Presto

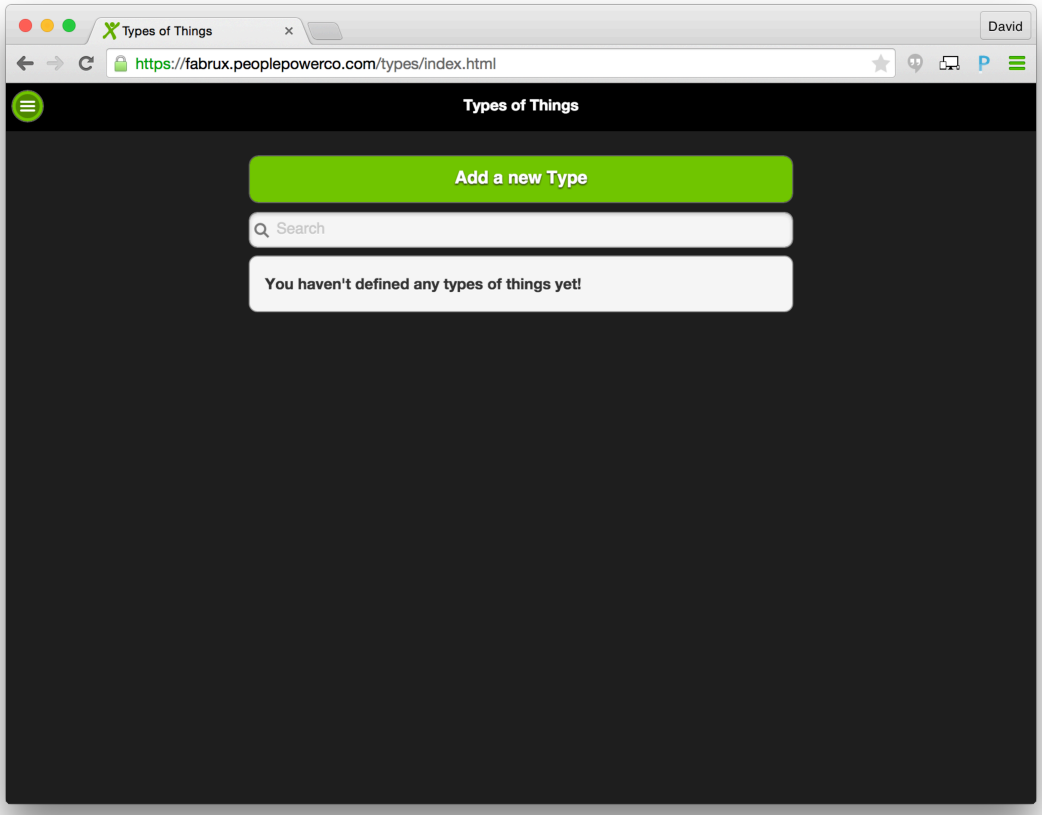
End-to-End Beginner's Example

We're going to show you, step-by-step and very quickly, how to get a new device completely connected to Presto, start to finish. We'll use a hypothetical 1-socket WiFi smart plug as an example, which may connect directly to the server.

Our smart plug will measure power in watts and energy in kilowatt-hours. It also has a switch to turn something on and off. Let's get it connected and run some API's.

1. Create a new Type of Thing

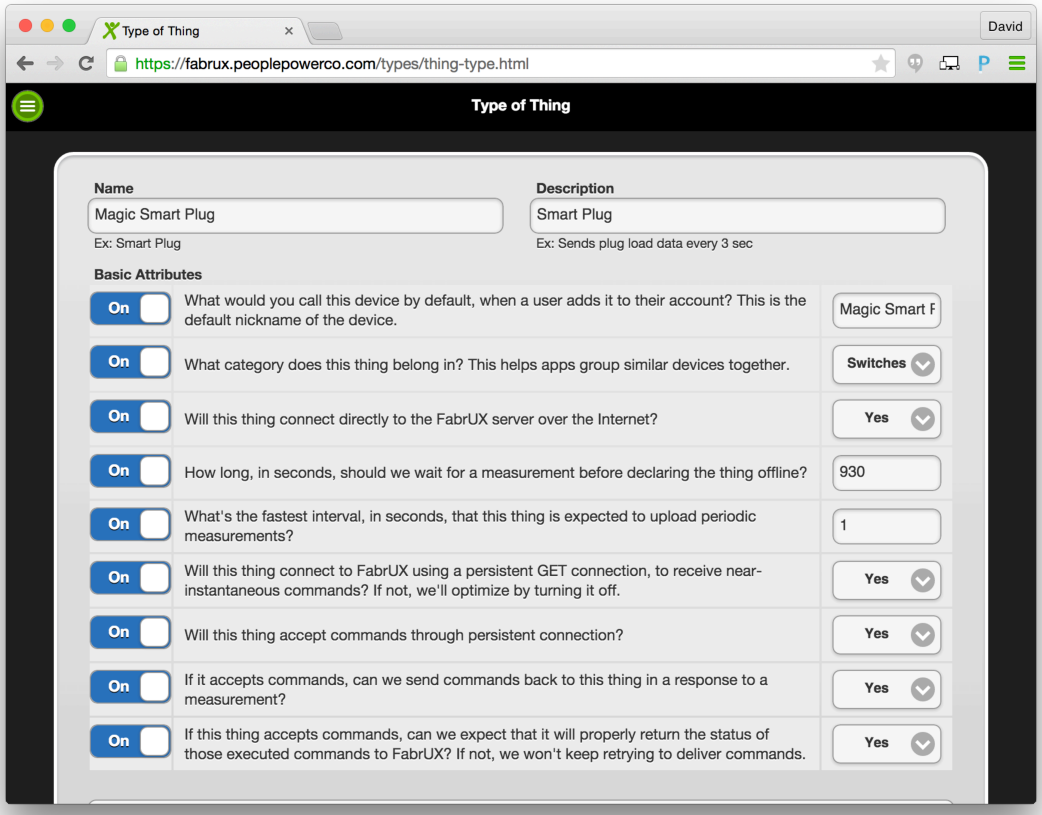
After signing in, tap into "Types of Things" and "Add a new Type".



