# **OpenH264 Encoder Conformance Test**

Table of Contents

[OpenH264 Encoder Conformance Test 1](#_Toc481580770)

[1. Conformance Test brief introduction 3](#_Toc481580771)

[1.1 Conformance test for one case 3](#_Toc481580772)

[1.2 Conformance for one YUV 3](#_Toc481580773)

[1.3 Conformance test for all YUVs 4](#_Toc481580774)

[1.4 Conformance test for SVC/SCC 4](#_Toc481580775)

[2. Conformance Test script basic info 5](#_Toc481580776)

[2.1 Brief introduction 5](#_Toc481580777)

[**Repos info** 5](#_Toc481580778)

[Support platform 5](#_Toc481580779)

[2.2 How to use 6](#_Toc481580780)

[**2.2.1 Linux single machine local test** 6](#_Toc481580781)

[2.3 Codec and test tools 12](#_Toc481580782)

[**2.3.1 Codec for test** 12](#_Toc481580783)

[**2.3.2 App tools for test** 13](#_Toc481580784)

[2.4 Test cases 14](#_Toc481580785)

[**2.4.1 Test case configuration files** 14](#_Toc481580786)

[**2.4.2 Test case generation** 15](#_Toc481580787)

[2.5 Test data 16](#_Toc481580788)

[**2.5.1** **Test data for one case during test** 16](#_Toc481580789)

[**2.5.2** **Test data for one test cases set** 17](#_Toc481580790)

[**2.5.3** **Test data for all YUVs** 20](#_Toc481580791)

[2.6 Test report 22](#_Toc481580792)

[3. Jenkins based conformance test 25](#_Toc481580793)

[3.1 Basic test architecture 25](#_Toc481580794)

[3.2 Test set and Jenkins slaves task assignment 25](#_Toc481580795)

[3.3 Test data and test report 26](#_Toc481580796)

[3.4 Failed cases reproduce and analysis 26](#_Toc481580797)

[4. SGE based conformance test 27](#_Toc481580798)

[4.1 Basic test architecture 27](#_Toc481580799)

[4.2 Test set and Jenkins slaves task assignment 27](#_Toc481580800)

[4.3 Test data and test report 28](#_Toc481580801)

[4.4 Failed cases reproduce and analysis 28](#_Toc481580802)

[5. SHA1 tables generation 29](#_Toc481580803)

[6. SGE install and configuration 30](#_Toc481580804)

# **Conformance Test brief introduction**

## **1.1 Conformance test for one case**

## **1.2 Conformance for one YUV**

## **1.3 Conformance test for all YUVs**

## **1.4 Conformance test for SVC/SCC**

# **Conformance Test script basic info**

## **2.1 Brief introduction**

### **Repos info**

* GitHub URL： <https://github.com/shihuade/Conformance-Test-Openh264.git>
* Branch: master
* Basic introduction

- This model is part of Cisco openh264 project for encoder conformance test.

In this test, all cases of all test sequences will be tested and check that whether

the reconstructed YUV is the same with JM decoder's YUV. if yes, the test case

will be marked as passed and SHA1 string will be generated, otherwise, marked as unpassed

and no SHA1 string for this test case in SHA1 table file(XXX.yuv\_AllCases\_SHA1\_Table.csv)

- The output of the test are those files in ./FinalResult, espectially the summary files named as XXX.Summary.log.

and cases passed status in files named as XXX\_AllCasesOutput.csv.And SHA1 table files can be found in

folder ./SHA1Table.

For those temp data generated during test, can be found ./AllTestData/xxx.yuv/

- For Cisco openh264 project,please refer to https://github.com/cisco/openh264.

