Technical White Paper

Video Encoding Cookbook and Profile Guidelines for the Adobe® Flash Platform

Video on Demand Encoding/Transcoding Guide for Desktop End Users

By Maxim Levkov, Adobe Systems Inc., encodingcookbook@adobe.com

VERSION 1.1

Table of Contents

Introduction

Video compression is indeed both an art form and a highly technical process. Many compressionists and website developers responsible for preparing video for web delivery today don't fully understand the details behind the settings in their encoding software; these details are often hard to find and are often only discussed in scholarly technical papers.

This comprehensive technical whitepaper is intended as a reference guide for those whose aim it is to improve the overall quality of their encoded video, producing the very best quality possible. Best practices, quality control testing techniques, and specific encoding considerations for delivery on the Flash Platform are discussed. While highly technical in nature, it is meant to introduce advanced encoding techniques to those who may not adequately understand what is happening "behind the UI" in their compression software, providing greater insight into the encoding process — and ultimately helping to deliver the very best playback experience.

Scope

This document addresses techniques, best practices, recommended settings, and tools for encoding video on demand (VOD) assets that will be streamed by Flash Media Server and played back in Flash Player or on AIR. The objective of this guide is not to reiterate the fundamental knowledge put forth in an encoding/transcoding software user manual, but rather to draw on the experience and technical benefits that make the Flash Platform the popular choice for video on the Internet and beyond, while advancing knowledge through explanation of key concepts that help compressionists and video site developers to achieve the highest quality playback possible.

Along with the guide the following elements will be made available to help solidify and reproduce concepts presented: optimized profiles, technical video and real video segments.*

The best practices and settings outlined in this document are designed to operate with Flash Media Server 3.5 and Flash Player 10.0.32.x. In order to be compliant with quality and performance optimizations, these versions of the server and player must be used.

The profiles presented in this document are targeted specifically at encoding of VOD (i.e. non-live) content.

While these profiles may be similar to those used to encode live content, profile options for live content are not specifically considered in this release. Encoding profiles for live hardware encoders, used primarily for live content, will be covered in live content encoding releases of this document.

NOTE: All abbreviations and acronyms referenced in this paper are defined in the Glossary on page 55.

This guide will be modified as new VOD encoding options are supported or new tools are identified. Content that is created with any publicly released version of this document is expected to be supported by future versions of Flash Media Server.

*The rights for content supplied with this guide is for non-commercial use only, rights for commercial use of the content must be obtained individually by the user.

History of Video in Flash

Video on the Flash Platform has been constantly evolving since the first videos were embedded in SWF files and played in Flash Player 6, back in 2002. Flash is now the defacto delivery method for video on the web, with over 75% of web video being delivered in the Flash player. Flash supports many different codecs and container formats, along with a flexible array of delivery protocols.

Codecs and Container Formats

Flash supports a variety of codecs and container formats. When video support was first introduced in Flash Player 6, a video could either be embedded directly in a SWF file, or it could be streamed using the Real Time Media Protocol (RTMP) via Flash Communication Server. In 2003, Flash Player 7 added support for loading external video files in the FLV file format.

Sorenson Spark/Nellymoser

These first videos in Flash Player 6 and 7 used the Sorenson Spark video codec (H.263 video standard). This codec is still used today for compression of webcam capture within the Flash Player. The audio codec used with Spark is Nellymoser.

On2 VP6/MP3

Flash Player 8 and later added support for the On2 VP6 video codec, which is generally used with the MP3 codec for audio. VP6 provided higher quality compression than Spark, but is more complex to decode, especially on older computers. It also supports alpha channels, which enables video shot on a blue screen or green screen to be layered on top of other content.

On2 VP6-S

Because VP6 tends to be processor-intensive to decode, Flash Player 9 update 3 (version 9.0.115.0) introduced support for a simplified version, VP6-S. This codec is targeted for older computers with slower processors, improving playback performance of high-quality video. Audio for VP6-S is generally MP3.

H.264 (MPEG-4 Part 10)/AAC-HE

Flash Player 9 update 3 (version 9.0.115.0) also introduced support for the open source H.264 codec, and a new container format: F4V. The standard supported is MPEG-4 Part 10. This codec can also be processor-intensive to decode, but offers better quality at lower bitrates. AAC-HE, a new audio codec also introduced at this time, is generally used with H.264 video.

Speex

Flash Player 10 and later introduced support for the Speex codec for audio recording in the browser. This codec is optimized for encoding speech, so it is ideal for VOIP and audio chat applications.

Delivery Protocols

There are a variety of protocols available for media delivery on the Flash Platform. The method you choose depends on the attributes of your content and the size of your audience.

Progressive Delivery (HTTP)

The simplest delivery method to implement is progressive delivery. In this method, a video file is downloaded and displayed just like any other web asset. In this method, the video content (FLV or F4V [an MP4 variant]) is kept external to the other content and the video playback controls. Because of this, it's relatively easy to add or change content without republishing the SWF file.

Streaming (RTMP)

The most complete, consistent, and robust delivery option is to stream video and audio files from a Flash Media Server. In streaming, each client opens a persistent RTMP connection back to the video server. This approach lets you deliver features such as realtime data sharing, live webcam chats, bandwidth detection, quality of service metrics, detailed tracking and reporting statistics, and a whole range of interactive features along with the video experience. Advanced features such as Dynamic Streaming and DVR functionality are available with Flash Media Server 3.5 and later.

With streaming, as with progressive download, the video content (FLV or F4V [an MP4 variant]) is kept external to the other content and the video playback controls. It is, therefore, relatively easy to add or change content without the need to republish the SWF file.

Streaming also has other advantages, including the following:

- The video starts playing sooner than it does using progressive delivery.
- Streaming uses less of the client's memory and disk space, because the clients don't need to download the entire file.
- It makes more efficient use of network resources, because only the parts of the video that are viewed are sent to the client.
- It provides more secure delivery of media, because media does not get saved to the client's cache when streamed. Flash Media Server also allows for encrypted streaming, providing an additional level of security.
- It provides better tracking, reporting, and logging ability.
- It allows you to deliver and record live video and audio, or capture video from a client's webcam or digital video camera.
- It enables multiway and multiuser streaming for creating video chat, video messaging, and video conferencing applications.
- It provides programmatic control of streams (server scripting) for the creation of server-side playlists, synchronization of streams, smarter delivery adjusted to client connection speed, and application creation.
- It provides advanced monitoring and reporting on traffic and throughput.

There are various flavors of RTMP available with Flash Media Server 3.5 and later:

- RTMP—Standard, unencrypted RTMP.
- RTMPT—RTMP "tunneled" over HTTP. The RTMP data is encapsulated as valid HTTP data. This protocol is used when a client's network blocks RTMP traffic.
- RTMPS—RTMP sent over an SSL. SSL utilizes certificates, and enables secure connections.
- RTMPE—Enhanced and encrypted version of RTMP. RTMPE is faster than SSL, and does not require certificate management as SSL does (supported with Flash Player 9,0,115,0 or later; and Adobe AIR).
- RTMPTE—Encrypts the communication channel, tunneling over HTTP (supported with Flash Player 9,0,115,0 or later; and Adobe AIR). The key benefits over SSL (RTMPS) are performance, ease of implementation, and limited impact on server capacity.
- RTMFP—Peer-to-peer networking protocol that enables Flash Player clients to share video, audio and data over a direct P2P connection. (Requires Flash Player 10 or later, and an introduction service such as Stratus or a future version of Flash Media Server.)

Utilizing the appropriate RTMP type, Flash Media Server can send streams through all but the most restrictive firewalls, and help protect rights-managed or sensitive content from piracy.

HTTP Dynamic Streaming (HTTP)
HTTP Dynamic Streaming, Adobe's multibitrate adaptive streaming delivery solution, enables delivery of video-on-demand and live streaming using standard HTTP servers, or from HTTP servers at CDNs, leveraging standard HTTP infrastructure. The addition of HTTP streaming support to Flash Player 10.1 and later enables expanded protocol options to deliver live and recorded media to Flash player, including DVR functionality and full content protection powered by Flash Access 2.0.

This delivery method requires that media files go through an additional level of processing. They are broken into fragments, which are then delivered in sequence; with Flash Player detecting the client's bandwidth and requesting the appropriate bitrate fragments from the server at playback time. This provides the client with the highest possible video quality for their resources and a smooth playback experience.

