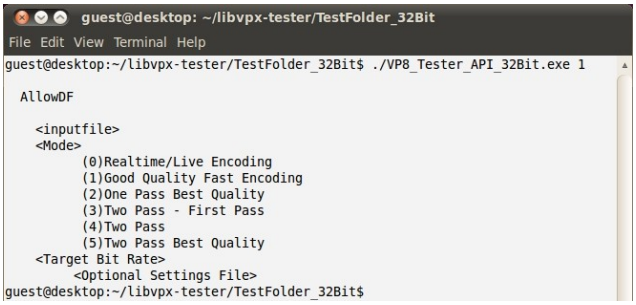
VP8 Codec-Tester v0.9.1

(08-02-2010)

Using the Tester

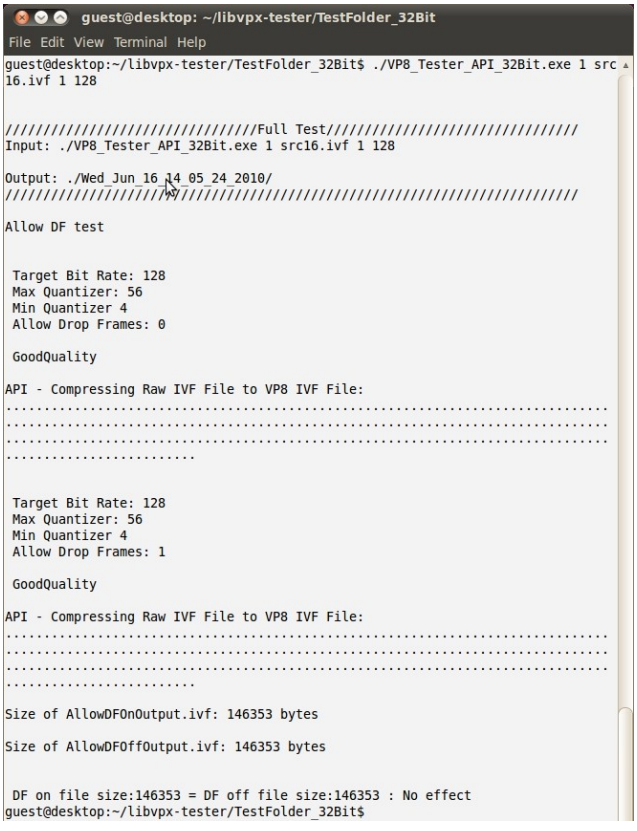
To run the executable open the command prompt, from there change the current directory to /libvpx-tester/TestFolder_32Bit or /libvpx-tester/TestFolder_64Bit depending on which tester you have built or wish to run. Run VP8_Tester_API_32Bit.exe or VP8_Tester_API_64Bit.exe.

When just the master executable is run, you will see the default screen shown on the right. The Tester is driven on an instance by instance basis. Tests are signified numerically with the number to the left of the test name representing the test. It is possible to run tests independently or from a properly formatted control text file. It is also possible to make use of tools built into the tester.



Running a Single Test

To run a single test run the executable with the number of the test you wish to run as the first command line argument. The Tester will output what further test specific input is necessary.



EXAMPLE- to run the test shown on the right type:

"./VP8_Tester_API_32Bit.exe 1 src16.ivf 1 128"

Where the first "1" is the test number, "src16.ivf" is the input file, the second "1" is the mode to run the compression in, and 128 is the target bit rate in KBps.

Running Multiple Tests

Multiple Tests are run by inputting commands from a user created text file specifying which tests are to be run and the conditions under which they are to be run. A sample control file can be created to the directory the executable is located in by using the Tester's "CreateSampleTextFiles" Tool (see "Using Tools" below). Multiple Tests may be run in four modes.

Mode 1: Run Compressions and Tests,

Mode 2: Run Compressions Only,

Mode 3: Run tests on preexisting compressions, or

Mode 4: Resume last mode.

To run multiple tests run the executable with the command line argument 0, for "RunTestsFromFile", the mode you wish to run it in, and finally the controlling text file or in the case of Modes 3 and 4, the directory that compressions were being saved to or that tests were being run in.

