

OIC Conformance Test Tool

USER MANUAL

TABLE OF CONTENTS

	Page#
1. GENERAL INFORMATION -----	2
1.1 Introduction	2
1.2 Organization of the Manual	2
2. PREREQUISITE -----	4
2.1 Platform	4
2.2 Java	4
2.3 Gradle	4
2.4 Jython	4
2.5 Robot Framework	4
2.6 Tshark	4
2.7 Eclipse Luna	5
3. BUILD -----	7
3.1 IoTivity Build (For Conformance Simulator)	7
3.2 GUI Build	7
3.2 CTT Library and Conformance Simulator Build	8
3.3 Conformance Simulator (standalone) Build	8
4. RUN -----	11
4.1 Prerequisite	11
4.2 Using GUI	11
4.3 Using CLI	16
4.4 Run Conformance Simulator	18

1. GENERAL INFORMATION

1. GENERAL INFORMATION

General information section introduces Conformance Test Tool and purpose for which it is intended.

1.1. Introduction

Open Interconnect Consortium (OIC) Conformance Test Tool (CTT) is a tool used in IoTivity and OIC product to ensure that an IoTivity or OIC product is compliant with OIC standard.

1.2. Organization of the Manual

The user manual consists of four sections: General Information, Prerequisite, Build, and Run.

General Information section explains in general terms the system and the purpose for which it is intended.

Prerequisite section describes installation process of required software and tools that is essential to use Conformance Test Tool.

Build section provides the build process of Conformance Test Tool and DUT Simulator.

Run section explains the execution process of DUT Simulator and Conformance Test Tool using CLI and GUI.

2. PREREQUISITE

2. PREREQUISITE

2.1. Platform:

Operating System: Ubuntu 12.04 or higher

Architecture: 32 bit

2.2. Java:

Java 1.8 or higher (To install jdk 1.8, run the following commands)

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
```

2.3. Gradle:

Gradle 2.3 or higher (To install Gradle on linux, run the following commands)

```
$ sudo add-apt-repository ppa:cwchien/gradle
$ sudo apt-get update
$ sudo apt-get install gradle
```

2.4. Jython:

Jython 2.7 or higher (To install Jython, follow these steps)

- Go to the following Link and Download Jython 2.7.0 installer
<http://www.jython.org/downloads.html>
- After download is complete, run the jar file using java-8
- Follow the installation procedure.

2.5. Robot Framework:

To install Robot Framework, follow these steps

- Go to the following link and download zip
<https://github.com/robotframework/robotframework>
- Extract the zip file
- Go to the root folder of robot framework, open terminal and run:

```
$ jython setup.py install
```

2.6. Tshark:

To install tshark, follow below commands

```
$ sudo apt-get install tshark
$ sudo chmod 4711 `which dumpcap`
```

2.7. Eclipse Luna:

- Download Eclipse Luna RCP from:
<http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/luna/SR2/eclipse-rcp-luna-SR2-linux-gtk.tar.gz>
- Extract it
- Start Eclipse
- Click on Help->Install New Software
- Under “work with:”, select '*Luna –http://download.eclipse.org/releases/luna*'
- Wait for the packages to load, then select everything under 'Linux Tools'
- Click “Next” and proceed with Installation.
- Again click on Help->Install New Software.
- Under “work with:”, click on Add
- Provide Name as 'RobotFramework', and Location as
http://sourceforge.net/projects/robotide/files/stable/
- Click OK
- Now from package list, select '*Robot Framework Eclipse IDE*'
- Click on Next and proceed with installation.
- Restart Eclipse

3. BUILD

3. BUILD

3.1. IoTivity Build (For Conformance Simulator)

- Go to the top (root) directory of 'iotivity' project.
- Follow the prerequisite steps described for IoTivity project.
- Run the below command:

Build with any of the following:

```
$ scons TARGET_OS=linux (non-secured resource):  
$ scons TARGET_OS=linux SECURED=1 (secured resource)
```

(Note: C sdk requires tinycbor. Please follow the instruction in the build message to install tinycbor).

3.2. GUI Build

Open Eclipse Luna RCP

- Import Project from git '*conformance-test-tool/res/ConformanceTestTool*'
- Build the project
- Now, Right click on the project, and then click Export.
- Under 'Plug-in Development', select Eclipse product, click on Next.
- As Root Directory, type 'ConformanceTestTool'

As Destination Directory, show '*conformance-test-tool/res/ConformanceTestTool/bin*'

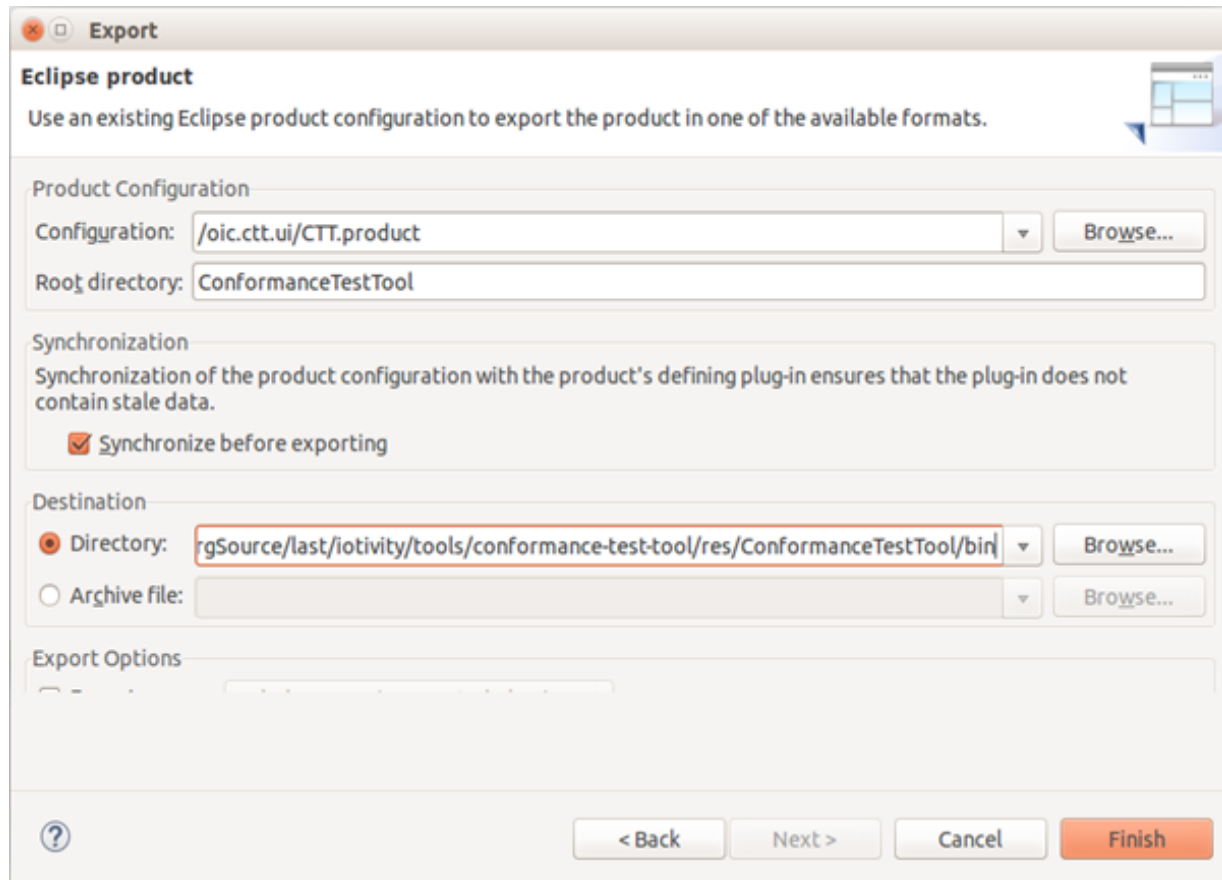


Figure 3.2: Export Settings Window (Root & Destination Directory must be same as described)

- Click on Finish.

3.3. CTT Library and Conformance Simulator Build

- Go to the root folder (*iotivity/tools/conformance-test-tool*)
- From the terminal, run:


```
$ sudo chmod 777 build.sh
$ ./build.sh
```

3.4. Conformance Simulator (standalone) Build

Conformance Simulator is an OIC client-server simulator which supports the basic client and server features mentioned in the OIC Core Spec. It is based upon IoTivity implementation of OIC specification. The server part supports discoverable, non-discoverable, observable, non-observable, secured and non-secured resources. Both the server and client support all CRUDN operations.

Resource, Platform, Device and Collection discovery is supported for the client. The client supports both CON & NON type messaging.

If only Conformance Simulator build is required, then follow the bellow steps:

- Perform steps described in 3.1
- Go to the top (root) directory of 'ConformanceSimulator' project.
- Run the below command to build Conformance Simulator
\$ `scons`

4. RUN

4. RUN

4.1. Prerequisite

- Create a description of device/resource representation data structure in json format. An example is provided in the same directory of this documentation. Name the file as 'DUTDescriptor.json'
- Copy the file to `<iotivity_root>/tools/conformance-test-tool/bin/linux/ConformanceTestTool/libs/` folder. (When using the Conformance Simulator as DUT, no need to make any DUTDescriptor.json)
- If DUT device is not available, then run Conformance Simulator by following section-4.4

4.2. Using GUI

- a) Start the DUT device or Conformance Simulator
- b) Go to the root folder of Conformance Test Tool, Then Run:

```
$ cd bin/linux/ConformanceTestTool
```
- c) To run the GUI App, execute the following:

```
$ ./CTT
```
- d) The GUI will start like this:

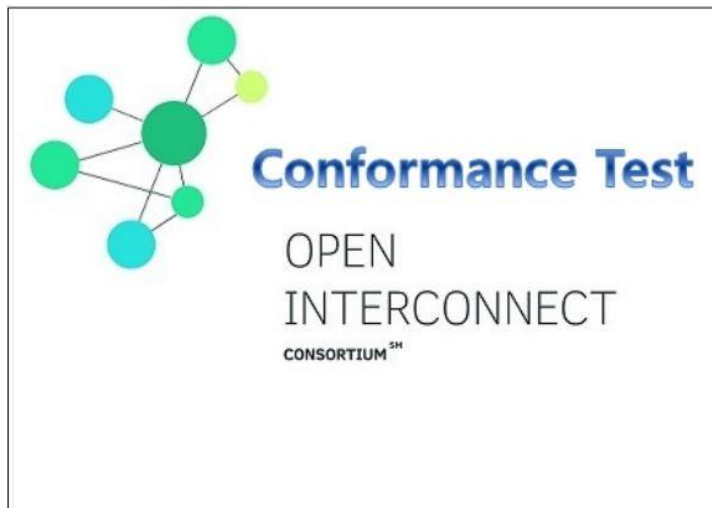


Figure 4.2.1: Splash Screen

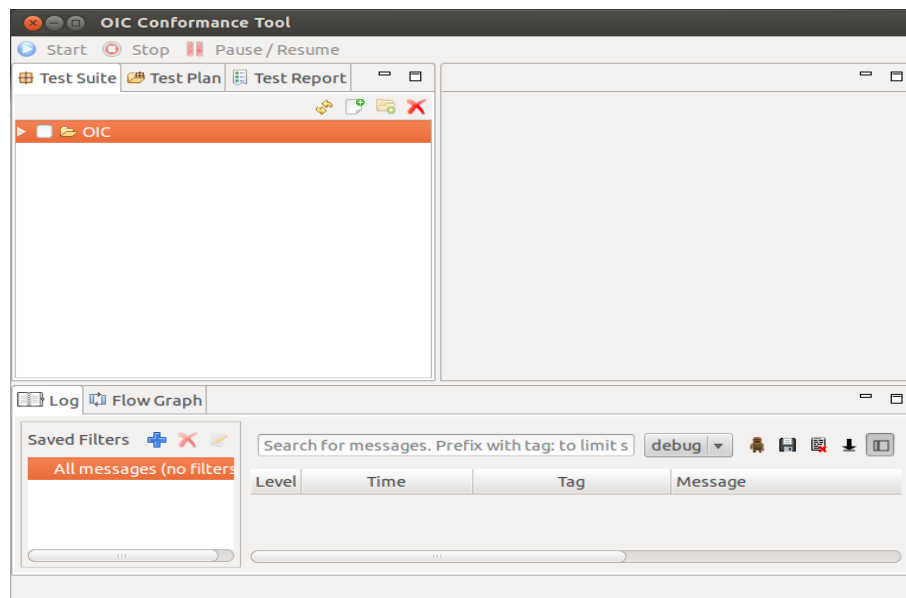


Figure 4.2.2: Conformance Tool Graphical User Interface (GUI)

e) How to make a Test Plan

- Select your desired test cases in the **Test Suite** tab. Click Create Test Plan

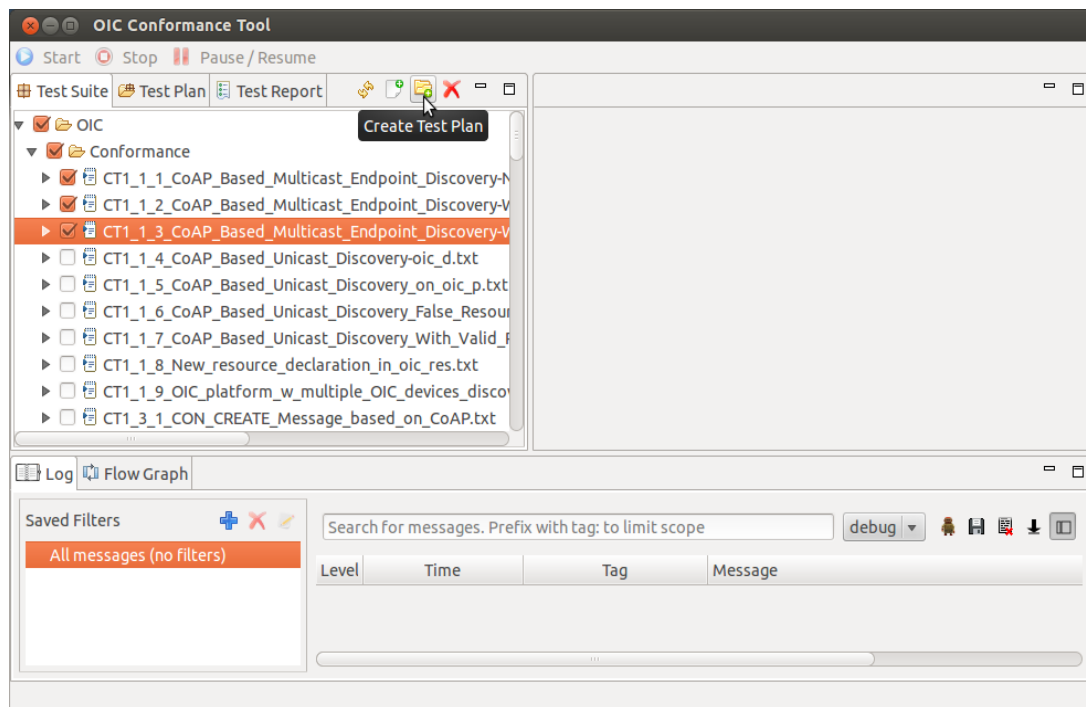


Figure 4.2.3(a): Conformance Tool Create Test Plan

- A **Create Test Plan** dialogue box will appear. Enter a name for it and click OK.

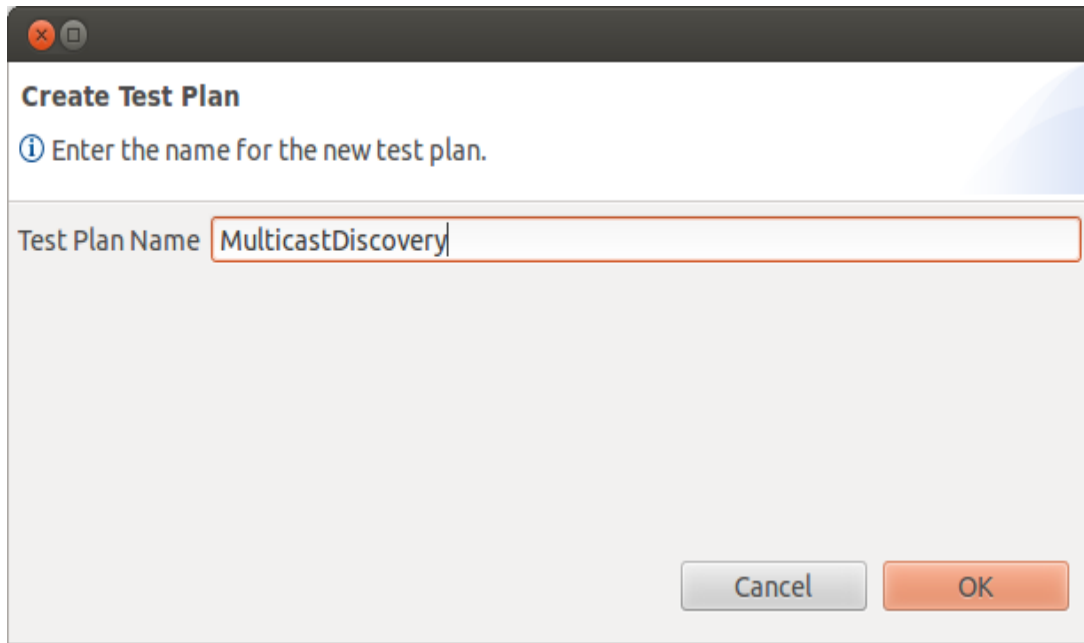


Figure 4.2.3(b): Conformance Tool Create Test Plan

- A test plan will be created. The created test plan will be visible in the **Test Plan** tab.

