StormTest Video Quality Analysis

StormTest® Development Center Application Note

Document ID: ST-15007

Revision Date: December 2015

Product Version: 3.2.5

Web: http://www.s3tvtechnology.com

Contents

1 Preface

1.1 StormTest

StormTest® Development Center is the leading automated test solution for digital TV services. It is designed to reduce the cost of getting high quality digital TV services to market faster.

StormTest Development Center greatly reduces the need for time-consuming, expensive and error-prone manual testing and replaces it with a more accurate and cost-effective alternative. It scales easily to large numbers and types of devices and integrates with existing infrastructure to give much greater efficiency in testing. It can be used to verify and validate services on a virtually every piece of consumer premises equipment (CPE), from set-top boxes to games consoles and from iPads to Smart TVs. It has been specifically designed to meet the needs of developers and testers of these CPE devices and the applications which run on them.

StormTest Development Center consists of:

- A choice of hardware units that can test 1, 4, or 16 devices. Each device under test can be controlled individually and independently and the audio/video from each device can be captured and analysed to determine the outcome of the test. The StormTest hardware supports capture of audio and video over HDMI interfaces and supports all HD resolutions up to 1080p. In addition there is a hardware upgrade option for the 16 device tester that will allow native capture of UHD content.
- Server software that controls all the hardware and devices in the rack as well as managing a central repository of test scripts and a central database of test results.
- A Client API that allows test scripts to interact with the server software
- A number of graphical tools that allow the user to directly control devices connected to StormTest Development Center, to create and schedule tests to run and to view the results of those test runs.

Test scripts can be run from any location – the tester needs only a network connection to the StormTest server. Video and audio output from the devices under test can be streamed over this network to any location, allowing remote monitoring and control of testing, either within a company LAN or across a WAN. Alternatively, scheduled tests can run directly on the server, negating the need for maintaining a continuous network connection to the StormTest server.

1.2 About This Document

This document describes how to use the StormTest Live Video Quality Analysis feature (or VQA).

1.3 Related Documentation

The StormTest user documentation set comprises of the following documents:

1) StormTest Developer Suite User's Manual
2) StormTest Programmer's Guide
3) StormTest Client API

4) StormTest Hardware Installation Guides (HV01, HV04, HV16)
5) StormTest Software Installation Guide
6) StormTest Server Monitor User's Manual
7) StormTest Administration Console User's Guide
8) StormTest Administration Tools User's Guide

The latest version of these documents can always be found on our support website, in the "Docs" section: https://larisa.engage.s3group.com/docman/?group_id=6.

1.4 Definitions, Acronyms and Abbreviations

Item	Description
DUT	Device Under Test
RPS	Remote Power Switch
UPS	Uninterruptible Power Supply
UI	User Interface
STB	Set Top Box
PVR	Personal Video Recorder
VQA	Visual quality analysis
SAMVIQ	Subjective Assessment Methodology for Video Quality
ACR	Absolute Category Rating
MOS	Mean Opinion Score

Table 1: Acronyms

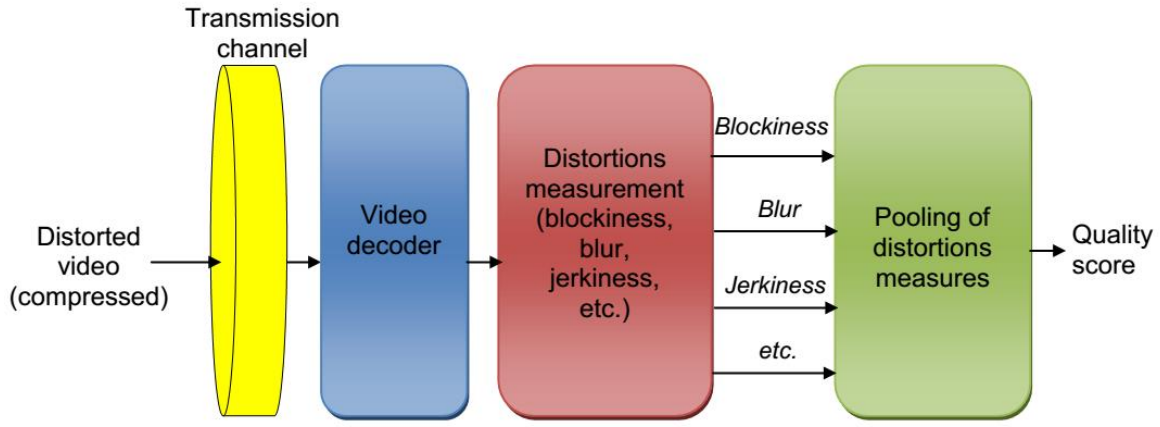
2 Introduction

The VQA, or Video Quality Analysis feature in StormTest is an algorithm working on audio/video content that can calculate quality parameters without using a reference stream.

No Reference (NR) metrics use only the data from the distorted video to produce an objective quality score for this video. These metrics don't have any data about the actual video. However these metrics use a priori information about the distortions. For example, they know the type of encoding scheme that was used so that they can look for codec-specific distortions: blockiness, blur, etc...

So No Reference metrics explore the video frames at pixel level in order to detect and measure expected distortions. These distortions measurements are then pooled to compute the video quality score.

The typical functioning of an NR metric is described in figure below.



In StormTest, the VQA engine is run using raw data from the capture cards.

The VQA engine can also measure audio quality based on bandwidth estimation, audio breaks, saturations and silences.

3 Pre-requisites

- VQA is only available on StormTest HD systems
- VQA requires at least StormTest version 3.2.3 for video analysis. At least version 3.2.5 is required for audio analysis.
- VQA is run on the video capture server. For an HV01HD or an HV04HD, this is the main unit. For an HV16HD, VQA is run on one or all of the video servers in the rack.
- Each video server that runs VQA requires a VQA license. So, for an HV01HD or and HV04HD a single VQA license is required. For an HV16HD, up to 4 VQA licenses may be required if all 4 video servers will need to run VQA tests.
- As this is no reference quality measurement, the type of encoding scheme of the stream should be known (MPEG2 or H264)

4 Restrictions

- The VQA engine can only be run on one slot per server at a time. (if you have an HV16-HD with 4 licenses, one on each video server, it can be run on 4 slots in parallel)
- Video should be analysed at its native resolution for accurate results (if the source content is SD, set to the STB output to an SD resolution)
- When VQA is running, the other 3 slots on that video server can be used for typical non VQA tests. The High Speed Timer API can be used, but as it shares memory with the VQA engine, it will reduce the available memory for the VQA engine.
- Only MPEG2 and H264 encoded content are supported for now. HEVC support is on the roadmap.

5 Using the Engine

5.1 Input

The VQA feature works using a VQA engine, which is a python object. This is the core engine. You can create this object directly or use `VQACreateEngine()` function. This engine can be passed to functions such as `VQAStart()` or you can call methods directly on this object.

The object parameters needs to be set before the analysis for Video and Audio.

5.1.1 Video Options

This holds all the options that can be configured for VQA video analysis. You can create a default object with the default parameters and adjust those that are necessary.

5.1.1.1 Enabled

Whether video analysis should be performed. This is a new property in 3.2.5 to allow the options of audio only VQA analysis. Default is True.

5.1.1.2 Format

The format of the source of the video that was supplied to StormTest. Most HD Video uses H.264 and this is the default value for the Format value. You should change this to MPEG2 ONLY if you know for certain that the video was originally encoded using MPEG2. StormTest processes raw video frames but these are supplied by a device - that device will have decoded the video from either H.264 or MPEG2 video. Getting this value 'wrong' will alter how the VQA engine assesses the quality of the video and thus may generate unexpected scores for the quality. If another format was originally used, then the VQA results from StormTest are not necessarily representative of human perception of quality.

5.1.1.3 Backlog

The number of frames used as a backlog queue. It is possible that video is captured more quickly than it can be analysed. In this case, StormTest keeps a queue of pending frames. When that queue is full, then frames are dropped. This affects the MOS scores. Setting a high value uses more memory but allows a longer period from the start of video analysis before frames are dropped. The minimum value is 16. The maximum depends on available memory: for an HD video server with 8GB of RAM then the maximum is around 1500 for 1920 x 1080 frames. The default value is 500.

