your product. To access this view and start exploring, click on the SIM card icon ( ).

Much like devices, SIM cards can now be associated with a Particle product instead of an individual developer account. It is important to be aware of 2 main implications of product ownership of a SIM card:

- The owning product takes billing responsibility of the SIM card and the data usage costs it incurs. This is useful to simplify product billing, which is covered more in the billing and invoicing section
- Any member of the owning product team gains access to view and manage the SIM (i.e. deactivation, changing the data limit). This makes it more accessible for your team to access and control SIM cards when necessary.

The SIMs page looks like this:



Here's a few things you will see on this page:

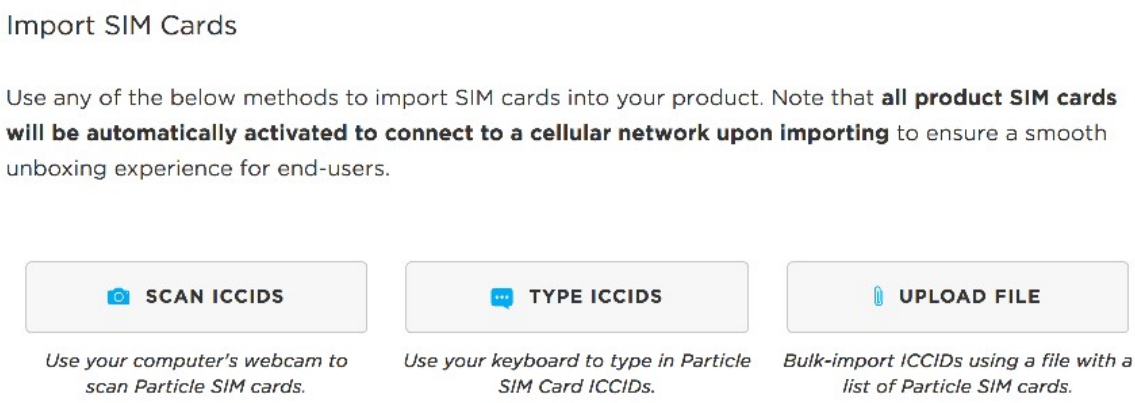
- A list of your product SIM cards, with details like its ICCID, connection status, associated device info, data limit, and base country
- Some statistics about your fleet's aggregate data usage
- Buttons to take action on your fleet, like importing new SIMs into the product, setting a default data limit, and more

Importing SIM Cards

You can import SIM cards into your product a variety of different ways:

- Typing in ICCIDs
- Scanning ICCIDs using your webcam
- Uploading a file containing a list of ICCIDs

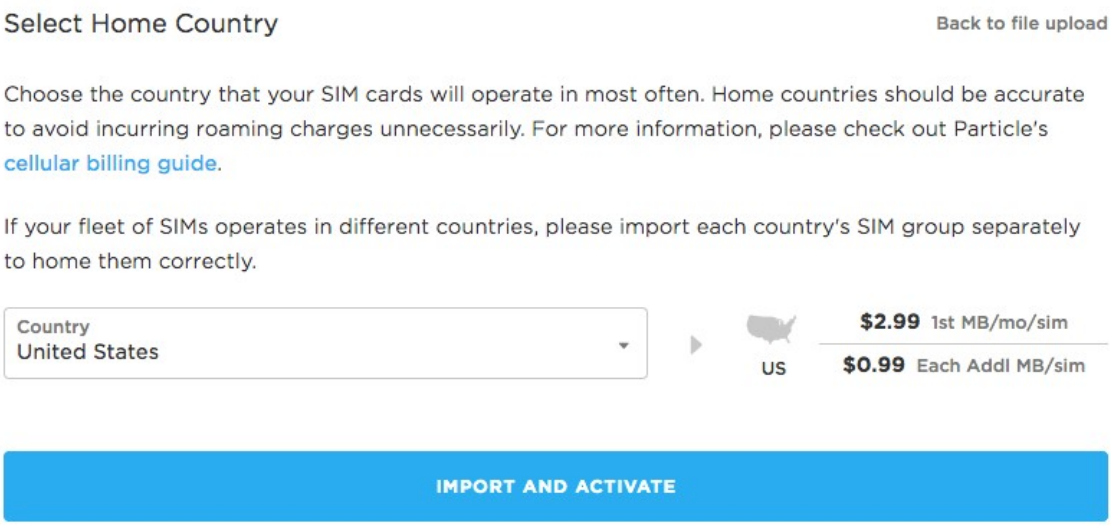
Regardless of the method you choose, **importing SIMs into a product also triggers SIM activation**. This is a key benefit of managing SIM cards on the product level: you are able to pre-activate large numbers of SIM cards to ensure a smooth unboxing and setup experience for end-users of your product.



You have methods to choose from when importing SIM cards into your product

SIM activation requires selection of a home country for billing purposes. For more information on home countries, please see the cellular billing guide. You

will be presented with a dropdown to select the home country after importing the desired ICCIDs:



You should be sure to choose the country that your SIMs will be operating in most often to avoid incurring unnecessary roaming charges. You also will see the SIMs' data plan cost details on this page. If you have a fleet of SIMs that operate in different countries, we suggest importing each country's SIM cards separately so they can be homed accurately.

