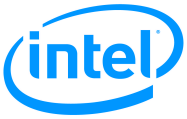
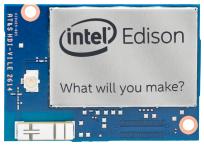
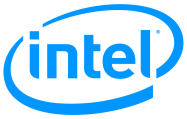




Intel® Cloud-based Analytics Part 2



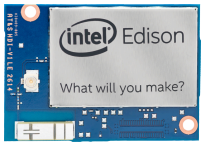
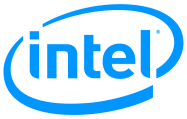
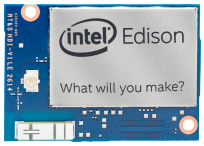


Table of Contents

Revision history		
Version	**Date**	**Comment**
1.0	10/11/2015	Initial release





Introduction

The previous tutorials introduced Intel Cloud-based Analytics. At the end of the tutorial, you should have become familiar with account creation, device activation, component registration, and sensor data transmission to the Cloud. In this tutorial, you will learn to:

1. Control actuators connected to the Intel Edison,
2. Set up rules on the Intel Cloud-based Analytics,
3. Interact with the Cloud via REST calls, and
4. Set up a remote system to control your Edison via Cloud.

Things Needed

1. An Intel Edison with Arduino-compatible breakout,
2. A micro USB cable,
3. A Grove Starter kit, and
4. A PC or Mac

Actuation

So far, we have only used sensor components. In this section, we will use actuators and learn about how to send actuation requests from the Cloud Analytics to the Intel Edison and how the Edison handles the requests. As briefly mentioned in the previous tutorial, the IoT Agent communicates with the Cloud Analytics via either HTTP or MQTT protocol. While sensor data transmission from the IoT Agent to the Cloud is done via either protocol, the Cloud can send actuation request to the IoT Agent via only MQTT protocol. The Cloud sends actuation requests as JSON messages to the IoT Agent. Then, a UDP client can receive the messages. We will skip the details for now. In the following demonstration, we will control the Edison's on-board LED from the Intel Cloud. Follow the steps below.

1. SSH into the Edison.
2. Start the IoT Agent by entering "**systemctl start iotkit-agent**".
3. On your Edison, change the protocol to MQTT by entering "**iotkit-admin protocol mqtt**".
4. Register an LED component by entering "**iotkit-admin register led_switch powerswitch.v1.0**".
5. Now, you can check whether the component registration was successful.
6. Go to https://dashboard.us.enableiot.com/
7. From the sidebar menu, click on **Devices**.
8. Click on **Component** to expand. Then, you should be able to see a component named **led_switch**.

Figure 1 Component

9. Now, go to the **Control** menu from the sidebar menu.
10. Select your device and then, select **led_switch**.

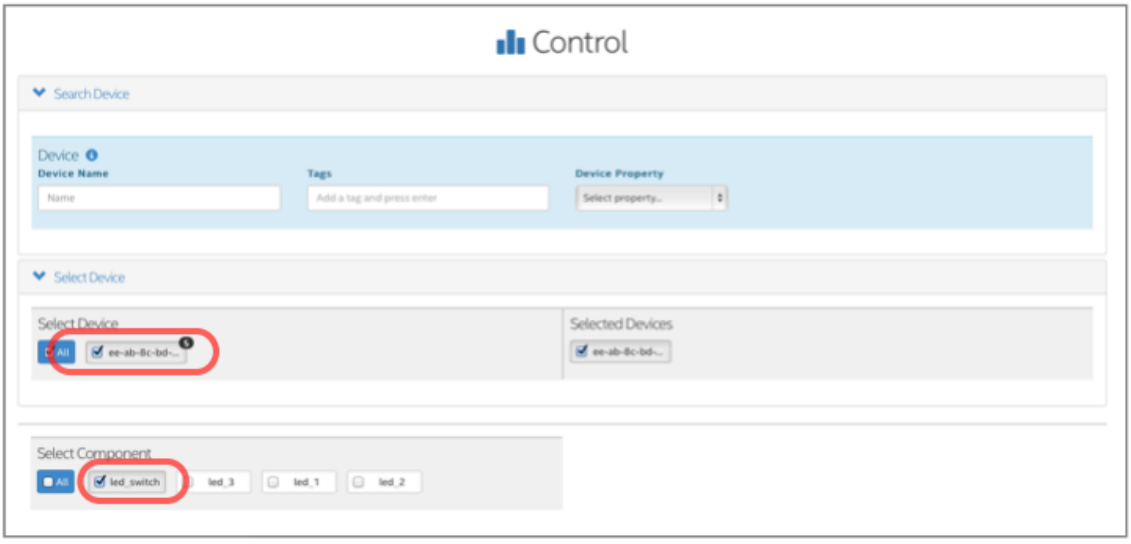


Figure 2 Control Menu

11. Then, "**Add action**" menu will appear below.

Figure 3 Add action

12. Select **LED (0,1)** for the **Parameter Name**, **1** for the **Parameter Value**, and **mqtt** for the **Transport type** and then, click **Add action**.
13. The actuation request is ready to be sent. Now click on **Send**.

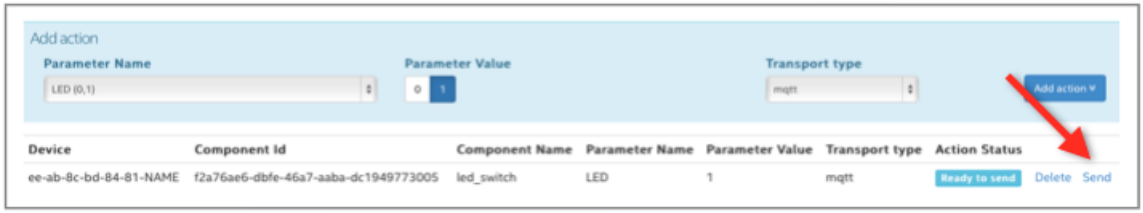
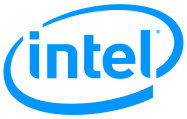
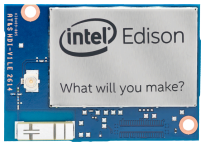


Figure 4 Send Actuation Request

14. **$ systemctl status iotkit-agent -l**
15. On the last line of the output, you should see the actuation request sent by the Cloud as a JSON message.

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





The general format of the message is:
{"component":"<registered name>","command":"<catalog ID>","argv":[{"name":"<catalog name>,"value":"<value sent from the cloud>"}]}

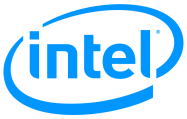
16. Now, we need a program that reads the JSON message and make actuation.
17. **$ git clone https://chrisIHbaek_reader:ucla_whi@bitbucket.org/chrisIHbaek/tutorial7.git**
18. **$ cp ./tutorial7/cloud_blink.c ~/cloud_analytics/cloud_blink.c**
19. **$ cd ~/cloud_analytics**
20. **$ gcc –lmraa –o cloud_blink cloud_blink.c**
21. **$ ./cloud_blink**
22. Repeat steps 9-13.
23. You should see that the on-board LED lights up when an actuation request whit value 1 is sent and turns off when a request with value 0 is sent.
24. Press **Ctrl-C** to quit.

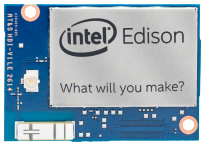
How does it work?
The Intel Cloud sends actuation requests in JSON format to the IoT Agent via MQTT protocol. Similar to how send_udp.js communicates with the IoT Agent, the cloud_blink program communicates with the IoT Agent via UDP protocol but with port 41235.

As shown in step 15, the JSON message contains the registered name, "**led_switch**" in this case, and the value we sent from the Intel Cloud Dashboard. We can extract these values. In the example code, some simple shell commands such as "**echo**", "**sed**", "**tr**", and "**awk**" are used. The "**echo**" command will send the JSON message to the stdout, which is then redirected as an input to "**sed**". The "**sed**" command (**sed 's/\"/ /g'**), then, replaces " with a blank space. The output of this looks like **{ component : led_switch , command : LED.v1.0 , argv :[{ name : LED , value : 1 }]}**. This is also redirected as an input to "grep". The "**tr**" command (**tr ' ' '\n'**) replaces all blank spaces with new line characters. Thus, the output looks like the following:

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

The "**awk**" command (**awk '/component/{getline; getline; print}'**) searches for "**component**" and skip to two lines below, then send the content, "**led_switch**" in this case, in that line to the



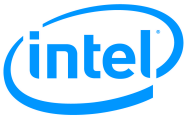


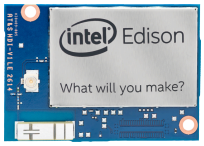
stdout. We can extract the value with the same idea. Based on the information extracted, we can control the LED using MRAA library.

The shell command trick may be useful to extract small number of words or values. However, it may not be feasible to parse much longer JSON message, from which you may need to extract a lot of information. In this case, you can use a third-party JSON library for C to parse the JSON messages. One of most commonly used JSON parser in C is JSMN. You can learn about how to install and use JSMN at http://zserge.com/jsmn.html.

Optional Practice

1. Implement a Cloud-based actuation to control an LED and a buzzer.
2. Implement a Cloud-based actuation to control the brightness of an LED (refer to "SPI, PWM, and More GPIO" tutorial for LED brightness control)





RULE

You can set up rules on the Intel Cloud-based Analytics system. With this functionality, the Intel Cloud-based Analytics is capable of sending notification or actuation requests based on sensor observation. For instance, you can set up a rule, which sends a notification email to you when a sensor reads low light intensity. In addition to the email notification, this system can turn on an LED. Let's build this system!

Note for Mac users: As reported in Part 1, Google Chrome makes errors. Please use Safari.

1. Create a new catalog item with attributes shown in the figure below.

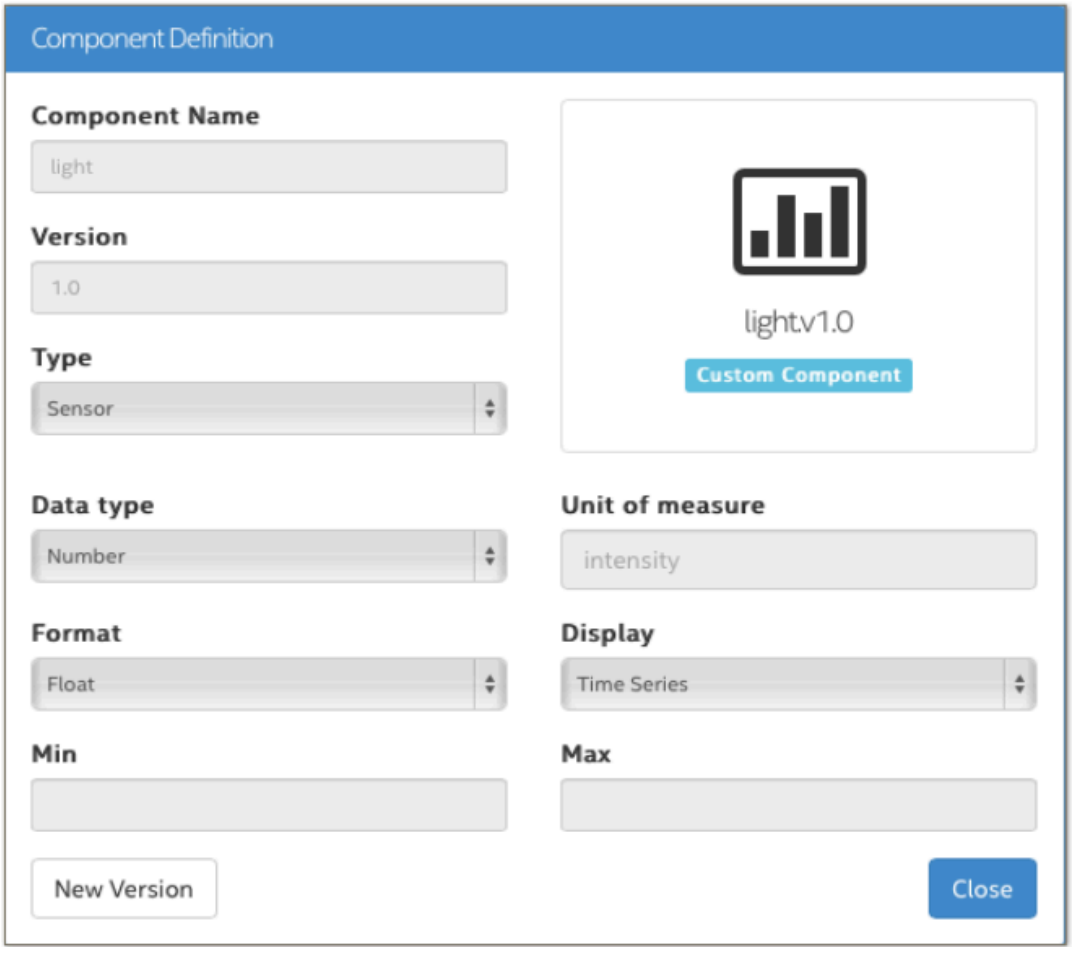


Figure 5 light.v1.0

2. On your Edison, register a light sensor entering "**iotkit-admin register light light.v1.0**".
3. As we did in the previous tutorial, we need to restart the IoT agent because it does not know of the component alias "light".
4. **$ systemctl restart iotkit-agent**
5. Now, we need to set up the hardware.
6. Insert the Grove Base shield into the Edison.
7. Connect the light sensor to A0 input of the Grove Base shield.
8. Now, we need a program that sends JSON messages to the cloud.

9. **$ vi cloud_light.c**
10. Type the following C code. This code is a modified version of cloud_rotary.c.

```
#include <stdio.h>
#include <unistd.h>
#include <mraa/aio.h>

int main()
{
      uint16_t value = 0;
      char cmd_buf[1024];
      mraa_aio_context light;
      light = mraa_aio_init(0);

      while(1){
            value = mraa_aio_read(light);
            snprintf(cmd_buf, sizeof(cmd_buf), "./send_udp.js light
%d", value);
            system(cmd_buf);
            printf("%d\n", value);
            sleep(1);
      }

      mraa_aio_close(light);
      return 0;
}
```

Figure 6 cloud_rotary.c

11. **$ gcc -lmraa -o cloud_light cloud_light.c**
12. Let's set up a rule on the Intel Cloud-based Analytics.
13. From the sidebar menu, click on **Rules**.
14. Click on **Add a rule**.

Figure 7 Rule Setup

15. Enter a rule name.
16. Select **Medium** for **Priority** and **Email** for **Notification type**.
17. **Notification to** sub-menu appears on the right.
18. Click **Choose one** and choose your email address. Then, click **Next**.
19. Select your device. Then, click **Next**.

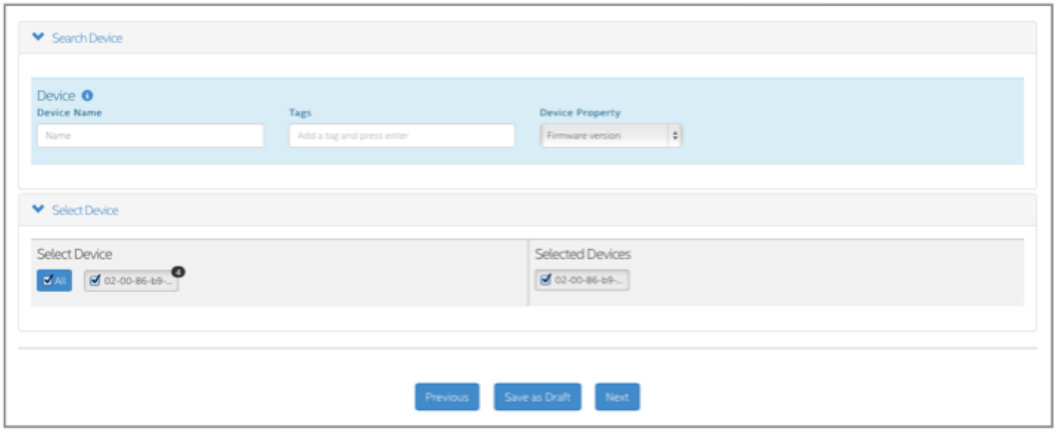


Figure 8 Device Selection

20. Choose **light (Number)** for **Monitor Measure**.

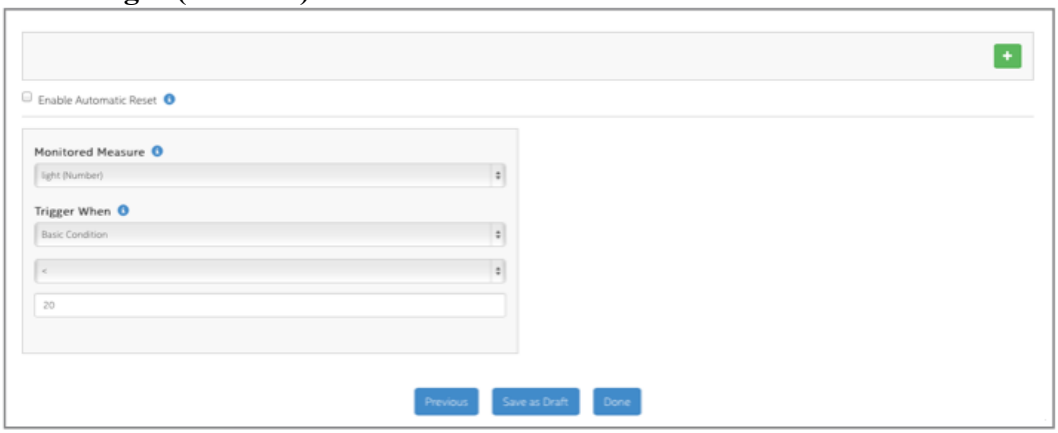


Figure 9 Rule Setup

21. Choose **Basic Condition** and then **<** for Trigger When.
22. Enter **20** in the blank.
23. Click **Done**.
24. Now you are ready to run cloud_light.
25. **$ ./cloud_light**
    a. What is printed on the screen is the light intensity.
    b. You will get an email when the reading goes below 20.
26. Cover the sensor with your hand for about one second. (Make sure the reading goes below 20)
27. Press **Ctrl-C** to quit.
28. You will receive an email from **IoT Analytics - Intel Corporation**.

a. Note: Be patient. This can take a few minutes.

29. Now, let's modify the rule setup to add the on-board LED control in addition to the email notification.
30. Before we modify it, we need to save a **complex command**.
    a. A **complex command** can be a series of actuation requests.
    b. In this demo, we are saving only one actuation request (send a parameter value of 1 for led_switch component via MQTT protocol) as a complex command.
31. Open the sidebar and then click **Control** menu.
32. Add an action to turn on the LED.
33. Click **save as complex command**.

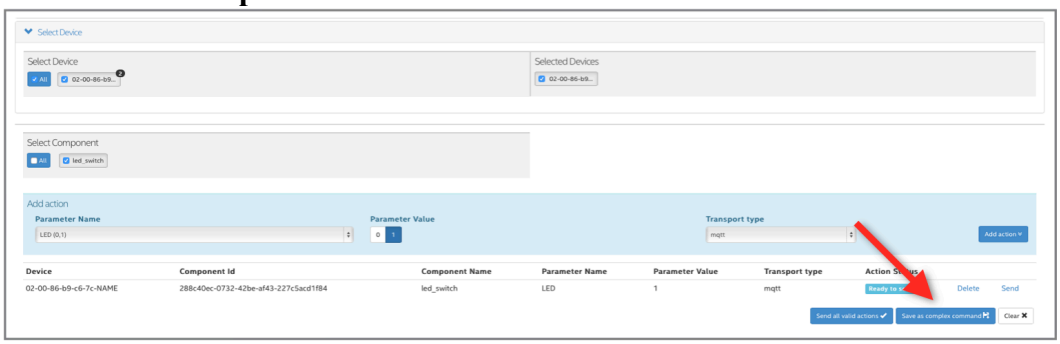


Figure 10 Complex Command Setup

34. Give a name to the complex command.

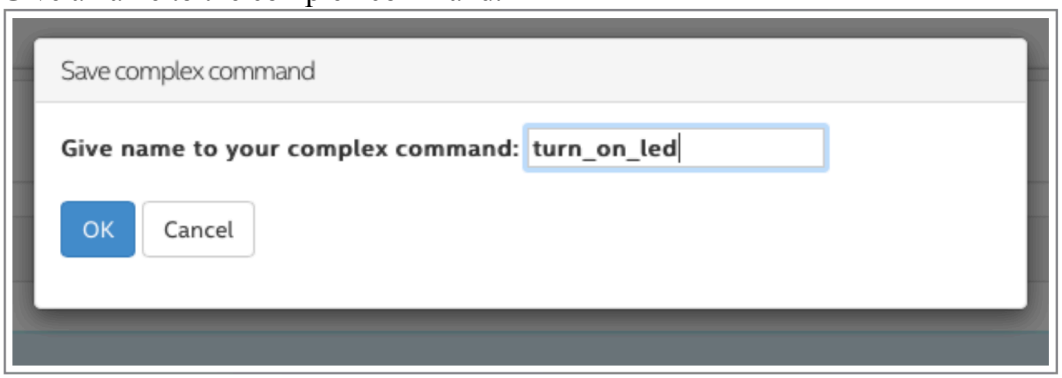


Figure 11 Complex Command Name

35. Now, you can see that the complex command is saved.

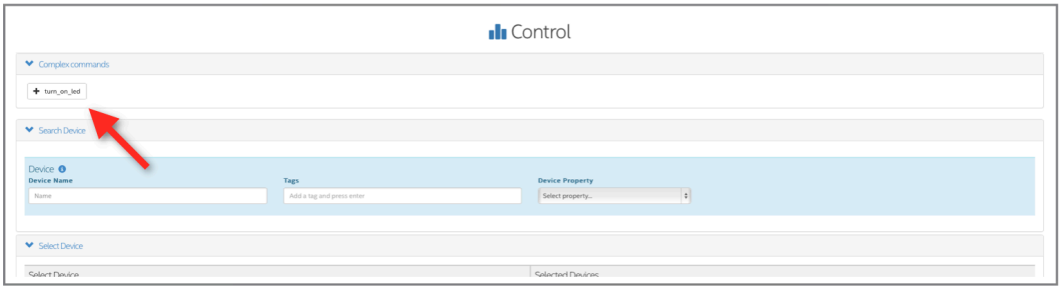


Figure 12 Saved Complex Command

36. Open the sidebar and then click **Rule** menu.
37. Click the rule we already have set up.
38. Choose **Actuation** for **Notification type**.
39. **Actuation notifications** is added to **Notifications to** sub-menu.

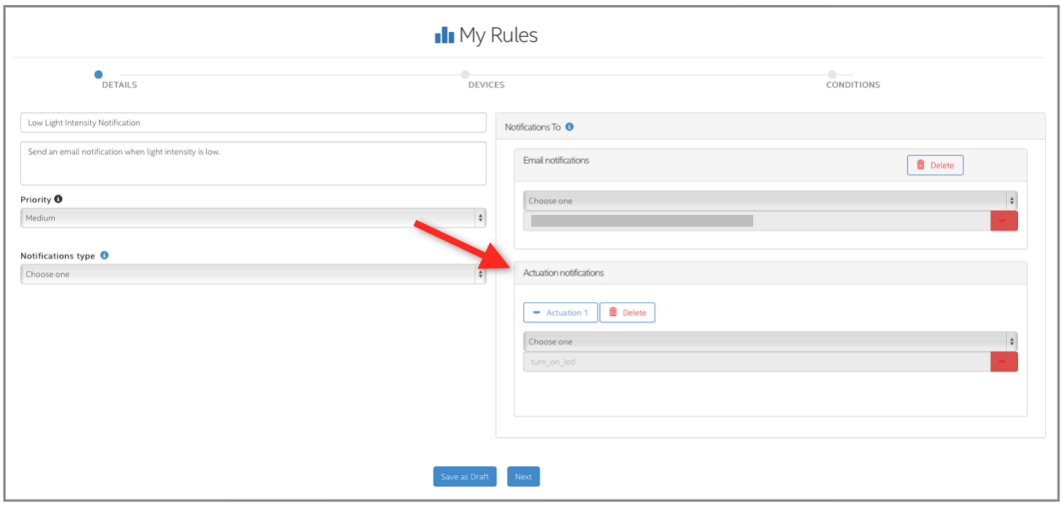
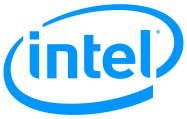
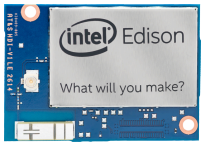


Figure 13 Actuation notifications

40. Click **Choose one** and choose the complex command we saved.
41. Click **Next**, **Next**, and then **Done** to save the change.
42. The rule is ready. Now, we need a program.
43. You already have a piece of C code in the repository you cloned in the previous section.
44. **$ cp ./tutorial7/rule.c ~/cloud_analytics/rule.c**
45. This code uses POSIX thread (pthread) for multithreading. You need to link the pthread library to compile.
    a. The main thread handles actuation.
    b. Another thread is created to send the light sensor observation data to the cloud.
    c. How this code works is not explained in this tutorial. However, if you understood how cloud_blink.c and cloud_light.c work, this code must be easy to understand as well.
46. **$ gcc -lmraa -lpthread -o rule rule.c**
47. **$ ./rule**
48. Cover the sensor with your hand for about one second. (Make sure the reading goes below 20)
49. You will receive a notification email and the on-board LED will turn on in a few minutes.
50. Press **Ctrl-C** to quit.





REST (Representational state transfer)

So far, we have been using the IoT Agent to interact with the Cloud. Part 1 briefly mentions that the IoT Agent sends data to the Cloud by making REST calls. If we write a program that makes REST calls to the Cloud, we can bypass the IoT Agent and directly interact with the Cloud. Another significant advantage of writing a custom program that makes REST calls is that we can set up a cloud interaction system that is not an Edison board. For instance, you can write an Android application that interacts with the Cloud.

What is REST anyway? REST is the software architectural style that is used in Web service development. Web services that use REST are called REST APIs, which allow users to send HTTP requests to perform CRUD (Create/Read/Update/Delete) operations.

We can use cURL to make HTTP requests to interact with the Cloud via REST API. cURL is a tool to transfer data to or from a server. Before we proceed, Windows users and Mac users must consider the following:

- **Windows users**: The demo presented in this tutorial assumes that you use Unix/Unix-like operating systems such as Linux and Mac OS X. You can install Cygwin, use a virtual machine to run Ubuntu, or simply use your Intel Edison.
- **Mac users**: You need development tools such as GCC installed on your Mac. By installing Xcode, you will have them installed. If you haven't installed Xcode already, please go to App Store and install Xcode.

1. On you own machine (or your Edison if you chose to use it instead), run the following command:
    a. **$ curl -X POST -H "Content-Type:application/json" https://dashboard.us.enableiot.com/v1/api/auth/token -d '{"username": "*your_username*", "password": "*your_password*"}'**
    b. Replace *your_username* with the email address you use to log into the Intel Cloud-based Analytics and *your_password* with your login password.
2. You will receive a response from the cloud that includes a JSON message with your user token, which contains user access information.

```
{"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJqdGkiOiIzMGRkMGRlYy01MGI4LTQ3NTgtOTU1NS02OWQxMzAzNjk
```

Figure 14 Response

3. This is an example of a POST request.
    a. There are other request types: GET, PUT, HEAD, and DELETE.
    b. You need to include "-X" option to specify the request type.
4. A request with the "-i" option includes the HTTP header in the response output.
5. Try with this option by entering "**curl -i -X POST -H "Content-Type:application/json" https://dashboard.us.enableiot.com/v1/api/auth/token -d '{"username": "*your_username*", "password": "*your_password*"}'**.
    a. Replace *your_username* with the email address you use to log into the Intel Cloud-based Analytics and *your_password* with your login password.

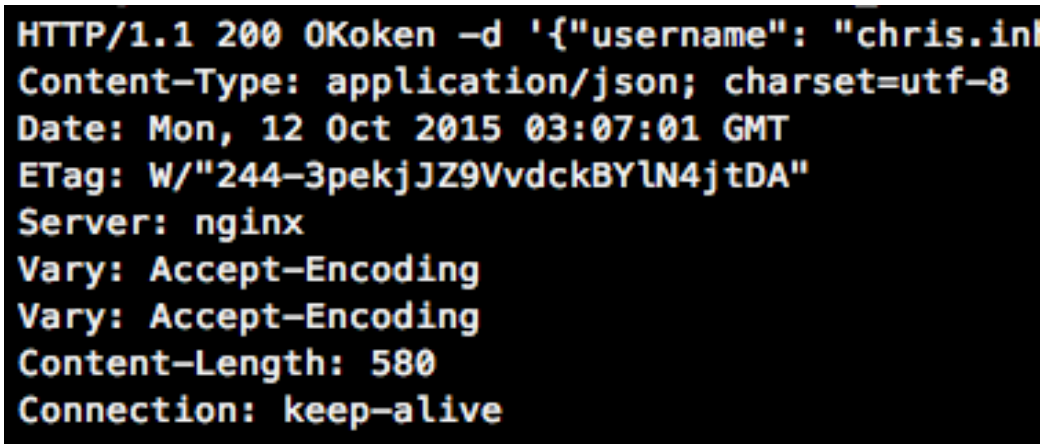


Figure 15 HTTP Header

6. You need to include "-H" option if there is a header to be passed to the server. In this case, the header is "**Content-Type:application/json**".
    a. You may include multiple headers as follows:
    curl -X POST -H "header1" –H "header2" url -d 'data'
7. What follows the header is the URL.
8. You need to include "-d" option if there is a data to send in a POST request. In this case, the data is a JSON message with your account login information.
9. You can find more options at http://curl.haxx.se/docs/manpage.html.

The HTTP request demo above is one of many operations you can perform with the REST API. You can learn about other operations at https://github.com/enableiot/iotkit-api/wiki/Api-Home. Other operations include very long headers, so we will not directly type cURL commands for them. Instead, we have a C program in the repository that makes system calls to run cURL commands. You may use this program to try other operations and find the cURL commands for them.

1. **$ cp -r ./tutorial7/rest_interface ~/cloud_analytics**
2. **$ cd rest_interface**
3. **$ gcc -o rest rest_main.c rest_api.c request.c**
4. **$ ./rest**
5. The program asks for the login information. Enter your Intel Cloud login information.

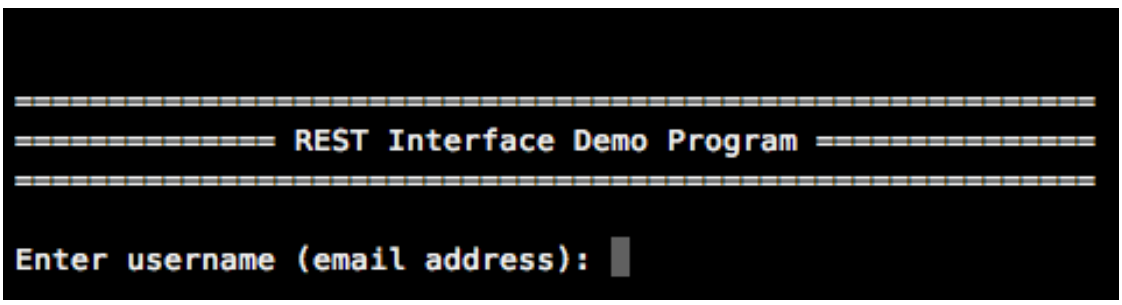


Figure 16 REST Interface Demo Program

6. The program displays operations you can make.

```
====================== Main Menu ========================
Choose a request type:
        1. Create device
        2. Generate activation code
        3. Activate device
        4. Create component
        5. List all devices
        7. Send observation
        8. Get observation
        9. Send actuation command
        10. Observatoin-triggered actuation demo
        11. Turn on/off output display (cURL commands, cloud responses)

Enter number to choose request type:
```

Figure 17 Main Menu

7. Enter "11" to display cURL commands and the server response.
8. Let's try some operations.
9. Enter "5" to list all registered devices and components.
10. You can see the cURL command and the server response. If you directly run the cURL command, you will receive the same response.
11. The output underneath the server response is the list of the registered devices and components.

The purpose of this program is to show the cURL commands and demonstrate HTTP requests. If you are to develop a cloud interaction system in pure C code, you can use libcurl library. We have a sample code in the repository under "rest_interface_libcurl" directory. Please refer to README file for more information. Also, you can write cloud interaction programs in Python or JavaScript. You can find sample codes at https://github.com/enableiot/iotkit-samples/tree/master/api/python and https://github.com/enableiot/iotkit-samples/tree/master/api/javascript.