## **Support platform**

* Linux:
* Mac OS/Unix

## **2.2 How to use**

### **2.2.1 Linux single machine local test**

* Step 1:

Change **Testplatform** to **Linux**,in one cfg file under CaseConfigure/\*.cfg which you want to test

#============================== Test platform =======================

TestPlatform:  Linux            #test platform, Mac or Linux

#============================== Test platform =======================

* Step 2:

For SVC, run below command like

./run\_Main.sh LocalTest ./CaseConfigure/case\_SVC.cfg

For SCC, run below command like

./run\_Main.sh LocalTest ./CaseConfigure/case\_SCC.cfg

* Step 3:

Wait final test result and test result will be under

SHA1Table/\*

FinalResult/\*

FinalTestReport/\*

**2.2.2 Linux Jenkins based test**

**Jenkins based test, will assign test cases to Jenkins slaves.**

**Jenkins job, please refer to:**

[**http://10.140.198.27:8080/view/ConformanceTest/job/OpenH264-Encoder-Conformance-Test/**](http://10.140.198.27:8080/view/ConformanceTest/job/OpenH264-Encoder-Conformance-Test/)

**For example,**

**in CaseConfigure/, there are 20 cfg files for SVC and SCC test case**

case\_Jenkins\_SCC\_TestSet0.cfg

case\_Jenkins\_SCC\_TestSet1.cfg

case\_Jenkins\_SCC\_TestSet2.cfg

case\_Jenkins\_SCC\_TestSet3.cfg

case\_Jenkins\_SCC\_TestSet4.cfg

case\_Jenkins\_SCC\_TestSet5.cfg

case\_Jenkins\_SCC\_TestSet6.cfg

case\_Jenkins\_SCC\_TestSet7.cfg

case\_Jenkins\_SCC\_TestSet8.cfg

case\_Jenkins\_SCC\_TestSet9.cfg

case\_Jenkins\_SVC\_TestSet0.cfg

case\_Jenkins\_SVC\_TestSet1.cfg

case\_Jenkins\_SVC\_TestSet2.cfg

case\_Jenkins\_SVC\_TestSet3.cfg

case\_Jenkins\_SVC\_TestSet4.cfg

case\_Jenkins\_SVC\_TestSet5.cfg

case\_Jenkins\_SVC\_TestSet6.cfg

case\_Jenkins\_SVC\_TestSet7.cfg

case\_Jenkins\_SVC\_TestSet8.cfg

case\_Jenkins\_SVC\_TestSet9.cfg

**For more detail about test set, please refer to:**

AboutJenkinsTestSet.txt

**For each Jenkins slaves, run below command, take SCC test set 0 for example:**

**./run\_Main.sh LocalTest case\_Jenkins\_SCC\_TestSet0.cfg**

**note: please make sure that there in test YUV desktop\_dialog\_1920x1080\_i420.yuv**

**under /home/Video/YUV on test slave or you can change the YUV location**

**For Jenkins job run for one test set, take SCC test set 0 for example,**

**you can refer to below Jenkins job URL for more detail.**

[**http://10.140.198.27:8080/job/OpenH264\_ConformanceTest\_SCC\_TestSet0/configure**](http://10.140.198.27:8080/job/OpenH264_ConformanceTest_SCC_TestSet0/configure)

**To Optimize overall test runtime for all test cases,**

* **Test set cfg file will need to optimize based on actual historical run time in future.**
* **And the script level optimization is also need to decrease run time for one test cases**

**2.2.3 Linux SGE based test**

**SGE test is based on SGE distribution system and assign test tasks to SGE slaves.**

**For more detail about SGE test, please refer to**

* Chapter SGE based conformance test
* Chapter SGE install and configuration

SGE based test will be transform to 2.2.2 test mode,

and will be removed in future

**For SGE test on SGE master:**

**Run below command:**

* SCC: ./run\_Main.sh  SGETest    ./CaseConfigure/case\_SCC.cfg
* SVC: ./run\_Main.sh  SGETest    ./CaseConfigure/case\_SVC.cfg

**For SGE test on SGE master based on Jenkins:**

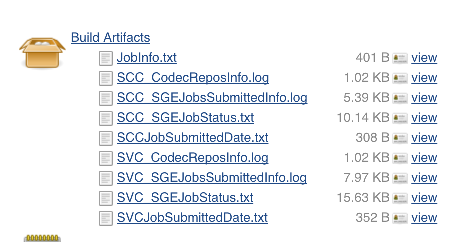
**Please refer to Jenkins job(will be removed)**

**1）Openh264-SGE-Job-Submit**

[**http://10.140.198.27:8080/view/SGE-Test/job/Openh264-SGE-Job-Submit/configure**](http://10.140.198.27:8080/view/SGE-Test/job/Openh264-SGE-Job-Submit/configure)

**after running this job, SGE test tasks will be assigned to SGE slaves**

**job artifacts will give more detail about SGE jobs submit, as show below:**

****

**And need to wait for final test result, sometimes may need 6~12 hours to compete all test cases on SGE which run in background**

**If you want to check the test status and result, please go to 2).**

**2）penh264-SGE-Job-status-And-TestResult**

<http://10.140.198.27:8080/view/SGE-Test/job/Openh264-SGE-Job-status-And-TestResult/configure>

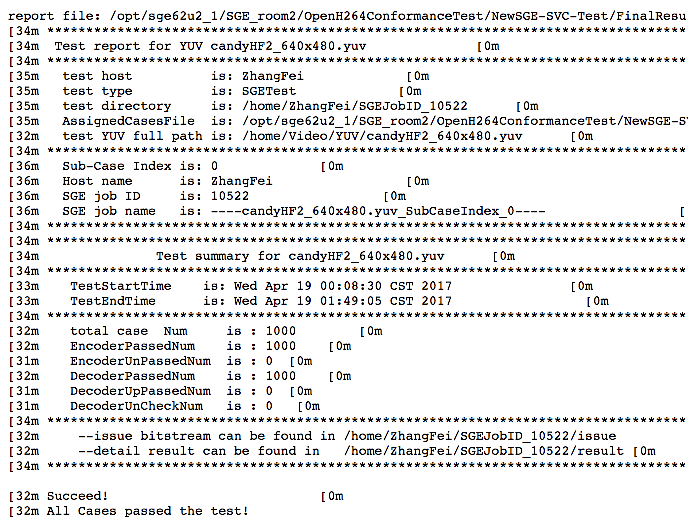
**if there are failed cases, this job status will be marked as failed, and the artifacts files will give more detail about which YUV in which case on which SGE slaves failed**

**for more detail about test status and result please refer to artifacts as show below:**

****

**take SVC\_JobReport.txt for example:**

**below shows that test YUV candyHF2\_640x480.yuv with 1000 test cases which running on ZhangFei SGE Slaves**

****

**2.2.4 Mac/Unix single machine local test**

**note:**

**#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

For downsample and extractor app tools,

which using for multiple layer cases,

currently, there are no mac version,

so, on mac, only support AVC cases.

**#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Below tools app only for Linux under**

Codec\_Linux/Tools32Bits/

Codec\_Linux/Tools64Bits/

* **DownConvertStatic**
* **JSVMDecoder**
* **extractor.app**

**#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**multi layer setting should set to 0**

#============================== Multiple Layer option===================

MultiLayer:     0   # 0 single layer  1 multi layer

#  2 mult layer and single layer

Multiple16Flag: 1  # all sub layers' resolution is multiple of 16

# 0:disable ;  1:enable

#============================== Multiple Layer option===================

**#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Usage:**

* Step 1:

Change **Testplatform** to **Mac**,in one cfg file under CaseConfigure/\*.cfg which you want to test

#============================== Test platform =======================

TestPlatform:  Mac            #test platform, Mac or Linux

#============================== Test platform =======================

* Step 2:

For SVC(AVC only), run below command like

./run\_Main.sh LocalTest ./CaseConfigure/case\_SVC.cfg

For SCC, run below command like

./run\_Main.sh LocalTest ./CaseConfigure/case\_SCC.cfg

* Step 3:

Wait final test result and test result will be under

SHA1Table/\*

FinalResult/\*

FinalTestReport/\*

**2.2.5 SHA1 table generation**

**SHA1 tables are used for encoder output bit stream validation.**

**For more detail, please refer to chapter:**

* + SHA1 tables generation

1. **Generate SHA1 tables on Mac/Linux local machine:**
   * SCC

./run\_Main.sh LocalTest case\_SHA1Table\_SCC.cfg

* + SVC:

./run\_Main.sh LocalTest case\_SHA1Table\_SVC.cfg

**Final SHA1 tables for test YUV can be found under folder:**

* + **./SHA1Table/**

1. **Jenkins Jobs:**

[**http://10.140.198.27:8080/job/Openh264\_Travis\_SHA1Table\_Generation\_weekly/configure**](http://10.140.198.27:8080/job/Openh264_Travis_SHA1Table_Generation_weekly/configure)

**Final SHA1 tables for 3 YUVs, you can download from artifacts files**



## **2.3 Codec and test tools**

### **2.3.1 Codec for test**

1. **switch to your codec branch and run conformance test**

if you change algorithm or you refactor code, you need to run conformance test for your code change. For example, you push your code change to your repos under branch Algorithm\_V1.0,

* + Branch=”Algorithm\_V1.0”
  + Repos=” <https://github.com/shihuade/openh264.git>”

and run conformance test on Linux,

you can:

1. modify below parameters in case configuration file

#====================== Git Repository setting ==============

GitAddress  <https://github.com/shihuade/openh264.git>

GitBranch   Algorithm\_V1.0

#============================================================

or

1. you can overwrite parameters by passing to run\_Main.sh

./run\_Main.sh  LocalTest  ./CaseConfigure/case\_SVC.cfg  **${ Branch } ${ Repos }**

1. **Update codec by scripts**

* checkout openh264 repos and switch to test branch

more detail, please refer to script files

./run\_CheckoutRepos.sh

* build codec with enable YUV dump macro

more detail, please refer to script file

./run\_UpdateCodec.sh

* Update flow list as below:

./run\_Main.sh

🡺run\_PrepareAllTestData.sh

🡺runUpdateCodec()

./run\_CheckoutRepos.sh

./run\_UpdateCodec.sh

### **2.3.2 App tools for test**

for more detail about test tools, please refer to below files:

* Codec\_Linux/Tools32Bits/AboutTools.txt
* Codec\_Linux/Tools64Bits/AboutTools.txt

For linux 32 bits test tools, list as below:

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tools app build on centos 6.5 32 bits version

i686

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    ----JMDecoder/DownConvertStatic

        git clone from: https://github.com/shihuade/JSVM.git

        bin file from JSVM/bin/

        with build from JSVM/JSVM/H264Extension/build/linux/

        with default make setting

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    ----JMDecoder

        git clone from: https://github.com/shihuade/JM.git

        JM19.0

        bin file from JM/bin/

        with build from JM/ with default make setting

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    ----extractor.app

        from train pangu project

        bin file: under pangu/extractor/build/linux/bin/

        build under pangu/extractor/build/linux/bld/

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Tools introduction:**

* DownConvertStatic

Downsample YUV for input test YUV

In Scripts/run\_PrepareInputYUV.sh

./DownConvertStatic ${OriginW} ${OriginH} ${OriginYUV}

${OutputW} ${OutputH} ${OutPutYUVFile}"

Truncate frame num

In Scripts/run\_TruncateYUV.sh

./DownConvertStatic ${PicW} ${PicH} ${InputYUV}

${PicW} ${PicH} ${OutputYUV}

0 0 0 ${OutputFrmNum}

* JMDecoder

It is for validating encoded bit stream which encoded by test cases, and, check if pass below iterms:

1. Bit stream decoder by JMDecoder successfully
2. Decoded YUV is the same with reconstruction YUV by test encoder

For more detail, please refer to Scripts/run\_CheckByJSVMDecoder.sh

* JSVMDecoder

The same with JMDecoder, and support SVC multiple layer cases

For more detail, please refer to Scripts/run\_CheckByJSVMDecoder.sh

* extractor.app

extract single layer bit stream from multiple layer cases’ bit stream.