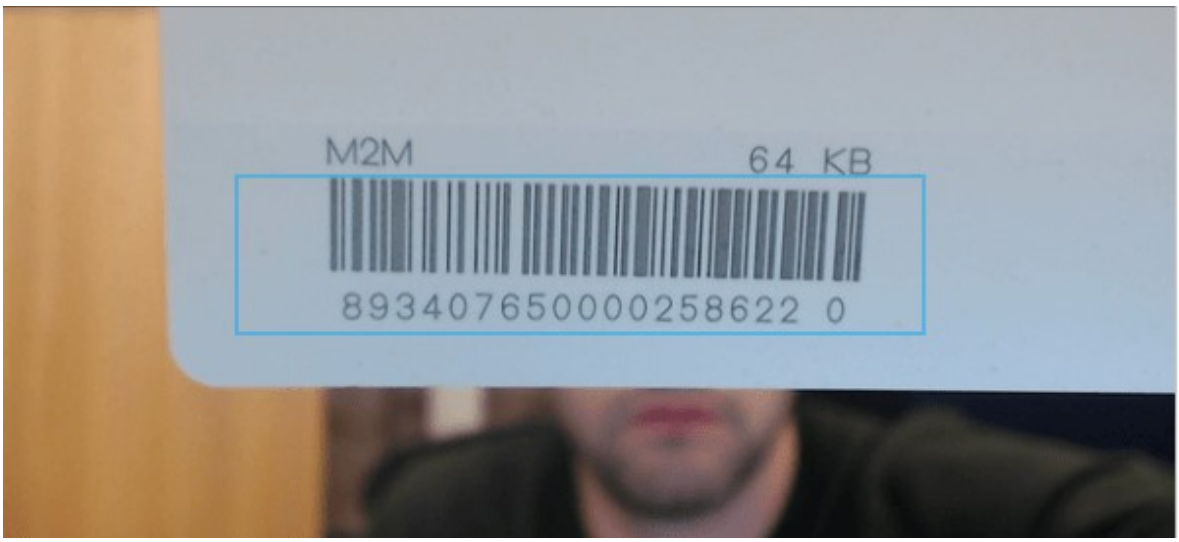
Upon importing, SIM cards will be activated immediately via Particle's MVNO service. For each SIM, you will receive a prorated charge for the first month's 1MB data plan on your next invoice.

Typing or scanning ICCIDs

For importing smaller numbers of SIM cards, it makes more sense to type or scan ICCIDs to import into your product. You can find the full ICCID number on the back of your Particle SIM carrier card, below the bar code. Alternatively, the ICCID is also printed directly on the micro SIM, if it has already been popped out of it's SIM carrier card.

You can find a Particle SIM's ICCID on the back of its carrier card

You also have the option to use your computer's webcam to scan a SIM card's ICCID for product importing. Note that **scanning works best with a webcam capable of 1080p resolution or higher**. As a result, many default laptop scanners will not work well for scanning. If you don't have this kind of webcam handy, we suggest typing ICCIDs or importing via a file.



To scan, position the bar code inside the cyan box, and ensure that it is in focus

Regardless of if you are typing or scanning ICCIDs, SIM cards will queue up for batch importing. When you have finished queueing up SIMs, click "Continue" to proceed to selecting a home country for the data plan.

Importing via files

For large numbers of SIM cards, it will be much easier to import and activate using a file. If you have purchased a large number of Particle SIM cards, you should have received a file containing a list of your ICCIDs from Particle.

The Console's UI makes it easy to drag-and-drop a file containing ICCIDs to use for importing. Your file must be a `.txt` and have one ICCID per line. Any other data in the file will be ignored.

Transferring SIMs owned by a developer

Often times, you will begin developing a cellular product as a prototype, and claim ownership of a SIM card as a Particle developer. You now might want to transfer this SIM card into your product to manage it as part of a larger fleet.

You have the ability to directly transfer ownership of a SIM owned by a Particle developer account into a product. Particle will seamlessly transition ownership from your account as a developer to your product. When importing via scanning or typing ICCIDs, the Console should give you a heads up that you are about to transfer ownership from your individual billing account to the product's billing account. In addition, be aware that these SIMs will:

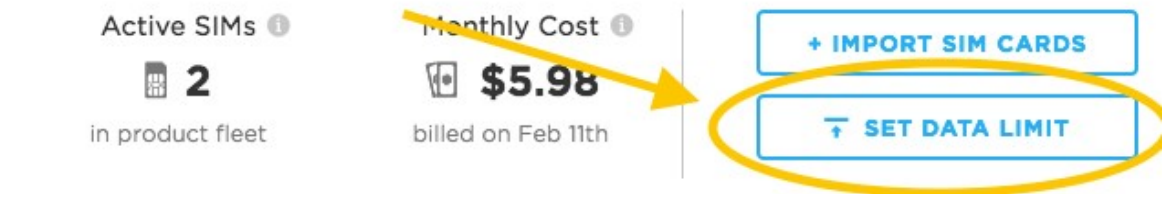
- Trigger immediate **charges for any data overages** incurred during this current month
- Be put on a new data plan where **existing promotions from the previous plan will no longer apply** (i.e. free 1MB data plan for 3 months)

Default Data Limits

Each SIM card has its own *data limit* or the maximum data that you'd like a particular SIM to be allowed to consume as a safety cutoff. For more information on data limits, please read the docs on SIM data limits.

For large product fleet, it can become cumbersome to individually set data limits for each SIM card. You have the ability to set a default data limit for your product. This data limit will be applied to any new SIM card imported into your product fleet moving forward. If your product does not have a default data limit set, Particle will apply a **5MB** limit to any SIM imported into your product.

To set or update your fleet-wide data limit, click on the "Set Data Limit" button on your product's SIMs page:

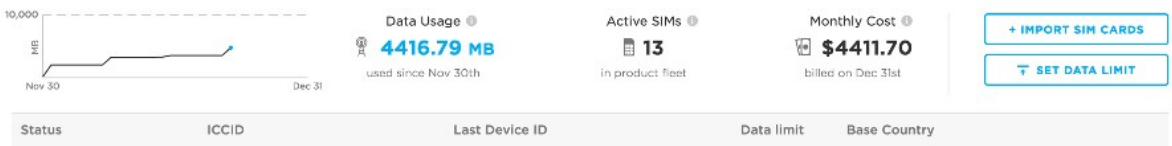


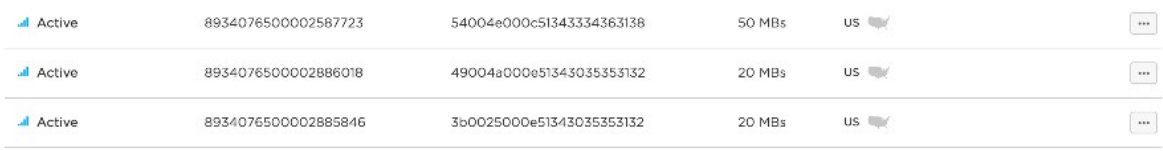
You also have the ability to override the default data limit on an individual SIM once it has been added to your product. This is useful if you know certain SIMs will consume more data, like a SIM used in a test device. To do this, click on the ... icon next to a SIM in the product list and select "Set data limit."