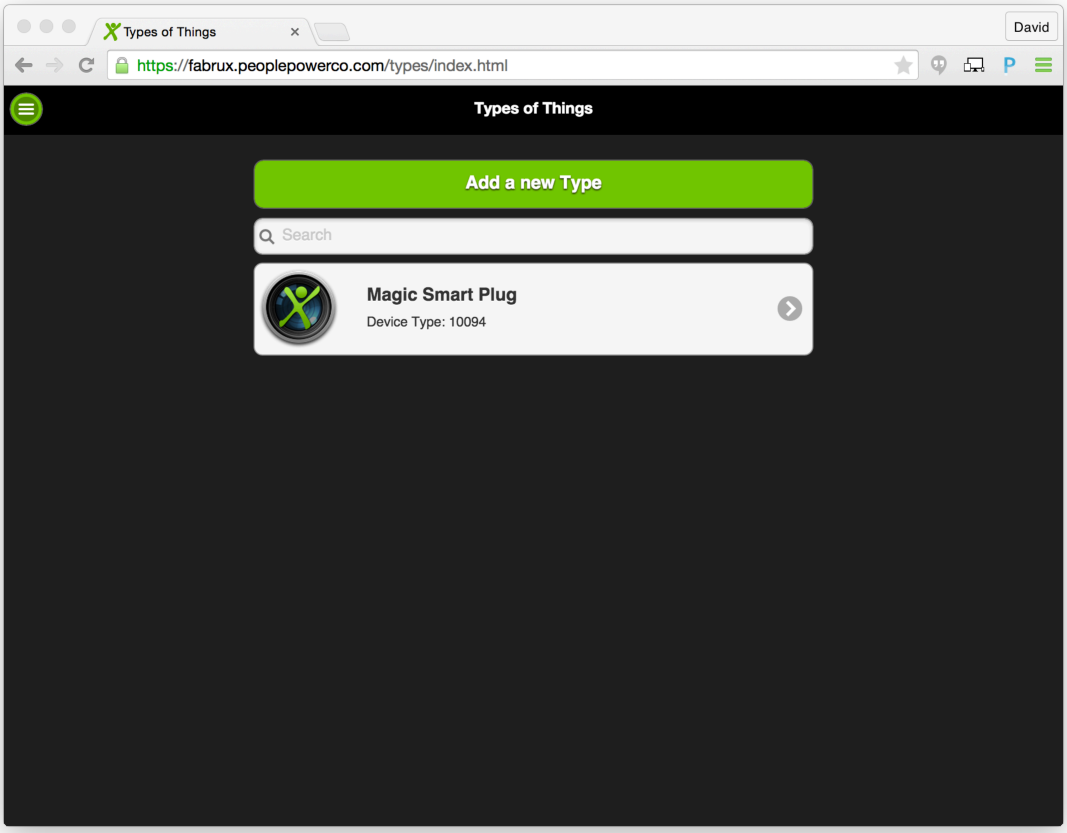
We want to create a new smart plug. Here is it's behavior:

- It's a switch, with a unique name "Magic Smart Plug".
- It will connect directly to Presto through an Internet connection.
- If Presto hasn't seen a measurement in over 15 minutes, the plug is probably offline.
- The fastest this smart plug will ever send measurements to Presto is once every second.
- The smart plug will use a persistent GET connection with Presto to receive commands instantaneously.
- The smart plug will accept commands through the HTTP POST as well.
- When the smart plug receives a command, it will properly acknowledge it got the command from the server.

We plug this into the Presto developer console. Give your plug a unique Name.



When you're done, hit Save at the bottom.



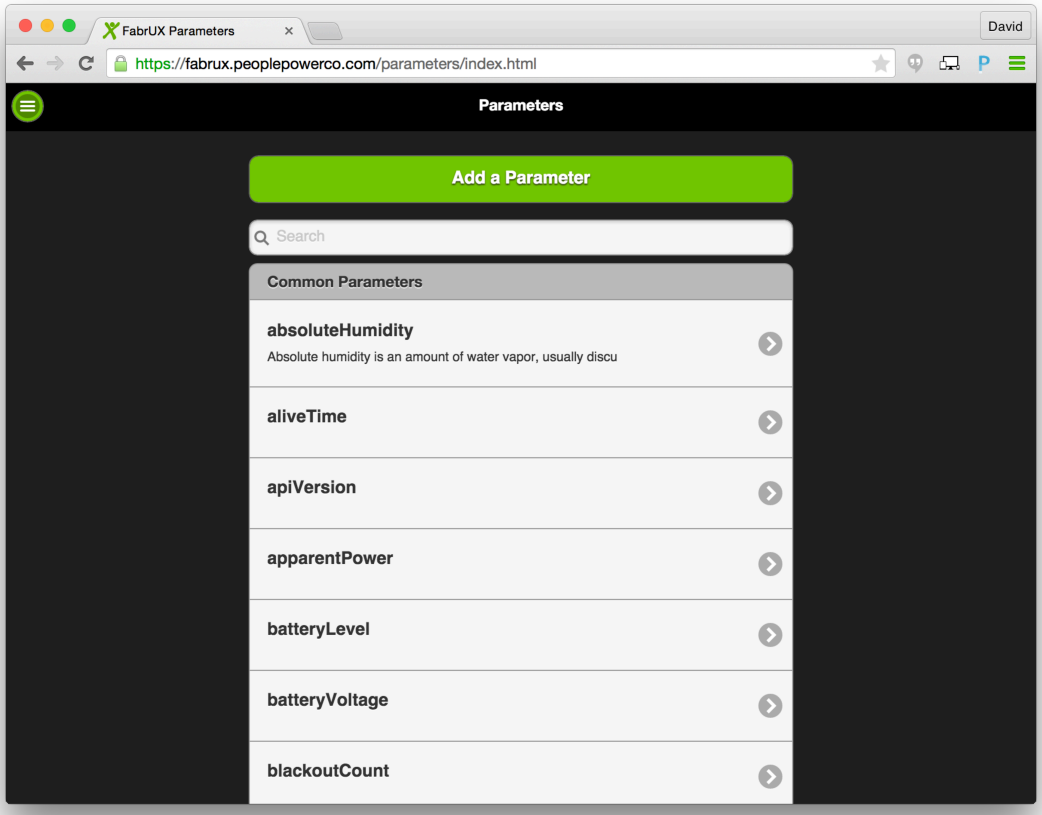
Make a note for later that Presto generated device type 10094 for this new smart plug. This device type is like a class, and a device instance is like an object.

2. Define Parameters

Parameters are the key-value messages we pass back and forth between Presto and a device.

Our hypothetical smart plug can measure power and energy, and turn on and off. So we need 3 parameters (power, energy, status). We'll choose "ppc" for the organization's initials on the front of our parameter names.

Tap into the Parameters section of the site.

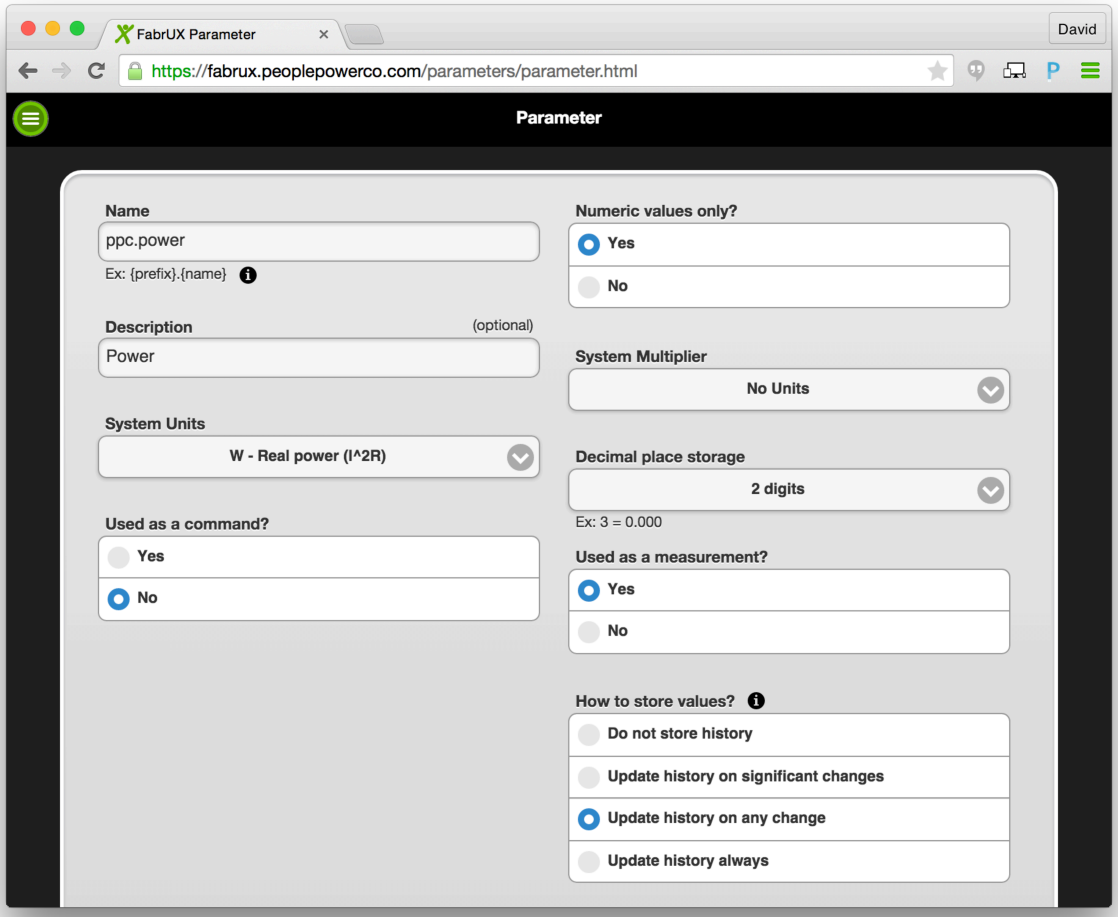


Then tap "Add a Parameter". Create 3 parameters.