For more detail, please refer to Scripts/run\_ExtractLayerBitStream.sh

## **2.4 Test cases**

### **2.4.1 Test case configuration files**

**For Jenkins based test sets assignment configure:**

**in CaseConfigure/, there are 20 cfg files for SVC and SCC test case**

case\_Jenkins\_SCC\_TestSet0.cfg

case\_Jenkins\_SCC\_TestSet1.cfg

case\_Jenkins\_SCC\_TestSet2.cfg

case\_Jenkins\_SCC\_TestSet3.cfg

case\_Jenkins\_SCC\_TestSet4.cfg

case\_Jenkins\_SCC\_TestSet5.cfg

case\_Jenkins\_SCC\_TestSet6.cfg

case\_Jenkins\_SCC\_TestSet7.cfg

case\_Jenkins\_SCC\_TestSet8.cfg

case\_Jenkins\_SCC\_TestSet9.cfg

case\_Jenkins\_SVC\_TestSet0.cfg

case\_Jenkins\_SVC\_TestSet1.cfg

case\_Jenkins\_SVC\_TestSet2.cfg

case\_Jenkins\_SVC\_TestSet3.cfg

case\_Jenkins\_SVC\_TestSet4.cfg

case\_Jenkins\_SVC\_TestSet5.cfg

case\_Jenkins\_SVC\_TestSet6.cfg

case\_Jenkins\_SVC\_TestSet7.cfg

case\_Jenkins\_SVC\_TestSet8.cfg

case\_Jenkins\_SVC\_TestSet9.cfg

**For local /SGE test mode, configure files:**

SVC: case\_SCC.cfg

SCC: case\_SCC.cfg

**For SHA1 table generation, configure files:**

SCC: case\_SHA1Table\_SCC.cfg

SVC: case\_SHA1Table\_SVC.cfg

**For default encode parameters, encoder configure files:**

welsenc.cfg

layer0.cfg

layer1.cfg

layer2.cfg

layer3.cfg

which copied from ${Openh264SrcDir}/testbin

you can refer to script in

scripts/run\_UpdateCodec.sh

🡺runCopyFile

all those encode parameters in welenc.cfg and layerx.cfg

will be **overwrited** by parameters in case\_xxx.cfg

you can refer to script in

Scripts/run\_TestOneCase.sh

🡺runEncodeOneCase()

### **2.4.2 Test case generation**

more detail, please refer to script file

./Scripts/run\_GenerateCase.sh

**Brief:**

Combine all test cases in configure file like case\_xxx.cfg

And output cvs file which contain all cases

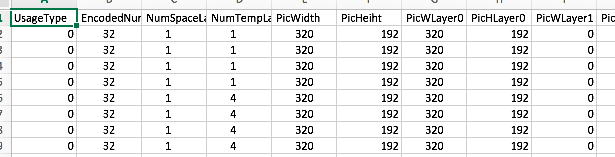
./run\_GenerateCase.sh $Case.cfg $TestSequence $OutputCaseFile

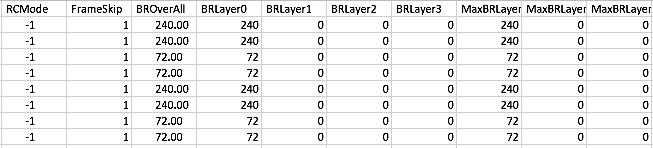
for example;

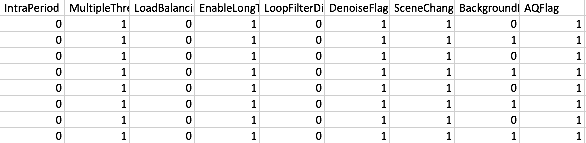
./run\_GenerateCase.sh case\_SVC.cfg ABC\_1920X1080.yuv AllCase.csv

will output all test cases for ABC\_1920X1080.yuv based on case\_SVC.cfg

output cases csv file may look like:







## **2.5 Test data**

### **Test data for one case during test**

When start to run test for one test set, script will go to test space,

* TestSpace=./AllTestData/${TestYUV}
* And all temp files will be output to:

${TestSpace}/${TempDataPath},

which empDataPath=” TempData”

* And all temp files list below

CheckLogFile="${TempDataPath}/CaseCheck.log";

EncoderLog="${TempDataPath}/encoder.log"

RecYUVFile0="${TempDataPath}/${TestYUVName}\_rec\_0.yuv";