You can easily override the default data limit for an individual SIM using the Console

Monitoring your fleet

Another major benefit of SIM management within a product is gaining fleet-wide visibility into data usage and estimated cost. Previously, data usage and cost was only displayed on a per-SIM basis. Now, you have the ability to monitor your entire fleet of SIMs in real time from the Console:





The usage sparkline aggregates data consumption from all SIMs in your product for the current month, and plots usage over time. This could allow you to identify patterns of data usage, realize the usage impact of a new version of firmware, or spot unexpected spikes in consumption.

Coupled with the sparkline is some additional information about your fleet:

- **Data Usage**: represents the total number of megabytes consumed across your fleet of SIM cards since the start of the current billing period. Usage reports may be delayed by up to 1 hour.
- **Active SIMs**: represent all the SIM cards in your fleet that are either active, or temporarily paused due to hitting their monthly data limit.
- **Monthly Cost**: an approximation of the amount to be invoiced for SIM data usage at this point in time. The actual amount invoiced at the end of the billing period may change as SIMs consume additional data.

Together, this information paints a picture of how your fleet of SIM cards is behaving at a high level in the current month.

Individual SIM actions

Although SIM management within a product provides fleet-wide control and insight, it does not come at the expense of individual SIM control. In the SIMs view, you can take the following actions on a SIM card:

- **Set Data Limit**: Override the default data limit applied to every SIM in your product. This is especially helpful for SIMs that might use more data than average, like a SIM used for internal testing. Service will be temporarily paused on this SIM if the data limit is reached
- **Deactivation**: Immediately stops any device using this SIM from sending or receiving data via a cellular network

- **Reactivation/Unpausing**: Resume cellular data transfer of a deactivated or paused SIM and immediately enable the SIM to be used with a Particle device
- **Removal from Product**: Deactivate the SIM card and disassociate it from your product's billing account

Billing and invoicing

This section will outline billing and invoicing information specific to associating a SIM with a product. For general information about how SIM data is billed, check out the cellular billing guide.

Data plans

Each SIM card will have its own individual data plan and overage cost, similar to what you might have experienced owning a SIM as a Particle developer. The particular base rate and overage cost per-MB applied to your SIMs will depend on 2 things:

- **Where it is "homed"**: When importing the SIM, you choose the home country for a SIM card(s) where it will operate most frequently. Different countries have different base and overage rates for data plans
- **How many SIMs are in the product**: Particle offers volume-based pricing for product creators with large fleets of SIM Cards

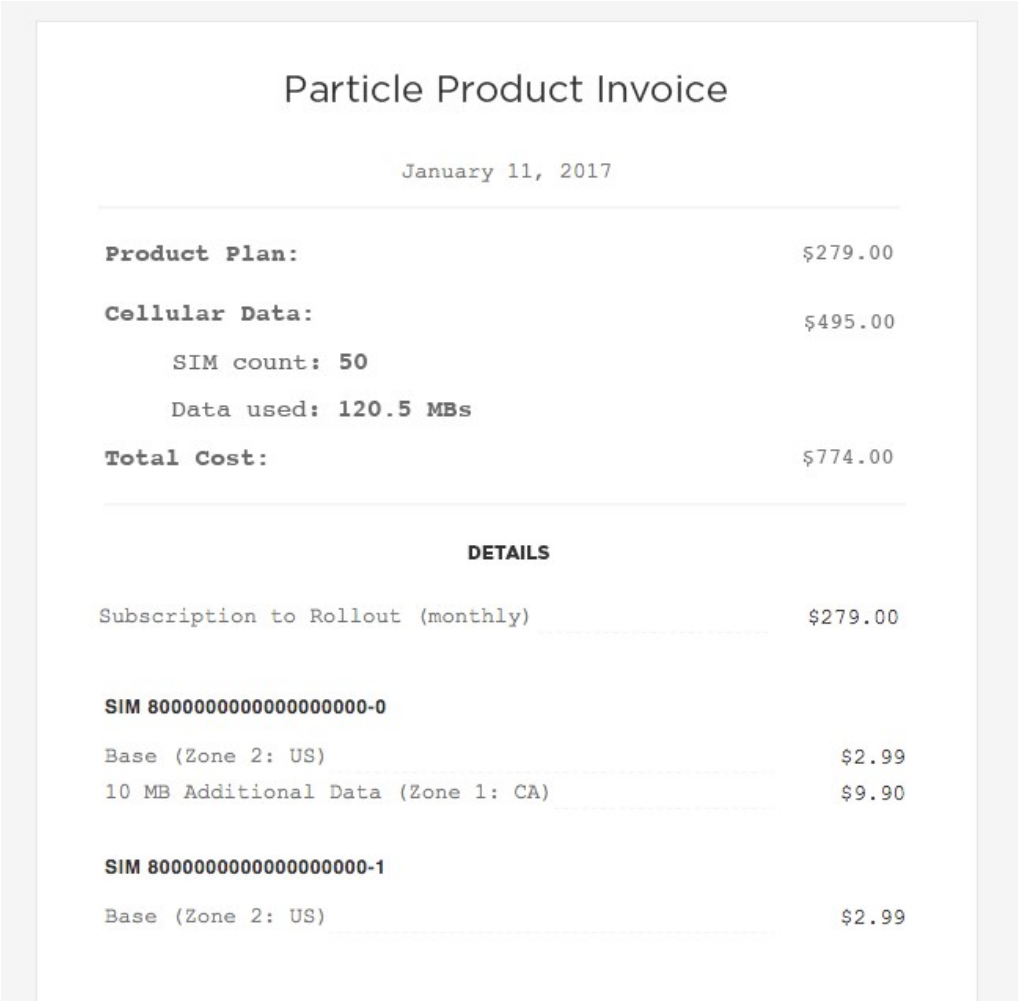
Note that product SIM cards are **not** eligible for promotional pricing given to developers (i.e. 3 free months of data). Data plan details will be communicated clearly during the import process as well as on monthly invoices.