EXAMPLE- to run the multiple test shown on the left type:

"./VP8_Tester_API_32Bit.exe 0 1 QuickTest_32Bit.txt"

The sample control file provided by the tester is configured under the assumption that input test .ivf files are located in a directory named "TestClips" that exists one folder level above the executable .

The Raw video files used as input for the sample multiple test can be downloaded from the ftp directory found here: _____________

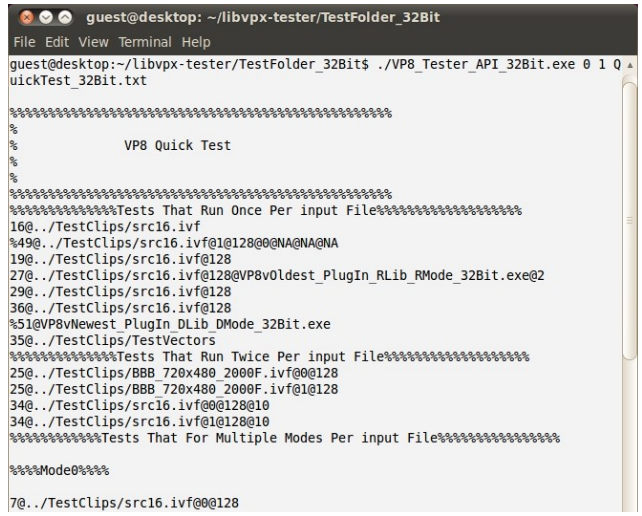
Using Tools

To use a tool input the name of the tool as the first command line parameter and press enter. If any further information is needed to run the tool it will be displayed in the same way test input is displayed.

EXAMPLE- to run the Create Sample Text Files Tool mentioned above type:

"./VP8_Tester_API_32Bit.exe CreateSampleTextFiles"

See the column named "Tools" under the testers default output or the section below named "Tools Overview" for a full list of supported tools.