"ppc.power" and "ppc.energy" are measurements only and not commands.

> Power will be in units of "watts", so it will have no units multiplier. Power will store up to 2 digits after the decimal place.
>
> Energy will be in units of kilowatt-hours, so it needs a multiplier of "kilo" and units of "watt-hours". Energy will also store up to 5 digits after the decimal place.

"ppc.outletStatus" will be a command to the device, and separately a measurement from the device. The command and measurement channels are completely separate, but use the same parameter name. This parameter represents the on/off state of the smart plug and does not need to store any history, if we're not doing any more complex analytics on it.

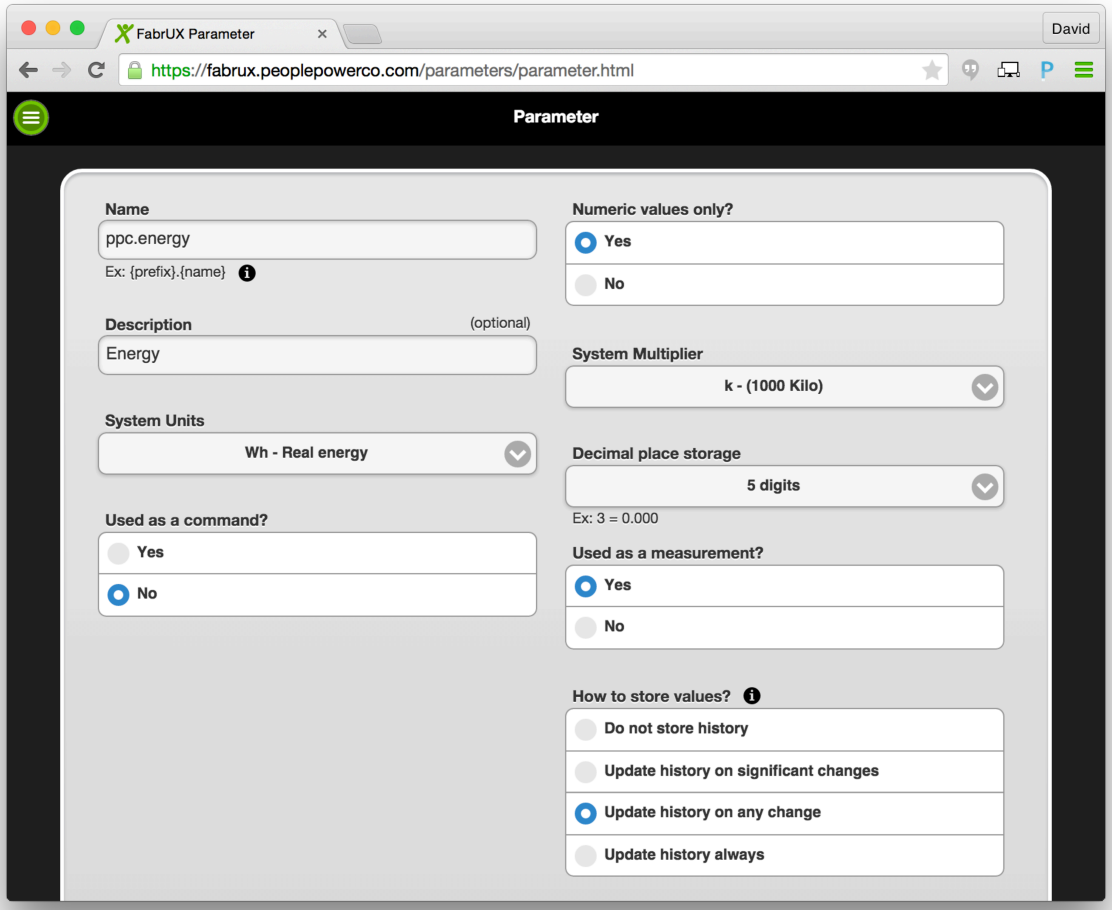


Presto by People Power

End-to-End Beginner's Example

Question? We're here. support@peoplepowerco.com

2015.04.20

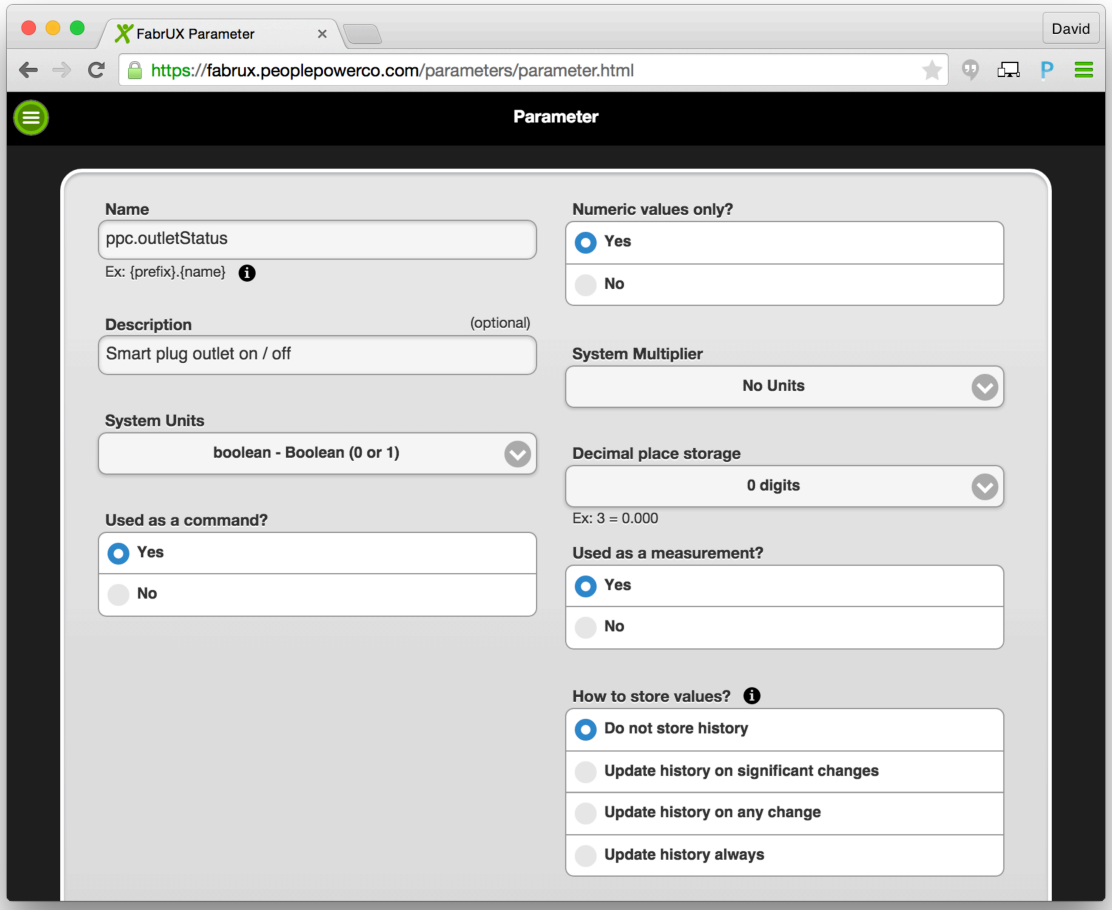


Presto by People Power

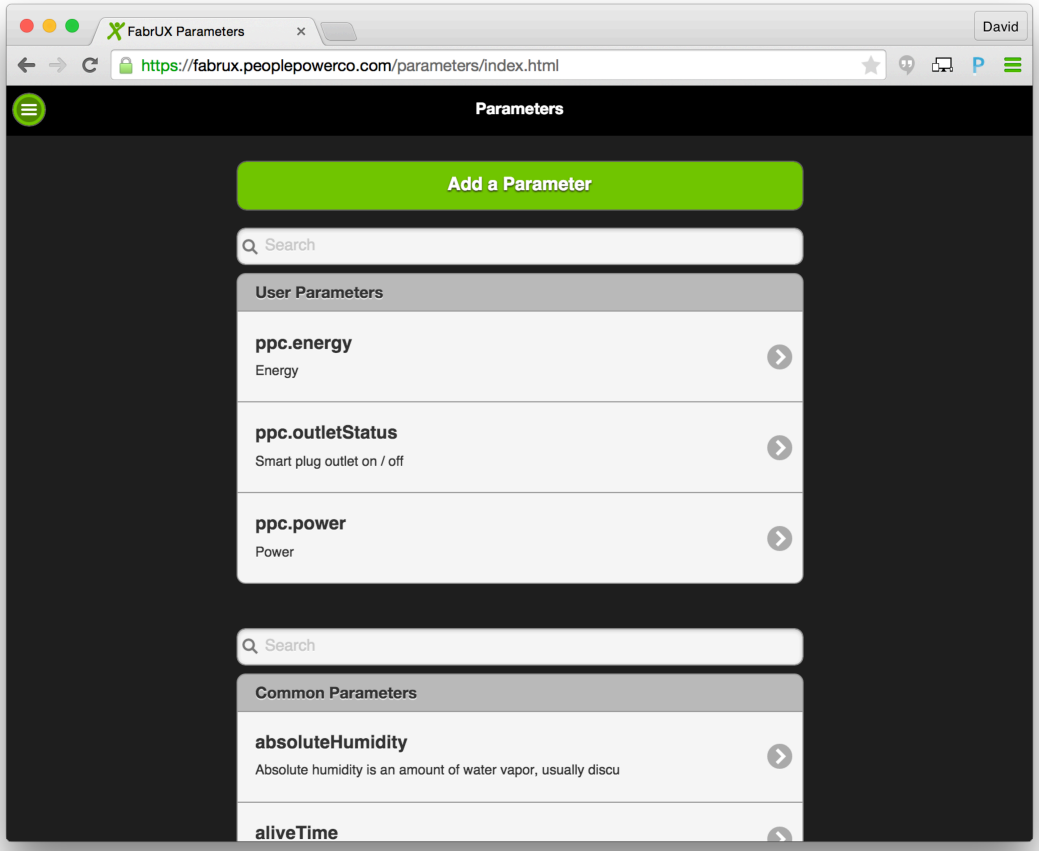
End-to-End Beginner's Example

Question? We're here. support@peoplepowerco.com

2015.04.20



Now we're ready. Let's take a look at all the parameters we created…



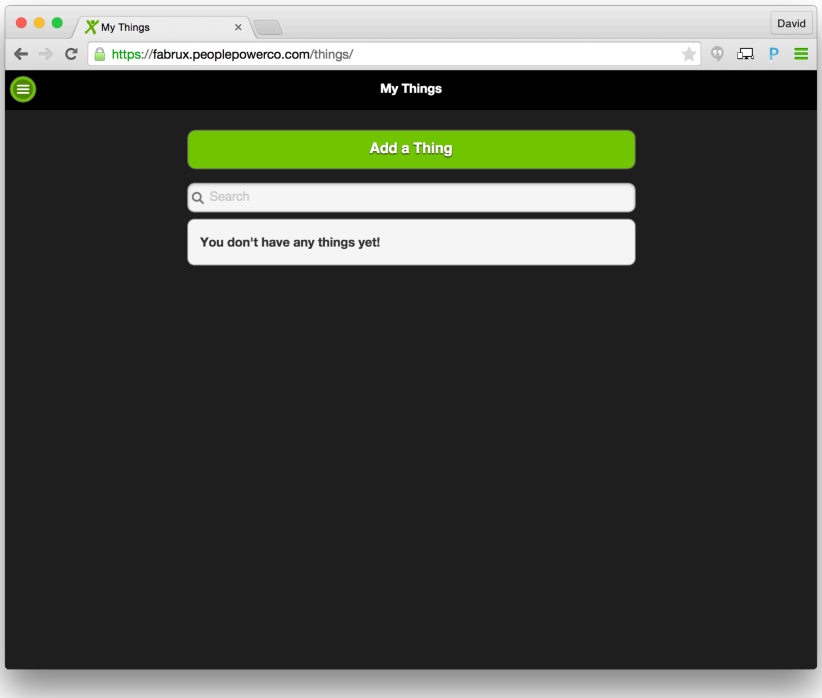
3. Register a device to your account

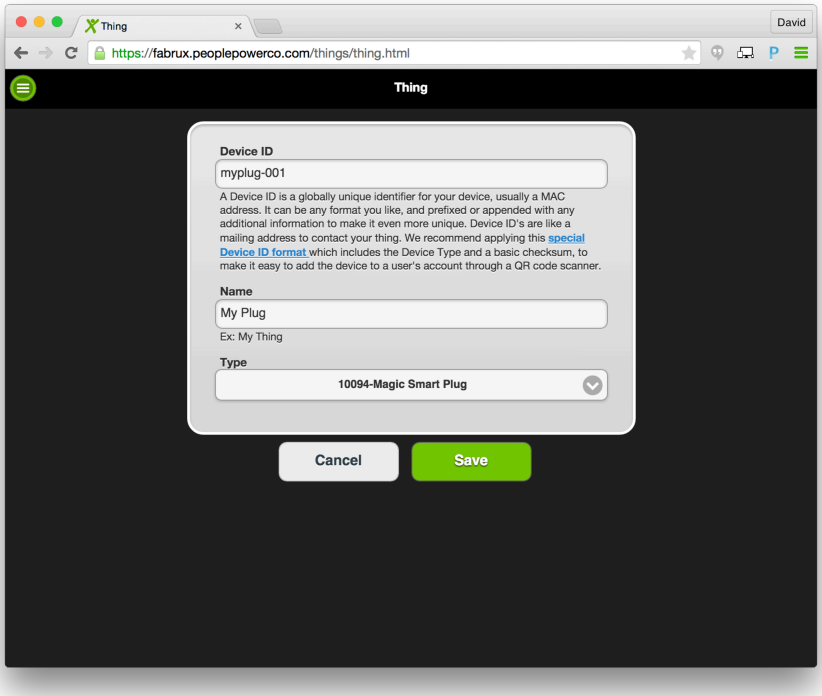
Tap into My Things and Add a Thing.