Data pooling, or purchasing a chunk of cellular data shared by a fleet of SIM cards can be arranged by contacting us. This is often useful to help product creators more accurately estimate cloud infrastructure costs over time.

Invoices

Cellular data usage will be calculated and charged on a monthly basis, and included as part of your regular invoice for your product plan. If you are on a monthly plan, this means you will receive an invoice each month that contains line items for both your platform plan and SIM costs. For those who have chosen to be billed for their product plan annually, you will receive monthly invoices for cellular data and still maintain your annual payment schedule for your product plan.

Your invoice will look something like this:



Particle Product Invoice

January 11, 2017

Product Plan:	$279.00
Cellular Data:	$495.00
SIM count: **50**	
Data used: **120.5 MBs**	
Total Cost:	$774.00

DETAILS

Subscription to Rollout (monthly)	$279.00
SIM 8000000000000000000-0	
Base (Zone 2: US)	$2.99
10 MB Additional Data (Zone 1: CA)	$9.90
SIM 8000000000000000000-1	
Base (Zone 2: US)	$2.99

You will notice a few things about your invoice:

- It will contain a summary at the top, that includes a cost for your product plan and cellular data, which add up to your monthly total cost
- You will see the total number of billable SIMs in your fleet as well as the total amount of data consumed for the month
- Each SIM's individual cost will be itemized in the invoice details

YOU DID IT!

Thanks for reading. Now that you've finished the tutorial, you should be ready to build your own product on the Particle platform.

If anything described here is unclear, please let us know. We are eager for your feedback; you can post a comment on our forums or send us an email at hello@particle.io.

If you need any development services, please contact sales; we would be glad to pair you up with one of our professional services partners.

Finally, here's a short list of things to consider in the design and delivery of your product:

- What will the unboxing experience be? How will you tell your customer how to set up their product?
- What happens if your product never comes online? Is connectivity necessary for the experience? If not, how will you support your offline customers?
- How might your product fail? What troubleshooting information do you need to provide, and how will your customers contact you when they can't solve their own problems?
- How can you take advantage of your product's connectivity to provide better support and customer service?
- How will you use over-the-air firmware updates to improve your product once you've shipped it to customers? What cheap sensors and actuators could you add to your product that might be useful in the future?
- How might your distributors and other business partners benefit from the fact that your product is connected?

Good luck and godspeed!

DEVELOPMENT DEVICES

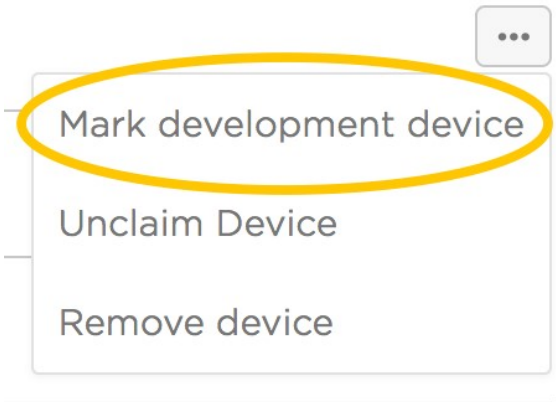
As a firmware engineer building a Particle product, it is important that you can rapidly iterate on new device firmwares, while still simulating the behaviors of production devices deployed in the field. Development devices allow you to do just this.

Development devices ( ) are special kinds of product devices marked specifically for internal testing, separate from the production fleet. Development devices are prevented from receiving any automatic product firmware updates from the Particle cloud. These devices will ignore both released product firmware as well as any firmware version it has been locked to run.

This allows you to freely flash new firmware to these devices from the development tool of your choice (i.e. the Web IDE, Local IDE, or the CLI), without having it immediately overwritten by the Particle cloud. These devices will continue to maintain product privileges, allowing you the flexibility to experiment with new firmwares while still simulating behaviors of deployed devices in the production fleet.

Marking a development device

You can mark a device as a development device from the Console. When viewing your product's Devices view ( ), mark a development device by clicking the ... icon and finding the "Mark development device" link as shown below:



Once you mark a device as a development device, you will see it marked with the icon in the "Firmware" column:

This illustrates that the device is no longer targeted for product firmware releases. You are now free to begin flashing the device from the Web IDE, or whatever your preferred tool is for sending firmware to Particle devices.

Quick tip: You can sort by firmware version on this page to quickly find all of your development devices. When sorting in descending order, development devices will appear at the top of your device list.

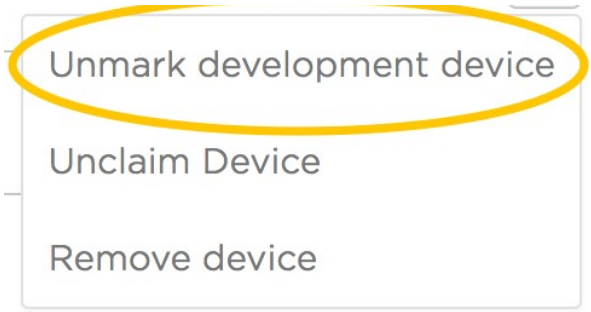
Quick tip: If you prefer, you can also call the Particle API directly to mark a device as a development device.

Unmarking a development device

If you would like to return a device to the pool of eligible product devices to receive automatic firmware updates, you can unmark it as a development device. Non-development devices will be targeted to receive released product firmware or any firmware version it has been locked to run.

This is useful if a device is no longer being used for active firmware development, and you would like it to behave just like any other device in the product fleet.

From the Console, you can find the "Unmark development device" link by clicking the ... icon from the Devices view:



You should see the development device icon disappear from the "Firmware" coulumn for this device once it has been successfully unmarked as a development device. Note that **unmarking a device as a development device may trigger an immediate over-the-air update**. This happens to return the device to its targeted version of product firmware.