Peer-to-peer (RTMFP)
Flash Player 10.0 introduced support for the Real Time Media Flow Protocol (RTMFP), which enables peer-to-peer connections between Flash player clients. This technology requires an "introduction service," which is currently known as Stratus. Stratus provides a way for clients to find each other and open direct connections to share video, audio and data.

RTMFP Groups
RTMFP Groups is an extension of the RTMFP technology, introduced in Flash Player 10. It enables application level multicast delivery, which allows one-to-many broadcasts over UDP-enabled networks without taxing the network infrastructure.

For additional details and comparison of delivery methods, refer to "Video Learning Guide for Flash: Progressive and streaming video," on the Adobe Developer Connection:
http://www.adobe.com/devnet/flash/learning_guide/video/part02.html#progressive.

Adobe Flash Player 10.x

Adobe Flash Player 10, originally codenamed Astro, was introduced in beta form in October, 2008. It featured many new capabilities, including:

- 3D object transformations
- Custom filters via Pixel Bender
- Advanced text support
- Speex audio codec
- Real Time Media Flow Protocol (RTMFP)
- Dynamic sound generation
- Vector data type
- Dynamic Streaming (Flash Media Server 3.5 required.)

The most important features relating to video include support for a new audio codec, Speex, which is optimized for voice encoding; support for the new RTMFP delivery protocol, which enables peer-to-peer communication in Flash Player; and Dynamic Streaming which, along with Flash Media Server 3.5, enables bandwidth detection on the client, switching between streams of various bitrates depending on the client's connection speed and processing power, ensuring the best possible playback experience.

Hardware acceleration was also enhanced in Flash Player 10, which improved full screen video playback performance. Encoding and player considerations that help make the mostof this feature are addressed in the *Video Player Design Considerations* section of this document.

Adobe® Flash® Player 10.1 is the first runtime release of the Open Screen Project (*http://www.openscreenproject.org/*) that realizes the promise of a consistent, cross-platform runtime across desktop and mobile devices. With support for a broad range of devices, including smartphones, netbooks, smartbooks and other Internet-connected devices, Flash Player 10.1 allows you to deliver consistent, high quality video content to a wide variety of screens.

The wealth of new video/audio playback features of Flash Player 10.1 include:

- **Hardware acceleration** for graphics and video along with many other performance improvements (i.e. rendering, scripting, memory utilization, start-up time, battery and CPU optimizations) for the very best performance on devices
- **New mobile-ready features** including multi-touch, gestures, mobile input models, and accelerometer support, as well as graphics acceleration and adaptive framerate for improved playback performance
- **HTTP Dynamic Streaming** support, Adobe's multibitrate adaptive streaming delivery solution
- **Content protection** powered by Adobe Flash Access, to support a wide range of business models, including video-on-demand, rental, and electronic sell-through, for streaming as well as download.
- **Smart seek** allows you to seek within the buffer and introduces a new "back" buffer so you can easily rewind or fast forward video without going back to the server, reducing the start time after a seek. Smart seek can speed up and improve the seeking performance of streamed videos and enable the creation of slow motion, double time, or "instant replay" experiences for streaming video. (Requires Flash Media Server 3.5.3.)
- **Stream reconnect** allows an RTMP stream to continue to play through the buffer even if the connection is disrupted. (Requires Flash Media Server 3.5.3.)
- **Peer-assisted networking** using the RTMFP protocol now supports application level multicast, providing one (or a few) -to-many streaming of continuous live video and audio (live video chat) using RTMFP groups. (Requires Stratus *http://adobe.com/go/stratus*) or a future release of Flash Media Server.
- **Buffered stream catch-up** allows developers to set a target latency threshold that triggers slightly accelerated video playback to ensure that live video streaming stays in sync with real time over extended playback periods.
- **Fast Switch** enhances the Dynamic Streaming capability introduced in Flash Player 10 and FMS 3.5 to improve switching times between bitrates, reducing the time to receive the best viewing experience for available bandwidth and processing speed. Users no longer need to wait for the buffer to play through, resulting in a faster bitrate transition time and an uninterrupted video playback experience, regardless of bandwidth fluctuations. (Requires FMS 4 server.)
- **Microphone Access** (desktop only) allows access to binary data of the live and continuous waveform coming from the microphone to create new types of audio applications, such as audio recording for transcoding, karaoke, vocoder voice manipulation, sonographic analysis, pitch detection, and more.

Assumptions

- Technical staff using this document is skilled in video coding technology field.
- Quality control tools, viewing and listening conditions are tested and calibrated as described in this document, using recommended test patterns and equipment.
- Coded content is destined for appropriate compatible software and/or hardware decoders.
- Coding software and hardware in use is functioning as stated.
- Coding software and hardware support at least some of the following coding elements mentioned throughout this document:

Image Formats	
Sizes	128x96 to 1920x1088
Frame Rates	23.976, 24, 25, 29.97, 30, 50, 59.94, 60 fps, or fraction thereof
Aspect Ratio	1.33, 1.78, 1.78 AN, 1.85, 2.35, including Letterbox and Pillarbox variants
Color Space	YUV 16 - 235, Color Matrix 601 or 709
Video Sampling Structure	4:2:0
H.264 Codec Parameter Set	
Coding Levels	Baseline, Main, High
Coding Profiles	1 through 4.2
At least one of the mixing formats	F4V, MP4, MOV
At least one of the audio coding formats	AAC (LC), HEAAC v1, HEAAC v2

Transcoding Workflow Setup and Verification

Typical content workflow is complicated with many creative decisions. Adding content transcoding workflow to the creative process complicates it even further, and making sure the creative work did not get compromised throughout the supply chain process is a daunting task.

Many audio-visual transcoding problems arise as a result of:

- A supply chain consisting of many cascaded processing stages
- Processors that are provided by many suppliers
- Many different formats are involved
- Use of suboptimal source, often in previously published form
- A semi-automated or manual profile creation process, resulting in a variety of human errors
- Profiles seldom accommodate a variety of content, generally they are recycled without regard to the specifics of the ingested media

Coding processes are no different than manufacturing processes in the factory making tangible goods.

One of the ways to minimize unpredictability in final perceptual quality, while maintaining consistency, is through a combination of synthetic AV test patterns and real AV media clips.

Encoding and transcoding is a transformational process, where each preceding element is an input to the next element, with the result being an output of this combination, thereby making the process irreversible. If a preceding element was compromised and is then used as input for the next coding process, then the final output will be compromised as well, resulting in suboptimal picture and audio quality (Figure 1).

The following are the key elements that are important to maintain throughout the encoding process in order to achieve the highest quality output and maintain the creative integrity of content.

- Video Levels scheme: 16-235 and 0-255 (Y,R,G,B)
- Audio levels
- Audio-Video Synchronization
- Video Scaling
- Video Color Matrix: 601/709 and YUV/RGB
- Bitrate: Source, intermediate, and output
- Frame Rate Conversion

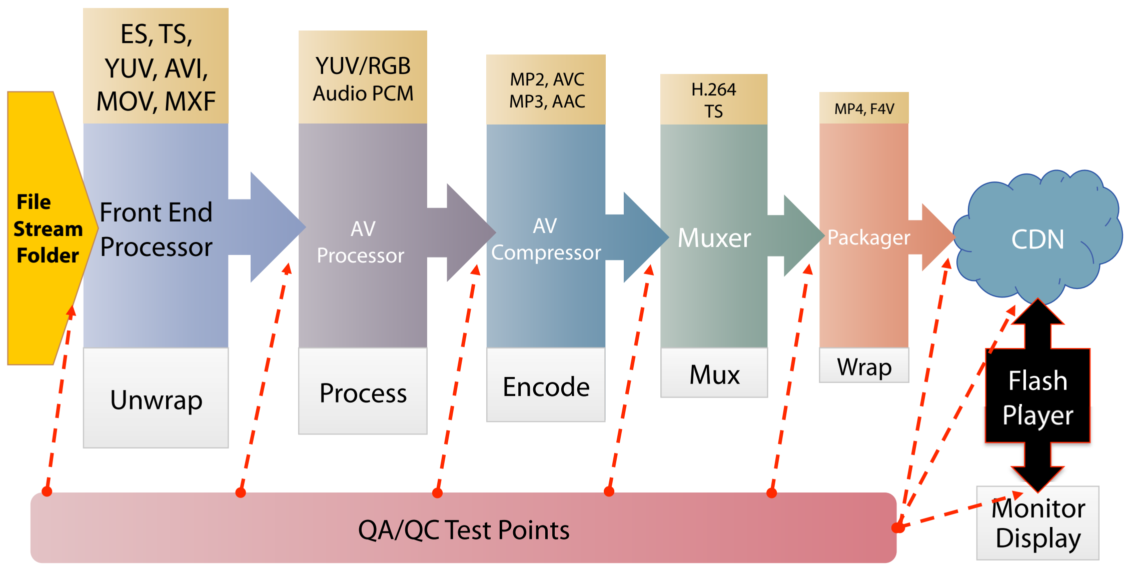


Figure 1

The transcoding process.