The memory used is shared with the High Speed Video Timing API. The total memory available varies slightly over time due to dynamic memory allocation/deallocation in the server. However, it has been verified that if the total memory requested between VQA and all slots of High Speed Timer on the slave is less than 4.6656 billion then the requests will be satisfied. (On HV16HD, there are four slaves, slots 1-4 being the first slave). Each image of video requires width x height x 1.5 bytes. So from the above calculation, you could allocate all memory to VQA and at 1920 x 1080 that would mean a maximum backlog of 4665600000 / (1920 x 1080 x 1.5) => 1500. Alternatively, you could allocate 375 to each slot of high speed timer and not use the VQA.

Real-time video quality analysis can be demanding on system resources. In some scenarios, further tuning of the VQA engine and options may be desired.

How to set this value?

This is one of the main tuning points of the VQA engine. In these sections it is explained that as each frame arrives for quality analysis, it is stored into a backlog queue until it can be processed. By default, this backlog is set to store up to 500 frames. This value can be changed by setting the desired option for the VQA engine as per examples above:

```
vqaOptions.Backlog = 1500                                    # increase backlog
```

As noted earlier, for 1920x1080 resolution, the backlog can hold up to a maximum of about 1700 frames. If the backlog setting calls for more RAM than is available, StormTest will generate an error similar to the following:

```
ERROR  : Exception in VQAStartAnalysis: StormTestException: Can't start VQA analysis: The
memory request to High Speed Timer/Video Analyzer exceeds available memory
```

Most StormTest systems capable of performing Video Quality Analysis are equipped with 8GB of RAM. When idle, a typical system uses about 2GB of RAM. The remaining 6GB are available for use by tests, High Speed Timer and VQA. StormTest by default allows the VQA engine access to these entire 6GB or free RAM. If each frame at 1920x1080 is roughly about 3.5MB, the maximum backlog queue can hold approximately 6GB/3.5MB which comes out to about 1700 frames. For lower resolution streams, the queue can be increased significantly to utilize the free RAM.

The default value of 500, can contain can contain roughly about 16 seconds worth of 1920x1080 frames worth of video consuming less than 2GB of RAM, while a Backlog of 1700 can hold about a minute worth of frames consuming about 6GB or RAM.

The rate at which each frame is processed will dictate the amount of backlog required in order to avoid dropping any frames from being processed for the duration of the test. It should be noted here that it is probably not the best idea to always set the Backlog to 1700. Depending on other activities on the system, the full 6GB of RAM may not always be available. In such scenario, if Backlog option is set to its maximum value, the results obtained from VQA after a certain point, when the RAM is completely utilized, will likely be distorted or otherwise useless.

5.1.1.4 FrameRate

The frame rate to use for analysis. If the actual frame rate is greater than this value then frames are dropped in a deterministic manner to bring the frame rate down to the FrameRate value. if you are monitoring video for an extended period and the engine cannot process all frames then a deterministic drop of frames will typically give better results (more consistent) than the non-deterministic dropping on a full queue. Best results are achieved where the number of frames dropped can be expressed as

1 in n where n is integer, for example 20 for a 25 fps input video (1 in 5 frames dropped) A value of 0 should be used to mean 'same as input rate'

How to set this value?

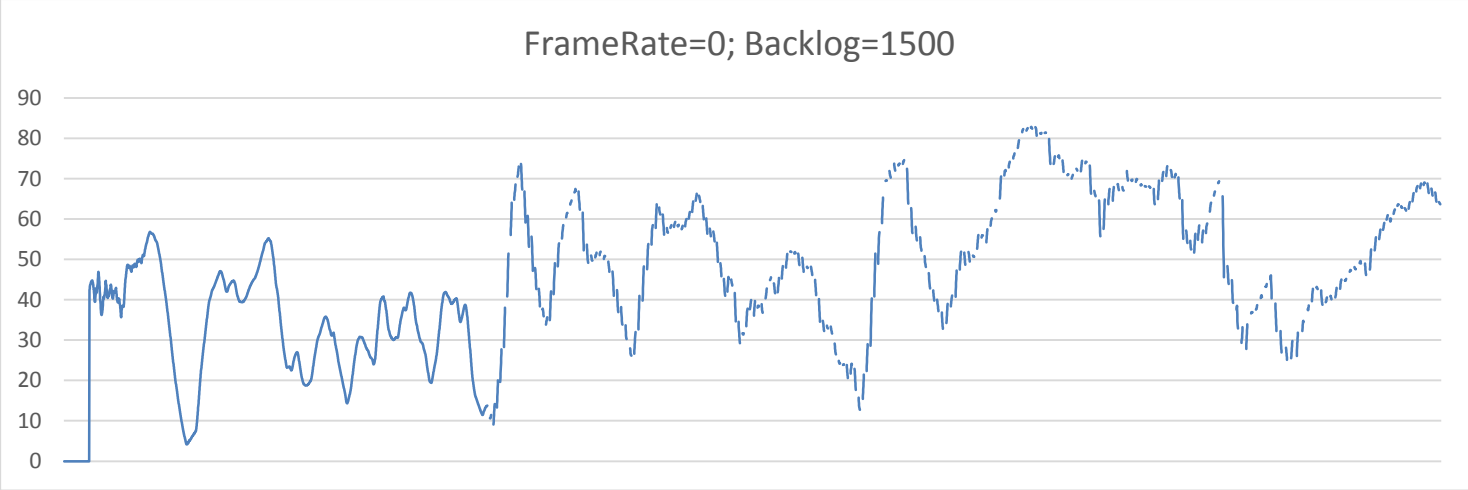
If after setting your backlog setting, the VQA is still struggling to keep up with video processing without dropping frames, further action may be possible.

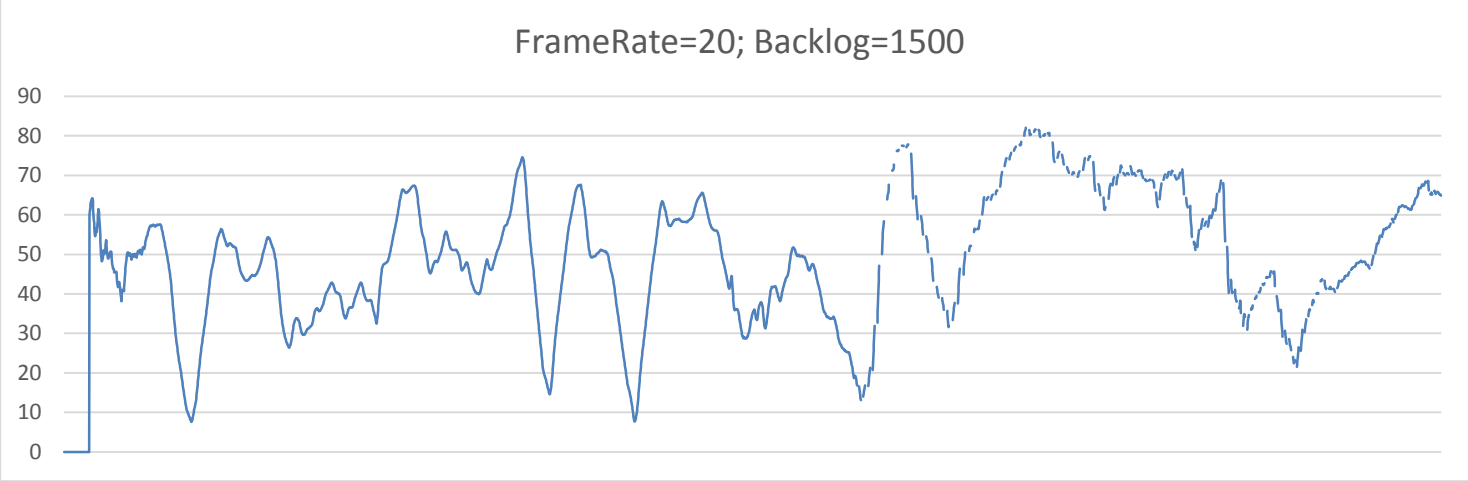
If the VQA engine has drop frames when it is unable to process them fast enough, it will do so indeterminately. If this occurs, it may be a good idea to decide to drop frames deterministically (e.g. every other frame). This can be achieved by varying the FrameRate option when setting up the VQA engine.