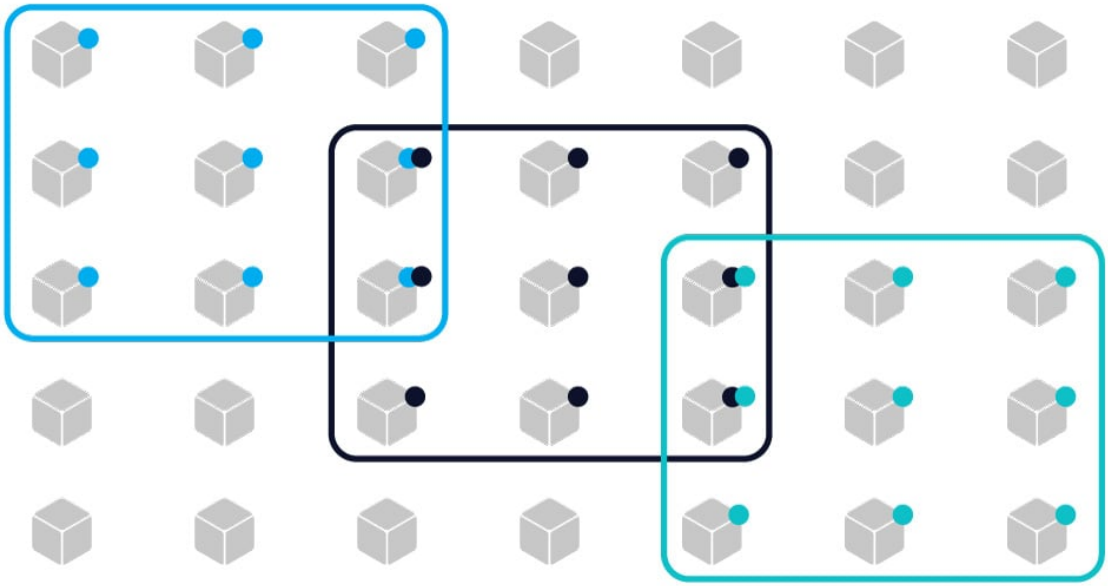
Quick tip: If you prefer, you can also call the Particle API directly to unmark a device as a development device.

DEVICE GROUPS

Beta feature

*Device groups are **currently in beta** and available to customers on an Enterprise plan. If you would like access to this feature, please contact us.*

Device groups allow you to define subsets of your IoT deployment that can be addressed separately from the rest of the fleet. Groups provide the flexibility and granularity you need to manage a connected product at scale.



Assign devices to groups for more granular control of your IoT deployment

Segmenting devices into groups allows you to manage an IoT fleet with more specificity, like the ability to rollout firmware by group. As an example, a business serving customers around the world can now target devices to run a different version of firmware depending on the local country's native language.

Stay tuned as we add more ways to leverage device groups in your product.

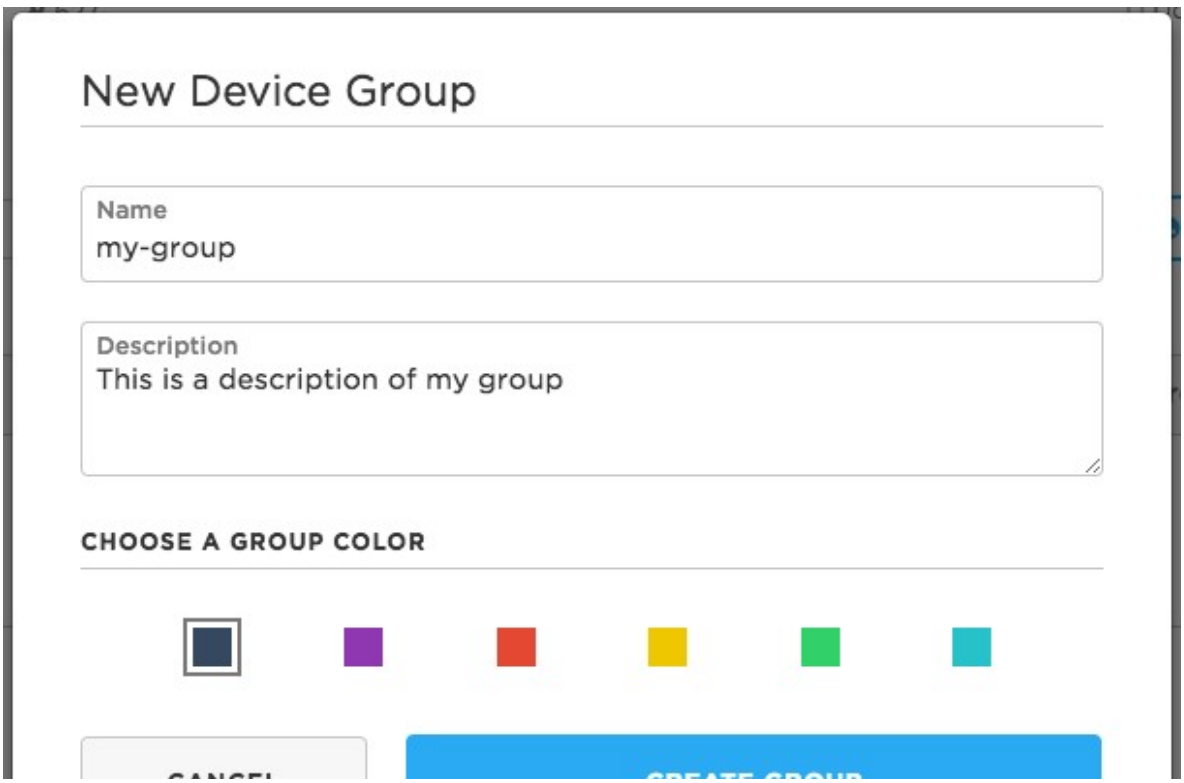
Creating Device Groups

In order to get the benefits of device segmentation, you will first need to create a group.

A group is scoped to a Particle product, and can be applied to any number of devices in the fleet. A group can be created using the Particle Console. Visit your product's devices view (📦) and click the "New Group" button:

You will find the 'New Group' button on your product's devices page. Note that this button will only be enabled if you are on an Enterprise plan. Contact us for access to this feature.

