Getting Started Guide for Ubuntu

Revision History

Revision number	Changes	Author	Revision date
0.5	Initial draft	Shamit Patel	8/28/2014
0.8	Tech Writer Review	Stephanie Liefeld	9/22/2014
0.9	Fixed Ubuntu 14.04 error	Shamit Patel	9/23/14

Table of Contents

Build the Services

1. SoftSensorManager

Once the source code is downloaded in your local specific folder, you may follow the steps to build and execute Soft Sensor Manager and its applications. In this context, we assume that the code was downloaded into 'oic' folder.

1. Download source code download

Once you download the codes, three main directories, resources, services, and utilities, are generated as follows;

```
~/oic/resource $_
~/oic/service$_
~/oic/utilities$_
```

Then, the path for Soft Sensor Manager is as following;

```
~/oic/ service/soft-sensor-manager$_
```

The SoftSensorManager directory includes following sub directories;

Directories	Description
/build	There are makefiles for different platform; Linux, Tizen, and Arduino.
/doc	SSM developer's guide and Getting started documents
/SampleApp	There are two types of sample applications; application for UI, and application for physical sensors. For UI application, there are SSMTesterApp in /linux, and /Tizen. For physical sensors, 1) Temperature and Humidity sensors, THSensorApp, in \linux and \arduino. In the two directories, in \linux and \arduino , there are two TemperaterHumiditySensor applications, *THSensorApp* and *THSensorApp1*, and they are for DiscomfortSoftSensor which aggregates two TemperaterHumiditySensors to calculate current discomfort index in the given room. 2) Trackee_Thing for *IndoorTrajectorySensor* in \linux and \arduino

/SDK	The SDK APIs for applications is located.
/SSMCore	The SSM service codes
/SoftSensorPlugin	The source codes for soft sensors can be located in this folder. Examples of soft sensors are *DiscomfortIndexSensor* and *IndoorTrajectorySensor*.

2. Refer readme files in each build directory for each module.

There are readme files in the build directories for each module (e.g. \SDK, \ SSMCore, \SampleApp). Please refer the files for specific setup.

3. Run make

3.1 Run make for SoftSensorManager and App in Ubuntu.

3.1.1 Before running make for SoftSensorManager & App in Ubuntu, resource should be built in advance. Please refer to 'Build the IoTivity project for Linux' in previous section.

3.1.2 If you type "make" at "soft-sensor-manager/build/linux", all packages will be pushed to "/soft-sensor-manager/build/linux/release". You can also found other packages in the folder.

```
~/oic/service/soft-sensor-manager/build/linux$ make
```

3.2 Run make for App in Arduino

3.2.1 If you want to build for Arduino, download Arduino IDE (Arduino 1.0.6) from following *url.* Extract 'arduino-1.0.6-linux32.tgz' and change folder name from 'arduino-1.0.6' to 'arduino' and then move to "/usr/share/".

url: http://arduino.cc/en/Main/Software

```
$ mv arduino-1.0.6 arduino
$ sudo cp -Rdp ./arduino /usr/share/
```

3.2.2 Download Time library (Time.zip, Click "The download") from following *url*. Unzip Time.zip and move them to /usr/share/arduino/libraries.

url: http://playground.arduino.cc/Code/Time

```
$ sudo cp -Rdp ./Time /usr/share/arduino/libraries/
```

3.2.3 Create file named 'local.properties' in "/oic/resource/csdk/" with the following definitions.

```
< local.properties >
ARDUINO_DIR = /usr/share/arduino
ARDUINO_TOOLS_DIR = $(ARDUINO_DIR)/hardware/tools/avr/bin
```

If you have a problem with compiling 'resource' when you compile arduino application, you may need to check 'local.properties' which is written in the readme file, '/oic/resource/csdk/README'.

3.2.4 Before running make for application, you need to build resource with arduino platform first. Please refer to below example.

```
~/oic/resource/csdk$ make PLATFORM=arduinomega ARDUINOWIFI=1
```

PLATFORM : arduinomega or arduinodue.

ARDUINOWIFI : 0 (Ethernet), 1(Wifi)

3.2.5 Now, you are ready to build sample arduino application. To build all sample applications for arduino, just do as below.

```
~/oic/service/soft-sensor-manager/ build/Arduino$ make PLATFORM=arduinomega
ARDUINOWIFI=1
```

If you want to build each sample application separately, go to build directory for sample application. Belowing is example for THSensor App.

```
~/oic/service/soft-sensor-manager/SampleApp/arduino/THSensorApp/build$ make
PLATFORM=arduinomega ARDUINOWIFI=1
```

3.2.6. To build and deploy the binary into the target hardware board, Aruino in this case, you need 'install' option.

Please refer to below example for THSensorApp ;

```
~/oic/service/soft-sensor-manager/SampleApp/arduino/THSensorApp/build$ make install
PLATFORM=arduinomega ARDUINOWIFI=1
```

Before ‘make install’, you need to check the file located at "/oic/service/soft-sensor-manager/SampleApp/arduino/THSensorApp/build/Makefile". Line 26, ARDUINO_PORT is the serial port path, which has to be aligned on your system.

4. Execute THSensorApp and SSMTesterApp

If you want to check how soft-sensor-manager is working, you can run simple applications - THSensorApp and SSMTesterApp.

5.1 To initiate THSensorApp, please enter as following;

```
~/oic/service/soft-sensor-manager/build/linux/release$ ./THSensorApp
~/oic/service/soft-sensor-manager/build/linux/release$ ./THSensorApp1
```

5.2 To initiate SSMTesterApp , please enter as following;

```
~/oic/service/soft-sensor-manager/build/linux/release$ ./SSMTesterApp
```

Note that the sequence of process initiations should be followed due to the process dependencies.

2.Protocol Plugin

Additional Libraries for Protocol Plugin:

Automake

Automake is a tool for automatically generating Makefile.in files compiliant with the GNU Coding Standards. This tool is used for compiling C-Pluff open source which used in Plug-in Manager.

$ sudo apt-get install automake

Libtool

GNU libtool is a generic library support script. This tool is used for compiling C-Pluff open source which used in Plug-in Manager.

$ sudo apt-get install libtool

gettext

GNU `gettext' utilities are a set of tools that provides a framework to help other GNU packages produce multi-lingual messages. This tool is used for compiling C-Pluff open source which used in Plug-in Manager.

$ sudo apt-get install gettext

Expat

Expat is a stream-oriented XML parser library. This library is used for compiling C-Pluff open source which used in Plug-in Manager.

$ sudo apt-get install expat

Building

Once the source code is downloaded into a specific folder, **oic** in this context, you may follow the steps to build and execute Protocol Plug-in Manager.

1. Make sure that the downloaded code structure is as followings;

Two directories for oic-resources; oic-resource and oic- utilities

```
~/oic/resource$_
```

```
~/oic/utilities$_
```

The path for Protocol Plugin is as following;

```
~/oic/service/protocol-plugin $_
```

The Protocol Plug-in directory includes following sub directories;

Directories	Description
/plugin-manager	Directory for Plug-in Manager
/plugins	Directory for Reference Plugins
/lib	Directory for Common Library
/sample-app	Directory for Iotivity Sample Application
/doc	Directory for Developers Document
/build	Directory for Building and Binary Release

2. Compiling C-Pluff library

Before building Protocol-Plugin Manager, C-Pluff library should be compiled as follows.

```
~/oic/service/protocol-plugin/lib/cpluff$ aclocal
~/oic/service/protocol-plugin/lib/cpluff$ autoconf
~/oic/service/protocol-plugin/lib/cpluff$ autoheader
~/oic/service/protocol-plugin/lib/cpluff$ automake
~/oic/service/protocol-plugin/lib/cpluff$ ./configure
~/oic/service/protocol-plugin/lib/cpluff$ make
```

3. Run make

By running make in the protocol-plugin path, protocol-plugin manager, all plugins and sample applications will be created.

```
~/oic/service/protocol-plugin/build/linux$make
```

3. Notification Manager

Once the source code is downloaded in your local specific folder, you may follow the steps to build and execute Notification Manager and its applications. In this context, we assume that the code was downloaded into 'oic' folder.

1. Download source code download

From the url, you can download NM source code;

https://www.iotivity.org/downloads

Once you download the codes, and Make sure that the downloaded code structure is as follows;

Two directories for oic-resources; oic-resource and oic- utilities

```
~/oic/resource$_
```

```
~/oic/utilities$_
```

The path for Notification Manager is as following;

```
~/oic/ service/notification-manager
```

The notification-manager directory includes following sub directories;

Directories	Description
/build	There are makefiles for different platform; Linux, Tizen.
/SampleApp	There are two types of sample applications; application for Provider, and application for Consumer.

	For Provider, there is the SampleProviderApp which is a TemperaterHumiditySensor in /linux, and /Tizen. For Consumer, there is the SampleConsumer which is a ClientApp in /linux, and /Tizen.
/NotificationManager	The NotificationManager's sources for Hosting is located .

2. Modify the the ROOT_DIR and BOOST path in the environment file

To build, there are two setting attributes for your system environment, ROOT_DIR and BOOST_BASE, in the environment file, *environment.mk*. The file is in each platform in the directory, "oic\build".

```
# root path of each PC.
ROOT_DIR=/home/iotivity/Desktop/Project/Iotivity-Candidate

# boost folder path.
BOOST_BASE=/home/iotivity/Desktop/boost_1_56_0.
.
.
```

3. Refer readme files in each build directory for each module.

There are readme files in the build directories for each module (e.g. \NotificationManager, \SampleApp) . Please refer the files for specific setup.

4. Run make

```
~/oic/ service/notification-manager/build/linux/makefile$ make
```

5. Execute SampleConsumerApp, SampleProvider and NotificationManager

To initiate SampleConsumerApp, please enter as following;

```
~/oic/ service/notification-manager/build/linux/release$ ./SampleConsumer
```

To initiate SampleProviderApp, please enter as following;

```
~/oic/ service/notification-manager/build/linux/release$ ./SampleProvider
```

To initiate NotificationManager, please enter as following;

```
~/oic/ service/notification-manager/build/linux/release$ ./NotificationManager
```

Note that the sequence of process initiations should be followed due to the process dependencies.

4. Things Manager

Once the source code is downloaded in your local specific folder, you may follow the steps to build and execute Things Manager and its applications. In this context, we assume that the code was downloaded into 'oic' folder.

1: Download source code

From the url, you can download Things Manager source code;

https://www.iotivity.org/downloads

Once you download the codes, and Make sure that the downloaded code structure is as follows;

Four directories for oic; extlib, resource, service, and tools.

```
~/oic/extlib$_
```

```
~/oic/resource$_
```

```
~/oic/service$_
```

```
~/oic/tools$_
```

The path for Things Manager is as following;

```
~/oic/service/things-manager/$_
```

The things-manager directory includes following sub directories;

Directories	Description
/sdk	The SDK APIs for applications is located. The main functionality of this SDK is to provide developer-friendly APIs of Things manager component to application developers.
/sampleapp	It is the sample application on Ubuntu. Basically, the input and output of application on Ubuntu are displayed in the console.
/build	Whole library files and binary files would be made in this folder

2: Build

```
~/oic/service/things-manager/build/linux$ make
```

5. Control Manager

Building Control Manager, Controllee, RESTFramework and sample applications in Ubuntu

This guide will help you setup your development environment, build the Control Manager, Controllee and RESTframework components with IoTvitiy stack and sample applications. Once successfully build the stack, the Developer's guides for each of these components will be a good reference to use Control Manager Service

This guide will provide instructions on how to build the stack. Instructions provided in this guide are tested against Ubuntu 14.10, however, Ubuntu version 12.0.4 and above are supported.

Build tools and libraries

Open the terminal window and follow the instructions below to install all the necessary tools and libraries in order to build IoTvity project.

Git

Git is a source code management software. IoTvitiy is set as a git project. Git is mandatory in order to get access to the IoTvitiy source code. Use the following command to download and install git.

```
$ sudo apt-get install git-core
```

ssh

Secure Shell is required to connect to GIT repository in order to checkout the IoTivity source code. SSH is typically part of the base operating system and should be include. If, for any reason, it is not available, it can be installed by running the following command in your terminal window.

```
$ sudo apt-get install ssh
```

G++

G++ is required in order to build the IoTvity stack. Download and install G++ by running following command in your terminal window

```
$ sudo apt-get install build-essential g++
```

Boost Version 1.55

Boost c++ library is required in order to build the IoTvity stack. Download and install Boost libraries by running the following command in your terminal window

```
$ sudo apt-get install libboost-all-dev
```

Doxygen

Doxygen is a documentation generation tool used to generate API Documentation for the IoTvity project. Download and install doxygen by running following command in your terminal window.

```
$ sudo apt-get install doxygen
```

Checking out the source code

Gerrit is a web-based code review tool built on top of the git version control system. Gerrit's main features are side-by-side difference viewing and inline commenting, streamlining code review. Gerrit allows authorized contributors to submit changes to the git repository after reviews are done. Contributors can have code reviewed with little effort, and get their changes quickly through the system.

Building Control Manager and its samples

Once the source code is downloaded in your local specific folder, you may follow the steps to build and execute Control Manager and its applications. In this context, we assume that the code was downloaded into 'oic' folder.

1. Download source code

From the URL, you can download CM source code;
https://www.iotivity.org/downloads

Once you download the codes, and Make sure that the downloaded code structure is as follows;
Two directories for oic-resources; oic-resource and oic-utilities

```
~/oic/oic-resource$_
```

```
~/oic/oic-utilities$_
```

The path for Control Manager is as following;

```
~/oic/oic-resource/service/control-manager$_
```

The control-manager directory includes following sub directories;

Directories	Description
/controlmanager	Control Manager source code
/controllee	Controlee source code
/RESTframework	REST Framework source code
/opensource	Open source components, 1. Jsoncpp 2. Sqlite3
/samples	There are 3 different samples. 1. controller-client 2. controllee-server 3. REST-client
/docs	Developer's guide and Getting started documents
/makefiles	Internal makefiles and build configuration files
/build_common	Platform specific Scons build configuration scripts
Makefile	Top level makefile to build all components
SConstruct	Scons based main build file
README.txt	README file

2. Modify the BOOST path in the Makefile

To build, set the BOOST_PATH environment variable in Makefile.

```
# boost folder path.
BOOST_PATH=/home/iotivity/Desktop/boost_1_56_0.
.
.
```

3. Refer README file

There is README file in top level control-manager folder

4. Run make

```
~/oic/oic-resource/service/control-manager$ make
```

5. Execute controllee-server, controller-client and REST-client

To run controllee-server, please enter the following command;

```
~/oic/oic-resource/service/control-manager/samples/controllee-
server/{release/debug}$./controllee-server
```

To run controller-client service, please enter the following command;

```
  ~/oic/oic-resource/service/control-manager/samples/controllee-
server/{release/debug}$./controller-client
```

To run controller-client service, please enter the following command;

```
  ~/oic/oic-resource/service/control-manager/samples/REST-client/{release/debug}$./REST-
client
```

Once the sample starts running send any http request using 'cURL or Python scripts.
Note: 'REST-client' runs at port 3115.

Build the API reference documentation

To build the API reference documentation:

a. Navigate to oic-resource/docs folder using the terminal window.
b. Run the following command:

```
$ doxygen
```

This command builds the API reference documentation in the output directory.

The output directory for this command is oic-resource/docs/html/index.html.