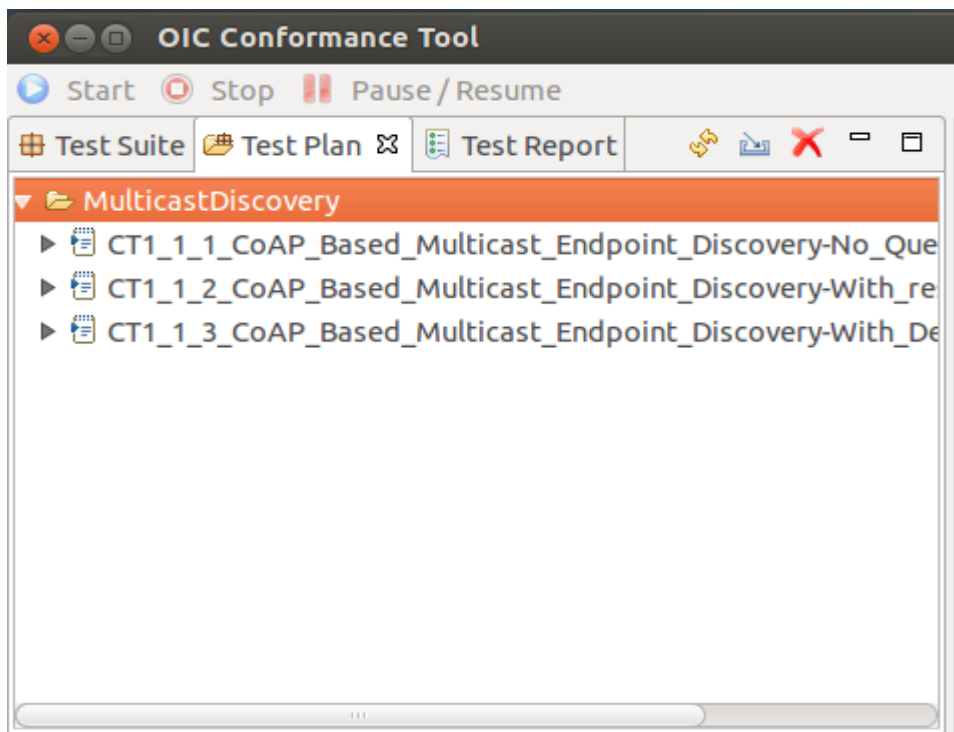


Figure 4.2.4: Conformance Tool Show Test Plan

- Double click on the test plan to select it.

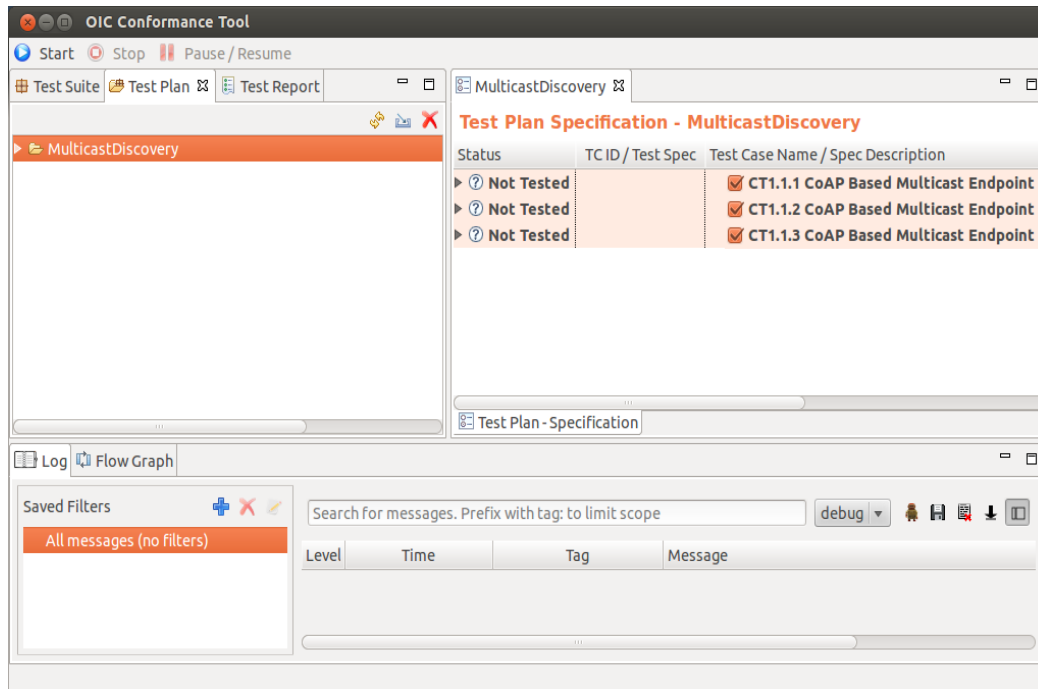


Figure 4.2.5: Conformance Tool Select Test Plan

f) How to run the Test plan

- Once the test plan is selected, click the **Start** button to start the test.

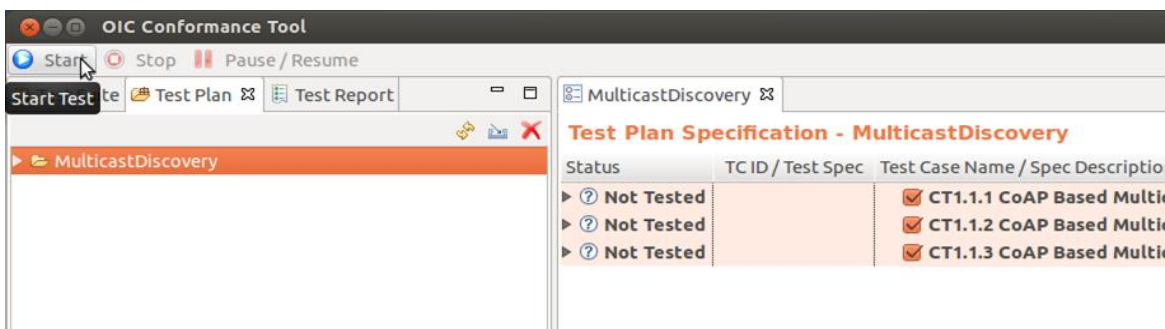


Figure 4.2.6: Conformance Tool Start Test Plan

- When the test is running, you can **Stop** or **Pause/Resume** the execution. The Flow Graph, Progress & Log windows will show real-time test execution information

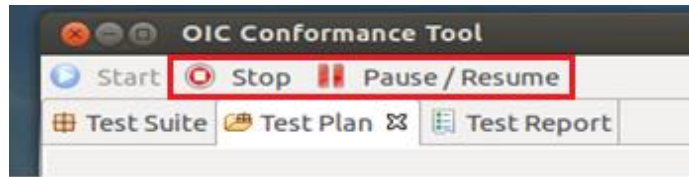