After clicking this button, the *New Device Group* modal will appear:



Give your group a unique name that allows you to quickly identify its purpose. You can optionally provide a description and a thumbnail color. Some examples of groups you may wish to create:

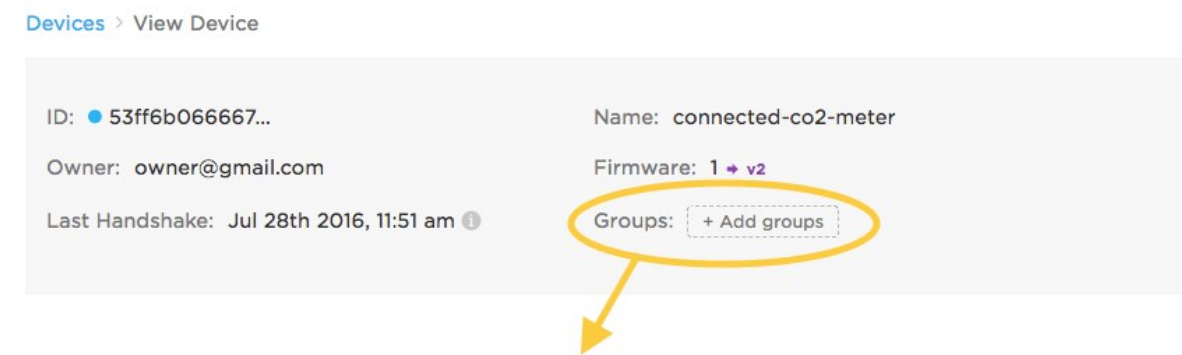
- **Prerelease Group**: Target this subset of devices to receive "bleeding-edge" firmwares for early testing. This can help you get user feedback and/or identify potential firmware bugs before releasing to the rest of the fleet.
- **Group by Geographic Region**: Devices that are deployed in various parts of the world may need to be addressed differently. Segment device behavior, or run different versions of firmware.
- **Group by Permissions**: You may have distributors, service partners, or customers that need access to certain device groups.

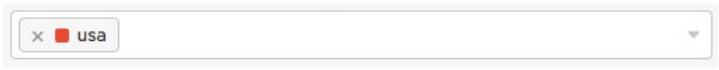
Quick tip: If you prefer, you can also call the Particle API directly to create a device group.

Assigning Groups to Devices

Now that you have created your group(s), you'll need to assign groups to devices in your fleet. Note that **a device can belong to multiple groups**, as you will likely need to segment your fleet along many different axes.

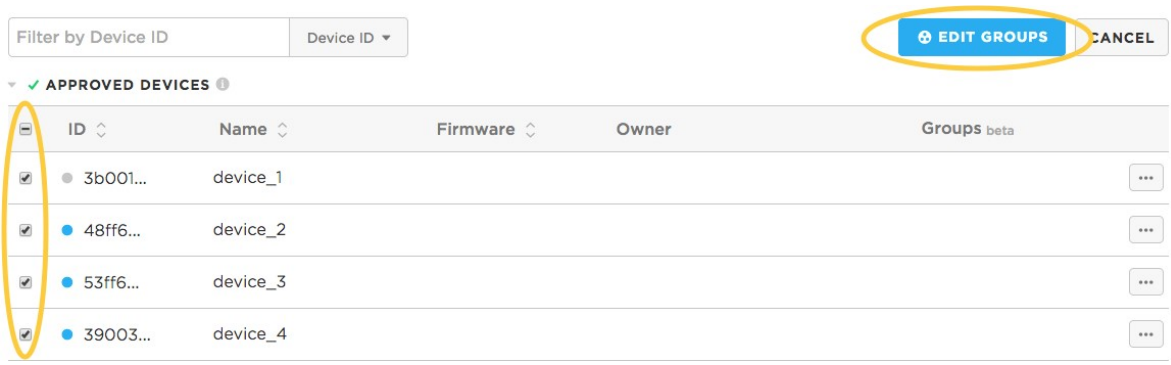
To assign a single device to a group, click on it from your product's devices view (📦). Then, click the **+ Add groups** button as shown in the device header:





The page will enter into an edit mode. Apply groups to the device using the dropdown, and click the **Done** button to save. You can assign devices to existing groups, or create a new group and assign it to the device. You can come back here and click the **Edit** button on a device to update its group memberships at any time.

You can also assign many devices to a group at the same time. If you have grouping enabled for your product, you should see checkboxes next to each approved device visible on your product's device view. Select some devices, and click the **Edit groups** button that appears:



Edit groups of many devices at once using the Console

You will then be presented with a modal that will allow you to batch update groups of the selected devices. **Note that editing groups for a device may result in the device being targeted for an OTA firmware update**, due to product firmware releases by group.

Quick tip: If you prefer, you can also call the Particle API directly to assign groups to a device.

Quick tip: After assigning groups to devices, you can use the Console or the Particle API to filter product devices by group.

Firmware Release by Group

With devices segmented into groups, it is now possible to gain more control and flexibility over the firmware release process. If you are unfamiliar with how to rollout firmware to a product fleet, please check out the Particle Console Guide before continuing.

With grouping enabled, you can *release a product firmware to one or more groups*, or choose to mark the version as the *Product default* release. Releasing firmware to a group will target a only subset of the product fleet to receive the binary (those devices that belong to that group).

Marking as the Product default, in contrast, sets the firmware as the default firmware version available to all devices in the fleet to download and run. The specific firmware chosen to be delivered to a given device is determined by *precedence rules*, which you can read about in the next section.