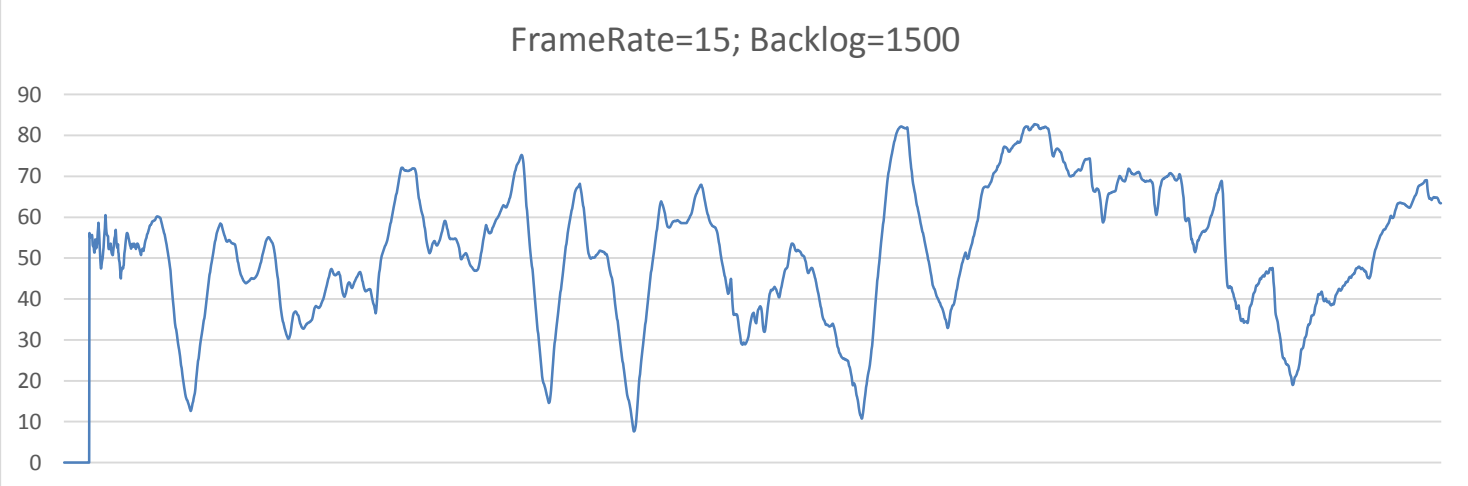
```
vqaOptions.FrameRate = 20
```

Note that changing the FrameRate option will impact the results of the video quality analysis. On the next page, take a note of the differences between results of the same stream analysed at incoming framerate, and when frames are being dropped deterministically using FrameRate option (i.e. every third frame being dropped, then every other frame being dropped).

In the following example, the same input stream was used for analysis. The stream in question is at resolution of 1920x1080 at 30fps. The exercise was performed a number of times with different FrameRate option settings. Note that FrameRate=0 setting means that the VQA will process incoming frames at the rate they arrive in. The following are MOS results observed from each test run. Note the gaps in the graphs due to dropped frames when the VQA engine is unable to process all frames fast enough. FrameRate option has a significant impact on this.







5.1.1.5 Jerkiness

Set to true to permit 'jerkiness' or 'image freezing' of the video to affect MOS score. If you expect smooth video then you should set this to True. If, however, you expect your video to be jerky with a lot of static frames then you may get a more realistic MOS score by setting this to False. The default is True. You can adjust the detection parameters of what is considered a 'static picture'. A picture is moving when it exceeds the Activity OR the ChangingPixels OR the MeanBrightness thresholds.

True: MOS values can be decreased by jerkiness

False: jerkiness will be ignored in MOS calculation

5.1.1.6 Activity

A threshold indicating temporal activity, above which, we consider the video to be moving. It ranges from 0 to 255. Values lower than 5 are consider low motion, 5 to 15 is normal motion. Default value for Activity is 0.9 - only very slow moving pictures are considered 'static'. This parameter has no effect if Jerkiness is False

5.1.1.7 ChangingPixels

The percentage of pixels (0 - 100) which have a significant change compared to the previous frame. If the number exceeds this value, then the picture is considered moving. The default is 0.01. This parameter has no effect if Jerkiness is False

5.1.1.8 MeanBrightness

The average value of brightness, above which the image is considered moving. The default value is 21 meaning that if the image is not black, it is considered to be moving. This means that jerkiness is likely to be detected only on very dark pictures or black screens. Setting this to a high value would allow freezing to be detected on any brightness of screen. This parameter has no effect if Jerkiness is False

5.1.1.9 Example

```
video_params = StormTest.VQACreateVideoOptions()

video_params.Enabled = True
video_params.Backlog = 350
video_params.FrameRate = 25
video_params.Format = "H264"
video_params.Jerkiness = True
video_params.Activity = 5
```

5.1.2 Audio Options:

This holds all the options that can be configured for VQA audio analysis. You can create a default object with the default parameters and adjust those that are necessary.

Audio quality analysis is performed on the audio captured from the DUT. Specifically, the audio from the currently selected audio channel is analysed. StormTest supports 2 channel stereo PCM audio input, so the user can select which channel to analyse by calling `SetAudioAnalysisChannel()` before starting the audio analysis. The default channel if this API is not called is the **left** channel.

5.1.2.1 Enabled

Whether audio analysis should be performed. Default value is True.

5.1.2.2 SilenceInfluence

The percentage influence (0 - 100) that the silence detector has on the MOS score. Silence is defined as 100mS or more of consecutive samples all having the same value. The final MOS score is reduced by SilenceInfluence MOS score. So a value of 0 disables the silence detector while 100 causes any silence detected to push the MOS score to 0. A value of 10 will remove 10% from a MOS score so if the raw MOS was 85 and silence was detected during the sample, the resulting MOS score is 76.5 (. Default value for SilenceInfluence is 0 (no effect).

5.1.2.3 BreaksInfluence

The percentage influence (0 - 100) that the break detector has on the MOS score. A break is defined as a significant difference between 2 consecutive samples that is not normal in audio. As with SilenceInfluence, the MOS score is reduced by BreaksInfluence MOS score. Default value for BreaksInfluence is 0 (no effect).

5.1.2.4 SaturationInfluence

The percentage influence (0 - 100) that the saturation detector has on the MOS score. Saturation is defined as 2 consecutive samples having either the maximum or minimum possible value. As with SilenceInfluence, the MOS score is reduced by SaturationInfluence MOS score. Default value for SaturationInfluence is 0 (no effect).

5.1.2.5 CustomSilenceInfluence

The percentage influence (0 - 100) that the custom silence detector has on the MOS score. As with SilenceInfluence the MOS score is reduced by CustomSilenceInfluence MOS score. Default value for CustomSilenceInfluence is 0 (no effect).

Note: All the Influence parameters work on the original MOS score. So if SilenceInfluence is 40 and BreaksInfluence is 40 and both occur on a MOS score of 85 then the MOS score will be 17. There is no compounding of influences and this means the MOS score can become negative.

5.1.2.6 CustomSilence

The type of custom silence detection used. Default value is disabled. The custom silence detector works when any sample meets the criteria for silence unlike the standard silence detection which requires 100mS of samples to be 'silent'.

Defines the valid values of custom audio silence detector supported by VQA

- Disabled : The custom audio silence detector is disabled.
- Amplitude: The custom audio silence detector examines the value of audio samples (the amplitude) looking for values below a threshold.
- StdDev : The custom audio silence detector examines the standard deviation of a group of samples (1 frame, equal in length to the video frame time) and calculates the standard deviation of the amplitudes and examines that in order to determine silence.

5.1.2.7 Threshold

The threshold for the custom silence detector. In amplitude mode, this is in dBu using the same scale as the other audio detection functions. If a sample is below or equal to this value then it is considered silence. In standard deviation mode, the value is in the range 0 - 1.0 indicating a standard deviation between 0 and maximum audio sample value (not possible in practice so a value of 1.0 will always detect silence as the deviation will always be lower than 1.0).