Testing Overview

Each step in the encoding process needs to be examined to assure integrity. Test patterns are helpful at each step to reveal artifacts that are undetectable to the human eye. Just because the final video and audio may look fine does not mean that all of the transformational stages were performed correctly. It simply means that you cannot see what is going on behind the scenes.

In some cases the entire supply chain needs to be tested in order to make sure that all of the integral elements of that chain are functioning properly. In other cases, when all stages have been tested and validated, it is necessary to test only some of the stages to establish confidence in the final output. Of course, human error is always a possibility, so validation of foundational elements (i.e. scaling, color transformation, audio/video sync, etc.) can be essential. The diagram above depicts, left to right, the additive process of establishing audio/video quality confidence. However, sometimes you may choose to process your confidence testing selectively, from right to left of the transcoding process, to discover errors through a process of elimination.

The audio/visual quality control process makes use of synthetic test patterns and other technical testing content (i.e. synthetic dynamic test patterns concatenated in sequence with real video), and is divided into four levels:

1. Subjective audio/visual observation and quality estimation (e.g. playback in Flash Player, Adobe Media Player, VLC, Elecard, etc.)
2. Operator driven objective measurement of audio and video performance through special instruments and test patterns (e.g. Tektronix, Interra Systems, Sencore, VideoQ etc.)
3. Fully automated robotic video quality control (e.g. VideoQ VQL/VQMA, Interra Systems Baton, Tektronix Cerify, etc.)
4. Fully automated robotic checking of syntax metadata, optionally combined with video and audio quality control (e.g. Interra Systems Baton)

The use of complex and comprehensive technical testing content in your workflow provides confidence in the results of transcoding processes, repeatable objective results, repeatable setup results, a foundation for verification and test procedures, and eliminates subjectivity of human interpretation. Of course, it does not eliminate the need for traditional syntax/functionality checks and subjective AV quality estimation on live clips.

NOTE: The technical testing content (static and dynamic test patterns) elements depicted in this section were acquired from VideoQ, which focuses on video test and measurement and video enhancement technologies, products, and services. This technical test content was carefully appended without the loss of perceptual quality, audible quality, and structural integrity to specially selected video clips provided by Artbeats, and HillmanCurtis, Inc.

Examples of Quality Problems

The following are some examples of issues with audio and video encoding that are ordinarily not immediately revealed without special instruments and technical testing content.

Domain Transformation

If you've transcoded a media clip and the results looked fine (e.g. size, color, audio quality), it does not mean that transformations were made within the same domain (ex. YUV in to YUV out, or 30 FPS in to 30 FPS out, or 1080p in to 1080p out) throughout the process. Just because your input is the same as the output, it doesn't mean that the interim processes were consistent. (See Figures 2-4 for examples.)

For example, in a typical audio processing operation, an audio engineer may increase the volume, later to find out that it was too loud. Then, another engineer decreases the audio volume supposedly back to the previous level, thinking that his adjustment set the perceivable audio loudness and throughput gain equal to the original. While it may seem fine throughout the process, neither of the engineers oriented their adjustments with special test signals reaching 0dBfs (typical testing reaches -6dBfs or -8dBfs, and at these levels perceived audible quality will sound great and critical sound loudness will be left unnoticed). The problem will only reveal itself at levels exceeding -6dBfs and not otherwise — and since such levels may rarely exceed -6dBfs in real video content, the error will therefore remain unnoticed.

In reality no one knows what the nominal normalization level is and should be, unlike in video (i.e. 16-235 levels) there is no globally accepted standard reference level of loudness, therefore, the need to test the system with entire range is highly recommended. Hence, any adjustments in the audio may result in clipping of critical levels and crippling overall audio performance.

- **Situation 1** — Levels are aligned and representative levels are preserved throughout the test duration (Figure 2).
- **Situtation 2** — Levels are excessive gain, somewhere in the processing chain, resulting in clipping and excessive loundness (Figure 3).
- **Situation 3** —Overall gain is too low, resulting in no clip, but loudness is too low (i.e. safe mode, and not fully utilizing system's potential) (Figure 4).

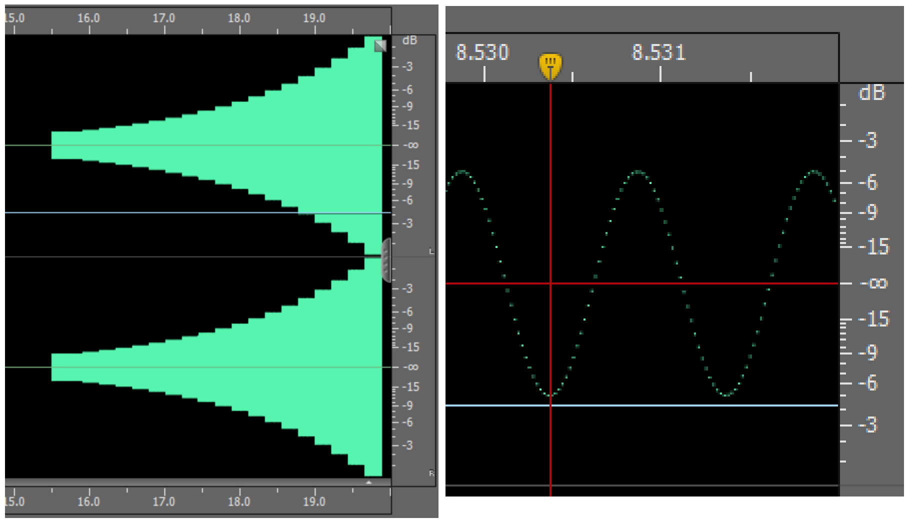


Figure 2

Example of reference audio levels. (Frequency: 1KHz, Levels Range: -18 dBfs to 0 dBfs, Step Increment: 1dB) Also, and example of correctly leveled audio processing.

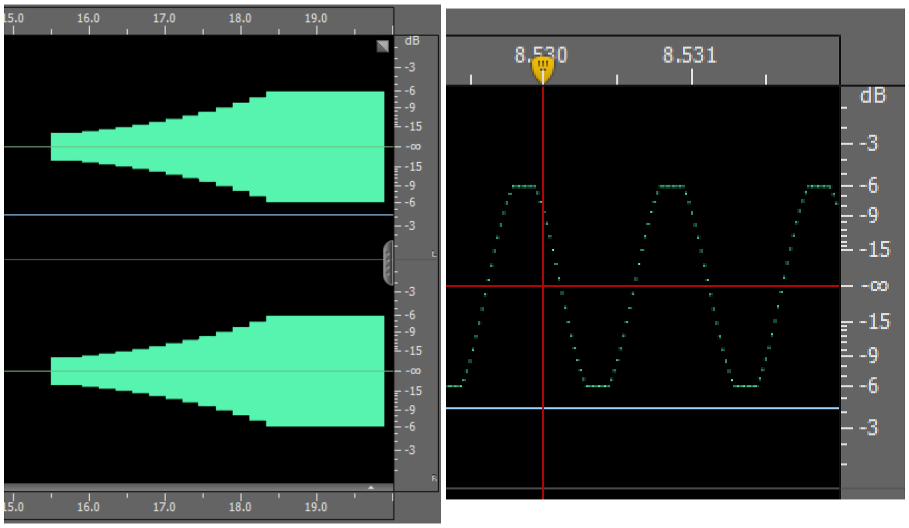


Figure 3

Example of audio levels from source being too high once processed, resulting in clipping. This figure illustrates a situation where audio was clipped by one operator when it reached 0 dBfs and then subsequently adjusted to -6 dBfs by another operator.

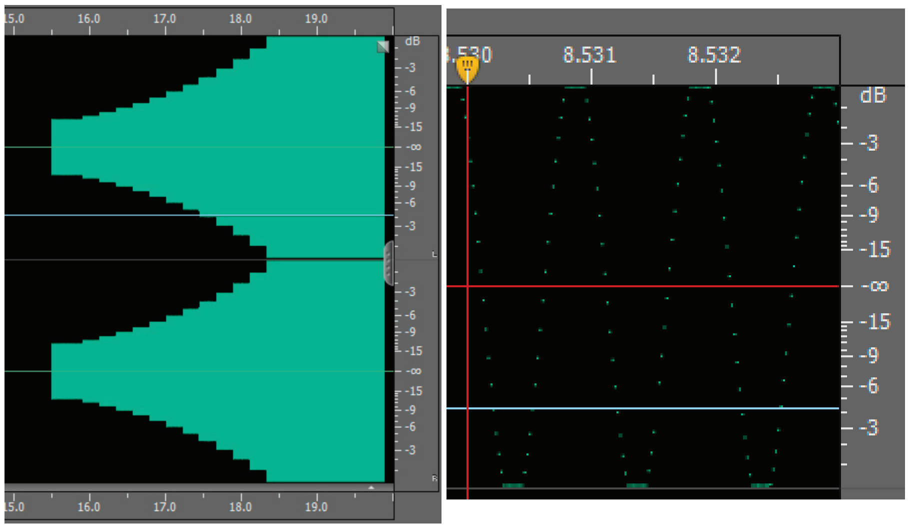


Figure 4

Example of audio levels from the source being too low once processed, resulting in clipping. This figure illustrates a situation where audio was clipped when it reached 0 dBfs. Notice that audio until -6 dBfs is fine and might remain fine unless a frequency of -6 dBfs to 0 dBfs is reached.

Color Transformation

If your input is in the YUV color matrix domain and your output is in the YUV color matrix domain, it is possible to set up the transcoder to achieve the output through conversion to RGB color matrix domain (YUV in -> RGB at processing time -> YUV out) and then back to YUV color matrix domain, while modifying video levels scheme (16-235 -> 0–255 -> 16-235). By doing so, the video changes its color matrix and video levels scheme unnecessarily and, therefore, loses its color matrix and video scheme levels fidelity. If this video undergoes several of these transformational stages, the picture at the final stage will end up looking entirely different from the original (Figures 5-7).

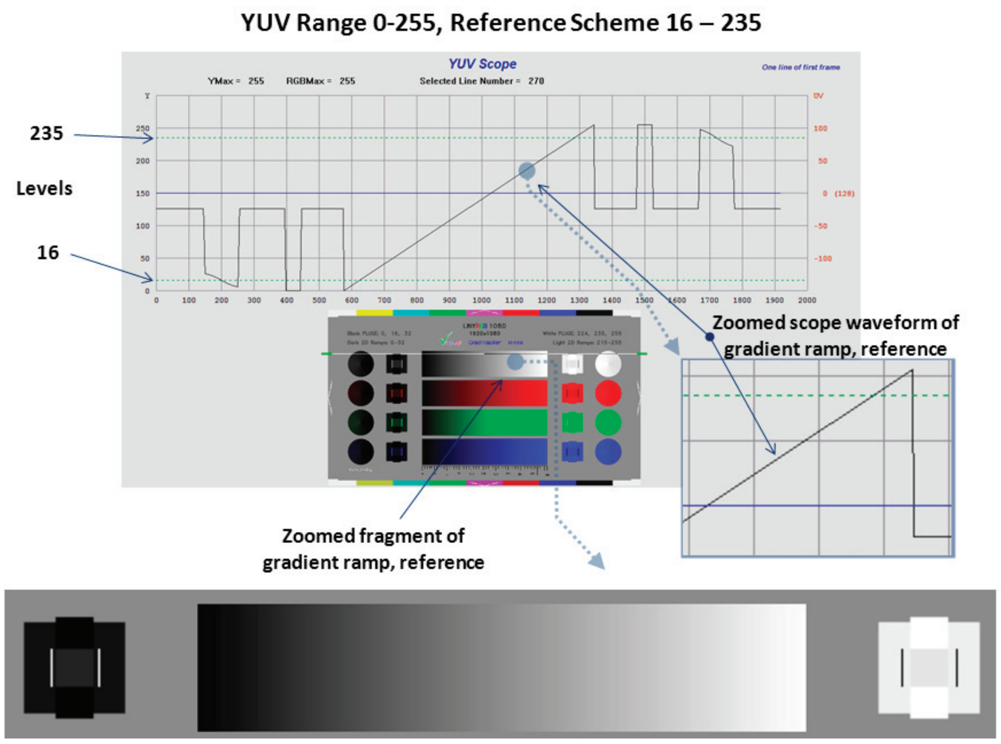


Figure 5

Depiction of linear YRGB test pattern: reference scheme of 16-235, ramp test range 0-255, inclusive sub range levels below black (16) and white (235). Notice clean gradient and no banding of color gradient ramp and clean waveform on the ramp.

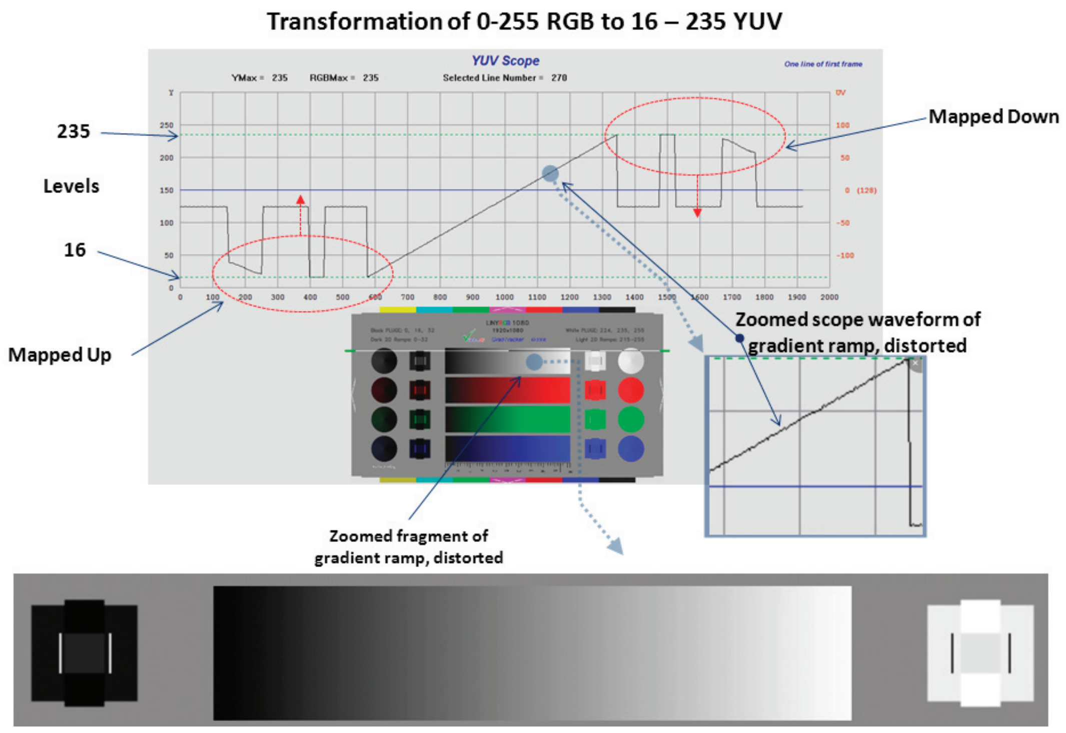


Figure 6

Depiction of Linear YRGB reference test pattern transformation from 0-255 RGB to 16-235 YUV, with color matrixing and range mapping. Notice banding of color gradient ramp and slightly distorted waveform on the ramp, no potential danger of clipping (i.e. safe mode, boost proof). Player will do final transformation of level mapping from 16 down to 0 and 235 up to 255.

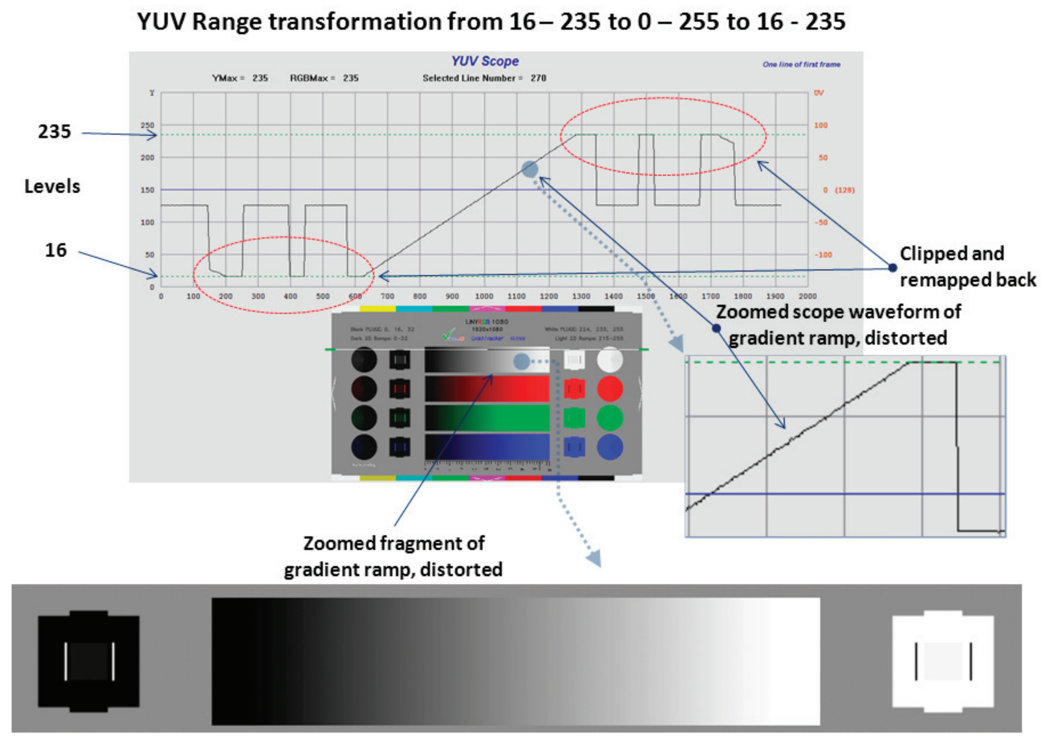


Figure 7

Depiction of result of two cascaded level mapping processes. Notice distorted gradient and banding of color gradient ramp and distorted waveform on the ramp, also visible gray limits of white on the right of Luma gradient. Black square on the left is no longer showing perpendicular rectangular shape, clearly visible in reference Figure 5. White square on the right is no longer showing white perpendicular rectangular shape, clearly visible in reference Figure 5. Hence, the clipping of white and black occurs. If further transformations take place, the entire color scheme will deteriorate significantly, resulting in irreversible reduction of picture quality.

Size Transformation

As demonstrated in the previous example, the encoding software may choose, without informing the operator, to scale the source from 1080 on input to 720 for processing and scaling back to 1080 for final output (1080 in > 720 at processing time -> 1080 out) (Figures 7 and 8).

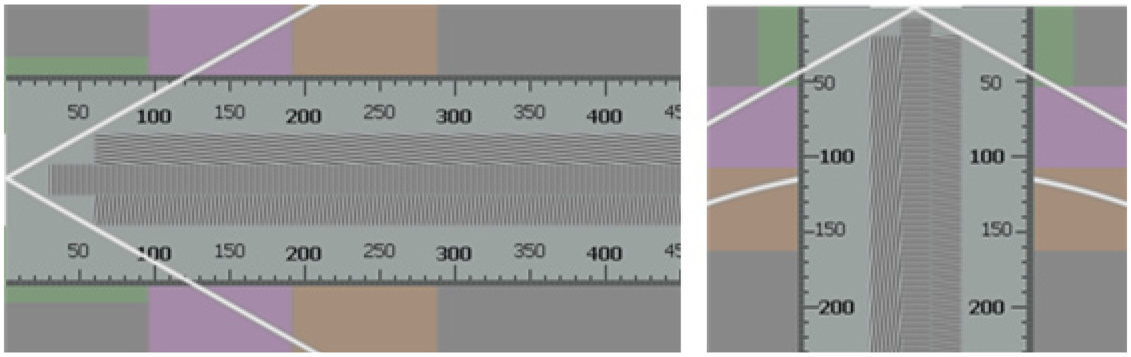


Figure 8

Above is a depiction of portion of the Tri-band combination from dynamic test pattern series "ST", by VideoQ. What is seen here is unnecessary scaling, causing beat waves on both horizontal and vertical Tri-band Patterns, similar to the process described in example 3 where internal processing is scaled to lower resolution and then scaled back up. This scaling process 1080 to 720 back to 1080 makes picture softer unnecessarily. While the human eye can forego that on soft pictures, the complex synthetic patterns will not be as forgiving and will reveal such transformation at glance. Even highest quality (10-15 taps) scaler will not keep Tri-band Patterns unchanged if video is scaled down.

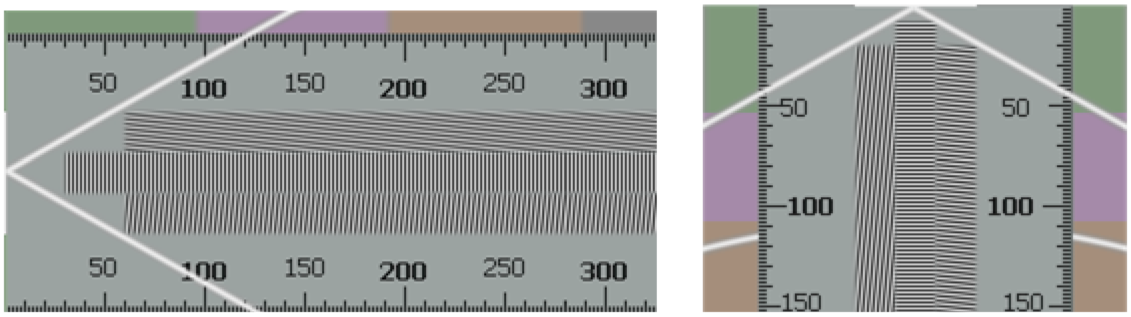


Figure 9

Above is a depiction of portion of Tri-band combination from dynamic test pattern series "ST", by VideoQ. What is seen here is a correct scaling setting, causing no visible beat waves on both horizontal and vertical Tri-band Patterns.

Frame Rate Transformation

Frame rate conversion can also be altered during processing without operator knowledge. If your input is 30 FPS and your output is 30 FPS, the system architecture may force you to transform it in the interim format for speed of processing (e.g. hybrid software/hardware/software, etc.), to 60 FPS sometimes without operator being informed (30 FPS in -> 60 FPS processing time -> 30 FPS out).

Bitrate Transformation

Bitrate can also be a factor that gets compromised. For example, the transcoding profile specifies 18Mbps output but the encoder may decide to treat it as 6 and pad the rest for speed of processing, or any other reasons, while outputting the result at 18Mbps (18Mbps set -> 6Mbps at processing time -> 18Mbps out).

"You will never have a second chance to do the first compression right."
—popular slogan

Cascaded Codecs

According to some broadcasters, the typical amount of cascaded codecs in today's modern production ranges up to five. The cumulative effect of this in transcoding operations is similar to the language translation of an original text through several interim languages and then back to original text. Imagine a translation of an article written in English as it passes through interim phases of other languages (e.g. English to Russian to German to Japanese to Spanish and back to English), the tone of the article is lost and the original idea is no longer there, regardless of how accurate the translation process is.

Why Test?

All in all, the picture may be perceived from to the human eye as being fine, however, it may not be good enough for downstream processing; thus the need for confidence testing with dynamic test patterns. Looking at the transcoded media clip for quality analysis is fine, but it doesn't tell the whole story, and professional "compressionists" need to know the whole story in order to make qualifying decisions about which variables to focus on in order to achieve optimum profile settings. Specially designed professional test patterns remain suitable for accurate measurements even after low bitrate coding, heavy scaling and/or cropping (e.g. after down-conversion for mobile devices).

If you are observing some sort of a problem on the synthetic test pattern and not seeing it in the actual clips, this means that in another clip — sooner or later — you will see that problem. If you don't see it now, it only means that the clip is not critical enough. However, if you are seeing a problem on the clip, but not seeing it on the synthetic test clip, it simply means that you are not using the appropriate test pattern to reveal the particular problem. If you are seeing a problem in the media clip, but cannot link this problem with specific reason of this issue , you need to disjoin, magnify, and isolate the problem. This is why it is not enough to have one type of synthetic testing, these tests must be complex and comprehensive to reveal the entire spectrum of problems — and sometimes need to be long enough and repetitive enough to reveal the issues, due to cumulative nature of some problems (e.g. AV Sync, Buffer occupancy and removal, etc.)