Firmware Precedence Rules

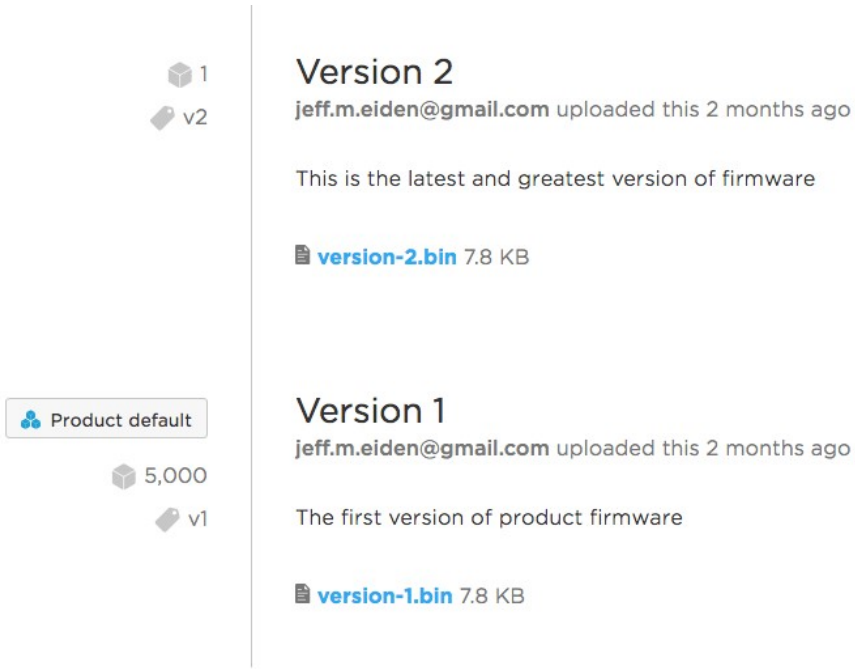
Devices in your fleet will be targeted to receive a version of product firmware according to these precedence rules:

- A **development device** never receives automatic updates of product firmware.
- A device that has been **individually locked** to a version of product firmware is respected above all else, and will not be overwritten by any released firmwares.
- If unlocked, devices **belonging to a group** will receive the corresponding group's released firmware (if a firmware has been released to the group). When a device belongs to multiple groups that each have released firmware, the *highest firmware version* will be preferred
- If a device is unlocked and **does not belong to any groups** with released firmware, it will receive the **Product default** released firmware (if a firmware has been released as the Product default)
- If none of the above conditions result in a device being targeted for a product firmware, it will not receive an automatic OTA update from the Particle cloud

Example Release Process

Let's walk through a real-life example of how you could use releasing product firmware by group. Imagine you have a fleet of 5,000 internet-connected widgets in the field. They are all currently running version 1 of your product firmware (v1 had been previously marked as the default product firmware).

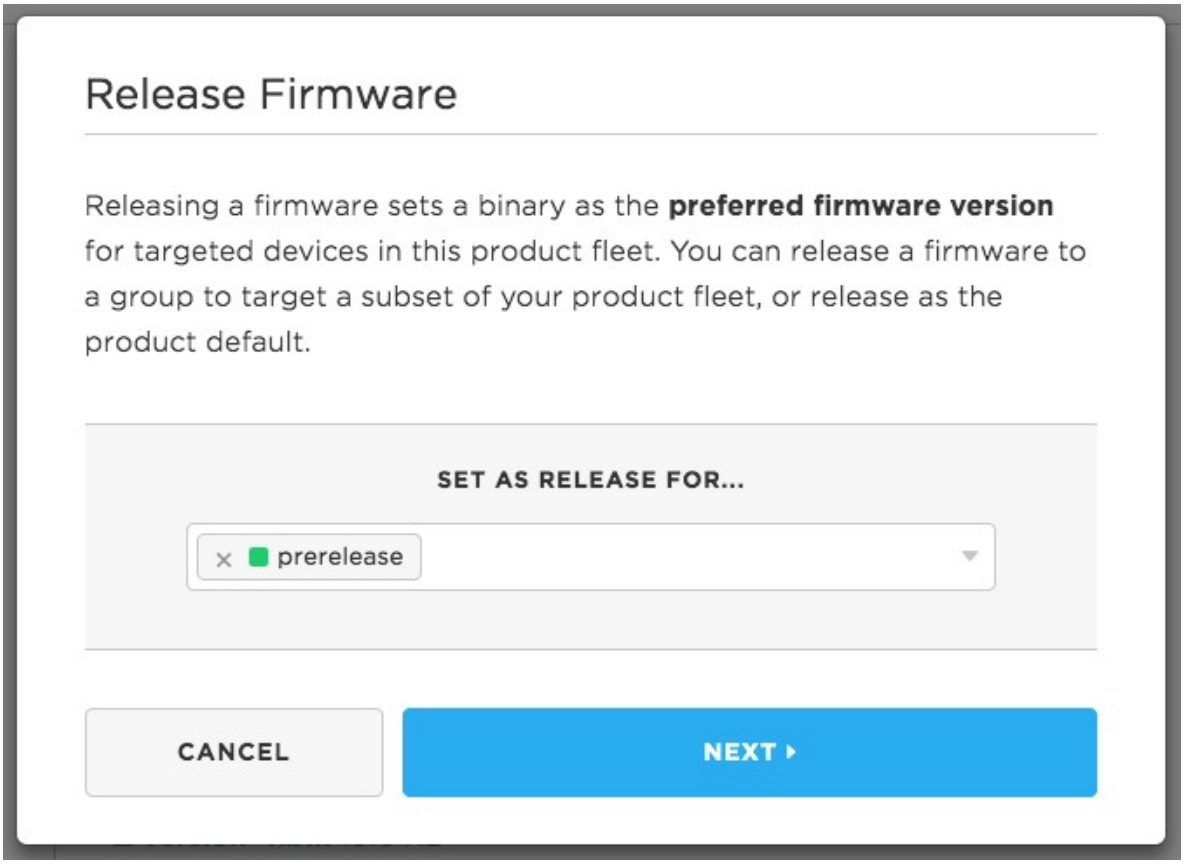
After some initial feedback from end-users, you and your engineering team add a few impressive new features. You upload this new version of firmware to your product:



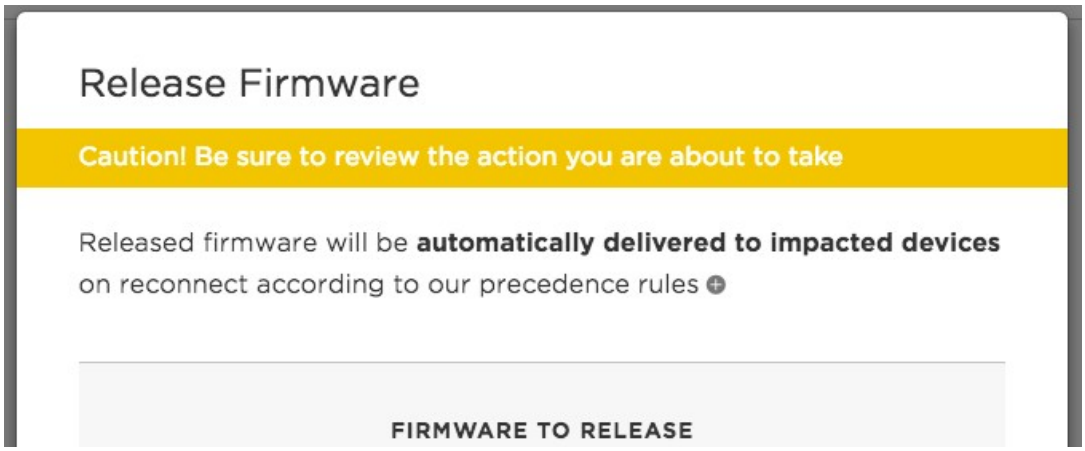
You've tested out the firmware on one of your internal development devices, and are ready to roll it out to devices in the field. However, you still aren't completely certain how this new code will perform in the wild and would like to mitigate risk as much as possible.

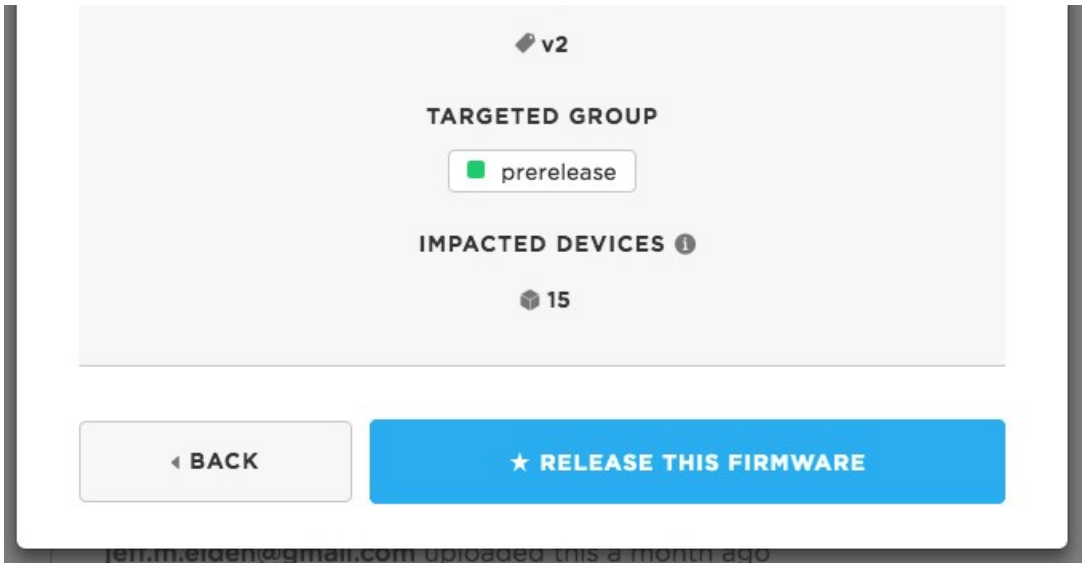
Luckily, you've defined a `prerelease` group. This group contains 15 devices belonging to customers that have opted-in to receiving bleeding-edge firmware. Let's go ahead and release version 2 to the `prerelease` group to target those eager customers.

To do this, click on ★ **Release this firmware** when hovering over v2. The *Release Firmware* modal will appear. Use the dropdown to select `prerelease` from the available groups:



Click **Next**. You will now be asked to confirm the action you are about to take. Releasing firmware to devices deployed in the field always comes with some risk, so use the confirmation screen to ensure you're releasing the right firmware to the right devices:

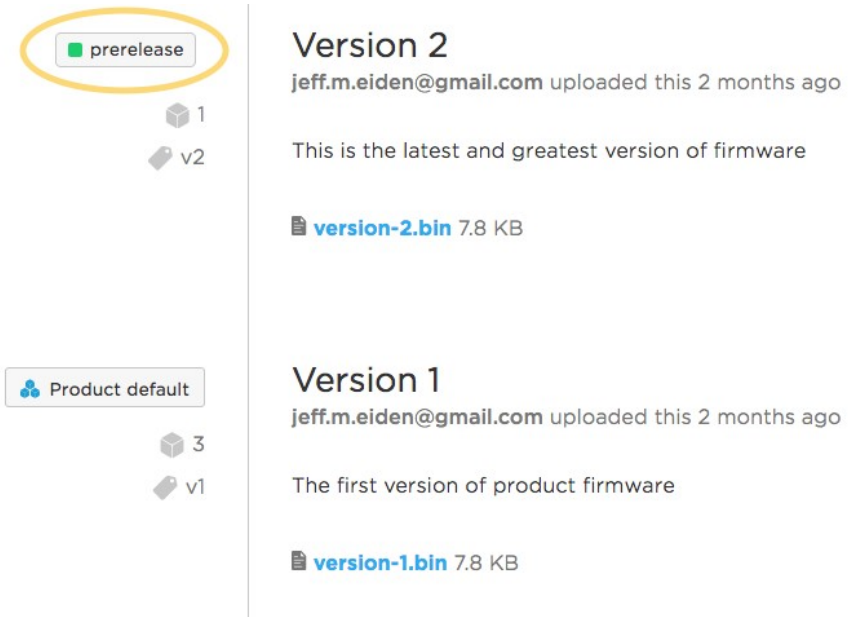




Always confirm the version, targeted group(s) and impacted devices before releasing firmware to your device fleet to ensure you are taking the desired action

Looks good! We see that we are releasing version 2 of firmware to the `prerelease` group, which should impact the 15 devices that belong to this group. Note that *impacted devices* refers specifically to the number of devices that will receive an OTA firmware update as a direct result of this action.

Click ★ **Release this firmware** to confirm the action. Nice! We've now released v2 to the `prerelease` group. You should now see the group tag appear next to the firmware version:



Note that the devices will not receive the firmware immediately; instead, they will be targeted for an over-the-air update the next time they start a new secure session with the cloud (this is called a *handshake*).