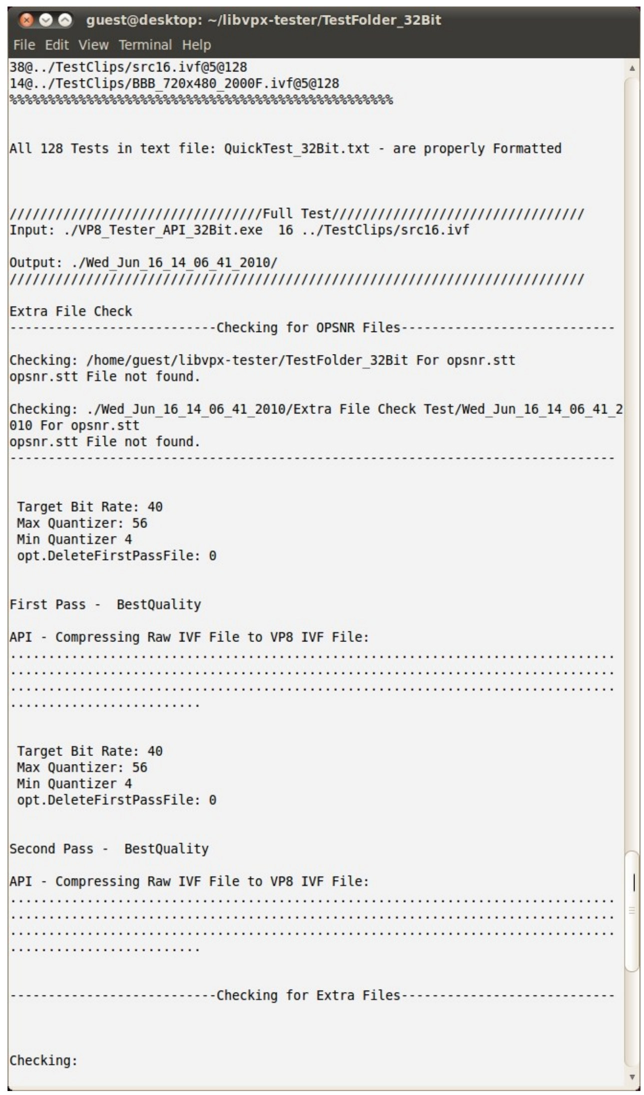


```
guest@desktop: ~/libvpx-tester/TestFolder_32Bit
File  Edit  View  Terminal  Help
guest@desktop:~/libvpx-tester/TestFolder_32Bit$ ./VP8_Tester_API_32Bit.exe 0 1 Q
uickTest_32Bit.txt

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%               VP8 Quick Test
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%Tests That Run Once Per input File%%%%%%%%%%%%%%%%%%%
16@../TestClips/src16.ivf
%49@../TestClips/src16.ivf@1@128@0@NA@NA@NA
19@../TestClips/src16.ivf@128
27@../TestClips/src16.ivf@128@VP8vOldest_PlugIn_RLib_RMode_32Bit.exe@2
29@../TestClips/src16.ivf@128
36@../TestClips/src16.ivf@128
%51@VP8vNewest_PlugIn_DLib_DMode_32Bit.exe
35@../TestClips/TestVectors
%%%%%%%%%%%%%%Tests That Run Twice Per input File%%%%%%%%%%%%%%%%%%%
25@../TestClips/BBB_720x480_2000F.ivf@0@128
25@../TestClips/BBB_720x480_2000F.ivf@1@128
34@../TestClips/src16.ivf@0@128@10
34@../TestClips/src16.ivf@1@128@10
%%%%%%%%%%%%Tests That For Multiple Modes Per input File%%%%%%%%%%%%%%%%

%%%%Mode0%%%%

7@../TestClips/src16.ivf@0@128
```



```
guest@desktop: ~/libvpx-tester/TestFolder_32Bit
File  Edit  View  Terminal  Help
38@../TestClips/src16.ivf@5@128
14@../TestClips/BBB_720x480_2000F.ivf@5@128
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


All 128 Tests in text file: QuickTest_32Bit.txt - are properly Formatted



//////////////////////////////////Full Test//////////////////////////////////
Input: ./VP8_Tester_API_32Bit.exe  16 ../TestClips/src16.ivf

Output: ./Wed_Jun_16_14_06_41_2010/
///////////////////////////////////////////////////////////////////////////

Extra File Check
---------------------------Checking for OPSNR Files----------------------------

Checking: /home/guest/libvpx-tester/TestFolder_32Bit For opsnr.stt
opsnr.stt File not found.

Checking: ./Wed_Jun_16_14_06_41_2010/Extra File Check Test/Wed_Jun_16_14_06_41_2
010 For opsnr.stt
opsnr.stt File not found.
--------------------------------------------------------------------------------


 Target Bit Rate: 40
 Max Quantizer: 56
 Min Quantizer 4
 opt.DeleteFirstPassFile: 0


First Pass -  BestQuality

API - Compressing Raw IVF File to VP8 IVF File:
................................................................................
................................................................................
................................................................................
.........................


 Target Bit Rate: 40
 Max Quantizer: 56
 Min Quantizer 4
 opt.DeleteFirstPassFile: 0


Second Pass -  BestQuality

API - Compressing Raw IVF File to VP8 IVF File:
................................................................................
................................................................................
................................................................................
.........................


---------------------------Checking for Extra Files----------------------------



Checking:
```

Checking a Multiple Test's Progress

To check the status of a currently running multiple test you can check both "TestsRun.txt" and "Mode1Results.txt" (or "Mode2" or "Mode3Results" if running modes 2 or 3).

TestsRun.txt tracks the testers progress through the original input control text file. Tests that have been fully completed are marked with a "+" to their left. Tests that have had compressions only completed are marked with a "-" and tests that have not been completed at all are marked with a " " to their left.

Mode1Results.txt records the result of each test the tester has run along with the folder name it is saved to.

Mode2Results.txt records if compressions have been made properly and the folder names they were saved to.

Mode3Results.txt records the outcome of the final tests run on pre-compressed material and the folder names they were saved to.

Resuming a Stopped Multiple Test

To resume a stopped multiple test input the following:

"./VP8_Tester_API_32Bit.exe 0 4 TestFolderName"

Where "TestFolderName" is the name of the folder containing the test data you wish to restart.

