FetchItGo Node-RED User Guide

Final

Dhiraj

6/19/2017

Contents

Contents 1
1 Revision History 1
2 Acronyms 1
3 Introduction 2
4 Architecture 2
5 Prerequisites 3
5.1 Hardware Requirement 3
5.1.1 Raspberry Pi 3
5.1.2 MicroSD Card for Raspberry Pi 3
5.1.3 Wi-Fi Router or Ethernet Switch 3
5.2 Software Requirement 3
5.2.1 FetchItGo Node 4
5.2.2 Californium CoAP Source code 4
5.3 Device Setup 4
5.3.1 Raspberry Pi setup 4
5.3.2 Configure FetchItGo for DIY mode 7
5.3.3 Californium CoAP server Executable 8
5.3.4 FetchItGo node 8
6 Execution 9
7 Limitation 10
8 References 10

1 Revision History

Version	Date	Author	Remarks
1.0	09-April-2017	Prasant	Initial
1.1	13-May-2017	Murali	Review
1.2	03-June-2017	Murali	Review
1.3	19-June-2017	Dhiraj	Review

2 Acronyms

FetchItGo – FetchItGo Device
Gateway – Gateway Device
Server – Tantiv4 Raspberry Pi Server
DIY – Do It Yourself
CoAP – Constraint Application Protocol

3 Introduction

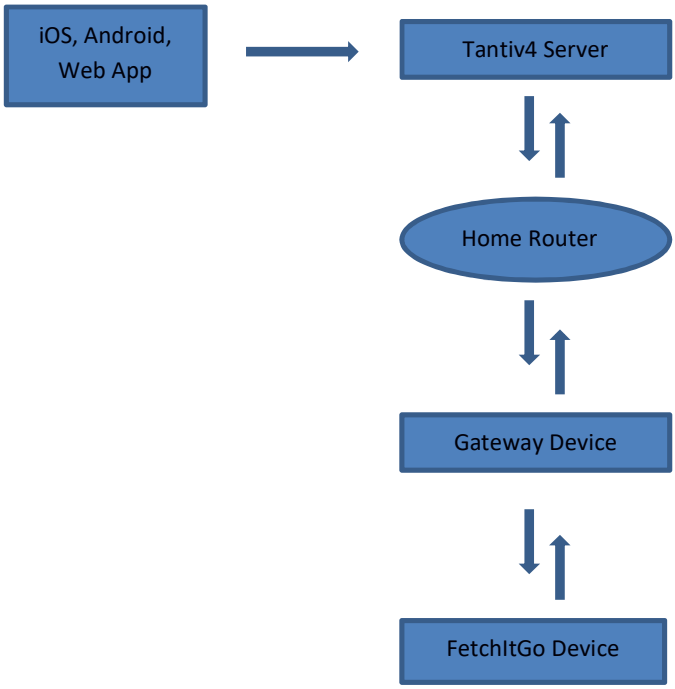
This document explains the requirement and steps required for handling requests from FetchItGo device in local network without any internet with the help of Raspberry Pi device.

Before starting the setup please download the package from github
https://github.com/Tantiv4/RaspberryPI/tree/master/FetchITGo%20Node

The package contains the java executable and supported files required to run a CoAP server with DTLS in Raspberry Pi device.

4 Architecture

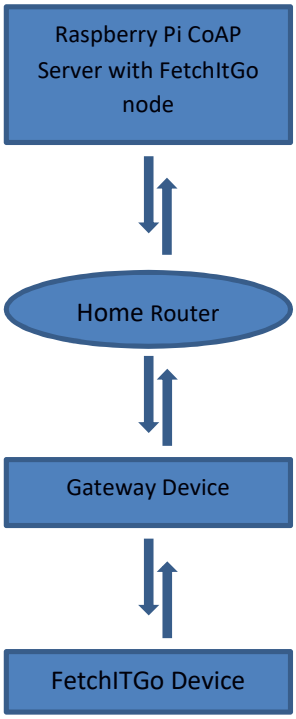
4.1 How to configure FetchItGo to use with Raspberry Pi?



FetchItGo device supports CoAP with DTLS protocol to communicate with any device in internet. To make Raspberry Pi device interact with FetchItGo, it needs a CoAP server running with DTLS protocol. The executable file for hosting a CoAP server in Raspberry Pi is provided as part of the package.
Our FetchItGo iOS, Android or Web App supports a mechanism to configure the FetchItGo device to interact with a CoAP server running in Raspberry Pi. Since FetchItGo is a battery powered device, the device needs to be paired with Raspberry Pi with the help of any FetchItGo Application. When the FetchItGo device wakes up, it will get the information from Tantiv4 server and configure itself to communicate with CoAP server running in Raspberry Pi. When a button is pressed in FetchItGo, corresponding device attached to Raspberry PI will be triggered based on the device configuration Raspberry Pi supports. Each button in FetchItGo can control one or multiple devices based on how Raspberry Pi is treating the request it gets from FetchItGo

4.2 How does FetchItGo communicate with Raspberry Pi Server?

FetchItGo device can't communicate with Raspberry Pi directly. It communicates through a Gateway device. The Gateway device and Raspberry Pi should be part of the same network. It doesn't require any internet connection to establish communication between FetchItGo and Raspberry Pi. The communication between Raspberry Pi and Gateway is without internet.
Gateway and Raspberry Pi should be connected to the home router. FetchItGo will receive commands from server through gateway via router and vice versa.



5 Prerequisites

5.1 Hardware Requirement

5.1.1 Raspberry Pi

Raspberry Pi device acts as a CoAP server with DTLS. The CoAP server running will get the request from FetchItGo and will forward it to FetchItGo node available in node red community.

5.1.2 MicroSD Card for Raspberry Pi

Raspberry Pi will require a micro SD card of size 16 GB or more since Ubuntu Mate needs 8GB memory space.

5.1.3 Wi-Fi Router or Ethernet Switch

The router acts as DHCP server to assign IP address to Raspberry Pi and Gateway. It forms a local network so that FetchItGo can control any devices connected to Raspberry Pi.

5.2 Software Requirement

First, we need to format the micro SD card (16 GB and more) using SD card formatter 4.0. Then download Ubuntu Mate (16.04 LTS) from the link- https://ubuntu-mate.org/download/ for Raspberry

Pi and write that Ubuntu Mate to micro SD card using Win32disk manager. Windows users can download it from https://sourceforge.net/projects/win32diskimager/

Mac users can use ApplePi Baker utility to create SD card with ubuntu mate OS.
https://www.tweaking4all.com/hardware/raspberry-pi/macosx-apple-pi-baker/

5.2.1 FetchItGo Node

A node is created in Node-RED repositories which gets the requests from Raspberry Pi and forwards to requested device attached.
Please follow the steps mentioned in the link below
https://www.npmjs.com/package/node-red-contrib-fetchitgo

5.2.2 Californium CoAP Source code

A CoAP server is instantiated in Raspberry Pi to get requests and send response to FetchItGo device. Only the jar file is provided not the entire source code. Californium code is used for this purpose.
For more information
https://github.com/eclipse/californium

5.3 Device Setup

5.3.1 Raspberry Pi setup

If your Raspberry Pi is already configured with Node.js, Node-Red, Java and Mosquitto, then the steps mentioned in this section can be skipped.

5.3.1.1 Install Node.js and npm

Ubuntu 16.04 makes it easy to install the latest long-term support (LTS) release of Node.js because it's included in the default repository.

```
$sudo apt-get install nodejs-legacy
```

This command installs Node.js v4.2.x LTS (long term support), which means the Node.js Foundation will continue to support this version for 30 months from its release date of October 12, 2015.

Note: It's important to install the -legacy version of the package because Node-RED's startup scripts expect your Node.js binary to be named node, but the standard package uses nodejs instead. This is due to a naming conflict with a pre-existing package.

Verify that the installation was successful by checking the version.

```
$ node -v
```

You'll get the version number of Node.js as: **v4.2.6**

Node Package Manager (npm) helps you install and manage Node.js software packages, and we'll use it to install Node-RED. Install npm using apt-get.

```
$ sudo apt-get install npm
```

To verify the install was successful, ask npm to print its version information:

```
$ npm -v
```

You'll get the version number of npm as**: 3.5.2**

If it prints a version number without error, we can continue to our next step, where we'll use npm to install Node-RED itself.

5.3.1.2 Install Node-RED

Use npm to install node-red and a helper utility called node-red-admin.

```
$ sudo npm install -g --unsafe-perm node-red node-red-admin
```

npm normally installs its packages into your current directory. Here, we use the -g flag to install packages 'globally' so they're placed in standard system locations such as /usr/local/bin. The --unsafe-perm flag helps us avoid some errors that can pop up when npm tries to compile native modules (modules written in a compiled language such as C or C++ vs. JavaScript).
After a bit of downloading and file shuffling, you'll be returned to the normal command line prompt. Let's test our install. First, we'll need to open a port on our firewall. Node-RED defaults to using port 1880, so let's allow that.

```
$ sudo ufw allow 1880
```

And now launch Node-RED itself. No sudo is necessary, as port 1880 is high enough to not require root privileges.

```
$ node-red
```

"Welcome to Node-RED" messages will be printed in the terminal. On your computer, point a web browser to port 1880 of the server. Type http://localhost:1880 OR http://127.0.0.1:1880 in your browser, then the main admin interface of Node-RED will load.

For more information on node red please check the link
https://nodered.org/

5.3.1.3 Install JDK and JRE

The easiest option for installing Java is using the version packaged with Ubuntu. Specifically, this will install OpenJDK 8, the latest and recommended version. First, update the package index.

```
$ sudo apt-get update
```

Install the Java Runtime Environment (JRE).

```
$ sudo apt-get install default-jre
```

There is another default Java installation called the JDK (Java Development Kit). The JDK is usually needed for compiling Java programs or if the software, that will use Java, specifically requires it.
Enter the following command in terminal

```
$ sudo apt-get install default-jdk
```

Install Oracle JDK if required

```
$ sudo add-apt-repository ppa:webupd8team/java


$ sudo apt-get update
```

To install JDK 8, use the following command:

```
$ sudo apt-get install oracle-java8-installer
```

Managing Java

If there are multiple Java installations in Raspberry Pi, then appropriate version needs to be selected. Run the following commands and select java appropriately

```
$ sudo update-alternatives --config java
```

There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-8-openjdk-armhf/jre/bin/java
Nothing to configure.

```
$ sudo update-alternatives --config command
```

update-alternatives: error- no alternatives for command

Setting the JAVA_HOME Environment Variable
Copy the path from your preferred installation and then type the below command it will go to another file. In that file at the end of the line add the following line
JAVA_HOME="/usr/lib/jvm/java-8-oracle"

```
$ sudo nano /etc/environment
```

Save and exit the file, and reload it.

```
$ echo $JAVA_HOME
```

***Note*:** Not showing anything, its blank

5.3.1.4 Install Mosquitto

Run the following commands in a terminal in Raspberry Pi to configure mosquito

```
$ curl -O http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key

$ sudo apt-key add mosquitto-repo.gpg.key

$ rm mosquitto-repo.gpg.key

$ cd /etc/apt/sources.list.d/

$ sudo curl -O http://repo.mosquitto.org/debian/mosquitto-wheezy.list

$ sudo apt-get update
```

Install the broker and command line clients:
mosquitto – MQTT broker
mosquitto-clients – command-line clients

```
$ sudo apt-get install mosquitto mosquitto-clients
```

Mosquitto broker starts straight away after download completes, so we will stop it to make some changes first:

```
$ sudo /etc/init.d/mosquitto stop
```

Now that the MQTT broker is installed on the Pi we will add some basic security. Create a config file:

```
$ cd /etc/mosquitto/conf.d/

$ sudo nano mosquitto.conf
```

Let's stop anonymous clients connecting to our broker by adding a few lines to your config file. To control client access to the broker, we also need to define valid client names and passwords. Add the following lines:

```
$ allow_anonymous false

$ password_file /etc/mosquitto/conf.d/passwd

$ require_certificate false
```

Save the file and exit your editor.

From the current /conf.d directory, create an empty password file:

```
$ sudo touch passwd
```

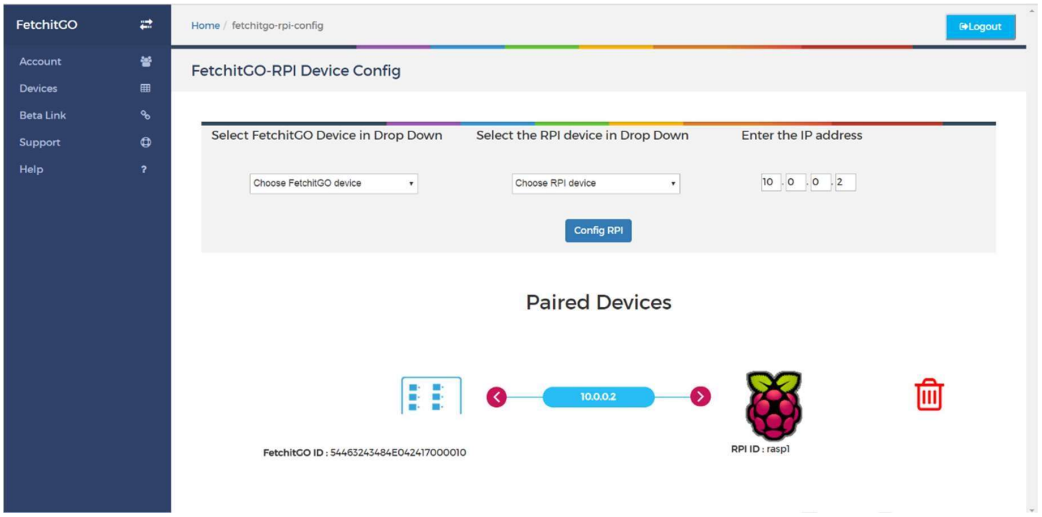
We will to use the mosquitto_passwd tool to create a password hash for user pi:

```
$ sudo mosquitto_passwd -c /etc/mosquitto/conf.d/passwd pi
```

You will be asked to enter your password twice. Enter the password you wish to use for the user you defined.

5.3.2 Configure FetchItGo for DIY mode

To support local network mode (DIY mode), FetchItGo device should know the IP address or hostname of the CoAP server running in Raspberry Pi. This mode will be initiated by the user through a provision provided in web, android and iOS application.
If user wants to check the status of Raspberry Pi, he can do that by controlling Raspberry Pi through SSH. But the remote system needs to be connected to any available LAN port in the router. Please make sure that raspberry pi and gateway are connected to same network.
The picture below gives an idea how FetchItGo application is used to configure FetchItGo device and Raspberry Pi.



Refer these below steps to pair a FetchItGo device with Raspberry Pi

1. After login in to app with valid credentials you can see an option called DEVICES at left hand side. If you click on that there will be two options like ADD FETCHIT GO DEVICE, ADD RPI DEVICE.
   Select an option based on which device you need to add (FetchItGo or Raspberry Pi).
2. Enter serial ID and device name (preferred name) for FetchItGo device and name or ID for Raspberry Pi.
3. If device is added in your account successfully, it will show in choose FetchItGo device drop down menu and is same for Raspberry Pi device. We need to enter IP address in which Raspberry Pi is connected. It will show FetchItGo device and Raspberry Pi got paired as shown in the above figure. (for example, if IP address 10.0.0.2 is assigned to Raspberry Pi then same IP address will be provided while configuring).
4. To disable communication between FetchItGo and Raspberry Pi, user should delete the configuration of Raspberry Pi in FetchItGo app by clicking the delete icon and then do a proper swipe action in FetchItGo device.

5.3.3 Californium CoAP server Executable

Execute the java executable file provided as part of the package. The payload for each client requests will have FetchItGo device id and button number which can be used by the user to differentiate which device to control.

5.3.4 FetchItGo node

Load the FetchItGo nodes in local directory.
Unzip the node-red-contrib-FetchItGo.
In the directory containing the node's package.json file, run below command in terminal

```
$ sudo npm link
```

Find FetchItGo node in the following path
/usr/local/lib/node-modules/

6 Execution

Start the FetchItGo server by typing below command in a terminal.

```
$ java -cp tantiv4-coap-raspberry-pi-0.0.1-SNAPSHOT.jar:lib/*:.
com.tantiv4.device.coap.T4CoAPRaspberryBiMain “SECURE” “/Add path of the key file”
```

1. Add the path at the end of the command where you have saved keys file (for example “/desktop/coap/keys”). The keys file will be available in our repository.

https://github.com/Tantiv4/RaspberryPI/blob/master/FetchITGo%20Node/T4RPiServer/coap-rpi.zip

2. Make sure that the above-mentioned command is running in the path where the folder is located. This directory contains jar file, library file, key files and californium properties files needed to execute the CoAP server.

The above command will host a secure CoAP server in Raspberry Pi with DTLS. To run a non-secure CoAP server run the following command.

```
$ java -cp tantiv4-coap-raspberry-pi-0.0.1-SNAPSHOT.jar:lib/*:.
com.tantiv4.device.coap.T4CoAPRaspberryBiMain
```

Note: FetchItGo device works only with secure CoAP server.

Check if CoAP server is up and running

If the setup is done and gateway can communicate to CoAP server running in Raspberry Pi, the server will receive health message requests from FetchItGo and gateway

Start the MQTT

Type below command in a terminal

```
$ mosquitto
```

Run the command line subscriber in another terminal

```
$ mosquitto_sub -v -t 'topic/device'
```

Start Node-red

```
$ node-red
```

Open the browser. Put in http://localhost:1880 OR http://127.0.0.1:1880, then the main admin interface of Node-RED will load.
Copy the FetchItGo.json file content.
CTRL+I in Node-RED instance window, and import the json file. Click the deploy button.

If you end up having the following issues
The workspace contains some unknown node types:

- coap request
- FetchItGO request
- FetchItGO mqtt out

To solve the above issues we need to install CoAP and FetchItGo nodes.

- To install this nodes in node-red terminal there is an option called manage palette.

- When we press that button, there will be two options at the right corner called nodes and install.
- Click that install button and search node-red-contrib-coap and install the nodes and it is same for all nodes.

Here you go. There will be some updates in MQTT subscriber window.
When a button is clicked on FetchItGo device, the request will be updated in subscriber window.

7 Limitation

8 References

1. Californium CoAP repository from Github.
2. Node red community and repository from Github.
3. FetchItGo user manual.
4. Raspberry Pi community.