You can come up with any device ID you want. Check out the "QR-code ready Device IDs" document for some suggestions on device ID's that can be recognized by a mobile app through a QR-code scan.

The device ID you register, will be the same ID that the device itself uses to report to Presto. MAC addresses or UDID's are usually used for the device ID. The device ID is a globally unique address that allows Presto to recognize the device.

For this example, we'll make up a device ID myplug-001. Remember, device ID's are case-sensitive! And they can't contain the character '/'.





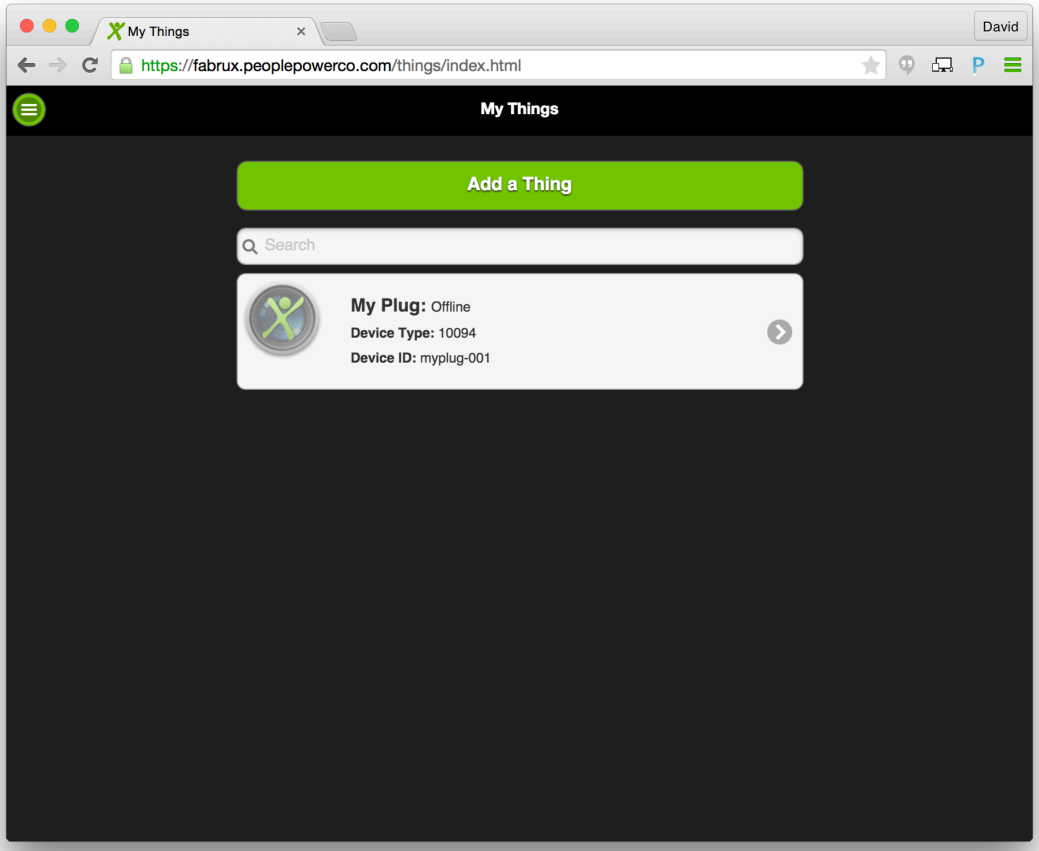
Presto by People Power

End-to-End Beginner's Example

Question? We're here. support@peoplepowerco.com

2015.04.20

After Saving your new device instance, it will appear in your account. Hit refresh if you don't see it.



Now let's make it talk.

4. Send measurements from the device.

We can send measurements using curl in a terminal window.

The device that owns the Internet connection, owns the proxy ID. The proxy ID should be the same as the device ID, if the measurement is coming from the same device that owns the Internet connection. That's the case here. It would also be true of a smart plug that's connected over Wi-Fi to the Internet. It wouldn't be true of a smart plug that routes its packets through a gateway, like a ZigBee router.

The two server end-points we have to choose from on our Presto developer server are:

https://esp.peoplepowerco.com:8443/deviceio/mljson for JSON
https://esp.peoplepowerco.com:8443/deviceio/mlxml for XML.

If you don't want HTTPS, then use port 8080 (great for getting started, and debugging):

http://esp.peoplepowerco.com:8080/deviceio/mlxml
http://esp.peoplepowerco.com:8080/deviceio/mljson

XML

```
curl --request POST \
-H "Content-Type: application/xml" \
--data '<?xml version="1.0" encoding="utf-8" ?>
  <h2s ver="2" proxyId="myplug-001" seq="1">
    <measure deviceId="myplug-001">
      <param name="ppc.power">0</param>
      <param name="ppc.energy">0</param>
      <param name="ppc.outletStatus">1</param>
    </measure>
  </h2s>
  ' \
--verbose \
https://esp.peoplepowerco.com:8443/deviceio/mlxml
```

JSON

```
curl --request POST \
-H "Content-Type: application/json" \
--data '{
   "version": "2",
   "proxyId": "myplug-001",
   "sequenceNumber": "1",
   "measures": [
      {
         "deviceId": "myplug-001",
         "params": [
            {
               "name": "ppc.power",
               "value": "0"
            },
            {
               "name": "ppc.energy",
               "value": "0"
            },
            {
               "name": "ppc.outletStatus",
               "value": "1"
            }
         ]
      }
   ]
}
 '\
--verbose \
https://esp.peoplepowerco.com:8443/deviceio/mljson
```

Here are the results in our Terminal window:

XML EXECUTION

```
curl --request POST \
> -H "Content-Type: application/xml" \
> --data '<?xml version="1.0" encoding="utf-8" ?>
>   <h2s ver="2" proxyId="myplug-001" seq="1">
>     <measure deviceId="myplug-001">
>       <param name="ppc.power">0</param>
```

```
>       <param name="ppc.energy">0</param>
>       <param name="ppc.outletStatus">1</param>
>     </measure>
>   </h2s>
>   ' \
> --verbose \
> http://esp.peoplepowerco.com:8080/deviceio/mlxml
* About to connect() to esp.peoplepowerco.com port 8080 (#0)
*   Trying 184.73.155.162... connected
* Connected to esp.peoplepowerco.com (184.73.155.162) port 8080 (#0)
> POST /deviceio/mlxml HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7
NSS/3.14.0.0 zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: esp.peoplepowerco.com:8080
> Accept: */*
> Content-Type: application/xml
> Content-Length: 275
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Credentials: false
< Access-Control-Allow-Methods: GET,POST,OPTIONS
< Access-Control-Allow-Headers: Content-Type,Content-Length,X-
Requested-With,accept,Origin,User-Agent,Referer,Access-Control-Request-
Method,Access-Control-Request-Headers,PPCAuthorization,x-bl-deviceid,x-
bl-access-code
< Access-Control-Max-Age: 10000
< Cache-Control: no-cache, no-store
< Content-Type: text/xml;charset=utf-8
< Content-Length: 74
< Date: Fri, 27 Mar 2015 04:01:31 GMT
<
<?xml version="1.0" encoding="UTF-8"?>
* Connection #0 to host esp.peoplepowerco.com left intact
* Closing connection #0
<s2h seq="1" status="ACK" ver="2"/>
```

JSON EXECUTION

```
curl --request POST \
> -H "Content-Type: application/json" \
> --data ' {
>     "version": "2",
>     "proxyId": "myplug-001",
>     "sequenceNumber": "1",
>     "measures": [
>         {
>             "deviceId": "myplug-001",
>             "params": [
>                 {
>                     "name": "ppc.power",
>                     "value": "0"
```