5.1.2.8 Example

```
audio_params = StormTest.VQACreateAudioOptions()

audio_params.Enabled = True
audio_params.SilenceInfluence = 0
audio_params.BreaksInfluence = 0
audio_params.SaturationInfluence = 0
audio_params.CustomSilenceInfluence = 0
audio_params.CustomSilence = "Amplitude"
```

5.2 Launching the Analysis - Trigger

The start of the analysis could be delayed, or subject to a condition. For example the appearance of a specific Image in the live stream. This is done using a Trigger condition object. It is possible to specify multiple conditions on one trigger object and in this case the condition is satisfied when any one of the conditions is met. If any property is None then it is ignored in considering the trigger. This is the default value for many properties.

The following condition parameter could be used for a trigger:

5.2.1 Delay

A time in seconds. Default value is None.

5.2.2 FrameCount

Number of frames as an integer. Default value is None.

5.2.3 DroppedFrames

Number of dropped frames. Useful to trigger on an error by settingto 1. Default value is None.

5.2.4 IRKey

An IRKey stroke to use to trigger the analysis. Default value is None.

5.2.5 IRKeyRepeat

The number of times that the IRKey must be present before it is considered the trigger. Default value is 1 meaning the first occurrence of the IRKey will cause the trigger to be considered triggered.

5.2.6 Sdo

A screen definition to match each frame. The Sdo may only include image, and color tests. Motion, audio and OCR are forbidden as they take too long. Youshould choose small regions to test so that the SDO can execute in less than 1 frametime. A full image compare cannot operate in real time. In this release, you cannot use NamedRegions, NamedColors or NamedImages within an SDO. Default value is None

5.2.7 SdoRepeat

Number of frames that must match the SDO for it to be considered a valid trigger. Default value is 1.

5.2.8 FromNow

True to consider the delay, frame count etc to start 'now'. If False, then the reference point is the most recent logical point in the past. This is not always sensible. A useful case is setting it to False on the Stop() call with a fixed delay. This means the Stop() will be exactly the specified value after Start() even if the script is a bit slow in calling Stop(). Useful for precise timing. Default value is True.

5.2.9 Example

```
#create a trigger to trigger after 1000 frames

trigger = stormtest.VQACreateTrigger()
trigger.FrameCount = 1000

vqa = stormtest.VQACreateEngine()
stormtest.VQAStartAnalysis(vqa , trigger)
```

```
# create the stop trigger to stop analysis after 60s of analysis

stop_trigger = stormtest.VQACreateTrigger()
stop_trigger.Delay = 60
# stop VQA on stopTrigger
stormtest.VQAStopAnalysis(vqa, stop_trigger)
```

5.3 Retrieving and understanding the data

The engine fires event giving status about the engine itself (if it started, stopped) and also the result of the audio and video analysis. A call-back mechanism is used to retrieve those information.

5.3.1 Video result

5.3.1.1 Frame Number

The frame number of this result. The first frame analysed is frame 0. Frame numbers are counted from the captured frame and include all dropped frames (whether intentionally or not). To be clear, if the queue is NOT overloaded and VQAVideoOptions.FrameRate is set to 15 for a 30 frame/sec source then you will get results with frameNumber value: 0, 2, 4, 6, 8 ... due to the deliberate dropping of frames.

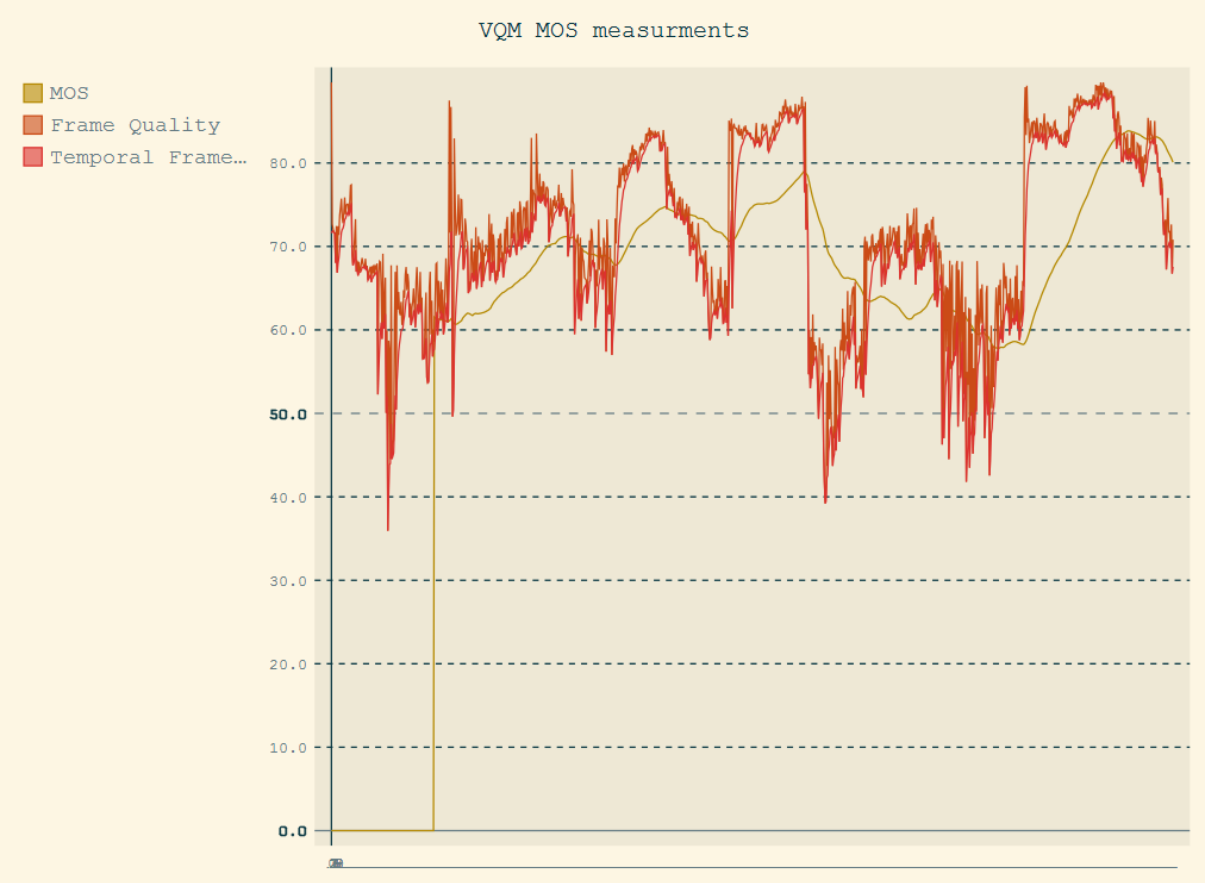
5.3.1.2 TimeStamp

The timestamp of the result. This is in seconds since the Unix epoch (1/1/1970 00:00:00 UTC) and is the time on the server when the result was generated. This may increase unevenly due to threading and CPU load. It is useful for debugging should problems occur.

5.3.1.3 MOS

MOS means Mean Opinion Score. The Mean Opinion Score (MOS) is the mean value of human observers' votes during subjective quality assessment tests (in which observers are asked to judge the quality of videos that are presented to them). The Mean Opinion Score of a video is its quality score. Therefore, if the MOS is between 80 and 100, it means that the distorted video has an excellent quality. And when MOS decreases, quality decreases.

Humans are not able to precisely assess the quality of each frame (because the frame rate is too high for that). Therefore, when human observers formulate a quality judgment, they perform a temporal integration over the feelings they had in the few previous seconds. That's what the MOS curve does too. Each point of this curve is an integration over the previous 5 seconds of frames quality scores (with temporal effect). That's why the MOS curve only starts at the 125th frame when your video has a frame rate of 25.0 fps (because 25.0 * 5 seconds = 125 frames, and 125 quality scores are required to compute a MOS). Figure below is an example of the temporal effect (**note the first 5 seconds of analysis where MOS is null**)



Typical values

Perceived video quality is expressed using a MOS scale. MOS (Mean Opinion Score) indicates the visual quality of the measured video. The MOS scale ranges from 0 to 100, as shown in figure below.

MOS=100 indicates that human observers would say the video quality is excellent.

MOS=0 indicates that observers would perceive so many distortions that they gave the lowest possible quality score to the measured video.

