FetchITGo Node User Guide

Original

Prasant Behera

4/9/2017

Contents

1 Revision History

Version	Date	Author	Remarks
1.0	09/04/2017	Prasant	Initial

2 Acronyms

FetchIT – FetchIT Device
Gateway – Gateway Device

3 Introduction

The 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 google drive
https://drive.google.com/drive/folders/0B4vGNja9mCnQLXN6N1ZWS0JVRnM
This package will be uploaded to github. The package contains FW_bin folder which won't be part of the package published for open community

4 Prerequisites

4.1 Hardware Requirement

4.1.1 Raspberry Pi

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

4.1.2 WiFi 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

4.2 Software Requirement

4.2.1 FetchitGo Node

A node is created in node red repositories which gets the requests from Raspberry Pi and forwards to requested device attached.
Please download the supported files from <provide github link here>

4.2.2 Californium CoAP Source code

A CoAP server is instantiated in Raspberry Pi to get requests and send response to Fetchitgo device. Californium code is used for this purpose.
Please download it from here
https://github.com/eclipse/californium

4.3 Device Setup

4.3.1 Raspberry Pi setup

If Raspberry Pi is already configured nodejs, node red, Java and mosquitto then the steps mentioned in this section can be skipped

4.3.1.1 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

The 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 see Node.js output its version number:
Output

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

Output

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.

4.3.1.2 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 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/

4.3.1.3 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 below 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, appropriate version needs to be selected. Run the following commands and select java appropriately

$ sudo update-alternatives --config java
$ sudo update-alternatives --config command

Setting the JAVA_HOME Environment Variable

Copy the path from your preferred installation and then open /etc/environment using nano or your favorite text editor.

$ sudo nano /etc/environment

Add the following line at the end

JAVA_HOME="/usr/lib/jvm/java-8-oracle"

Save and exit the file, and reload it.

echo $JAVA_HOME

4.3.1.4 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-repo.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 completed 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 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.

4.3.2 Router Setup

Setup a dhcp server in the router with the following information
IP address – 10.0.0.1
IP Pool Range – 10.0.0.2 to 10.0.0.255
DNS Server – 10.0.0.1
Instead of IP address, host name will be supported in future.
Please make sure Raspberry Pi is connected to 10.0.0.2 IP address. Connect gateway board to another available LAN port in the router.
If user wants to check the status of Raspberry Pi, he can use a screen or can control Raspberry Pi through SSH, but the laptop needs to be connected to any available LAN port in the router

4.3.3 Californium CoAP server Executable

Execute the java executable file provided as part of the package. If anyone needs to customize the server code, then the user needs to modify the java source code T4RPiServer.java and T4RPiClientThread.java files.
To compile the user needs to add these java files to cf-helloworld-server example folder. Please refer to Californium repository for how to compile
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.
Scandium is not supoorted

4.3.4 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/

4.3.5 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. This feature is currently not supported and more information will be shared when the feature is implemented. To test the concept Fetchitgo starts in DIY mode by a different firmware.
To use Fetchitgo device, the IP address of CoAP server is fixed at 10.0.0.2
The device id is fixed to t4rpifetchit001
Flash Fetchit_rpi.bin to Fetchitgo and Gateway_rpi.bin to gateway device and do a power cycle.

5 Execution

Start the FetchItGO server by typing below command in terminal.

$ java -jar cf-T4RPiServer-1.1.0-SNAPSHOT.jar

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 Fetchit 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 http://localhost:1880 in a browser
Copy the FetchItGO.json file content.
CTRLl+I in node red instance window , and import the json file. click the deploy button.
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.

6 Limitation

1. IP address of Raspberry Pi is fixed at 10.0.0.2
2. Fetchit and Gateway need to run the firmware which is part of the package.
3. DTLS is not supported, only Californium is tested. Scandium support will be provided in future
4. The package is not uploaded to github

7 References

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