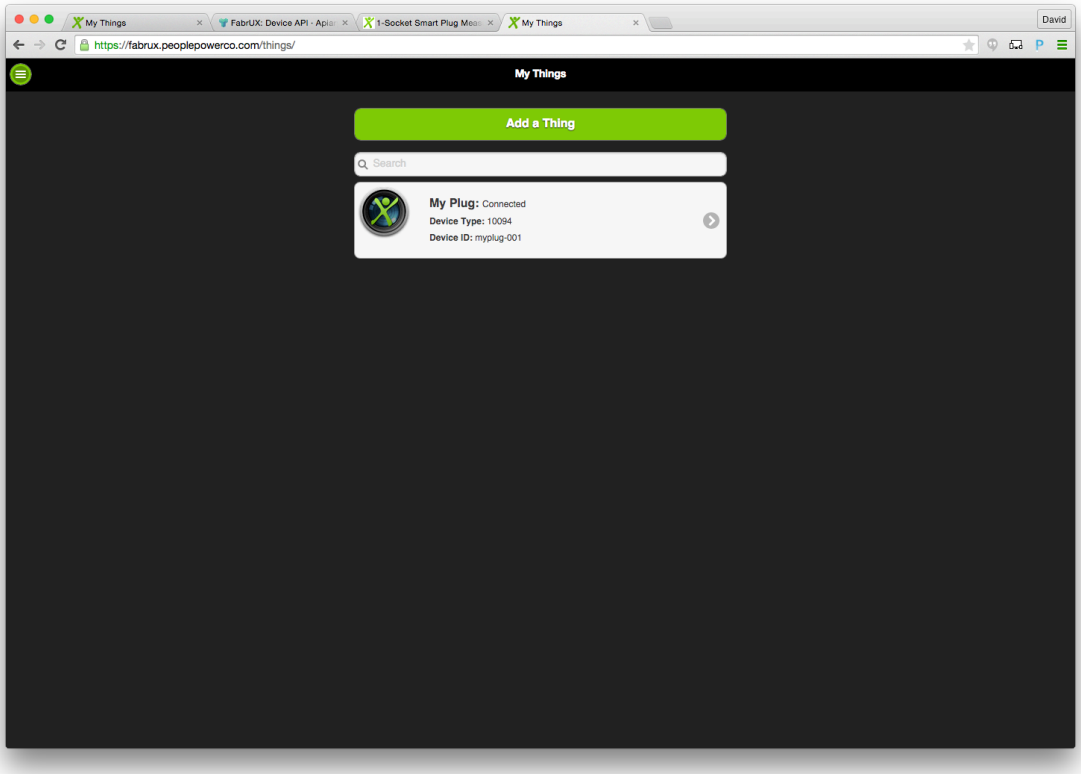
```
>                 },
>                 {
>                     "name": "ppc.energy",
>                     "value": "0"
>                 },
>                 {
>                     "name": "ppc.outletStatus",
>                     "value": "1"
>                 }
>             ]
>         }
>     ]
> }
>   ' \
> --verbose \
> https://esp.peoplepowerco.com:8443/deviceio/mljson

*   Trying 50.16.244.212...
* Connected to esp.peoplepowerco.com (50.16.244.212) port 8443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: *.peoplepowerco.com
* Server certificate: Go Daddy Secure Certificate Authority - G2
* Server certificate: Go Daddy Root Certificate Authority - G2
> POST /deviceio/mljson HTTP/1.1
> Host: esp.peoplepowerco.com:8443
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 555
>
* upload completely sent off: 555 out of 555 bytes
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Credentials: false
< Access-Control-Allow-Methods: GET,POST,PUT,OPTIONS
< Access-Control-Allow-Headers: Content-Type,Content-Length,X-
Requested-With,accept,Origin,User-Agent,Referer,Access-Control-Request-
Method,Access-Control-Request-Headers,PPCAuthorization,x-bl-deviceid,x-
bl-access-code
< Access-Control-Max-Age: 10000
< Cache-Control: no-cache, no-store
< Content-Type: application/json;charset=utf-8
< Content-Length: 40
< Date: Wed, 20 Apr 2016 15:07:39 GMT
<
* Connection #0 to host esp.peoplepowerco.com left intact
{"version":"2","seq":"1","status":"ACK"}
```

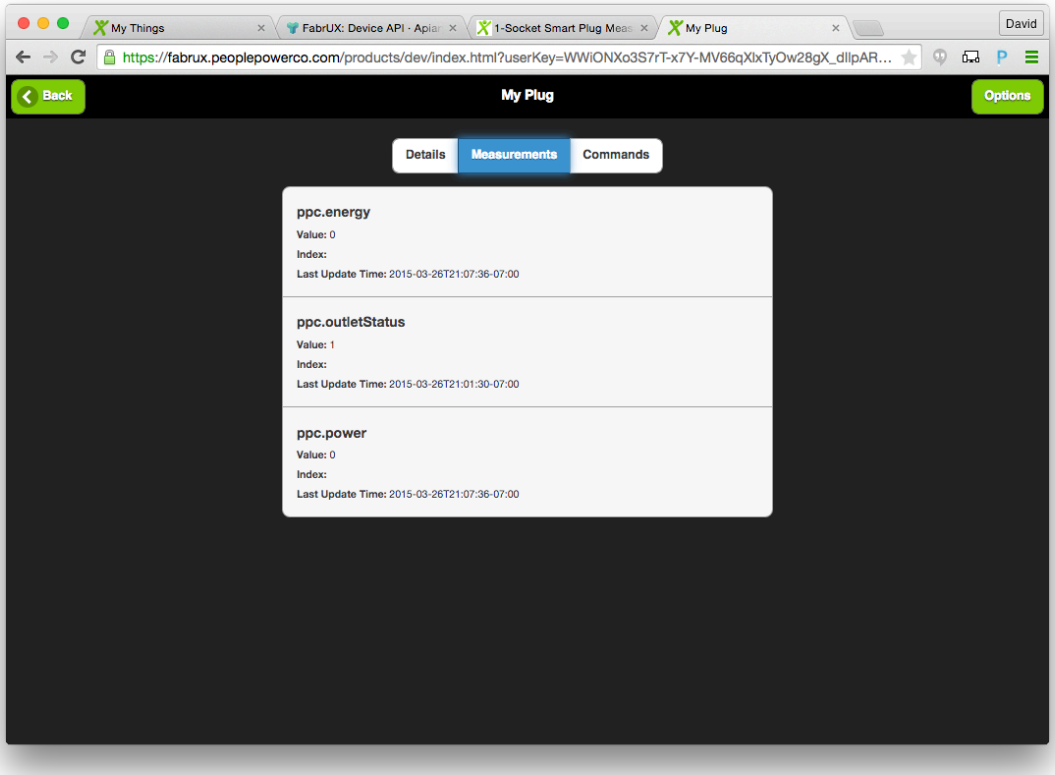
The "ACK" means the server got your message.

Let's refresh the "My Things" UI in the Presto developer console.

We see our device is online!



We can tap into the Measurements section of the generic developer device UI to see our measurements reached the server ok.



Use the Ensemble Application API at http://docs.iotapps.apiary.io to extract measurements from the server directly.

5. Send a command over a persistent connection

A persistent connection works by opening an HTTP GET with the server, and leaving it open for a long time (~2 minutes). Every time the HTTP GET connection closes, it can immediately reconnected by the device.

As soon as Presto needs to send the device a command, it has an open GET connection waiting, over which it can reply.