MOS ranges from 0 to 100. It corresponds to these quality levels:

between 100 and 80: excellent quality

between 80 and 60: good quality

between 60 and 40: rather good quality

between 40 and 20: poor quality

between 20 and 0: bad quality

These adjectives: excellent, good, etc. are defined by ITU standards (like BT.500 and BT.710) and normalized subjective voting protocols (like SAMVIQ and ACR). They were the adjectives shown on

the voting scale used by the human observers during the subjective video quality assessment tests whom results were employed to validate VQA's objective video quality metrics.

These terms are defined in standards such as BT.500 and BT.710. The MOS score is an average of the frame qualities over time (as no human can determine the quality of a single frame at 30 frames per second). **It uses a 5 second window for the averaging so the first 5 seconds worth of values after starting are not valid.**

5.3.1.4 MOSDescription

The String equivalent of the MOS score without the word Quality (i.e. Bad, Poor, Rather Good, Good or Excellent)

These terms are defined in standards such as BT.500 and BT.710. The MOS score is an average of the frame qualities over time (as no human can determine the quality of a single frame at 30 frames per second). **It uses a 5 second window for the averaging so the first 5 seconds worth of values after starting are not valid.**

5.3.1.5 FrameQuality

The quality of a single frame. The range and meaning is the same as MOS score. MOS score is the average of the raw frame qualities. The FrameQuality ignores the effect of prior frames.

5.3.1.6 FrameQualityTemporal

The quality of a single frame taking into account the previous frames' scores. This takes into account a property of the human visual system which is that we are "*quick to criticize, slow to forgive*". In order to understand this sentence, consider human observers watching a good quality video. Now imagine that a visible distortion appears. When the distortion appears, the quality judgment of the human observers decreases nearly instantly *("quick to criticize*"). However, when this distortion disappears, their human judgment takes several seconds to come back to its previous state that was judging that the video has a good quality ("*slow to forgive*"). This is the temporal effect implemented in this score computation. It indicates the quality of each frame, taking into account the previous judgments.

5.3.1.7 Jerkiness

Jerkiness can also be called Image freezing. In VQA, the video is considered as "moving" when:

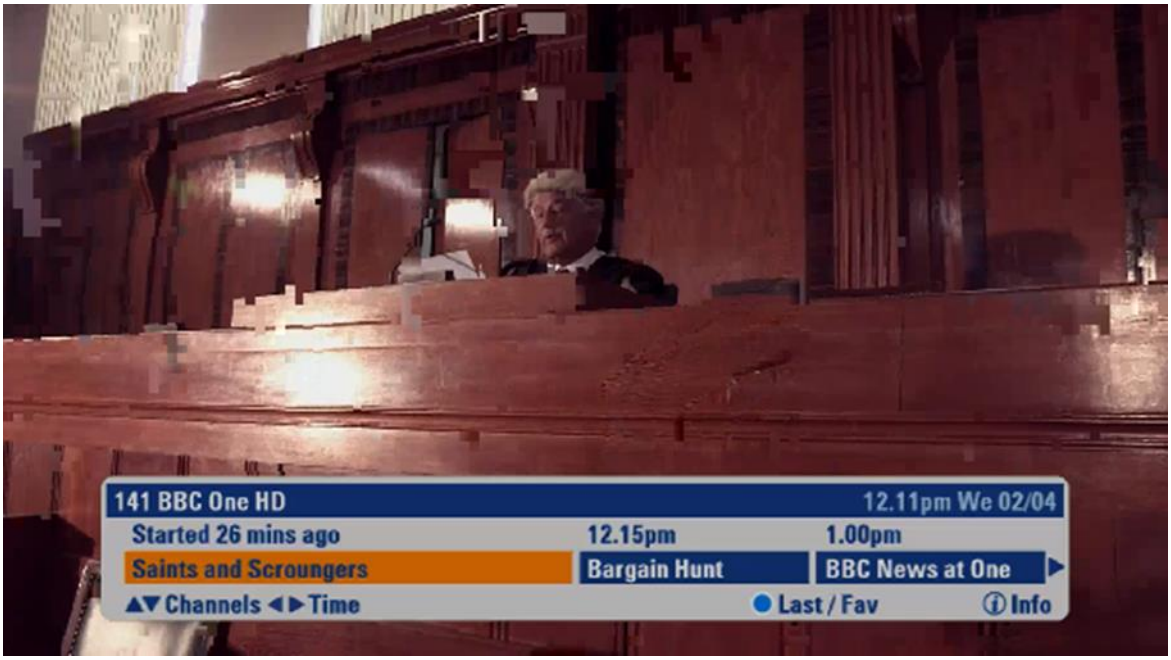
- temporal activity > 0.9
- or more than 0,01 % of the pixels have a significant change (more than 25) with the previous frame
- or the mean value of Y > 21 (to avoid black frames)

If the video is not considered as "moving", then jerkiness is detected. These different conditions aim at detecting jerkiness even when a few pixels move (as happens sometimes).

The minimum value is 0 and no maximum. It is the duration in milliseconds during which video was frozen. In the example above there are 3 frozen frames, so for 25 fps video, the returned value would be 120ms of frozen video.

5.3.1.8 Blockiness

Blockiness, also known as tiling effect, is a common artefact in MPEG video. It is sometimes called macro blocking. In the picture below, blockiness can be clearly observed on the upper part of the Image



Blockiness is quite prevalent in MPEG2 encoded video, however not so in H.264 encoded video due to the deblocking filter.

The one situation where blockiness occurs in H.264 video is due to transmission errors, such as in the frame below. This means that a high blockiness scores in H.264 video indicates transmission errors in the test system (i.e. poor signal quality for RF or high packet loss for IP).

Blockiness ranges from 0 to 255. It expresses the importance of the frontier between a block (which is suspected as distorted) and its neighbourhood.

Typical values are:

- between 0 and 3: OK
- above 3: distorted

5.3.1.9 Blur

Blur ranges from 0 to the diagonal length of the picture sqrt(width*width + height*height). It is similar to the width of contours, or (in other terms) to the width of the line spread function.

See the difference between blurred and not blurred frame.

Typical values

- between 0 and 2 : OK
- Above 2: distorted (e.g: in previous example (blurred frame) blur value is around 3.7). This of course, may be the intention of the source video for artistic effect.

5.3.1.10 Contrast

This ranges from 0 to 2 and expresses the importance of details in the video, especially around macroblock boundaries where they are submitted to the H.264 deblocking filter. This value only has meaning when the original video is H.264.

In the image frame below we see that there has been a loss of contrast due to the H.264 deblocking filter

Typical values

- less than 0.15: indicate not enough detail
- above 0.15: good video

5.3.1.11 TemporalActivity

Temporal activity is an indication of how pixel values changes between two successive images.

Temporal activity ranges from 0 to 255. It indicates how the pixel values changes between two successive images.

Typical values are:

- lower than 5: low motion
- Between 5 and 15: normal
- Above 15: important motion

5.3.1.12 MotionVector

A tuple of the global motion vectors in (X,Y) directions indicating the general motion of this frame relative to the previous frame.

5.3.1.13 YLevels

Tuple of minimum Y, maximum Y, average Y and standard deviation of Y.

Y is the brightness (luminance) of the video frame. All values are doubles between 0 and 255. The mean values can be interpreted as:

- 0 - 50 : dark content,
- 50 - 150 : normal content
- Above 150 : light content.

For the standard deviation values less the 20 indicate a homogeneous content.

5.3.1.14 ULevels

Tuple of minimum U, maximum U, average U and stardard deviation of U. The U value is the difference between Blue and the Brightness (part of YUV color space). In this release the average and standard deviation will always be None as it is not supported.

5.3.1.15 VLevels

Tuple of minimum V, maximum V, average V and stardard deviation of V. The V value is the difference between Red and the Brightness (part of YUV color space). In this release the average and standard deviation will always be None as it is not supported.

5.3.2 Audio results

Audio quality analysis is performed on the audio captured from the DUT. Specifically, the audio from the currently selected audio channel is analysed. StormTest supports 2 channel stereo PCM audio input, so the user can select which channel to analyse by calling `SetAudioAnalysisChannel()` before starting the audio analysis. Unlike for video quality analysis, the codec used to encode the audio originally is not significant to the analysis and the function will work with any codec.

The properties here indicate various metrics about the audio analysed. Audio is analysed in chunks of 1 second, not video frame lengths. However, not all properties are necessarily valid for all result records - in that case the value will be *None* instead of the value described here. If there is a long period of pure silence, the MOS score is *None* (a human would not say silence is 'good' or 'bad' but rather 'there is no audio'. The low level hardware supplies audio in packets of 8192 bytes (42.67 mS at 48 kHz audio) so 1 second is not a complete number of packets. For this reason, you do not get exactly 1 result per 1 second of 30 frames of video (at 30 frames per second). The audio is processed continually without error - the first audio result is generated when 24 packets of 42.67 mS have been received (1.024 seconds)

5.3.2.1 FrameNumber

The frame number of this result. The first result analysed is frame 30 +/-1 at a video rate of 30 frames per second. Frame numbers are counted from the captured video frame and will increment at the raw video capture frame rate (+/- 1), due to 1 result per second.

5.3.2.2 TimeStamp

The timestamp of the result. This is in seconds since the Unix epoch (1/1/1970 00:00:00 UTC) and is the time on the server when the result was generated. This may increase unevenly due to threading and CPU load. It is useful for debugging should problems occur.

5.3.2.3 MOS

The raw MOS score. This is a number normally between 0 and 100. However, if the XXXInfluence parameters are set to high values and lots of errors occur, the result can be negative. The scores are:

MOS Score Meaning

< 20 Bad Quality

20 - 40 Poor Quality

40 - 60 Rather Good Quality

60 - 80 Good Quality

80 - 100 Excellent Quality

These terms are defined in standards such as BT.500 and BT.710 and normalized subjective voting protocols (like MUSHRA and ACR).

The MOS score is an average of the qualities over time. It uses a **10 second window** for the averaging so the first 10 seconds worth of values after starting are not valid.

This duration of 10 seconds has several explanations:

- Human judgment of overall audio quality is generally an integration of the judgments over the last seconds.
- Process of human judgment of audio quality may take quite a long time before reaching a stable judgment. This is way the audio samples used in subjective listening tests are generally quite long (from 10 seconds to about 1 minute) whereas the video sequences used in subjective quality assessment tests are shorter (about 10 seconds). One could say that at medium quality level, video distortions are faster to perceive and evaluate than audio distortions

5.3.2.4 MOSDescription

The String equivalent of the MOS score without the word Quality (i.e. Bad, Poor, Rather Good, Good or Excellent) They were the adjectives shown on the voting scale used by the human listeners during the subjective audio quality assessment tests whom results were employed to validate VQA's objective audio quality metric.

5.3.2.5 AudioQuality

The quality of a single 1 second piece of audio. The range and meaning is the same as MOS score. MOS score is the average of the raw audio qualities. The AudioQuality ignores the effect of prior frames.

5.3.2.6 AudioQualityTemporal

The quality of a single 1 second piece of audio taking into account the previous pieces' scores. This takes into account a property of the human audio system which is that we are "quick to criticize, slow to forgive". In order to understand this sentence, consider human observers listening to a good quality audio track. Now imagine that an audible distortion appears. When the distortion appears, the quality judgment of the human listeners decreases nearly instantly ("quick to criticize"). However, when this distortion disappears, their human judgment takes several seconds to come back to its previous state that was judging that the audio has a good quality ("slow to forgive"). Humans are generally more sensitive to audio distortion than video, so a longer (10 second) window is used for temporal calculations. This is the temporal effect implemented in this score computation. It indicates the quality of each piece of audio, taking into account the previous judgments.

5.3.2.7 CustomSilences

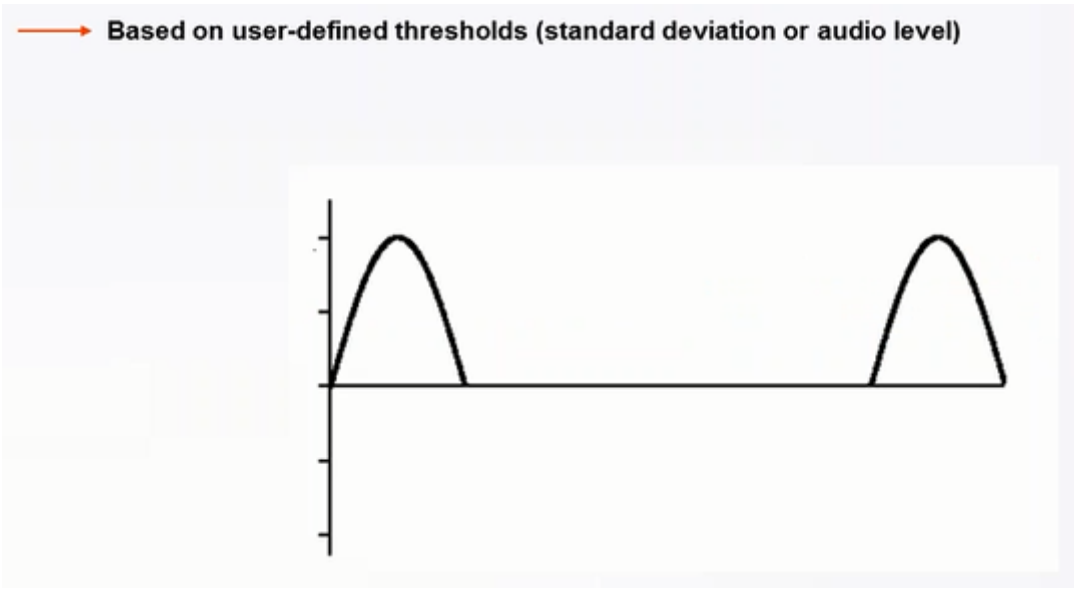
Number of silences detected based on the custom silence detection algorithm selected. It will be a number >= 0.

5.3.2.8 Silences

The number of audio silences detected using the standard silence detection algorithm. (CustomSilence input parameter set to "Disabled")

The number of silences in an audio segment. This can be:

- 0: no silence
- 1: 1 detected silence on a channel
- 2: 2 detected silences in processed channels
- N>2: N detected silences in processed channels



5.3.2.9 SilenceDuration

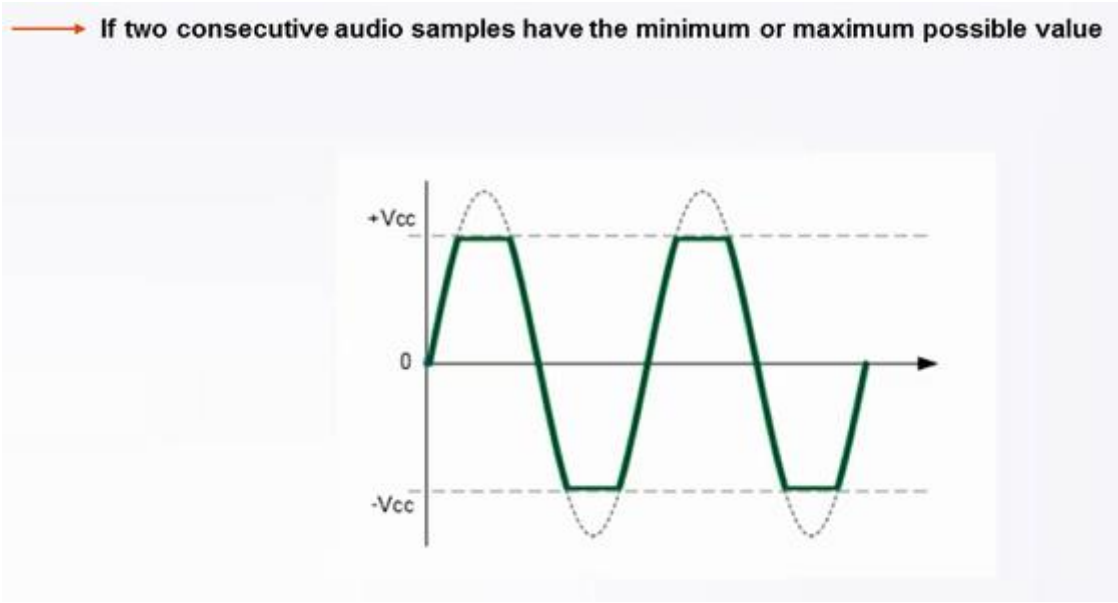
The duration of audio silences detected using the standard silence detection algorithm (CustomSilence input parameter set to “Disabled”). Each silence is >= 100mS, so this value is the sum of all silences.