Figure 4.2.7: Conformance Tool Stop/Pause Test Plan

g) How to view & interpret Reports

- The **Log** tab will show all the detailed logs generated by the test cases.

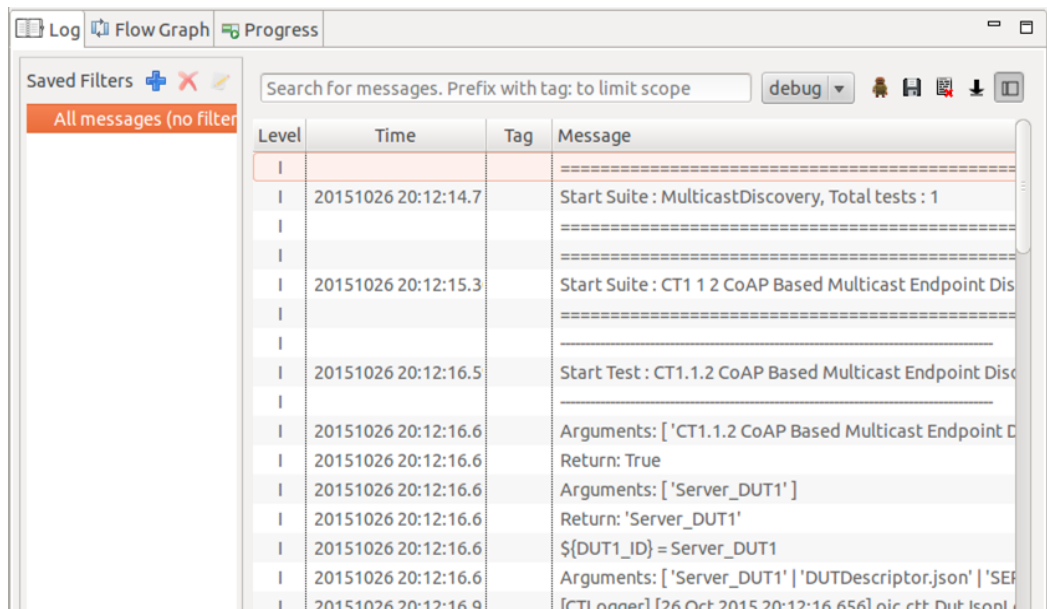


Figure 4.2.8: Conformance Tool Test Log

- When the test execution is finished, a **Test Report** window will open with a complete report on the test plan.

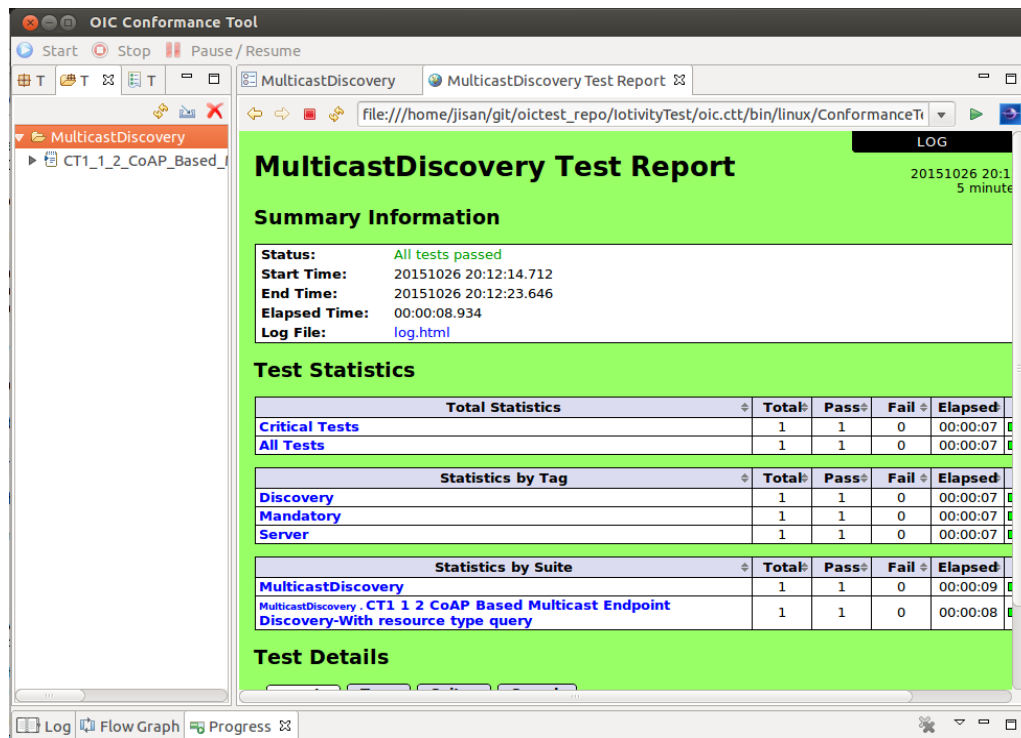


Figure 4.2.9: Conformance Tool Test Report

- The reports will be saved in the storage drive. You can view all the generated test reports from the **Test Report** tab.

4.3. Using CLI

- Go to the root folder of Conformance Tool
- Then go to the Conformance Test Directory


```
$ cd bin/linux/ConformanceTestTool/ConformanceTool/testsuite/OIC/Conformance/
```
- To run a TC file, execute `./tc_run.sh testsuite/<TC_File_Name>`, e.g.


```
$ ./run_tc.sh
testsuiteCT1_1_1_CoAP_Based_Multicast_Endpoint_Discovery-No_Query.txt
```
- The result report will be saved at `<path_to_conformance_tool>/bin/linux/ConformanceTestTool/testreport/<timestamp>/`

```

antnu@antnu-VirtualBox: ~/git/oictest_repo/IotivityOrgSource/ctt/Iotivity/tools/conformance-test-tool/bin/linux
=====
CT1.1.2 CoAP Based Multicast Endpoint Discovery - With resource ty... ..
TE -----> Multicast Discover with query -----> DUT
Query : rt=core.light

TE <----- Response for Discover <----- DUT
Response Code : 2.05
Source Address : 10.0.2.15:46362
payload[{"di":"1Qi6GcBmRWmpFinUXizXng==","links":[{"href":"/device/light-1","rt":"core.light
","if":"oic.if.baseline","p":{"bm":3}}]]}

TE <----- Response for Discover <----- DUT
Response Code : 2.05
Source Address : fe80:0:0:0:a00:27ff:fe85:b783%2:46881
payload[{"di":"1Qi6GcBmRWmpFinUXizXng==","links":[{"href":"/device/light-1","rt":"core.light
","if":"oic.if.baseline","p":{"bm":3}}]]}

TE -----> Multicast Discover with query -----> DUT
Query : rt=core.fan

TE <----- Response for Discover <----- DUT
Response Code : 2.05
Source Address : fe80:0:0:0:a00:27ff:fe85:b783%2:46881
payload[{"di":"1Qi6GcBmRWmpFinUXizXng==","links":[{"href":"/device/fan-1","rt":"core.fan","i
f":"oic.if.baseline","p":{"bm":1}}]]}