The HTTP GET and HTTP POST should always be run exclusively (never at the same time). This allows the Presto server to scale efficiently by only requiring half the number of ports at the server to support the device.

Do not allow your device to perform a POST and GET simultaneously.

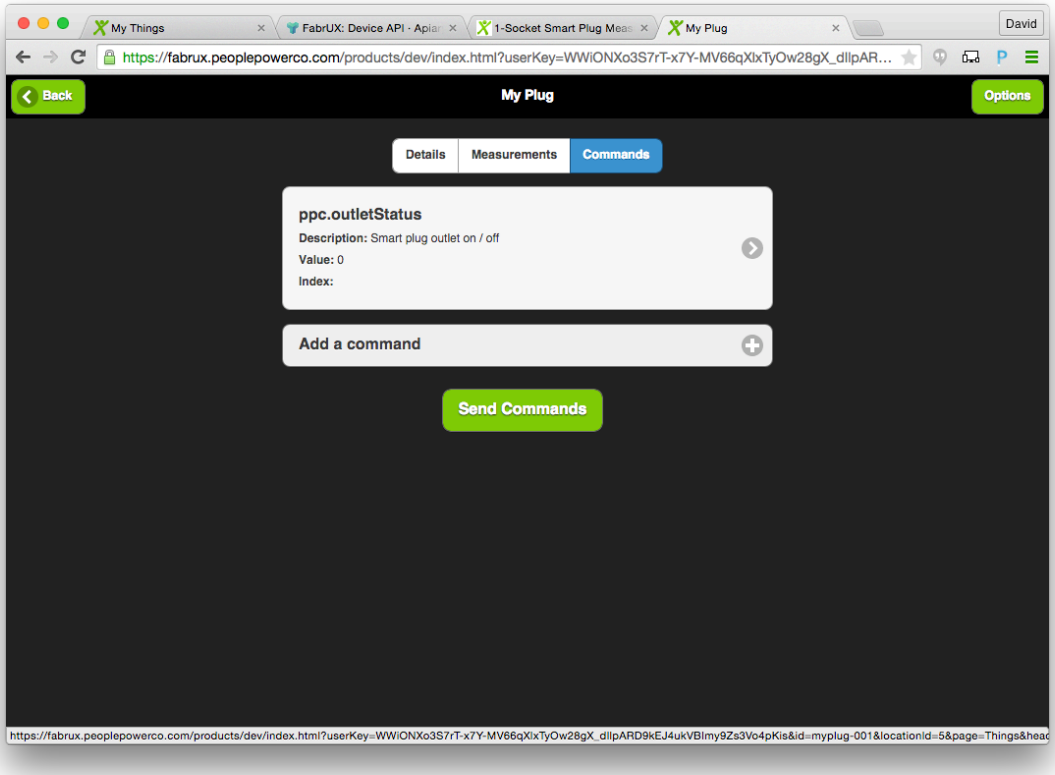
Start the command in your terminal window. Pass in your device ID and an HTTP GET timeout in seconds in the URL parameters:

```
curl --include \
    --header "Content-Type: application/json" \
  'https://esp.peoplepowerco.com:8443/deviceio/mlxml?id=myplug-001&timeout=120'
```

It will sit there silently.

Now send a command through your Presto developer console, or through the Presence mobile app in developer mode, or even through the Application API.

Tap into Commands, and then Add a Command. Select "ppc.outletStatus" and set its value to 0.



When you're ready, send the command. You should instantly see your terminal window exit curl with a response from the server.

XML EXECUTION

```
curl --include \
>      --header "Content-Type: application/json" \
>   'https://esp.peoplepowerco.com:8443/deviceio/mlxml?id=myplug-
001&timeout=120'
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: false
Access-Control-Allow-Methods: GET,POST,PUT,OPTIONS
Access-Control-Allow-Headers: Content-Type,Content-Length,X-Requested-
With,accept,Origin,User-Agent,Referer,Access-Control-Request-
Method,Access-Control-Request-Headers,PPCAuthorization,x-bl-deviceid,x-
bl-access-code
Access-Control-Max-Age: 10000
Cache-Control: no-cache, no-store
```

```
Content-Type: text/xml;charset=utf-8
Content-Length: 177
Date: Wed, 20 Apr 2016 15:55:08 GMT
```

```
<?xml version="1.0" encoding="UTF-8"?>
<s2h status="ACK" ver="2"><command cmdId="138696840" deviceId="myplug-
001" type="set"><param name="outletStatus">1</param></command></s2h>
```

JSON EXECUTION

```
curl --include \
>      --header "Content-Type: application/json" \
>   'https://esp.peoplepowerco.com:8443/deviceio/mljson?id=myplug-
001&timeout=120'
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: false
Access-Control-Allow-Methods: GET,POST,PUT,OPTIONS
Access-Control-Allow-Headers: Content-Type,Content-Length,X-Requested-
With,accept,Origin,User-Agent,Referer,Access-Control-Request-
Method,Access-Control-Request-Headers,PPCAuthorization,x-bl-deviceid,x-
bl-access-code
Access-Control-Max-Age: 10000
Cache-Control: no-cache, no-store
Content-Type: application/json;charset=utf-8
Content-Length: 151
Date: Wed, 20 Apr 2016 15:56:27 GMT
```

```
{"version":"2","status":"ACK","commands":[{"commandId":138696840,"type"
:0,"deviceId":"myplug-
001","parameters":[{"name":"outletStatus","value":"1"}]}]}
```

As you can see, we got command **138696840** for device with ID myplug-001, to switch "ppc.outletStatus" to a 0.

6. Acknowledge the command

We should always acknowledge to Presto that we received the command (because that's what we said we'd do when we set up the device type). If we don't do this, then Presto will keep sending the command to the device whenever it has the opportunity to do so.

XML

```
curl --request POST \
-H "Content-Type: application/xml" \
--data '<?xml version="1.0" encoding="utf-8" ?>
 <h2s ver="2" proxyId="myplug-001" seq="1">
  <response cmdId="138696840" result="1" />
 </h2s>
 ' \
--verbose \
https://esp.peoplepowerco.com:8443/deviceio/mlxml
```

JSON

```
curl --request POST \
-H "Content-Type: application/json" \
--data '{
   "version": "2",
   "proxyId": "myplug-001",
   "sequenceNumber": "1",
   "responses":
     [
       {
         "commandId": 138696840,
         "result": 1
       }
    ]
}
 ' \
--verbose \
https://esp.peoplepowerco.com:8443/deviceio/mljson
```