5.3.2.10 Saturations

The number of times audio saturation was detected in the 1 second piece of audio (two consecutive samples which both have the minimum or maximum possible sample value).

Typical values are:

- 0: no detected saturation
- Above 0: N detected saturations (N couple of consecutive audio samples have the same min or max value)

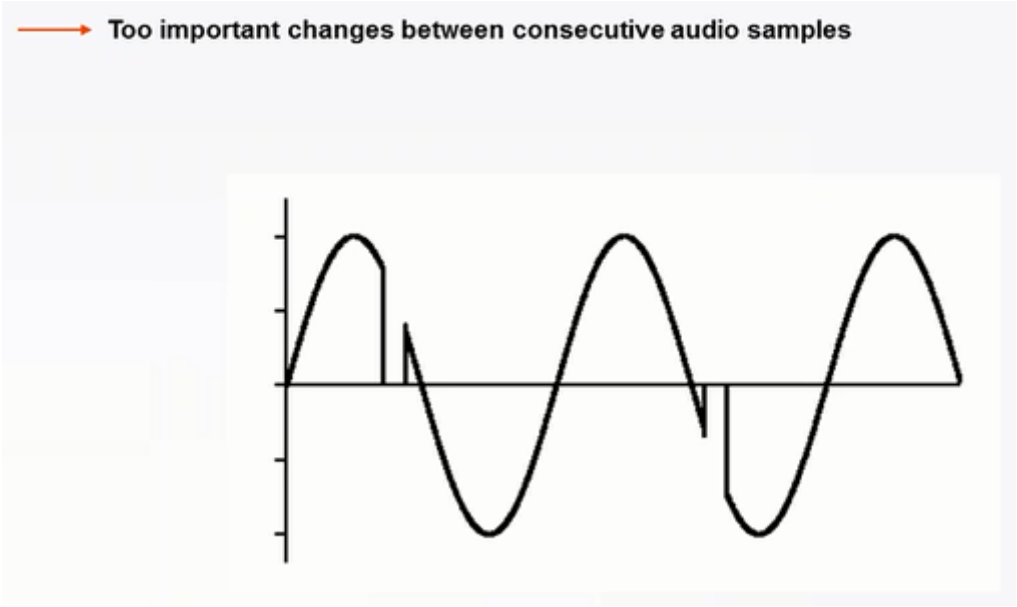


5.3.2.11 Breaks

Audio signal is normally a continuous signal. If the difference between two consecutive samples is too important, a break is detected. The number of breaks detected in the 1 second piece of audio.

Typical values are:

- 0: no detected signal break
- Above 0: N detected signal breaks

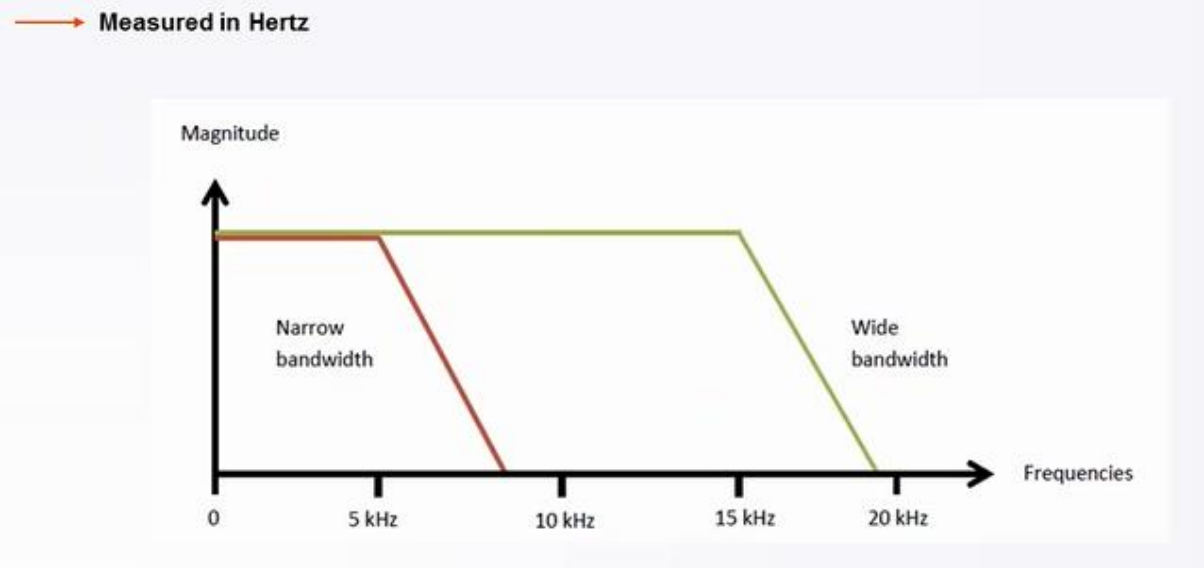


5.3.2.12 EstimatedBandwidth

The estimated bandwidth of the audio sample. The maximum bandwidth is the audio sample rate/2. A low value of the estimated bandwidth indicates possible distortion (but it may also simply be a feature of the source audio). The wider the bandwidth, the better the quality.

Typical values are:

- between 0 and 15000: distorted
- Above 15000: OK



5.3.2.13 HighFrequencyPresent

A value indicating the presence of high frequencies in the 1 second of audio. It is an adaptive calculation which takes into account the ratio between low and high frequencies.

$$Presence\ of\ high\ frequencies\ = \frac{\sum magnitudes\ of\ high\ frequencies}{\sum magnitudes\ of\ low\ frequencies}$$

This measurement allows the user to detect cases where there is a wide bandwidth detected, but actually there are very few occurrences of high frequencies, so the full bandwidth is not really being used by the audio.

Typical values are:

- Lower than 0.0005: heavily distorted
- Between 0.0005 and 0.01: distorted
- greater than 0.01: OK

5.3.3 Example

```
vqa = stormtest.VQACreateEngine()
stormtest.VQAStartAnalysis(vqa ,callbackFunction = VQACallback)

def VQACallback(slotNo, engine, event):
    """
    Description: sample Callback of the VQA Engine.
    """

    if event.Value and event.Type=="VideoResult":
        for i in event.Value:
            print "VideoResult"
            print "FrameNumber",i.FrameNumber
            print "MOS",i.MOS
            print "FrameQuality",i.FrameQuality
            print "FrameQualityTemporal",i.FrameQualityTemporal
            print "Jerkiness",i.Jerkiness
            print "Blockiness",i.Blockiness
            print "Blur",i.Blur
            print "Contrast", i.Contrast
            print "TemporalActivity",i.TemporalActivity

    if event.Value and event.Type=="AudioResult":
        for i in event.Value:
            print "AudioResult"
            print "FrameNumber",i.FrameNumber
            print "TimeStamp",i.TimeStamp
            print "FrameNumber",i.FrameNumber
            print "MOS",i.MOS
            print "MOSDescription",i.MOSDescription
            print "AudioQuality",i.AudioQuality
            print "AudioQualityTemporal",i.AudioQualityTemporal
            print "CustomSilences",i.CustomSilences
            print "Silences",i.Silences
            print "SilenceDuration",i.SilenceDuration
            print "Saturations",i.Saturations
            print "Breaks",i.Breaks
            print "EstimatedBandwidth",i.EstimatedBandwidth
            print "HighFrequencyPresent",i.HighFrequencyPresent
```

6 Tutorial

6.1 Hello World

The simplest way to get started with the VQA is with the following HelloWorld example.

```
import stormtest.ClientAPI as StormTest

# instantiate the VQA engine
engine = StormTest.VQACreateEngine()

# start analysis
StormTest.VQAStartAnalysis(engine)

# stop analysis
StormTest.VQAStopAnalysis(engine)

# close the engine
StormTest.VQACloseAnalyzer(engine)
```

Although completely functional, this example would yield no data about video quality.