Profiling does not guarantee the highest quality output, it simply means that someone has put an effort to make it easier and faster for others to create a desired output.

Profiling of typical, common-used settings is the best approach for achieving optimum output results. However, such an approach may lead to creation of a large number of profiles, which can be difficult to manage. Because of this, many people choose to maintain as few profiles as possible, serving most typical situations. It is recommended, however, that many specialized profiles be maintained for the highest quality output.

Using Technical Testing Content

The following pages will detail the use of synthetic dynamic test patterns specially designed by VideoQ to diagnose problems in various elements of profile creation, workflow setup, and verification. The following test patterns are discussed:

Levels, Colors, and Color Matrix

- **LINYRGB** — Y,R,G,B Gradation & PLUGE
- **CB** — Color Bars: SMPTE (60 fps), EBU (50 fps)

Scaling, Cropping, and Sharpness

- **ST** — Horizontal & Vertical Rulers, Static Radial Mire & Crop Markers

Interlacing, Frame Rate Conversion, and Cadence

- **FRC** — Moving Frequency Bursts and Frame Counter

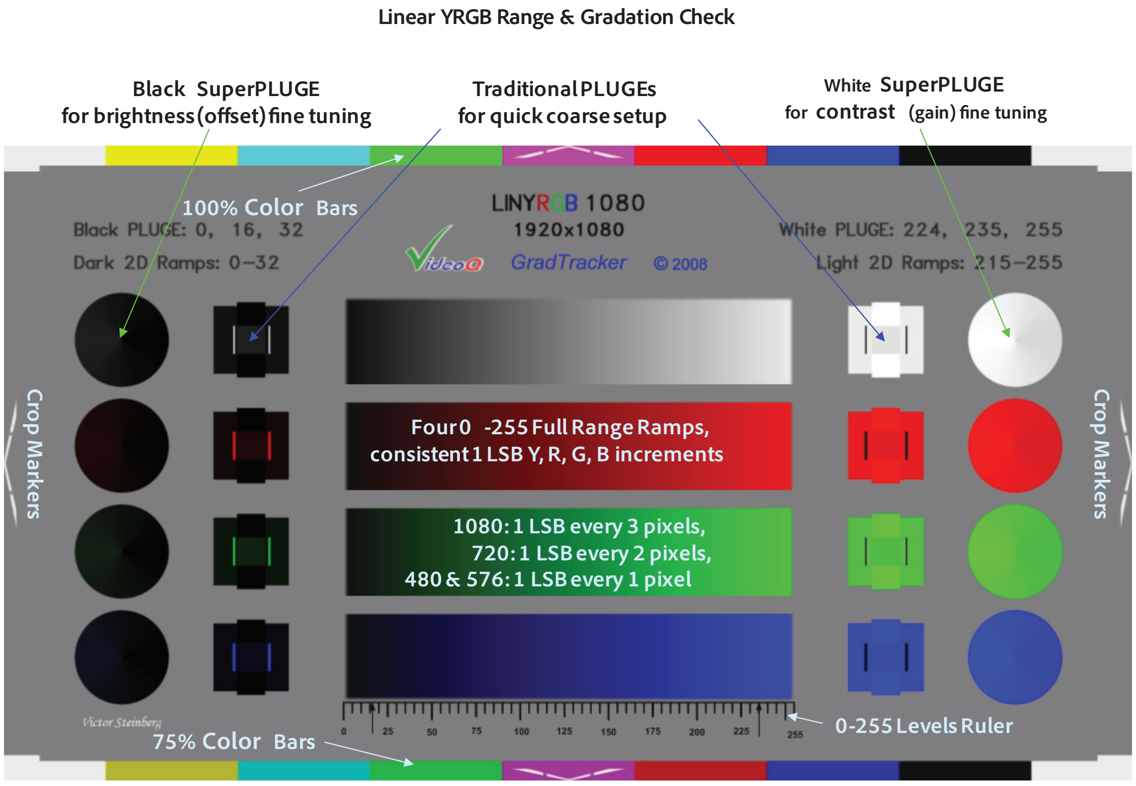
Audio Video Synchronization

- **CNT** — Timeline Markers & Frame Counter
- **AV Sync Audio Test** — Claps Pulses & 1 KHz Bursts

Levels, Colors, and Color Matrix

LINYRGB — Static Test

Y, R, G, B Range and Gradations Linearity Check



Linear YRGB Range & Gradation Check
Black SuperPLUGE for brightness (offset) fine tuning
Traditional PLUGEs for quick coarse setup
White SuperPLUGE for contrast (gain) fine tuning
100% Color Bars
LINYRGB 1080
1920x1080
GradTracker
© 2008
Black PLUGE: 0, 16, 32
Dark 2D Ramps: 0–32
White PLUGE: 224, 235, 255
Light 2D Ramps: 215–255
Crop Markers
Four 0 -255 Full Range Ramps, consistent 1 LSB Y, R, G, B increments
1080: 1 LSB every 3 pixels, 720: 1 LSB every 2 pixels, 480 & 576: 1 LSB every 1 pixel
Victor Steinberg
75% Color Bars
0-255 Levels Ruler
0 25 50 75 100 125 150 175 200 225 255

Figure 10

Depiction of a static test pattern of YRGB range and gradation linearity check. Colored lines with arrows and text are not part of the test pattern and are superimposed for explanation purposes only.

How to Use This Test (Figure 10):

General Use Note: This test can inserted as a frame into a media file for transcoding and/or as a live signal input through the use of special external devices.

YRGB Range Check:

- By observing YRGB (YUV) levels in Video Editor Scope or similar software tool.

Note that Color Space Conversion, such as 16-235 <-> 0-255, YUV <-> RGB and/or 601 <-> 709, may cause significant YRGB (YUV) level errors

- By checking the appearance of black and white PLUGEs and superPLUGEs components (Figures 10 and 11).

Black PLUGE & SuperPLUGE

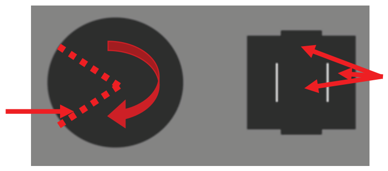
Usage

Fine Tuning

Brightness (Y Offset) is too low

Coarse Tuning

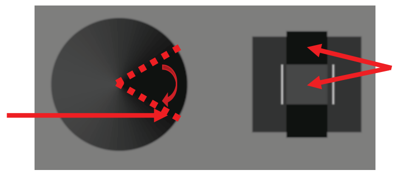
Clipped sector (with no shades of gray) is much more than 180 degrees



Both central super-black vertical band and central small square are almost the same brightness as big black square

Brightness is too high

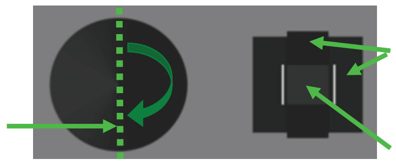
Clipped sector (with no shades of gray) is much less than 180 degrees



Both central super-black vertical band and central small square are clearly visible

Brightness is correct

Conicalgrayscaleis clipped exactly half-circle (180 degrees), no shades of gray on the right half



The super-black vertical band is almost the same brightness as big black square

Central small square is clearly visible

Figure 11

Depiction of black PLUGE & SuperPLUGE elements from static test pattern of YRGB range and gradation linearity check. Colored lines with arrows and text are not part of the test pattern and are superimposed for explanation purposes only.

White PLUGE & SuperPLUGE

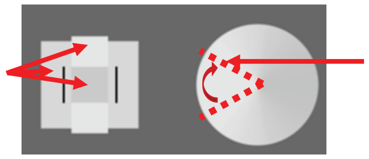
Usage

Coarse Tuning

Contrast (Gain) is too low

Fine Tuning

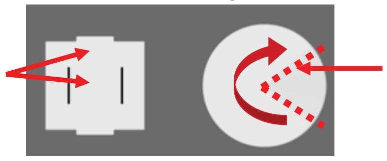
Both central super-white vertical band and central small square are clearly visible



Clipped sector (with no shades of gray) is much less than 180 degrees

Contrast is too high

Both central super-white vertical band and central small square are almost the same brightness as big white square

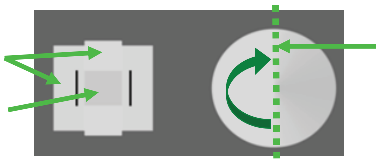


Clipped sector (with no shades of gray) is much more than 180 degrees

Contrast is correct

The super-white vertical band is almost the same brightness as big whitesquare.

Central small square is clearly visible



Conicalgrayscaleis clipped exactly half-circle (180 degrees), no shades of gray on the left half

Figure 12

Depiction of white PLUGE & SuperPLUGE elements from static test pattern of YRGB range and gradation linearity check. Colored lines with arrows and text are not part of the test pattern and are superimposed for explanation purposes only.

YRGB Linearity Check:

- **By observing YRGB (YUV) levels in Video Editor Scope or similar software tool** — if linearity is preserved (correct mode of operation) reconstructed YRGB Ramps still have *regular* 1 LSB increments every 1, 2, or 3 pixels depending on video image width (it should be different, but display a *regular* spatial level increments pattern, if image is scaled)
- **By checking the appearance of full range Ramps** — if linearity is preserved (correct mode of operation) there should be no visible "*banding*", (i.e. gradation gradient should be constant)

The transformation of YUV 16-235 to RGB 0-255 needs to happen, but it is important that it occur at the end of the processing chain, as the display side (RGB 0-255) transformation will happen anyway. By making certain that this transformation does not happen earlier than at the end, you are avoiding chance of further clipping and irreparable damage to the color scheme transformations. Also, similar damage can be done by boosting gain. However, ultimately, anything outside of 16-235 will be lost or clipped, so don't place any valuable picture data outside of 16-235 scheme.

RGB scheme of 0 – 255 does not foresee levels below black and above white, which exist in YUV scheme 16-235. Because of this, transformation of 16-235 to 0-255 done once, simply clips levels and no major distortions introduced besides banding. However, if 0-255 gets interpreted erroneously as 16–235 then black and white crash happens (16 levels of close to black and 20 levels of close to white will be clipped). This issue is unrecoverable, unfortunately, even though one would try to reverse it by converting it to 16-235; that data is gone. This would make the picture worse and add banding.

Even if your image was originally shot as 0-255, keeping it as such may result in a chance of severe clipping because it may be erroneously interpreted as a 16-235 scheme. Therefore, it makes sense to transform it to a 16-235 scheme and maintain that scheme throughout the process, thus minimizing the chance of inadvertent clipping during the remaining encoding processes.

In summary, it is recommended that the 16-235 color scheme be kept for as long as possible throughout the entire duration of transcoding processing workflow.

Scaling, Cropping, and Sharpness

ST — Static Test
Image Scaling, Cropping, and Sharpness Check

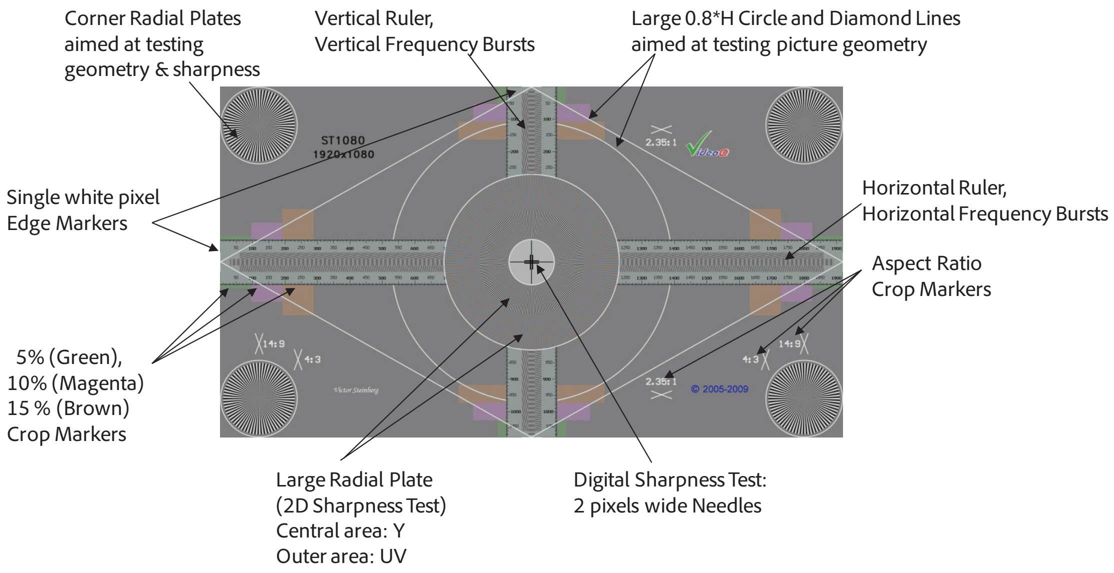

Corner Radial Plates aimed at testing geometry & sharpness
Vertical Ruler, Vertical Frequency Bursts
Large 0.8*H Circle and Diamond Lines aimed at testing picture geometry
ST1080 1920x1080
2.35:1
Single white pixel Edge Markers
Horizontal Ruler, Horizontal Frequency Bursts
Aspect Ratio Crop Markers
14:9
4:3
5% (Green), 10% (Magenta) 15 % (Brown) Crop Markers
Victor Steinberg
© 2005-2009
Large Radial Plate (2D Sharpness Test) Central area: Y Outer area: UV
Digital Sharpness Test: 2 pixels wide Needles

Figure 13

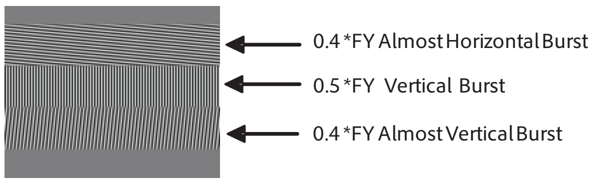
Static Test pattern. Black lines with arrows and text are not part of the test pattern and are superimposed for explanation purposes only.

How to Use This Test:

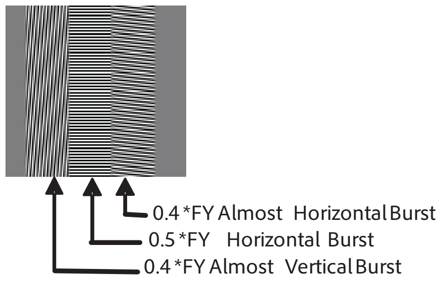
General Use Note: This test can inserted as a frame into a media file for transcoding and/or as a live signal input through the use of special external devices.

Scaling and sharpness is paired, because of scaling sharpness loses its integrity. When scaling is compromised, sharpness is compromised as well.

Horizontal and Vertical Tri-band Combination Burst Patterns



0.4 *FY Almost Horizontal Burst
0.5 *FY Vertical Burst
0.4 *FY Almost Vertical Burst



0.4 *FY Almost Horizontal Burst
0.5 *FY Horizontal Burst
0.4 *FY Almost Vertical Burst

Figures 14, 15

Illustrating cut-out section from ST test depicting the use of Tri-band patterns.

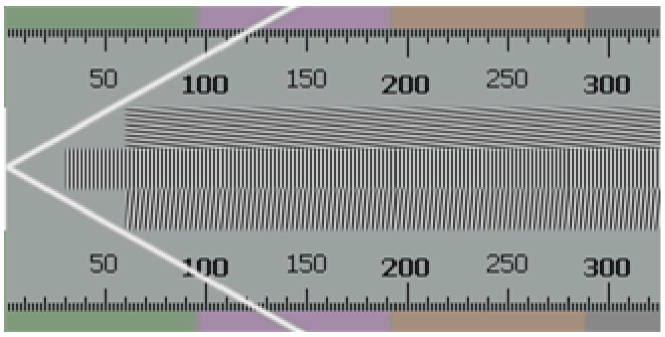
There are two groups of bursts with frequencies proportional to luma pixels rate FY: full length horizontal and full height vertical bursts bands, each consisting of maximum luminance frequency of exactly 0.5 FY in the middle with slightly oblique bands of 0.4 FY surrounding the middle burst.

The central 0.5 FY bands are especially sensitive to any errors in pixels clock (e.g. when analog interfaces such as VGA, YPrPb are involved), mapping or scaling. Two other bands allow differentiation between horizontal and vertical distortions thru the whole picture area — from the left picture edge to the right picture edge and from top to bottom.

Vertical and almost vertical burst lines test horizontal frequencies, whilst horizontal and almost horizontal lines test vertical frequencies.