XML EXECUTION

```
curl --request POST \
> -H "Content-Type: application/xml" \
```

```
> --data '<?xml version="1.0" encoding="utf-8" ?>
>   <h2s ver="2" proxyId="myplug-001" seq="1">
>     <response cmdId="1996" result="1" />
>   </h2s>
>   ' \
> --verbose \
> https://esp.peoplepowerco.com:8443/deviceio/mlxml
* About to connect() to esp.peoplepowerco.com port 8443 (#0)
*   Trying 184.73.155.162... connected
* Connected to esp.peoplepowerco.com (184.73.155.162) port 8443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
*   CAfile: /etc/pki/tls/certs/ca-bundle.crt
  CApath: none
* SSL connection using TLS_DHE_RSA_WITH_AES_128_CBC_SHA
* Server certificate:
*     subject: CN=*.peoplepowerco.com,OU=Domain Control Validated
*     start date: Mar 03 04:40:49 2013 GMT
*     expire date: Mar 10 03:15:43 2016 GMT
*     common name: *.peoplepowerco.com
*     issuer: serialNumber=07969287,CN=Go Daddy Secure Certification
Authority,OU=http://certificates.godaddy.com/repository,O="GoDaddy.com,
Inc.",L=Scottsdale,ST=Arizona,C=US
> POST /deviceio/mlxml HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7
NSS/3.14.0.0 zlib/1.2.3 libidn/1.18 libssh2/1.4.2
> Host: esp.peoplepowerco.com:8443
> Accept: */*
> Content-Type: application/xml
> Content-Length: 137
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Credentials: false
< Access-Control-Allow-Methods: GET,POST,OPTIONS
< Access-Control-Allow-Headers: Content-Type,Content-Length,X-
Requested-With,accept,Origin,User-Agent,Referer,Access-Control-Request-
Method,Access-Control-Request-Headers,PPCAuthorization,x-bl-deviceid,x-
bl-access-code
< Access-Control-Max-Age: 10000
< Cache-Control: no-cache, no-store
< Content-Type: text/xml;charset=utf-8
< Content-Length: 74
< Date: Fri, 27 Mar 2015 04:27:50 GMT
<
<?xml version="1.0" encoding="UTF-8"?>
* Connection #0 to host esp.peoplepowerco.com left intact
* Closing connection #0
<s2h seq="1" status="ACK" ver="2"/>
```

JSON EXECUTION

```
curl --request POST \
```

```
> -H "Content-Type: application/json" \
> --data ' {
>     "version": "2",
>     "proxyId": "myplug-001",
>     "sequenceNumber": "1",
>     "responses":
>         [
>             {
>                "commandId": 138696840,
>                "result": 1
>             }
>        ]
> }
>   ' \
> --verbose \
> https://esp.peoplepowerco.com:8443/deviceio/mljson
*   Trying 50.16.244.212...
* Connected to esp.peoplepowerco.com (50.16.244.212) port 8443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: *.peoplepowerco.com
* Server certificate: Go Daddy Secure Certificate Authority - G2
* Server certificate: Go Daddy Root Certificate Authority - G2
> POST /deviceio/mljson HTTP/1.1
> Host: esp.peoplepowerco.com:8443
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 218
>
* upload completely sent off: 218 out of 218 bytes
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Credentials: false
< Access-Control-Allow-Methods: GET,POST,PUT,OPTIONS
< Access-Control-Allow-Headers: Content-Type,Content-Length,X-
Requested-With,accept,Origin,User-Agent,Referer,Access-Control-Request-
Method,Access-Control-Request-Headers,PPCAuthorization,x-bl-deviceid,x-
bl-access-code
< Access-Control-Max-Age: 10000
< Cache-Control: no-cache, no-store
< Content-Type: application/json;charset=utf-8
< Content-Length: 40
< Date: Wed, 20 Apr 2016 15:58:36 GMT
<
* Connection #0 to host esp.peoplepowerco.com left intact
{"version":"2","seq":"1","status":"ACK"}
```

The server got our response, and won't send the command again.

In a real device, the HTTP GET connection would turn back on whenever there is not an HTTP POST happening.

7. Update the measurements for the new state of the device

Now, our (virtual) smart plug would have updated its "ppc.outletStatus" state. It needs to reflect the new state immediately at the server. We do another HTTP POST to send an updated measurement.

XML

```
curl --request POST \
-H "Content-Type: application/xml" \
--data '<?xml version="1.0" encoding="utf-8" ?>
 <h2s ver="2" proxyId="myplug-001" seq="1">
  <measure deviceId="myplug-001">
   <param name="ppc.outletStatus">0</param>
  </measure>
 </h2s>
 ' \
--verbose \
https://esp.peoplepowerco.com:8443/deviceio/mlxml
```

JSON

```
curl --request POST \
-H "Content-Type: application/json" \
--data ' {
   "version": "2",
   "proxyId": "myplug-001",
   "sequenceNumber": "1",
   "measures": [
      {
         "deviceId": "myplug-001",
         "params": [
            {
               "name": "ppc.outletStatus",
               "value": "0"
            }
         ]
      }
   ]
}
 ' \
--verbose \
```

https://esp.peoplepowerco.com:8443/deviceio/mljson

XML EXECUTION

```
curl --request POST \
> -H "Content-Type: application/xml" \
> --data '<?xml version="1.0" encoding="utf-8" ?>
>   <h2s ver="2" proxyId="myplug-001" seq="1">
>     <measure deviceId="myplug-001">
>       <param name="ppc.outletStatus">0</param>
>     </measure>
>   </h2s>
>   ' \
> --verbose \
> https://esp.peoplepowerco.com:8443/deviceio/mlxml
*   Trying 50.16.244.212...
* Connected to esp.peoplepowerco.com (50.16.244.212) port 8443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: *.peoplepowerco.com
* Server certificate: Go Daddy Secure Certificate Authority - G2
* Server certificate: Go Daddy Root Certificate Authority - G2
> POST /deviceio/mlxml HTTP/1.1
> Host: esp.peoplepowerco.com:8443
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/xml
> Content-Length: 194
>
* upload completely sent off: 194 out of 194 bytes
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Credentials: false
< Access-Control-Allow-Methods: GET,POST,PUT,OPTIONS
< Access-Control-Allow-Headers: Content-Type,Content-Length,X-
Requested-With,accept,Origin,User-Agent,Referer,Access-Control-Request-
Method,Access-Control-Request-Headers,PPCAuthorization,x-bl-deviceid,x-
bl-access-code
< Access-Control-Max-Age: 10000
< Cache-Control: no-cache, no-store
< Content-Type: text/xml;charset=utf-8
< Content-Length: 74
< Date: Wed, 20 Apr 2016 16:00:04 GMT
<
<?xml version="1.0" encoding="UTF-8"?>
* Connection #0 to host esp.peoplepowerco.com left intact
<s2h seq="1" status="ACK" ver="2"/>
```

JSON EXECUTION

```
curl --request POST \
> -H "Content-Type: application/json" \
```

```
> --data ' {
>     "version": "2",
>     "proxyId": "myplug-001",
>     "sequenceNumber": "1",
>     "measures": [
>         {
>             "deviceId": "myplug-001",
>             "params": [
>                 {
>                     "name": "ppc.outletStatus",
>                     "value": "0"
>                 }
>             ]
>         }
>     ]
> }
>   ' \
> --verbose \
> https://esp.peoplepowerco.com:8443/deviceio/mljson
*   Trying 50.16.244.212...
* Connected to esp.peoplepowerco.com (50.16.244.212) port 8443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate: *.peoplepowerco.com
* Server certificate: Go Daddy Secure Certificate Authority - G2
* Server certificate: Go Daddy Root Certificate Authority - G2
> POST /deviceio/mljson HTTP/1.1
> Host: esp.peoplepowerco.com:8443
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 329
>
* upload completely sent off: 329 out of 329 bytes
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Credentials: false
< Access-Control-Allow-Methods: GET,POST,PUT,OPTIONS
< Access-Control-Allow-Headers: Content-Type,Content-Length,X-
Requested-With,accept,Origin,User-Agent,Referer,Access-Control-Request-
Method,Access-Control-Request-Headers,PPCAuthorization,x-bl-deviceid,x-
bl-access-code
< Access-Control-Max-Age: 10000
< Cache-Control: no-cache, no-store
< Content-Type: application/json;charset=utf-8
< Content-Length: 40
< Date: Wed, 20 Apr 2016 16:00:59 GMT
<
* Connection #0 to host esp.peoplepowerco.com left intact
{"version":"2","seq":"1","status":"ACK"}
```

You can check your generic developer device UI to see that the "ppc.outletStatus" state is now updated: