IoTivity: 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

Introduction

This guide provides instructions and resources to help developers set up the development environment, build the IoTivitiy stack and build sample applications on Ubuntu for a Linux target. Developers should also read the IoTivity Programmer's Guide before starting development to better understand IoTivity architecture and use cases.

Tools and libraries

The following tools and libraries are necessary to build IoTivity code for Linux. The commands and instructions provided in this section are specifically for Ubuntu LTS 12.04.

Open the terminal window use the following instructions to install all the necessary tools and libraries to build an IoTivity project.

Ubuntu LTS 12.04

Ubuntu LTS version 12.04 is the supported OS for building the IoTvity stack. The instructions may be different for other versions of Ubuntu and Linux.

Git

Git is a source code management software. Git is necessary to gain access to the IoTivitiy 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 the git repository to check out the IoTivity source code. Secure Shell is typically part of the base operating system and should be included. 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++ version 4.6.1

G++ is required to build the IoTivity 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 necessary to build the IoTivity stack. Download and install Boost libraries by following steps below.

Step 1: download boost_1_55_0.tar.gz by clicking here [http://sourceforge.net/projects/boost/files/boost/1.55.0/boost_1_55_0.tar.gz/download]

Step 2: in your terminal window, run the following command

```
$ tar xzvf boost_1_55_0.tar.gz
```

Step 3: Navigate to the boost directory

```
$ cd boost_1_55_0/
```

Step 4: add libraries by running following command in the terminal window

```
$ ./bootstrap.sh  --with-libraries=system,filesystem,date_time,thread,regex,log,iostreams  --prefix=/usr/local
```

Step 5: Install by running following command

```
$ ./b2 install
$ sudo sh –c ‘echo “/usr/local/lib” >> /etc/ld.so.conf.d/local.conf’
$ sudo ldconfig
```

Doxygen

Doxygen is a documentation generation tool used to generate API documentation for the IoTivity 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.

The following five steps describe how to check out the source code on the development machine.

Note: skip Step 1 to use existing ssh keys.

Step 1: Create ssh keys

On the terminal, type the following (replace "your name <your_email_address>" with your name and email address):

```
$ ssh-keygen -t rsa -C "your name
<your_email_address_here>"
```

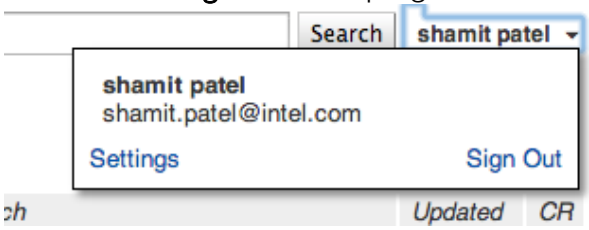
For example John Doe with an email address john.doe@example.com would type:

```
$ ssh-keygen -t rsa -C "John Doe john.doe@example.com"
```

After pressing the Enter key at several prompts, an ssh key-pair will be created at ~/.ssh/id_rsa.pub.

Step 2: Upload and register an ssh public key

a. Log in to **OIC Gerrit.**
b. Click on **Settings** on the top right side as shown here:



c.
d. Click on **SSH Public Keys** and add key.
e. Open **~/.ssh/id_rsa.pub**, copy the content, and paste the content in the **"Add SSH Public Key"** window.
f. Click **Add.**

Step 3: Setting up ssh

a. Open **~/.ssh/config** in a text editor.
b. Add the following lines:

```
Host oic
     Hostname oic-review.01.org
     IdentityFile ~/.ssh/id_rsa
     User <registered username on 01.org>
     Port 29418
```

c. To connect behind the proxy, add the following line after **IdentityFile ~/.ssh/id_rsa** with the appropriate proxy address and port:

```
ProxyCommand nc -X5 -x <proxy-address>:<port> %h %p
```

Step 4: Verify your ssh connection

Execute the following command in the terminal window:

```
$ ssh oic
```

Upon successful connection, the following message should appear indicating proper ssh and configuration connection.

```
****    Welcome to Gerrit Code Review    ****
```

If the connection is not established, check for the proxy and use the proxy settings described in Step 3.

Step 5: Cloning the project source

To build the IoTivity resource stack:

a. Using your terminal window, browse to the directory where code will be checked out.
b.] Execute the following command in the terminal window to clone the oic-resource and oic-utilities repositories:

```
$ git clone oic:oic-resource
$git clone oic:oic-utilities
```

This command clones the repository in your current working directory.

Note: the “oic-resource” and “oic-utilities” must be checked out in the same directory to successfully build the project without any modifications to the build scripts and make files.

Build the IoTivity project for Linux

To build the whole project, including the core, C SDK, C++ SDK and samples:

a. Navigate to the root of the oic-resource directory using the terminal window.
b. Execute the make all command from the oic-resource directory in the terminal window:

```
$ make all
```

This command builds all the components of the IoTivity project. Component outputs are described in the section below.

Build the C SDK

To build only the C SDK:

a. Navigate to the root directory of the oic-resource folder using the terminal window.
b. Executing following command in the terminal.:

```
$ make csdk
```

This command builds the core and the C SDK for the IoTivity project.
The output directory for this command is oic-resource/csdk/release/liboctbstack.a.

Build the C++ SDK

The C++ SDK is a wrapper around the C.
Note: liboctbstack.a must be built prior to building the C++ SDK.

To build the C++ SDK:

From the root of the oic-resource folder, execute the following command in the terminal window.

```
$ make OCLib.a
```

This command, assuming that liboctbstack.a is built, will only build the OCLib.a, which is a static library for the IoTivity C++ SDK.

The output directory for this command is oic-resource/release/OCLib.a.

Build the C++ samples

To build the C++ sample applications:

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

```
$ make examples
```

This command, assuming OCLib.a is built, will only build the sample applications in the examples directory under the oic-resource directory.

The output directory for this command is oic-resource/examples/release/.

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

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.

4.1 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
```

4.2 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.

5. Build with SCons.

If you want to use Scons to build soft-sensor-manager, build the source code and execute sample applications as following steps.

5.1 After downloading the source codes, type "scons" at project root directory "oic/"

```
~/oic$ scons
```

5.2 To execute Soft Sensor Manager and its applications, some shared libraries must be exported

5.2.1 Go to "oic/out/<target_os>/<target_arch>/release", and check if there are following .so files

liboc.so

liboctbstack.so

liboc_logger.so

5.2.2 To export the shared libraries, please enter as following;

liboc.so

liboctbstack.so

liboc_logger.so

```
~/oic/out/<target_os>/<target_arch>/release$ export
LD_LIBRARY_PATH=~/oic/out/<target_os>/<target_arch>/release
```

5.3 Generated outputs from building soft-sensor-manager will be in "oic/out/<target_os>/<target_arch>/release/service/soft-sensor-manager".

5.4 To initiate THSensorApp, please enter as following;

```
~/oic/out/<target_os>/<target_arch>/release/service/soft-sensor-manager$ ./THSensorApp
~/oic/out/<target_os>/<target_arch>/release/service/soft-sensor-manager$ ./THSensorApp1
```

5.5 To initiate SSMTesterApp, please enter as following;

```
~/oic/out/<target_os>/<target_arch>/release/service/soft-sensor-manager$ ./SSMTesterApp
```

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
```

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.