Intel® Internet of Things (IoT) Developer Kit

IoT Cloud-Based Analytics User Guide

November 2014

Table of Contents

LEGAL

1

Introduction

Intel provides a cloud-based analytics system for the Internet-of-Things (IoT) that includes resources for the collection and analysis of sensor data that the Intel® IoT Developer Kit provides. Using this service, Intel Galileo/Edison device developers can jump-start data acquisition and analysis without having to invest in large-scale storage and processing capacity.

This guide shows IoT developers how to access and take advantage of this valuable resource. Several elements are necessary in this multi-step process. Setting up an IoT Analytics account first and then proceeding with connectivity and sensors is the best pathway to success. Your device must be able to access the internet so that it can connect to the cloud. Networks with ports blocked, or firewalls may see difficulty in connectivity. Two pieces of software are involved on the device, one to collect data, the other to send data. This document will emphasize connectivity to the IoT Analytics site first (with device), then to collect data to send.

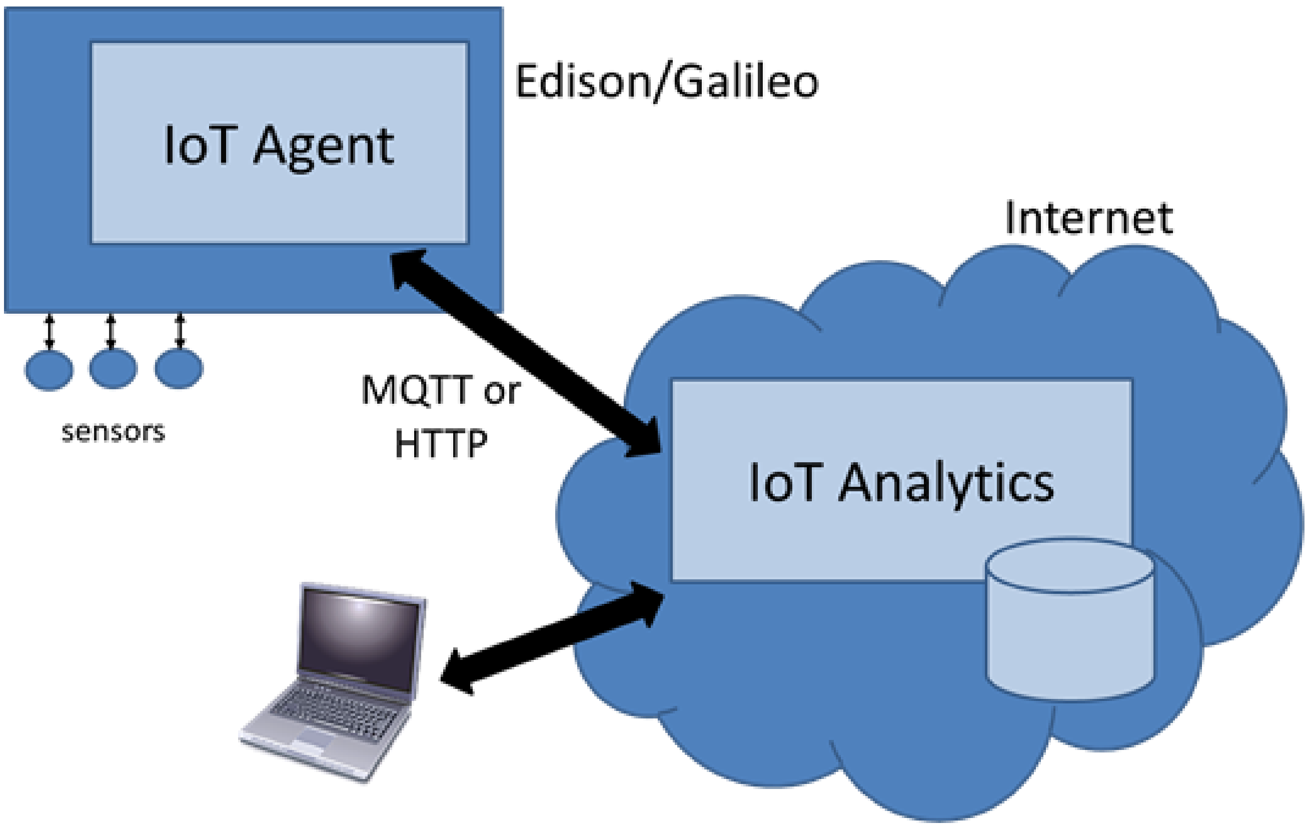


Figure 1. A basic model of IoT Analytics use

Considerations and Warnings

Prior to proceeding with the activities below it is important to have established connectivity and overall functionality with your Intel IoT device. Both an Internet and a serial connection are used at times to address the device. If you need assistance prior to proceeding with the setup of IoT Analytics, please visit the Intel IoT website and use the various getting started guides to solve any problems (https://software.intel.com/IoT)

The IoT Cloud Analytics site is provided as a service to the IoT development community. Keep the following considerations and warnings in mind when using the site:

- Using the iotkit-admin program to change your device ID (from the default of MAC address) does not make the device anonymous. The device IP address will still be available to Intel and the IoT Analytics site. In addition, if a device collects and submits GPS coordinates, Intel will have access to that information, as well. Please carefully review the Terms and Conditions and the Intel Online Privacy Notice.
- Do not connect a Intel® IoT Developer Kit board to any kind of actuator that could cause harm or damage. It is possible that users could unintentionally activate these devices.

2 Accessing and Using IoT Analytic Functionality

IoT Analytics: Login and Account Creation

To begin using the IoT Analytics cloud site, create a new administrative user account that will have the ability to register devices, manage alerts, create accounts, and perform other tasks on the site.

Go to the IoT Analytics Dashboard website (https://dashboard.us.enableiot.com). Figure 1 shows the **Sign in** screen that will be displayed.

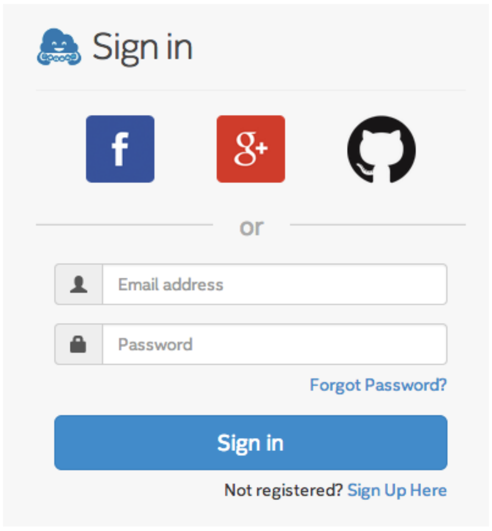
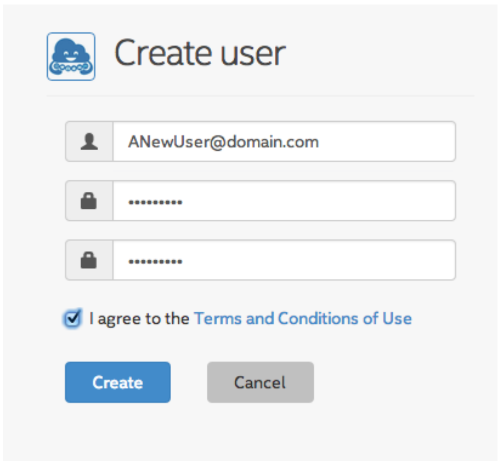


Figure 2. Creating an Account on the IoT Analytics site

You have the option of using your Facebook, Google+, or GitHub authentication, or you can create a local account with an email address and password. If you click **Sign Up Here**, the **Create user** page appears (see Figure 2).


Create user

ANewUser@domain.com

•••••••••

•••••••••

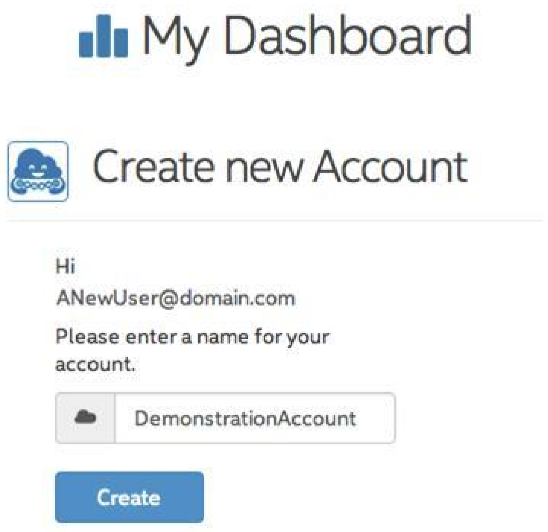
☑ I agree to the Terms and Conditions of Use

Figure 3. Create a New User

The **Create user** screen requires a valid email address (used to send the activation link to complete account creation) and a user-provided password. Review the Terms and Conditions, and if you agree, select the **I agree to the Terms and Conditions of Use** check box, and then click **Create**.

After the user login has been created, you are prompted to create an account name (see Figure 3).

My Dashboard provides access to all system functions, including the number of devices, the number of observations received in the past hour, and the percentage of devices that are "healthy," meaning they have transmitted data recently. You can customize the lower section to show a "favorite" chart, which contains a chart of specific time series from specific devices. This is described more fully in the User Manual.


My Dashboard

Create new Account

Hi
ANewUser@domain.com

Please enter a name for your account.

DemonstrationAccount

Figure 4. Create a New Account

This name appears on the site when you log in. When you have authenticated, the **My Dashboard** page appears (see Figure 4).

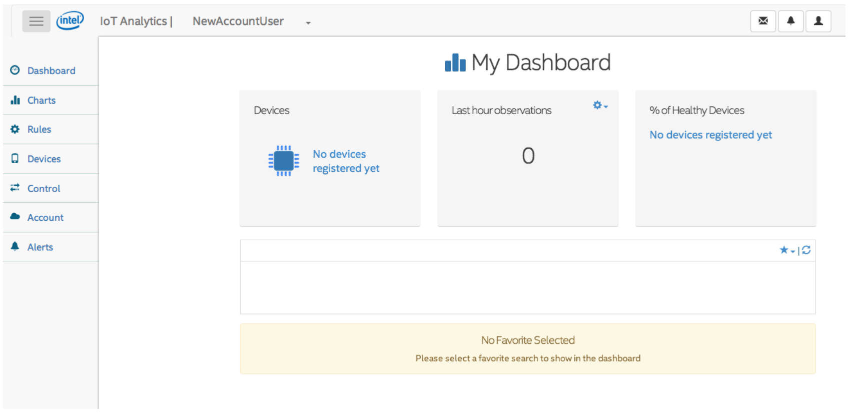

IoT Analytics | NewAccountUser
Dashboard
Charts
Rules
Devices
Control
Account
Alerts
My Dashboard
Devices
No devices registered yet
Last hour observations
0
% of Healthy Devices
No devices registered yet
No Favorite Selected
Please select a favorite search to show in the dashboard

Figure 5. User Dashboard and Navigation Page

Account Administration

Each account that you create on the IoT Analytics site is a separate workspace. You administer the accounts from the **My Account** page (see Figure 5).

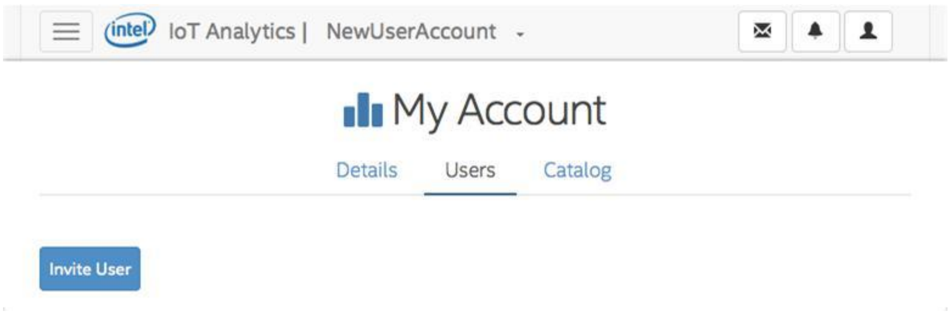


Figure 6. Account Administration

From this page, you can view the account details, invite additional users to participate in the analytics site (by registering their email address) and manage the catalog of Component Types.

Component Types are the templates which define time series and actuators to be used by your devices. Note that the catalog already contains a Component Type named “temperature.v1.0” which we will use in the following examples. This defines a time series where each observation is a floating point number, and the unit-of-measures is “Degrees Celsius”.

IoT Kit Agent

The IoT Analytics site supports a REST interface, making it possible to write client software which runs on the device and sends observations to the cloud. However, to simplify this process, there is also an Agent. The agent runs as a daemon on the device, listening for simple messages from other processes and handling the necessary message formatting and security to send observations to the cloud. The agent comes with another program, iotkit-admin, which provides many utility functions, such as tesing the network, activating a device, registering time series, and sending test observations.

The next steps depend on whether the iotkit-agent and iotkit-admin program are pre-installed:

Galileo and Edison Device with the Agent pre-installed

If you have a Galileo or Edison, you can see if the iotkit-agent and iotkit-admin are pre-installed by typing the following in the shell:

```
iotkit-admin
```

If this results in “Usage: iotkit-admin …”, then the program is pre-installed. This means:

- You do not need to install Node.js, npm or the agent code.
- The programs are in the path and you can run them from any directory.
- You do not need to add the “.js” suffix to the program names. For example, you can type “iotkit-admin” rather than “iotkit-admin.js” (without the quotes).
- The agent is controlled by “systemctl”, the systemd service manager.

If this is the case, type the following in the shell to stop the agent. We will start it again later.

```
systemctl stop iotkit-agent
```

Agent is not pre-installed

If you are working on a device other than a Galileo or Edison, or the agent is not pre-installed:

- You may need to install Node.js and npm first. Please see the Node.js web site for instructions.
- You will need to install the agent and it’s dependencies. To do that, “cd” to the directory where you want the agent installed and type:

  ```
  npm install iotkit-agent
  mv node_modules/iotkit-agent/ .
  rm –rf node_modules
  ```

- The programs will not be in your path, so you will need to “cd” to the iotkit-agent directory and run them from there.
- You will need to preface all commands with the prefix ./ (this is the standard Bash shell indicator to run a program from the local directory) and include the “.js” suffix. For example, you must type “./iotkit-admin.js” rather than “iotkit-admin” (without the quotes).
- run a program from the local directory). You may also need to add the “.js” suffix to the script name.

Network Test

To test network connectivity, run the following command from the command line:

```
iotkit-admin test
```

This command should produce the following message:

```
2014-11-10T21:50:54.062Z - info: Trying to connect to host ...
2014-11-10T21:50:54.570Z - info: Connected to dashboard.us.enableiot.com
2014-11-10T21:50:54.571Z - info: Environment: prod
2014-11-10T21:50:54.571Z - info: Build: 0.10.6
```

If you are trying to connect from behind a firewall with a proxy server, add the proxy server to the iotkit-admin configuration:

```
iotkit-admin proxy http://proxy.company.com 8080
```

Repeat the above test command. If you are still having connection problems, you must correct them before you can proceed with the installation.

Note: This is specific to the REST protocol. MQTT will not operate via a proxy.

Getting and Changing the Device ID

You should know your device ID because you may need it to find the device in the device list if you have more than one device. You can display the device ID with the following command:

```
iotkit-admin device-id
```

You can change this device ID (if desired) by running the following command.

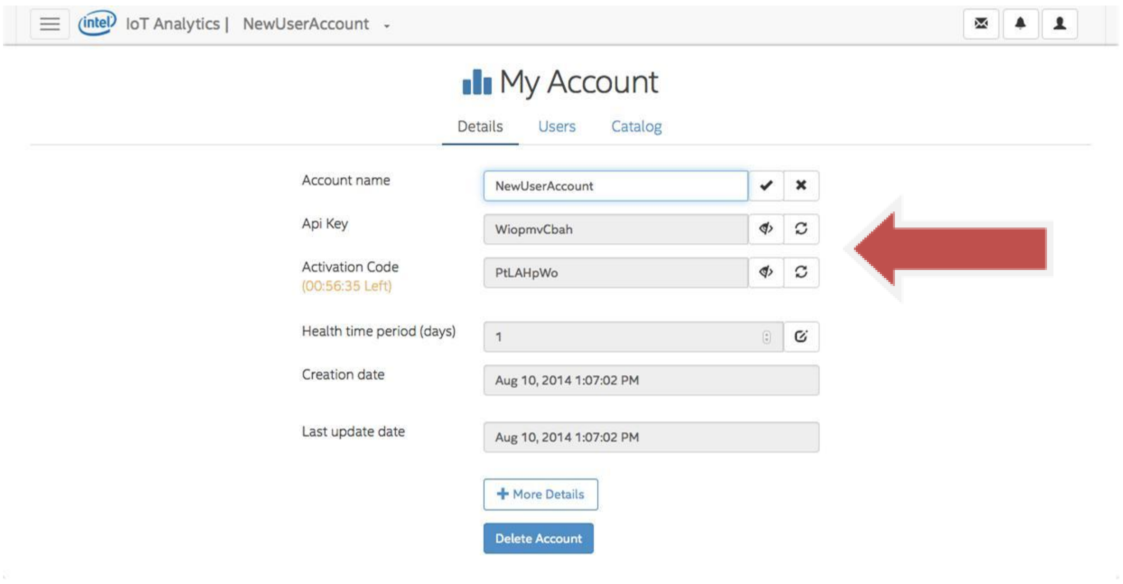
Note: Changing the ID does not provide anonymity. Device ID's should be unique. The source TCP/IP address will still be available to the cloud server.

```
iotkit-admin set-device-id «device_id»
```

It is also important to note that the device activation step returns a security token which will only work with the device ID which was used at activation. The device ID should not be changed after activation.

Activating the Device

To complete the device activation, you must supply a specific account activation code from the IoT Analytics account **Details** page (see Figure 6).



Account name	NewUserAccount
Api Key	WiopmvCbah
Activation Code (00:56:35 Left)	PtLAHpWo
Health time period (days)	1
Creation date	Aug 10, 2014 1:07:02 PM
Last update date	Aug 10, 2014 1:07:02 PM

Figure 7. My Account Agent Activation Code

The code is valid for only 60 minutes. After that, you will have to regenerate the code by clicking the button indicated in Figure 6. You must supply this code when activating the device:

```
iotkit-admin activate «activation_code»
```

The activation process creates the device record in the cloud, associates the device with the account (where the activation code came from), and provides the device with security credentials.

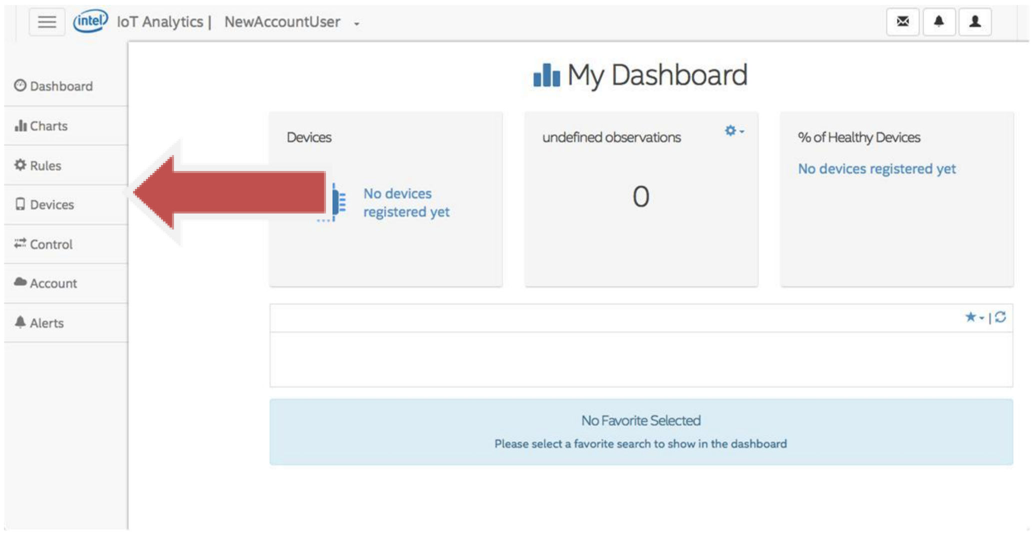
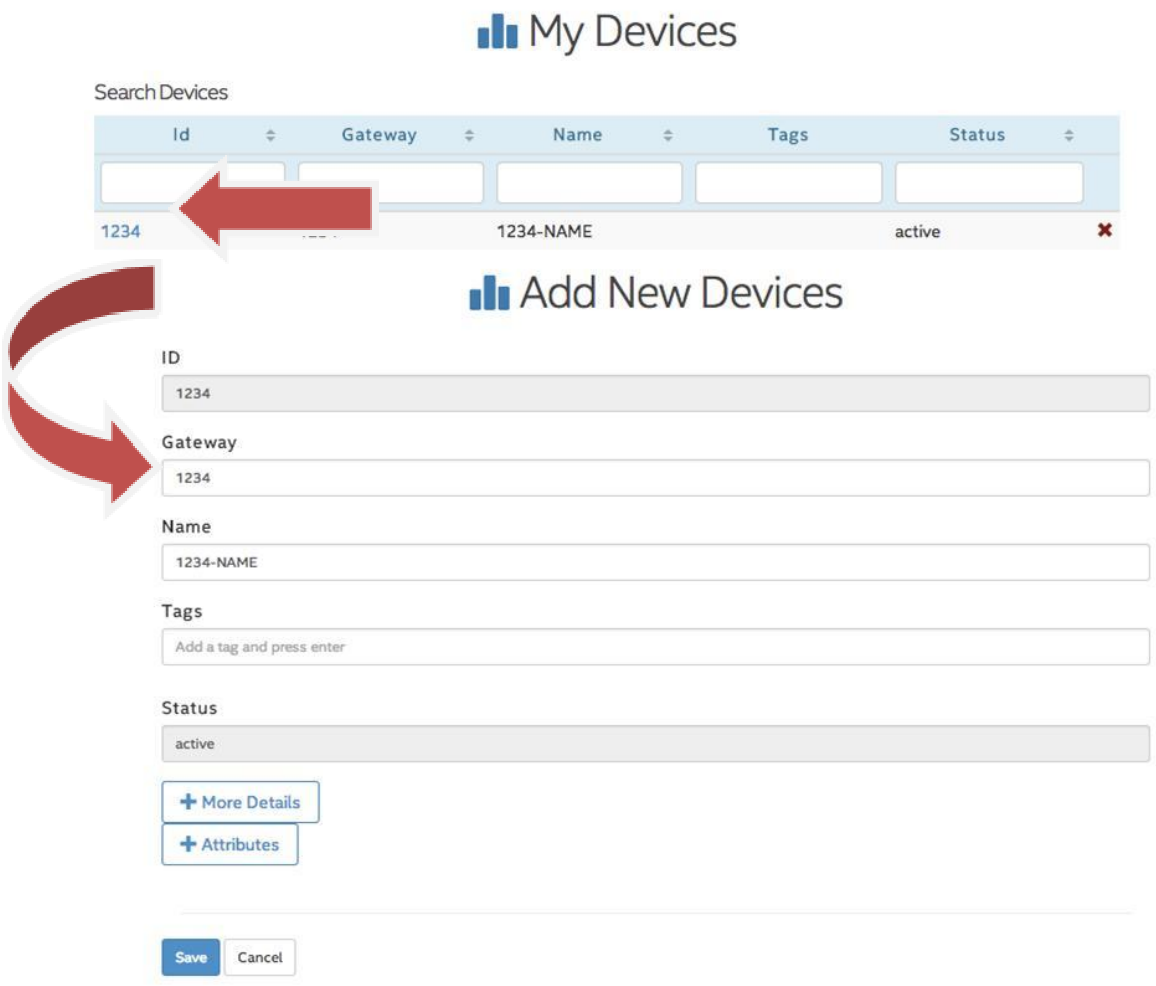

IoT Analytics | NewAccountUser
My Dashboard
Dashboard
Charts
Rules
Devices
Control
Account
Alerts
Devices
No devices registered yet
undefined observations
0
% of Healthy Devices
No devices registered yet
No Favorite Selected
Please select a favorite search to show in the dashboard

Figure 8. Accessing the Devices page

The **My Devices** page will be displayed where you can see the newly activated device. Click the Device ID (as shown in Figure 8) to see the device's detailed information.



Id	Gateway	Name	Tags	Status
1234	1234	1234-NAME		active

ID: 1234
Gateway: 1234
Name: 1234-NAME
Tags: Add a tag and press enter
Status: active

Figure 9. The My Devices Page and the Add New Devices Dialog Box

Sensor Registration and Data Submission

Now that the device has been activated, the next step is to register components. A component is either a time series which consists of sensor observations sent from the device, or an actuator, which allows commands to be sent to a device. For example, if you want to send both temperature and humidity, you would register two components.

Each component you reference for a device must reference a “Component Type” within the account Catalog. Why Component Types? If you had many devices throughout a city that all measured the same thing, such as Nitrogin dioxide (NO_2) content, Component Types save you the trouble of defining the details of the time series over and over for each device. Instead, the Catalog would hold a Component Type for Nitrogen dioxide, which would define a time series of integer values, with the unit-of-measure of “NO_2 pphm” (parts per hundred million). The unit-of-measure is printed on the Y access whenever the time series is charted.

You access the Catalog from My Account, as shown in Figure 9.

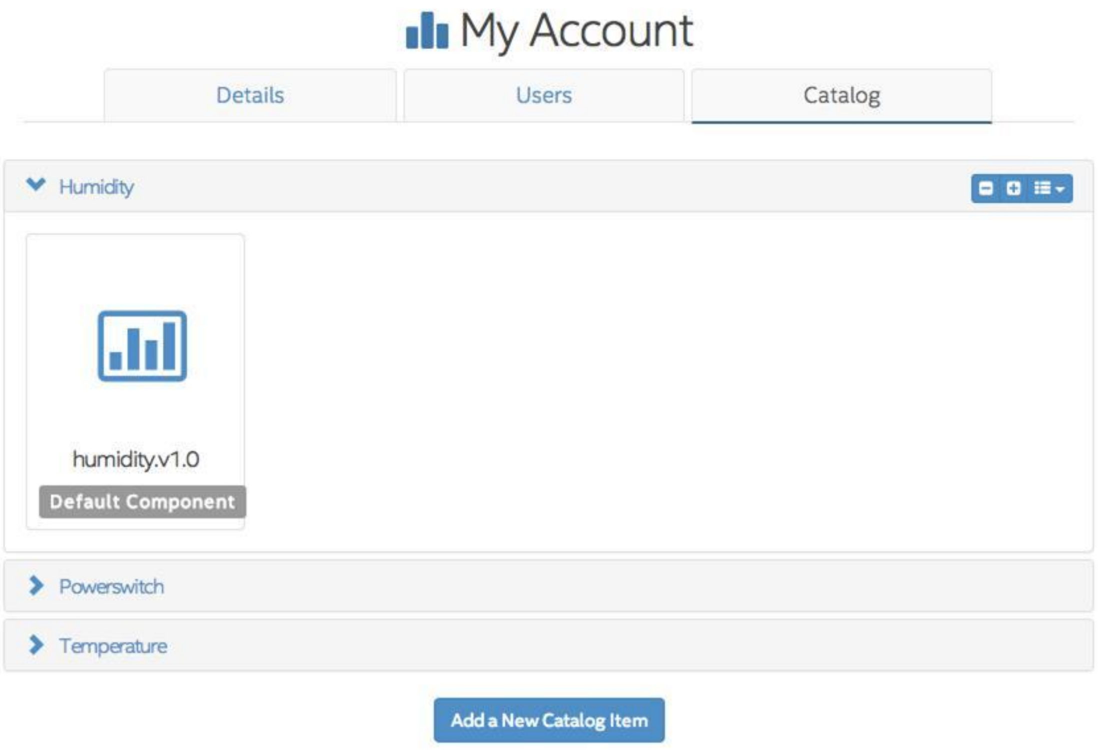


Figure 10. The Component Catalog from the My Account Page

There are three default components in each account: humidity, power, and temperature. You can also add new Component Types from the **Add a New Catalog Item** dialog box. If a new component type is added, the local agent must be restarted before that type can be used to register a new component.

You perform the registration step only once for each Component (i.e. time series) — for example:

```
iotkit-admin register temp temperature.v1.0
```

Here we are registering a time series that references the Component Type named “temperature.v1.0” in the Catalog. Here we have created an alias “temp” which will be used to send temperatures observations to the agent.

You can find the Component Types which have been registered in the Catalog using the following command on the device:

```
iotkit-admin catalog
```

Note: Any catalog changes made in the dashboard will not be reflected locally until the agent is restarted or a 'sync' command is issued manually.

Now that the component has been registered, it is possible to begin sending observations to the IoT Analytics site for this component. There are several ways to do this, but to test that everything is working, run the observation command to send individual observations to the IoT Analytics site:

```
iotkit-admin observation temp 35
iotkit-admin observation temp 30
```

Each of these two commands cause one observation to be sent to the IoT Analytics site. The "temp" alias indicates these observations should be added to the "temp" time series which was previously registered.

Switch back to the website and select **Chart** from the navigation pane to select the device and component and see the newly sent data elements (Figure 10). Hover over the ends of the lines to see the specific values and the timestamps when the observations were sent.

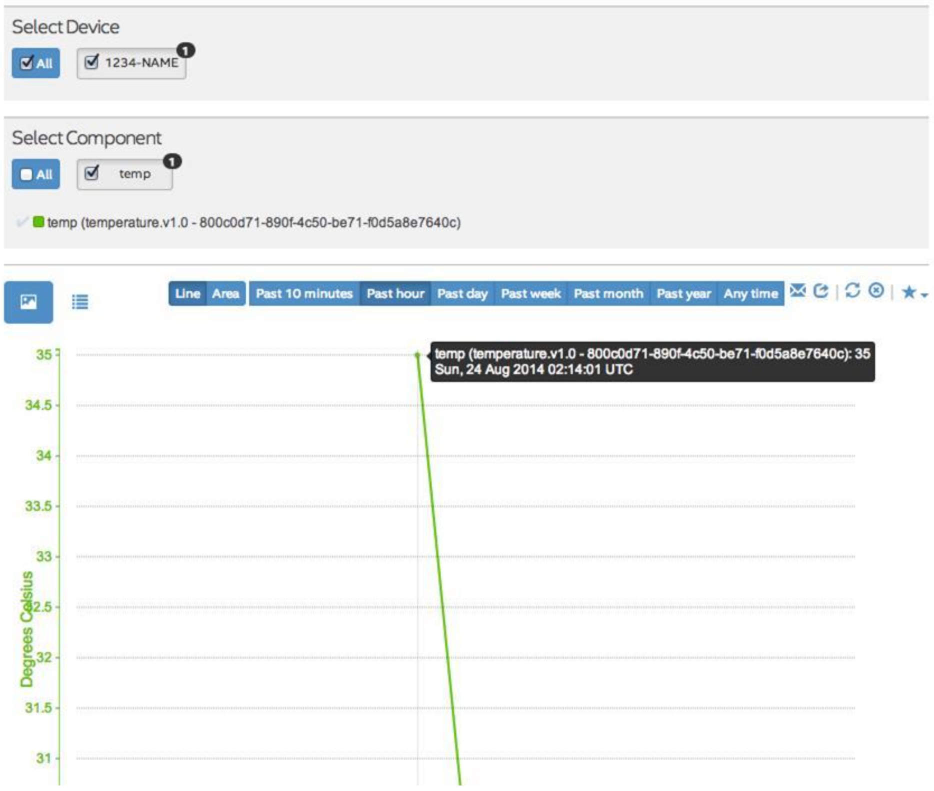


Figure 11. Chart for the Selected Device and Component

Sending Multiple Data Packets

The "iotkit-admin observation" command is intended to facilitate testing, not for continuous data capture. There are two additional ways to send observation data to the IoT Analytics site:

- A client can use the REST interface documented at https://github.com/enableiot/iotkit-agent/wiki/Api-Home to interacttly directly with the cloud.
- A client can send User Datagram Protocol (UDP) packets to the local Agent. The Agent converts these packets into REST calls and sends the request on behalf of the client.

This section explains how to use the Agent. First, the agent must be running. On the Galileo or Edison with the software pre-installed, you can use the following command:

```
systemctl start iotkit-agent
```

If you installed the agent yourself, just open a terminal window, “cd” to the iotkit-agent directory, and run:

```
./iotkit-agent.js
```

and leave it running. Open a second terminal to run the following commands.

You can send UDP-formatted messages to the Agent through the device localhost on port 41234 (e.g., UDP://localhost:41234). First, create a JavaScript Object Notation (JSON)–formatted document, where the key “n” is the name of a time series and “v” is the value:

```
{ "n": "<sensor name>", "v": "<value>" }
```

```
Examples:
{ "n": "temp", "v": 21.0 }
{ "n": "humidity", "v": 71.5}
```

Most programming languages can transmit this message, but on Mac OS X* and Linux*, use the nc program as follows:

```
echo -n '{"n": "temp", "v": 21.0}' | nc -u -w1 127.0.0.1 41234
```

This command sends the document to the agent. The agent (which is listening on port 41234 for the UDP messages) will convert the observation to a REST web service call (or MQTT messages) and forward it to the IoT Analytics site.

Windows does not include the “nc” command, and the –u argument is not supported on the Yocto build used on the Galileo and Edison boards. For this reason we have included a utility with the Agent that you can use to do the same thing.

```
test/send-udp.js temp 21.0
```

This simple program will create the JSON document and send it to port 41234 on localhost using UDP. If you cannot find the program, run the following command to get the latest release of the iotkit-agent, and try it again.

```
npm update iotkit-agent
```

3 Cloud Connectivity using IoT Kit Arduino Library and Samples

You can download an Arduino library and code samples from GitHub at https://github.com/enableiot/iotkit-samples/blob/master/arduino/IoTkit.zip that illustrate how to connect to the iotkit-agent.

These samples assume that you have already installed (or have access to) the iotkit-agent and that it is running. To install the IoTkit library and examples, complete these steps:

1. From the Arduino integrated development environment, click **Sketch > Import Library > Add Library**.
2. Browse to the IoTkit.zip file that was downloaded.
3. Click **Open**.

When installation is complete, you should see IoTkit listed under **File > Examples**. Open the "IoTkitSimpleExample.ino" file. Note that the first argument in the iotkit.send() call must be the alias that you used when you registered the component. For example, if you used :

```
iotkit-admin register temp temperature.v1.0
```

to register the component, the Arduino code would be:

```
iotkit.send("temp", temp);
```

4 Cloud Connectivity via REST API

You have seen how to send data to the IoT Analytics site via the iotkit-agent using either, the command-line or Arduino library and samples. Users can also directly send data to the cloud using C/C++, JavaScript* or Python.

The following sample code illustrates the use of pushing data to the cloud via HTTP.

Python – https://github.com/enableiot/iotkit-samples/tree/master/api/python

For JavaScript – https://github.com/enableiot/iotkit-samples/tree/master/api/javascript

Python Usage

The Python example on Github (iotkit_client.py) performs the following: authenticate against iotkit-dashboard, create a new device, register a new sensor and submits data.

Configuration

The Python sample code contains several variables to be configured to use your iotkit-dashboard account including your login (email), password, account name and a unique device name.

```
host = "dashboard.us.enableiot.com"

proxies = {
   # "https": "http://proxy.example.com:8080"
}

username = "my_email_address@example.com"
password = "myPassword"
account_name = "myAccountName"

#this will create a device with this id - error if it already exists
device_id = "myDevice"
```

REST API Commands

The Python sample contains several helper functions where the REST API calls are constructed, sent and parse the response for errors.

JavaScript Usage

The Javascript examples on Github perform the following: authenticate against iotkit-dashboard, create a new device, register a new sensor and submit data. The examples support two usage patterns:

- Simple get data and submit results (single)
- Event-based subscription model (ongoing)

Simple

Simply setup the cloud client, retreave results from observation provider and submit:

```
// error handling and validation omitted for clarity
cloud.setup(config, function(err, state){
    var observations = provider.get(state);
    cloud.submitObservations(state, observations, err_handler);
});
```

See examples for working example.

Event-based

When ongoing submission is required you can use the event-based pattern with subscription to the data event:

```
// error handling and validation omitted for clarity
cloud.setup(config , function(err, state){
    provider.sub(state).on("data", function(observations){
      cloud.submitObservations(state, observations, err_handler);
    });
});
```

See examples for working example.

Provider

Both of these usage models are implemented using provider pattern. You can easily implement your own provider from Database, Message Queue or simply by tailing a log by implementing get and sub functions.

4.1.1.1. Get

```
var get = function(config){
    //TODO: Implement your code here
}
```

4.1.1.2. Sub

```
var sub = function(config){
    //TODO: Implement your code here
}
```

Both of these functions must return an array of observations.

Observation

Observation is a simple construct representing an event in time:

```
{
    "componentId": "my-registered-component-1",
    "on": 1406685882807,
    "value": "96.6"
}
```

The above object represents only the required elements. See enableiot documentation for more details about the supported types.

Example Configuration

When you look at the code of the JavaScript example, you will see a fair amount of set up necessary to reach the cloud. Of course you need your username and other login information as well as where you will connect to. The Javascript examples read their configuration settings from the configs/app.json file:

```
{
   "target": {
     "host": "dashboard.us.enableiot.com",
     "username": "me@myEmail.com",
     "password": "MyPassword",
     "account": "MyAccount"
   },
   "source": {
     "deviceId": "myDeviceName",
     "gatewayId": " myDeviceName ",
     "name": "myMockedDevice",
     …
     }
   },
   "component": {
     "type": "temperature.v1.0",
     …
   }
}
```

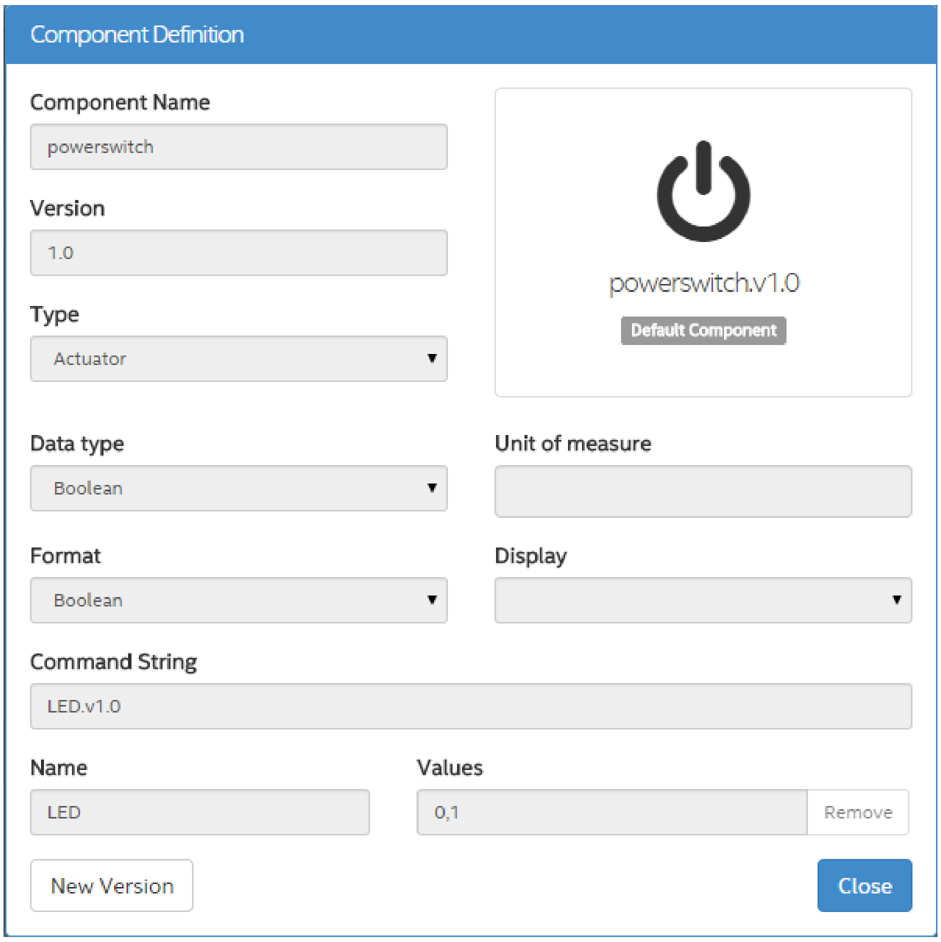
Note that the device name must be unique across all IoT Analytics users

5 Actuation

So far we have only covered “sensor” component types. Components can also be an “actuator”. Actuator components receive data from the cloud that client software can use to perform some local action. This section explains the various ways an actuator request is initiated and how that request is handled on the device. Note that actuation requests are sent to the device via MQTT protocol.

Actuator Component Types

An actuator component is designated by the “Type” property of the component type. The other type properties are similar to sensor components except for the “command String” and parameter Name/Value pairs.


Component Definition

Component Name: powerswitch

Version: 1.0

Type: Actuator

Data type: Boolean

Unit of measure:

Format: Boolean

Display:

Command String: LED.v1.0

Name: LED

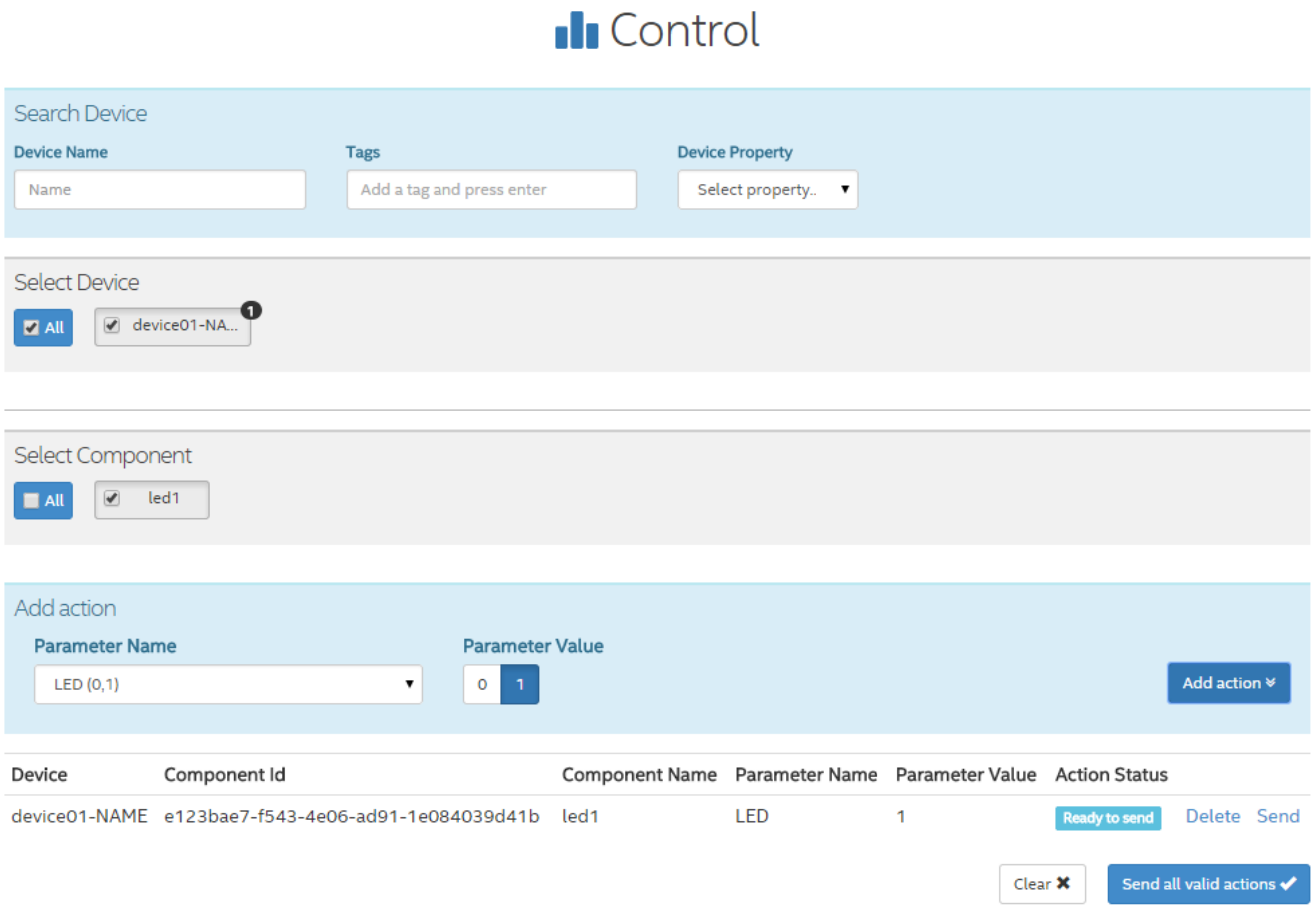
Values: 0,1

Figure 12. Component Type for Actuator

Actuator components have a fixed command string that may have some significance to the client software that processes the actuator request on the device. There may be one or more parameters that are sent to the client as well. These parameter values are what the user selects in the dashboard when sending an actuator request. Typically, the parameter value would indicate a pin value (0 or 1, “on” or “off” etc.)

Submitting Actuator Request

Actuator requests can be submitted through the ”Control” menu on the IoT Analytics Dashboard. The Control menu is similar to Chart menu – where you can select one or more devices and one or more of the “actuator” components on those devices. The menu allows the user to then set the desired parameter values before sending the request.



Control

Search Device

Device Name	Tags	Device Property
Name	Add a tag and press enter	Select property..

Select Device

All | device01-NA... 1

Select Component

All | led1

Add action

Parameter Name	Parameter Value
LED (0,1)	0 1

Device	Component Id	Component Name	Parameter Name	Parameter Value	Action Status
device01-NAME	e123bae7-f543-4e06-ad91-1e084039d41b	led1	LED	1	Ready to send Delete Send

Figure 13. Control Menu on Dashboard

Actuator Request Processing

When an actuator request is generated on the IoT Analytics Dashboard, it is sent to the device via MQTT protocol. The Agent must be configured to use the MQTT protocol to handle actuation requests. The Agent then processes the incoming request, reformats it as a JSON message and sends it to local UDP port 41235. The user must provide the software to listen to this port, handle the requests and perform the desired action based on the parameter value (turn LED on/off, set motor speed, etc.)

Below is an example of the local UDP actuation message sent to UDP port 41235

```
{
        "component": "led1",
        "command": "LED.v1.0",
        "argv": [{
                "name": "LED",
                "value": "1"
        }]
}
```

Examples

Two examples are provided to illustrate how Actuation requests are handles on the device. Both samples use the LED component parameter value to determine whether to turn pin 13 (onboard LED) on or off.

- Adruino library and sample sketch
- Node.js script

Bypassing the Agent

TBD

6 Start Working with the Cloud

You are now ready to write scripts, sketches and apps to send data to the cloud. Write the software on your computer and then download them to the board and run them. After a few minutes, check My Dashboard, and then the sensor you set up to capture to view the data.