TE <----- Response for Discover <----- DUT
Response Code : 2.05
Source Address : 10.0.2.15:46362
payload[{"di":"1Qi6GcBmRWmpFinUXizXng==","links":[{"href":"/device/fan-1","rt":"core.fan","i
f":"oic.if.baseline","p":{"bm":1}}]]}

..
TE -----> Multicast Discover with query -----> DUT
Query : rt=core.dummydummy
log4j:WARN No appenders could be found for logger (CTLogger).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

CT1.1.2 CoAP Based Multicast Endpoint Discovery - With resource ty... | PASS |
-----
CT1 1 2 CoAP Based Multicast Endpoint Discovery-With resource type... | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
=====

```

Figure 4.3.1: Execution of a Test Case using Command Line Interface of ConformanceTestTool

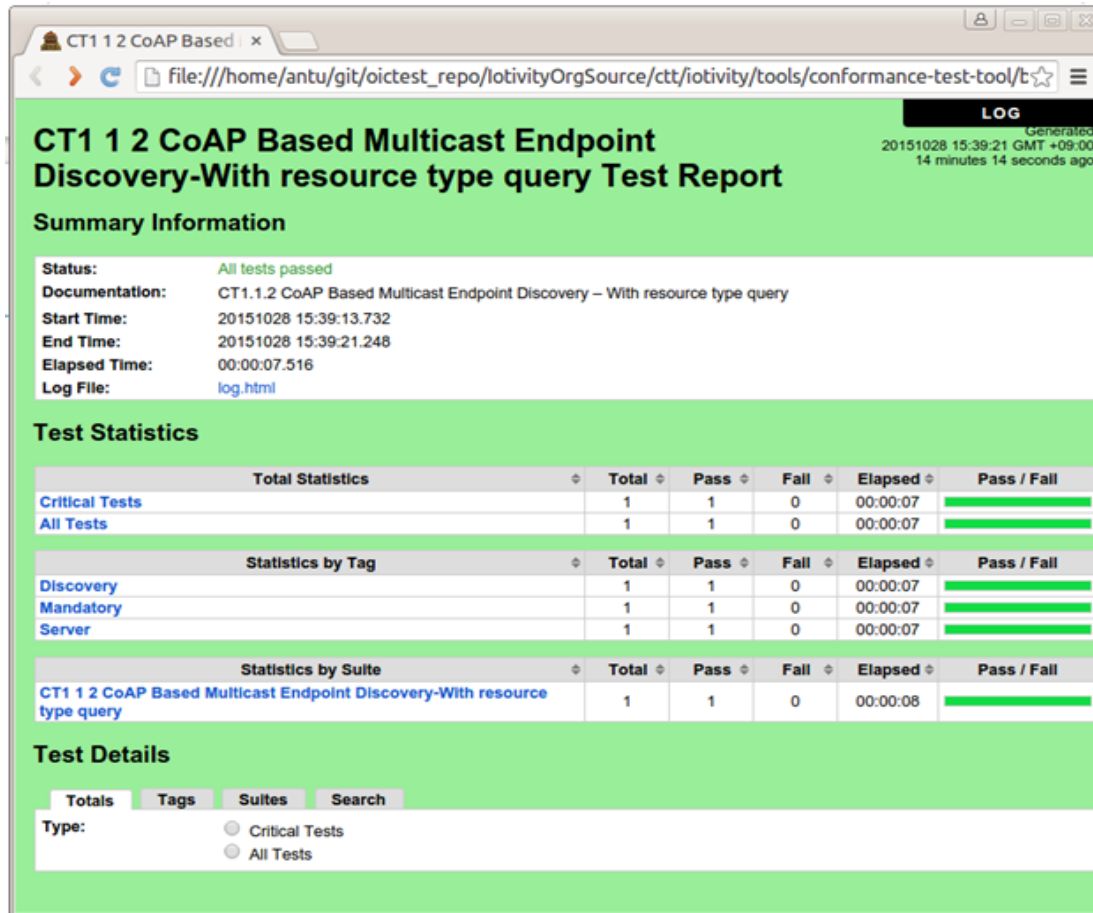


Figure 4.3.2: View of Conformance Test Tool TC Execution Result

4.4. Run Conformance Simulator

- From the top(root) directory of ConformanceSimulator, go to linux binary folder:

```
$ cd bin/linux/
```
- Append the library location of iotivity to system library path:

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<path-to-iotivity-root>/out/linux/x86/release/
```
- To run Conformance Simulator using default settings(Non-conformable, IPv6, non-secure server):

```
$ ./ConformanceSimulator
```
- Run Conformance Simulator with customize settings:

```
$ ./ConformanceSimulator [QoS<0/1>] [IP_Version<4/6>]  
[Security<0/1/2>]
```

Message Type:

QoS: 0 = NON (Non-Confirmable Message), 1 = CON (Confirmable Message)