Creating a Multiple Test Input Text File

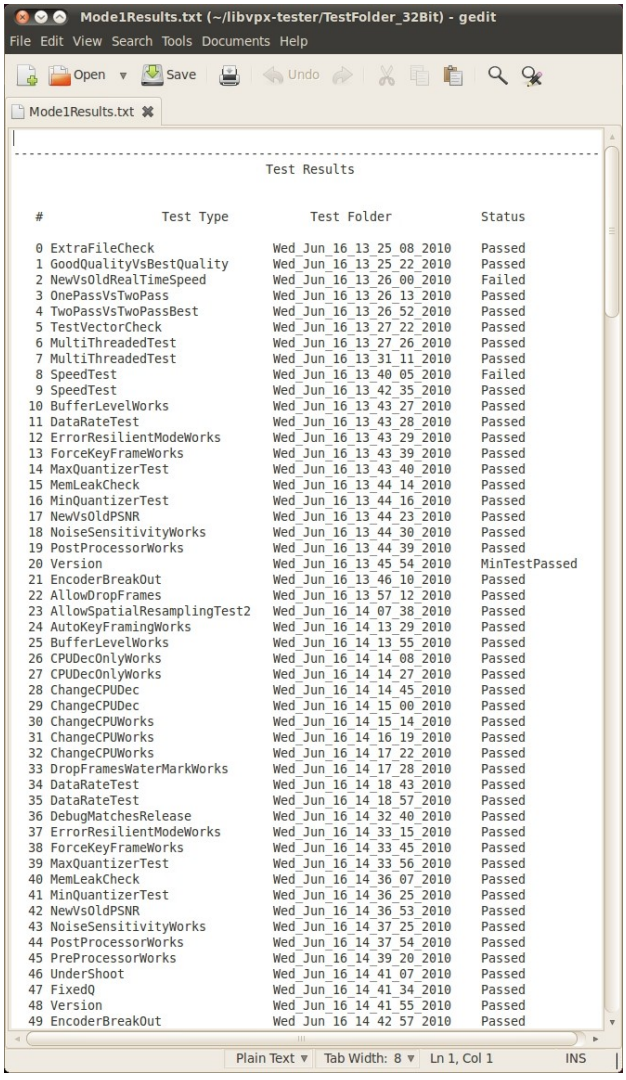
To create a new multiple test input text file create a new text file. Any line that starts with a "%" will be considered to be commented out so if you would like to add an identifying header at the beginning of the text file or notes through out the file you can. The "@" symbol serves as a delineator for text file arguments so to write a new test go to a new line and type the number of the test you wish to run followed by an "@" then type in the test specific input you wish to have run using an "@" between each new parameter. When finished run the file in the same way the sample file is run.

Evaluating a Multiple Test's Results

When a Multiple Test finishes, the results summary will be saved in two formats. A condensed results file (Mode1Results.txt, Mode2Results.txt, Mode3Results.txt) and an expanded results file (Mode1Results_Expanded.txt, Mode2Results_Expanded.txt, Mode3Results_Expanded.txt).

The condensed results summary contains the type of test that was run, the order it was run in, the folder name of the in depth test output, and final Pass/Fail status of the test. This file lists test run in chronological order. (See picture below)

The expanded results file groups results for individual test types together. In addition to the information contained in the condensed results file the expanded file also lists the parameters under which the test was run and a summary of how many total passes and fails occur for each type of test.



Tests Overview

0	RunTestsFromFile	The test runs tests from an input file and summarizes the results. The test can be run in four modes: Mode 1 - Create compressions and run tests, Mode 2 - Create compressions only, Mode 3 - Run tests on pre-existing compressions, and Mode 4 - Resume tests in progress. To create a template driver text file use the command: CreateSampleTextFiles.
1	AllowDropFrames	The test creates two files; the first with Drop Frames on, the second with Drop Frames off. It then records and compares the number of frames each file has. If Drop Frames on has fewer frames than Drop Frames off; the test passes.
2	AllowLagTest	The test creates two compressions; the first with Allow Lag equal to 0, the second with Allow Lag equal to 1. If the correct number of lagged frames are detected via quantizer output, alternate reference frames exist for Allow Lag on, Allow Lag on has the same number of visible frames as Allow Lag off, and Allow Lag on is not identical to Allow Lag off; the test passes.
3	AllowSpatialResampling	The test creates two files the first with Spatial Resampling off the second with Spatial Resampling on. The test then records the number of resized frames for each and computes the PSNR for Spatial Resampling on.
4	AltFreqTest	
5	AutoKeyFramingWorks	The test creates two files with identical parameters setting Auto Key Frame equal to 6. The test then records the placement of each files key frames. If both files key frames occur in identical locations and at least as frequently as Auto Key Frame dictates; the test passes.
6	BufferLevelWorks	The test creates a compression and runs CheckPBM on it. If no buffer under run is detected; the test passes.
7	CPUDecOnlyWorks	The test creates a compression of the user input version (0-3) and then decompresses it for ON2_SIMD_CAPS values ranging from 0 to 11. The test then compares them against one another. If all compressions are identical and the times to decompress them are not; the test passes.
8	ChangeCPUWorks	The test creates compressions of the user input version (0-3) with ON2_SIMD_CAPS values 0 1 3 7 and 15. If all compressions are identical and compression times are not the test passes.
9	DropFramesWaterMark-Works	The test creates 6 compressions with DFWM values of 100 80 60 40 20 and 0 and records their sizes. If each successively lower DFWM compression has a equal or larger size than the previous the test passes.
10	DataRateTest	The test creates a compression and records it data rate. If the compressions data rate is within 30% of the input target bandwidth; the test passes.
11	DebugMatchesRelease	The test creates two compressions the first using an executable built using the newest release library, the second using an executable built using the newest debug library. If the two compressions are identical; the test passes.
12	EncoderBreakOutTest	The test creates four compressions. The first with an EncodeBreakout of 0, the second with an EncodeBreakout of 100, the thrid with an EncodeBreakout of 500 and the fourth with an EncodeBreakout of 1000. Decompressions of the encoded files are then carried out and PSNR values are calculated. If the decompressions run successfully and the PSNR values of each successive EncodeBreakout trial are with in 2 dB the test passes. If the PSNRs are greater than 2 dB but less than 5 dB the test is inconclusive and if the PSNRs have greater than a 5 dB difference the test fails.
13	ErrorResilientModeWorks	The test creates two compressions the first with Error Resilient Mode off the second on. The test then records their PSNR values.If their PSNRs are with in 10% of one another the test passes.
14	ExtraFileCheck	The test creates a two pass compression and checks the current directory, the directory the executable is located in and the directory the output file is written to for extra files. If no extra files are found the test passes.

15	FixedQ	The test creates two compressions each with user input Fixed Quantizers and records the quantizers used to encode each frame. The test then records the compressions sizes. If all quantizers for each compression match the input Fixed Quantizer and the smaller quantizer's compression has a larger file size; the test passes.
16	ForceKeyFrameWorks	The test creates a compression using a user input value for a Force Key Frame Interval. The compressor forces a key frame for every Force Key Frame Invervalith frame. The Test then records the placement of all key frames in the compression. f key frames occur only when Force Key Frame dictates; the test passes.
17	GoodQualityVsBestQuality	The test creates six compressions. The first and fourth compressions for 30% less than the input target bandwidth at good quality and best quality, the second and fifth compressions for the input target bandwidth at good quality and best quality, and the thrid and sixth at 30% more than the input target bandwidth at good quality and best quality. The test then records each files data rate and PSNR and computes the area under the curve for the common interval between the good quality curve and best quality curve. If the area under the best quality curve is greater than the area under the good quality curve; the test passes.
18	LagInFramesTest	The test creates three compressions one with Allow Lag set to 0 the second and third with Allow Lag set to 1. The second compression uses the first user input Lag in Frames value for its Lag in frames and the third uses the second user input value for its Lag in Frames. The test outputs each files quantizer values for each encoded frame. If none of the files are identical, the PSNRs of each successive file are within 10% of the last and the quantizer output shows that the proper number of frames were lagged; the test passes.
19	MaxQuantizerTest	The test creates nine files the first with a WorstAllowedQ equal to 3 and each subsequent file with a WorstAllowedQ eight greater than the last until 63. The test records the individual quantizer values for each encoded frame. If the PSNRs of each WorstAllowedQ compression from 3 to 63 increase as Worst AllowedQ decreases and the recorded quantizers for each file do not exceed their corresponding WorstAllowedQ for all compressions; the test passes.
20	MemLeakCheck	The Test creates a compression using the debug executable to check memory usage and records the results to an output file. If no memory leaks are found the test passes.
21	MemLeakCheck2	The test uses the debug executable to open and close 10,000 instances of the encoder and open and close 10,000 instance the decoder and then checks to make sure there are no memory leaks. If there are no leaks the test passes.
22	MinQuantizerTest	The test creates two files the first with a MinQ equal to 10 the second with a MinQ equal to 60 and records the quantizer used for each compressions frames. If the first file has a higher PSNR than the second file and every quantizer for both files is above the corresponding MinQ; the test passes.
23	MultiThreadedTest	The test creates two compressions the first using a MultiThreaded equal to 2 the second using a MultiThreaded equal to 0. The test then compares the times to compress each. If MultiThreaded 2 is faster than 0; the test passes.
24	NewVsOldPSNR	The test creates two compressions the first using the newest version of VP8 and the second using a separate executable built using an older version. It then computes and records PSNR values for each. If new PSNR is greater than olds PSNR or is at least within 1% of the old; the test passes.
25	NewVsOldRealTimeSpeed	The test creates two compressions the first using the newest version of VP8 and the second using a separate executable built using an older version. The test records the time that each compression took. If the new compression's time is at least 10% faster than the old compression's time; the test passes.
26	NoiseSensitivityWorks	The test compresses seven files with Noise Sensitivity values from 0 to 6 and computes PSNR values for each. If all compressions have differing PSNR values and Noise Sensitivity 0 has a higher PSNR than Noise Sensitivity 6; the test passes.

27	OnePassVsTwoPass	The test creates six compressions. The first and fourth compressions for 30% less than the input target bandwidth at one pass good quality and two pass good quality, the second and fifth compressions for the input target bandwidth at one pass good quality and two pass good quality, and the thrid and sixth at 30% more than the input target bandwidth at one pass good quality and two pass good quality. The test then records each files data rate and PSNR and computes the area under the curve for the common interval between the one pass good quality curve and the two pass good quality curve. If the area under the two pass good quality curve is greater than the area under the one pass good quality curve; the test passes.
28	PlayAlternate	The test creates two compressions the first with Play Alternate equal to 0 the second with Play Alternate equal to 1. The test then records the placement of alternate reference frames and visible frames for both compressions. If alternate reference frames exist for Play Alternate = 1 and not for Play Alternate = 0, visible frames for Play Alternate 1 and Play Alternate 2 are equal, and the files are not identical; the test passes.
29	PostProcessorWorks	The test creates a compression then creates a No Filtering decompression, decompressions for Deblock and Noise levels ranging from 0 to 15. If all Deblock and Noise decompressions return a different PSNR than the No Filtering Decompression but are within 10%; the test passes.
30	ReconBuffer	The test creates a compression and internally compares the compressor's preview frames to the decoded output produced by decompressing the compressor's encoded frame. The state of each frame is recorded to a text file. If the contents of all preview frames are identical to the content of all decoded frames; the test passes.
31	ResampleDownWaterMark	The test creates two files the first with resample-down-watermark set to 90 the second with resample-down-watermark set to 10. The test then records the frames at which the file buffer reaches the designated thresholds, the location of key frames and location of resized frames for both files. If the first resized frame occurs on the first instance where the frame prior to a key frame reaches the correct buffer saturation for both compressions; the test passes.
32	SpeedTest	The test works for RealTime Mode and Good Quality Mode. For Real Time Mode the test creates compressions for CpuUsed Values from -1 to -16 and 0 to 16. For Good Quality Mode the test creates compressions for CpuUsed Values from 0 to 5. If compression speed increases as CpuUsed increases and all PSNRs are within 10% of the previous; the test passes.
33	TestVectorCheck	This test decodes each VP8 Test Vector and Checks its MD5 checksum against the expected value. f all Test Vectors decode properly and all MD5 checksums match their expected values; the test passes.
34	TwoPassVsTwoPassBest	The test creates six compressions. The first and fourth compressions for 30% less than the input target bandwidth at two pass good quality and two pass best quality, the second and fifth compressions for the input target bandwidth at two pass good quality and two pass best quality, and the third and sixth at 30% more than the input target bandwidth at two pass good quality and two pass best quality. The test then records each files data rate and PSNR and computes the area under the curve for the common interval between the two pass good quality curve and the two pass best quality curve. If the area under the two pass best quality curve is greater than the area under the two pass good quality curve; the test passes.
35	UnderShoot	The test creates two files the first with an undershoot equal to 10 the second with an undershoot equal to 100. If the Undershoot 100 compressions file size is greater than the Undershoot 10 compressions file size; the test passes.
36	Version	The test creates four compressions the first with Version equal to 0 the second with version equal to 1 the third with version equal to 2 the fourth with version equal to 3. The test then decodes each and records the time it took to do so. If each successive version takes less time than the prior to decode and has a lower PSNR; the test passes.

37	WindowsMatchesLinux	The test can be run in two test modes. The first Mode, 0 creates platform specific compressions and decompressions to be tested on another platform. The second Mode creates platform specific compressions and decompressions and then compares them to previously encoded and decoded files created by test mode 0. If the files are identical the test passes.

Tools Overview

IVFEnc	IVFEnc.exe clone
IVFDec	IVFDec.exe clone
IVF2IVFCompr	This utility will take in a raw ivf file and produce an encoded ivf file using the given mode and bitrate. Default encode settings can be overridden by specifying a parameter file.
IVF2IVFDec	This utility will take in an encoded ivf file and output a decoded ivf file.
IVF2RawDec	This utility will take in an encoded ivf file and output a decoded raw file.
IVFDataRate	This utility will take in an ivf file and compute its average, min, max, and file data rates.
IVFPSNR	This utility will compute an encoded files psnr using the encoded file's ivf source file.
IVFCheckPBM	This utility will run CheckPBM to make sure a buffer under run wont occur.
Raw2IVF	This utility will take in a raw file and produce a raw ivf file.
IVF2Raw	This utility will take in an ivf file and produce a raw file.
IVF2RawFrames	This utility will take in an ivf file and produce individual raw frames for each frame that exists in the user specified directory.
CombineIndvFrames	This utility will combine all individual decoded frames in a folder into a single raw file in numerical order.
CompareIVF	This utility will compare the video content of two ivf files and will display if they are identical, or if they differ the first frame they differ at
CompIVFHeader	This utility will compare the file and frame headers of two ivf files.
DispIVFHeader	This utility will display the file and frame headers of an ivf file.
DispKeyFrames	This utility will display the location of key frames within an ivf file.
DispResizedFrames	This utility will display the location of resized frames within an ivf file.
DispVisibleFrames	This utility will display the location of visible frames within an ivf file.
DispAltRefFrames	This utility will display the location of alternate reference frames within an ivf file.
CutIVF	This utility will cut a portion of an ivf file starting at Starting Frame and ending at Ending Frame to a new output file
PasteIVF	This utility will paste the contents of Inputfile2 into Inputfile1 starting at Inputfile1's First Paste Frame to a new Outputfile.
PlayDecIVF	This Tool will convert an uncompressed ivf file to a raw yuv file and play it using tmnplay or mplayer.
PlayCompIVF	This Tool will convert a compressed ivf file to a raw yuv file and play it using tmnplay or mplayer.
CreateSampleTextFiles	This utility will create sample text files.
PrintVersion	This utility will print the version of vp8 being used by the tester.
RandParFile	This utility will create a valid vp8 random parameter file.
RandIVFComp	This utility will create a compression using random parameters for an input ivf file.
GraphPSNR	The utility creates compressions from user input starting to user input ending bit rates at user input steps. The utility then computes and outputs the data rates and PSNRs of the resultant files.
Help	Displays Tester Help instructions