RecYUVFile1="${TempDataPath}/${TestYUVName}\_rec\_1.yuv"

RecYUVFile2="${TempDataPath}/${TestYUVName}\_rec\_2.yuv";

RecYUVFile3="${TempDataPath}/${TestYUVName}\_rec\_3.yuv"

RecCropYUV0="${TempDataPath}/${TestYUVName}\_rec\_0\_cropped.yuv";

RecCropYUV1="${TempDataPath}/${TestYUVName}\_rec\_1\_cropped.yuv"

RecCropYUV2="${TempDataPath}/${TestYUVName}\_rec\_2\_cropped.yuv";

RecCropYUV3="${TempDataPath}/${TestYUVName}\_rec\_3\_cropped.yuv"

For more detail, please refer to script

*run\_TestAssignedCases.sh*

*run\_TestOneCase.sh*

### **Test data for one test cases set**

1. **One test cases set is sub-set of all test cases for one YUV.**

For example, for desktop\_dialog\_1920x1080\_i420.yuv

* In case\_SCC.cfg, oveall test cases is num 10368
* In case\_Jenkins\_SCC\_TestSet0.cfg, test case num is 2592

When run below command in local test machine:

./run\_Main.sh LocalTest case\_Jenkins\_SCC\_TestSet0.cfg

only sub-set of 2592 cases of all 10368 cases.

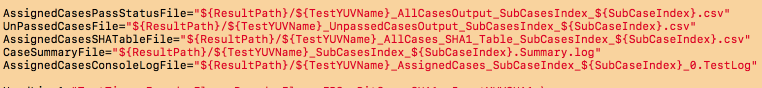
1. **Test data files for one test set is under ResultPath and IssueDataPath,**

* TestSpace=./AllTestData/${TestYUV}
* ResultPath=${ TestSpace }/result
* IssueDataPath=${ TestSpace }/issue

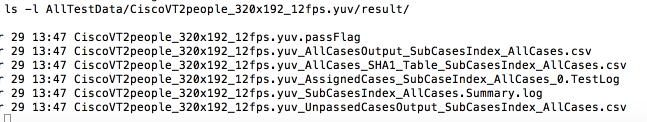
For more detail, please refer to script

*run\_TestAssignedCases.sh*

Below is part of script for test data path setting in *run\_TestAssignedCases.sh*



Below is test result files for CiscoVT2people\_320x192\_12fps.yuv



For subcase index,

* SGE test mode: index = 0, 1, 2, …
* Local test mode: index = AllCases

For more detail, please refer to run\_CasesPartition.sh

1. **For all sub test cases console output, you can refer to**

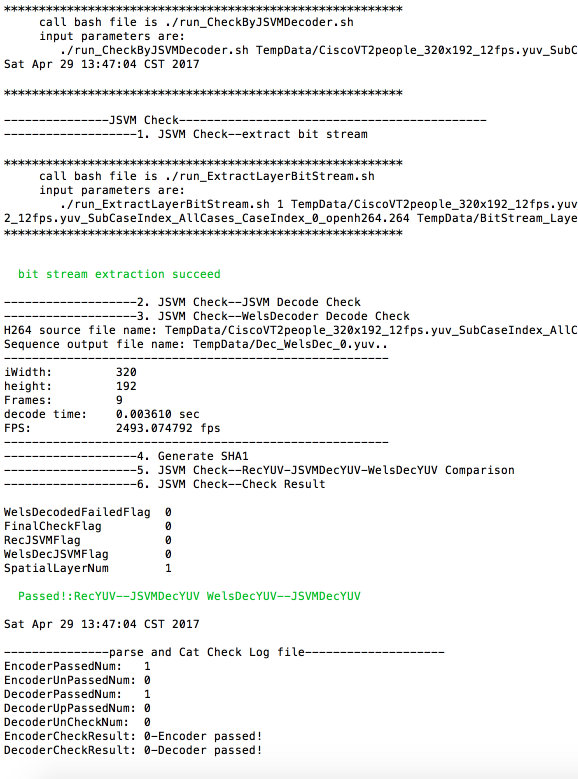
AssignedCasesConsoleLogFile

Below is test console output for CiscoVT2people\_320x192\_12fps.yuv

For each cases, there is detail console output log include:

* Test cases detail parameters
* Encoder log, include encoder command line, encoder output
* Encoder reconstruction YUV’s check
* JSVM/JM check log
* Wlsdecoder check log
* Final check summary info





1. **How to reproduce failed case**
2. **Reproduce based on console output log**

As mentioned in #3, there are console output log for each case,

you can find out the root cause for failed cases, and run that failed case step by step based on log

1. **Reproduce based on csv file**

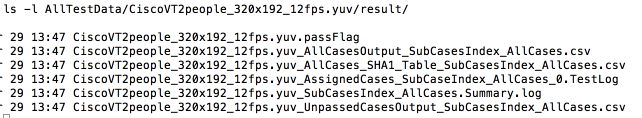
Below is test result files for CiscoVT2people\_320x192\_12fps.yuv

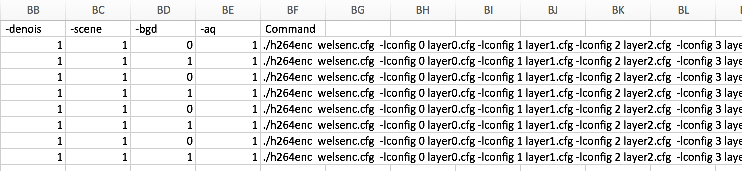
Scv file: CiscoVT2people\_320x192\_12fps.yuv\_AllCasesOutput\_SubCasesIndex\_AllCases.csv

Include all test cases status and encoder command line.

You can run that cases with:

* encoder command
* using JSVM/JM to decode bit stream
* check decoded YUV which decoded by JSVM/JM/Welsdecoder
* validate bit stream and YUVs

****

****

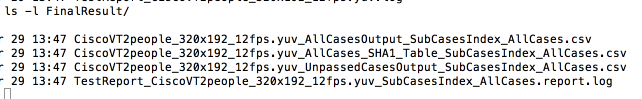
### **Test data for all YUVs**

During running test cases and after completed all test cases for one YUVs,

test result files are in ./FinalResult directory.

1. **Below is example for only one YUV in test**

* test YUV is CiscoVT2people\_320x192\_12fps.yuv
* test set num is 1



if there are 4 set for CiscoVT2people\_320x192\_12fps.yuv,

the result file may looks like:

CiscoVT2people\_320x192\_12fps.yuv\_AllCasesOutput\_SubCasesIndex\_0.csv

CiscoVT2people\_320x192\_12fps.yuv\_AllCasesOutput\_SubCasesIndex\_1.csv

CiscoVT2people\_320x192\_12fps.yuv\_AllCasesOutput\_SubCasesIndex\_2.csv

CiscoVT2people\_320x192\_12fps.yuv\_AllCasesOutput\_SubCasesIndex\_3.csv

CiscoVT2people\_320x192\_12fps.yuv\_AllCases\_SHA1\_Table\_SubCasesIndex\_0.csv

CiscoVT2people\_320x192\_12fps.yuv\_AllCases\_SHA1\_Table\_SubCasesIndex\_1.csv

CiscoVT2people\_320x192\_12fps.yuv\_AllCases\_SHA1\_Table\_SubCasesIndex\_2.csv

CiscoVT2people\_320x192\_12fps.yuv\_AllCases\_SHA1\_Table\_SubCasesIndex\_3.csv

CiscoVT2people\_320x192\_12fps.yuv\_UnpassedCasesOutput\_SubCasesIndex\_0.csv

CiscoVT2people\_320x192\_12fps.yuv\_UnpassedCasesOutput\_SubCasesIndex\_1.csv

CiscoVT2people\_320x192\_12fps.yuv\_UnpassedCasesOutput\_SubCasesIndex\_2.csv

CiscoVT2people\_320x192\_12fps.yuv\_UnpassedCasesOutput\_SubCasesIndex\_3.csv

TestReport\_CiscoVT2people\_320x192\_12fps.yuv\_SubCasesIndex\_0.report.log

TestReport\_CiscoVT2people\_320x192\_12fps.yuv\_SubCasesIndex\_1.report.log

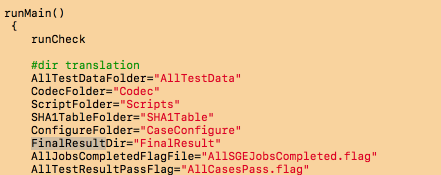
TestReport\_CiscoVT2people\_320x192\_12fps.yuv\_SubCasesIndex\_2.report.log

TestReport\_CiscoVT2people\_320x192\_12fps.yuv\_SubCasesIndex\_3.report.log

1. **Below is the final test result setting in scripts**

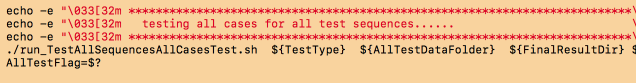
*run\_Main.sh*

*🡺runMain()*



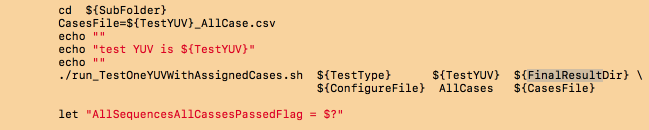
*run\_Main.sh*

*🡺runMain()*



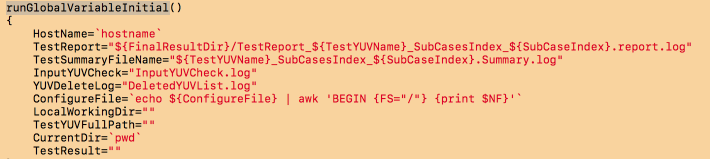
*run\_AllTestSequencesAllCasesTest.sh*

*🡺runLocalTest*



*run\_TestOneYUVWithAssignedCases.sh*

*🡺runGlobalVariableInitial*



## **2.6 Test report**

After all test cases for all YUVs have been completed,

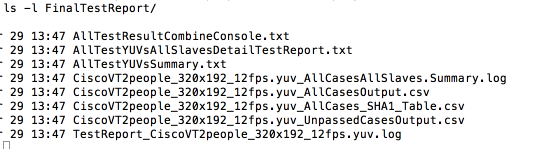
final test result and report for all YUVs will be generated

and create under directory

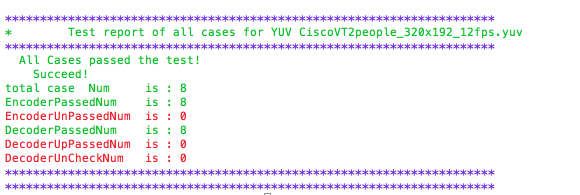
./FinalTestReport

1. **Below is example for only one YUV in test,**

test YUV is CiscoVT2people\_320x192\_12fps.yuv



output of TestReport\_CiscoVT2people\_320x192\_12fps.yuv.log



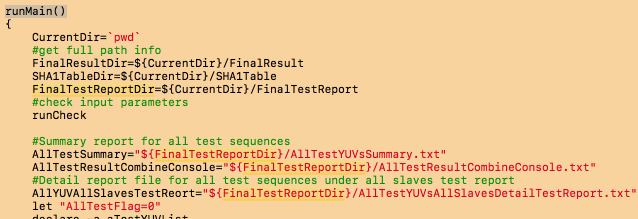
output of AllTestYUVsAllSlavesDetailTestReport.txt



1. **Below is the final test report directoty setting in scripts**

*run\_GetAllTestResult.sh*

*🡺runMain()*



*run\_GetAllTestResult.sh*

*🡺runGetAllYUVTestResult*



# **Jenkins based conformance test**

## **Basic test architecture**

## **Test set and Jenkins slaves task assignment**

## **Test data and test report**

## **Failed cases reproduce and analysis**

# **SGE based conformance test**

## **Basic test architecture**

## **Test set and Jenkins slaves task assignment**

## **Test data and test report**

## **Failed cases reproduce and analysis**

# **SHA1 tables generation**

# **SGE install and configuration**

5.1