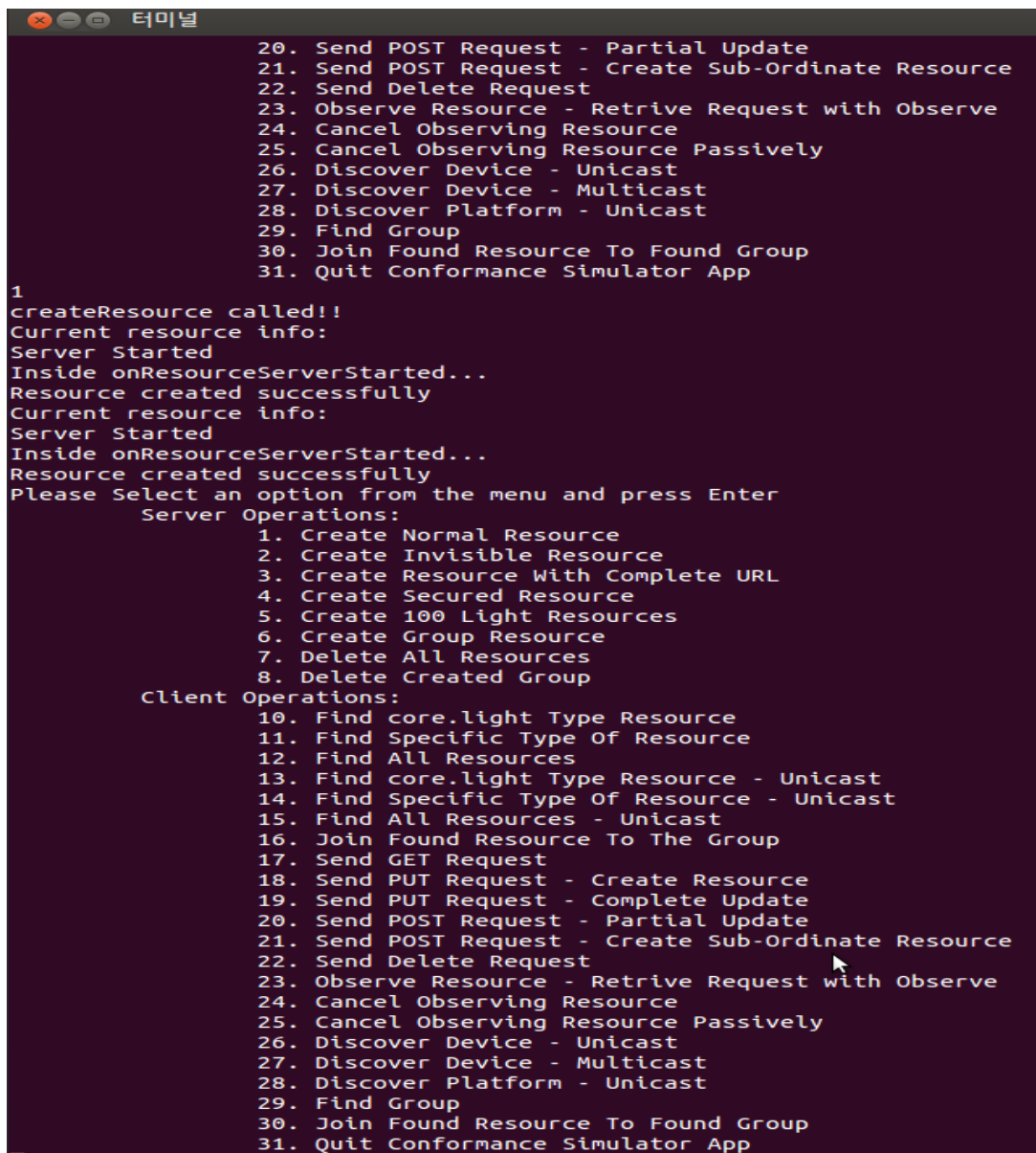
IP_Version: 4 = IPv4, 6 = IPv6

Security: 0 = non-secure, 1 = Secure Client, 2 = Secure Server

Example:

```
$ ./ConformanceSimulator 0 4 0
```

It's for, IpV4 non-secure NON-type server



```

20. Send POST Request - Partial Update
21. Send POST Request - Create Sub-Ordinate Resource
22. Send Delete Request
23. Observe Resource - Retrive Request with Observe
24. Cancel Observing Resource
25. Cancel Observing Resource Passively
26. Discover Device - Unicast
27. Discover Device - Multicast
28. Discover Platform - Unicast
29. Find Group
30. Join Found Resource To Found Group
31. Quit Conformance Simulator App

1
createResource called!!
Current resource info:
Server Started
Inside onResourceServerStarted...
Resource created successfully
Current resource info:
Server Started
Inside onResourceServerStarted...
Resource created successfully
Please Select an option from the menu and press Enter
Server Operations:
  1. Create Normal Resource
  2. Create Invisible Resource
  3. Create Resource With Complete URL
  4. Create Secured Resource
  5. Create 100 Light Resources
  6. Create Group Resource
  7. Delete All Resources
  8. Delete Created Group
Client Operations:
  10. Find core.light Type Resource
  11. Find Specific Type Of Resource
  12. Find All Resources
  13. Find core.light Type Resource - Unicast
  14. Find Specific Type Of Resource - Unicast
  15. Find All Resources - Unicast
  16. Join Found Resource To The Group
  17. Send GET Request
  18. Send PUT Request - Create Resource
  19. Send PUT Request - Complete Update
  20. Send POST Request - Partial Update
  21. Send POST Request - Create Sub-Ordinate Resource
  22. Send Delete Request
  23. Observe Resource - Retrive Request with Observe
  24. Cancel Observing Resource
  25. Cancel Observing Resource Passively
  26. Discover Device - Unicast
  27. Discover Device - Multicast
  28. Discover Platform - Unicast
  29. Find Group
  30. Join Found Resource To Found Group
  31. Quit Conformance Simulator App
  
```

Figure 4.4: Conformance Simulator Execution