6.2 Getting Data

In order to obtain data on video quality, the example above needs two things. Firstly, it needs to run for a period of time over which it will gather data about the video quality. This can be achieved by adding a runtime delay for the duration of the required analysis period.

```
# Analysis period
StormTest.WaitSec(30)
```

Secondly, the VQA engine needs a way to provide the data it gathered back to the script. This is achieved by using a callback function. This callback function will be called every time the VQA engine has something to report on. Following is an example of a callback function which will print out video quality data for each frame gathered over a period of time defined above:

```
# VQA Callback
def VQACallback(slotNo, engine, event):
    # if there are any results, log them
    if event.Value and event.Type=="VideoResult":
        for i in event.Value:
            print i.FrameNumber, i.MOS, i.FrameQuality, i.Jerkiness, i.Blockiness
```

Finally, to put this together into the example above, this callback function needs to be passed as an argument into the VQAStartAnalysis function.

6.3 Learning to Walk

The following example will perform video quality analysis of the current content for a period of 30 seconds and print results to console.

```
import stormtest.ClientAPI as StormTest

# instantiate the VQA engine
engine = StormTest.VQACreateEngine()

# start analysis
StormTest.VQAStartAnalysis(engine, callbackFunction=VQACallback)    # callback passed

# Analysis period
StormTest.WaitSec(30)

# stop analysis
StormTest.VQAStopAnalysis(engine)

# close the engine
StormTest.VQACloseAnalyzer(engine)

# VQA Callback
def VQACallback(slotNo, engine, event):
    # if there are any results, log them
    if event.Value and event.Type=="VideoResult":
        for i in event.Value:
            print i.FrameNumber, i.MOS, i.FrameQuality, i.Jerkiness, i.Blockiness
```

6.4 Options and Triggers

The previous section provided a simple example of how the StormTest VQA API can be used to provide video quality analysis data over a period of time. This section will provide some options allowing the tuning of the VQA engine as well as ability to start and stop the analysis at specific times.

6.5 Freedom of Choice

As covered in the StormTest Programmer's Guide, before the VQA object is instantiated, the developer can select specific options to be used by the VQA engine. If no options are specified, the defaults are used.

The available options and the default values are as follows:

Option	Default Value
Format	'H264'
Backlog	500
FrameRate	0
Jerkiness	True
Activity	0.9
ChangingPixels	0.01
MeanBrightness	21

The meaning and range of each of these values is detailed in section 3.35 of the StormTest Client API document.

If non-default options are desired, they can be set in the following way.

```
# create the options object for VQA
```

```
vqaOptions = StormTest.VQACreateVideoOptions()

vqaOptions.Format = "MPEG2"                          # change video format
vqaOptions.Backlog = 1500                            # increase backlog
vqaOptions.Jerkiness = False                         # ignore jerkiness
```

To apply a set of options to the example from previous section, the VQA Options object needs to be passed into the VQA engine constructor:

```
engine = StormTest.VQACreateEngine(videoOptions = vqaOptions)
```

6.6 Happy Triggers

Video Quality Analysis in StormTest can now be started or stopped by a specific event. This event is called a trigger and can be an image, color, icon or any of the following options:

Trigger Condition	Default Value
Delay	None
FrameCount	None
DroppedFrames	None
IRKey	None
IRKeyRepeat	1
Sdo	None
FromNow	True

To illustrate with an example, the following creates a trigger which will be used to delay the start/stop of the Video Quality Analysis by 30 seconds.

```
VqaTrigger = StormTest.VQACreateTrigger()
VqaTrigger.Delay = 30
VqaTrigger.FromNow = False
```

To apply a trigger, it must be passed into the call to start or stop the video quality analysis:

```
StormTest.VQAStopAnalysis(engine, trigger = VqaTrigger)
```

Another example is a trigger which can be used to start (or stop) video quality analysis when a specific frame is seen:

```
import Image                                          # imports images to memory

# create the start trigger
pil = Image.open("triggerImage.png")                  # needed to create an image object
triggerImage= StormTest.Imaging.StormTestImageObject(pil)   # create an image object

startTrigger = StormTest.VQACreateTriggerFromImage(triggerImage,rect=[0, 0, 1920,
1080],threshold=96)
```

```
# start VQA using the startTrigger
StormTest.VQAStartAnalysis(engine, trigger = startTrigger)
```

Note that if multiple triggers are defined for a single event (e.g. start of analysis), then the first trigger to occur will trigger that event.

6.7 Callback events

In the simple example of a callback here, it can be seen that if event.Type is “VideoResult”, then event.Value contains video quality analysis data.

```
# VQA Callback
def VQACallback(slotNo, engine, event):
    # if there are any results, log them
    if event.Value and event.Type=="VideoResult":
        for i in event.Value:
            print i.FrameNumber, i.MOS, i.FrameQuality, i.Jerkiness, i.Blockiness
```

Events can also signify the beginning or the end of video quality analysis, loss of frames or change of resolution. Complete documentation on types of callback events can be found in section 5.3

In the event that dropped frames or change of resolution need to be reported as well as the ones that contain video analysis data, the callback would need to be modified to look similar to this:

```
# VQA Callback
def VQACallback(slotNo, engine, event):
    # if there are any results, log them
    if event.Value and event.Type=="VideoResult":
        for i in event.Value:
            print i.FrameNumber, i.MOS, i.FrameQuality, i.Jerkiness, i.Blockiness

    # if any frames are dropped, report them
    if event.Type=="DroppedFrame":
        for i in range(event.Value[0]):
            print "dropped"

    # if resolution changes, report that
    if event.Type=="ResChange":
        print "Resolution changed to: ", event.Value[0], " x ", event.Value[1]
```

Note that in this example event.Value may not be the same for all types of events. For example, an event of type VideoResult, returns a list of VideoResult objects. DroppedFrame and ResChange events return a tuple of values.

6.8 Putting It All Together

The triggers and options covered in this sections can be added to previous example in the following way.

```
import stormtest.ClientAPI as StormTest
import Image                                                # imports images to memory
```

```
# create the options object for VQA
vqaOptions = StormTest.VQACreateVideoOptions()
vqaOptions.Format = "MPEG2"                                    # change video format
vqaOptions.Backlog = 1500                                      # increase backlog
vqaOptions.Jerkiness = False                                   # ignore jerkiness

# instantiate the VQA engine
engine = StormTest.VQACreateEngine(videoOptions = vqaOptions)

# create the start trigger
pil = Image.open("triggerImage.png")                           # needed to create an image object
triggerImage= StormTest.Imaging.StormTestImageObject(pil)    # create an image object
startTrigger = StormTest.VQACreateTriggerFromImage(triggerImage,rect=[0, 0, 1920,
1080],threshold=96)

# create a trigger at which to stop
stopTrigger = StormTest.VQACreateTrigger()
stopTrigger.Delay = 30                                         # perform VQA for 30s
stopTrigger.FromNow = False                                    # from start of VQA

# start analysis
StormTest.VQAStartAnalysis(engine, trigger = startTrigger, callbackFunction=VQACallback)

# stop analysis
StormTest.VQAStopAnalysis(engine, trigger = stopTrigger)

# close the engine
StormTest.VQACloseAnalyzer(engine)

# VQA Callback
def VQACallback(slotNo, engine, event):
    # if there are any results, log them
    if event.Value and event.Type=="VideoResult":
        for i in event.Value:
            print i.FrameNumber, i.MOS, i.FrameQuality, i.Jerkiness, i.Blockiness

    # if any frames are dropped, report them
    if event.Type=="DroppedFrame":
        for i in range(event.Value[0]):
            print "dropped"

    # if resolution changes, report that
    if event.Type=="ResChange":
        print "Resolution changed to: ", event.Value[0], " x ", event.Value[1]

    # if the VQA engine is just starting, chime cheerfully
    if event.Type == "Start":
        print "VQA Engine: Started"

    # if stopping, no new frames will be analysed
    if event.Type == "Stop":
        print "VQA Engine: Stopping"

    # if the VQA engine is done processing all remaining frames in queue
    if event.Type == "Finish":
        print "VQA Engine: Finished"
```

7 Tips

If you need to be able to detect errors on a single frame, then you should use the intrinsic frame quality score.

If you need to measure or monitor the perceived video quality of a video along time, then you should use the MOS value.

To get the global quality score of a video, you must compute the mean value of all the MOS values.

Full example using the VQA engine are available as part of the StormTest developer guide and the StormTest python framework.