Tri-band Combination Burst Pattern Usage

No Scaling



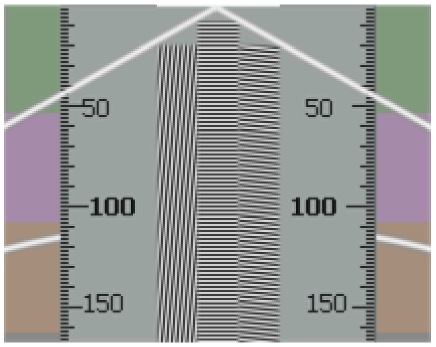


Figure 16

Zoomed in cut-out of section from ST test depicting the use of Tri-band patterns and example of correct setting with no scaling or high quality scaling.

There are no visible beat waves on both horizontal and vertical Tri-band patterns.

With Scaling

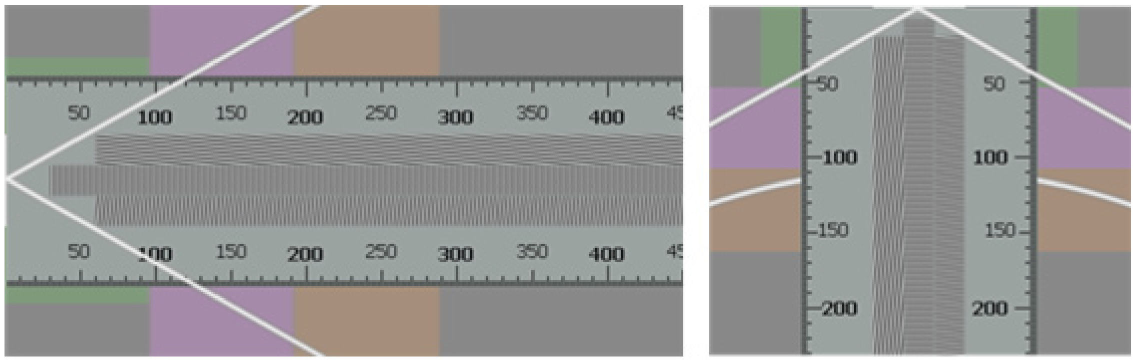


Figure 17

Zoomed in cut-out of section from ST test depicting the use of Tri-band patterns and example of use of incorrect setting with scaling or poor quality scaling.

There are visible beat waves on both horizontal and vertical Tri-band patterns, caused by scaling.

Diamond Pattern and Crop Markers Usage

No Cropping

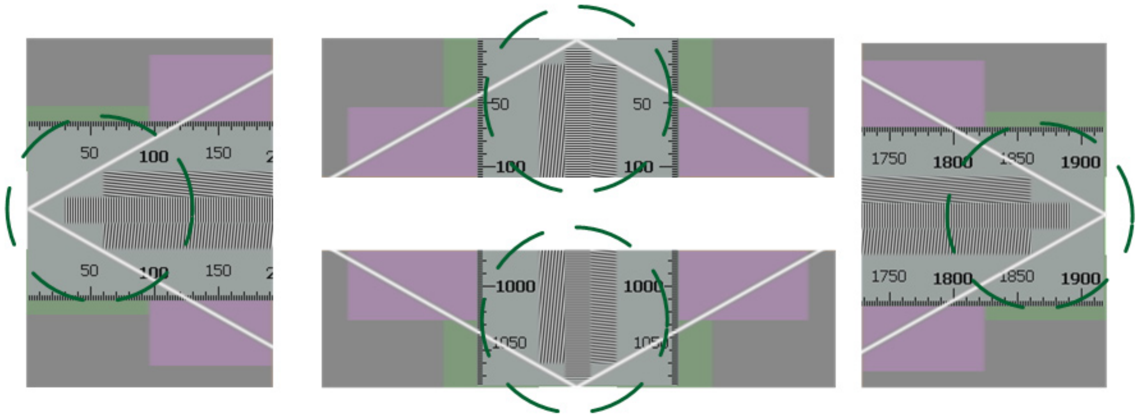


Figure 18

Zoomed in cut-out of sections from ST test depicting the use of Diamond Pattern and Crop Marker and example of use of correct setting with no cropping. Colored circles are not part of the test pattern and are superimposed for explanation purposes only.

All picture edges are not cropped and single pixel white markers are visible.

With Cropping

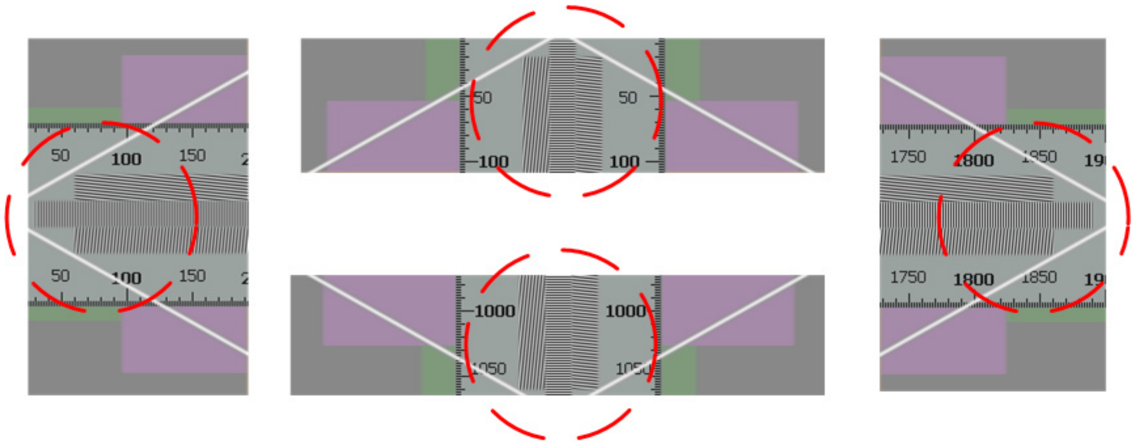


Figure 19

Zoomed in cut-out of sections from ST test depicting the use of Diamond Pattern and Crop Marker and example of use of incorrect setting with cropping. Colored circles are not part of the test pattern and are superimposed for explanation purposes only.

All picture edges are cropped and single pixel white markers are not visible.

Frame Aspect Ratio Markers
2.35:1, 4:3, 14:9

Crop Markers

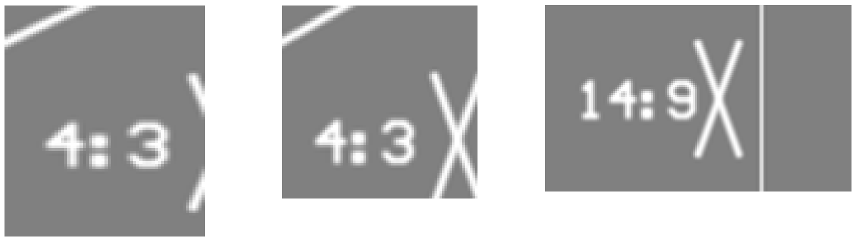


Figure 20

Zoomed in cut-out of sections from ST test depicting the use of Frame Aspect Ratio Crop Markers.

Correct 4:3 Crop

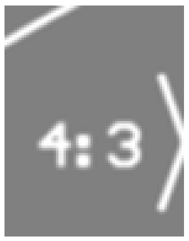


Figure 21

Zoomed in cut-out of sections from ST test depicting the use of 4:3 Frame Aspect Ratio Crop Marker and example of use of correct crop setting.

Wrong 4:3 Crop

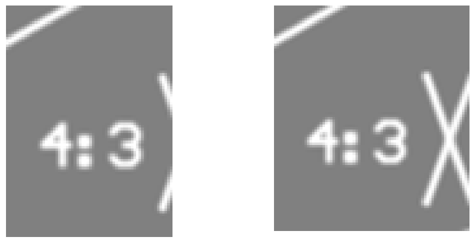


Figure 22

Zoomed in cut-out of sections from ST test depicting the use of 4:3 Frame Aspect Ratio Crop Markers and example of use of incorrect crop setting.

720 and 1080 scan line patterns are designed for measurement in 16:9 format, as well as in 4:3, 14:9, and 2.35:1 frame formats. Cross-shaped Frame Format Markers indicates precise crop area for each corresponding frame format.

The following are the most popular scale and crop modes:

- 4:3 crop is used to display 16:9 content on legacy standard definition television sets, smaller web sites, or in-page video segments
- 14:9 is a compromise (non-letterboxed) mode used in simulcast broadcasting to present 16:9 content on 4:3 and 16:9 screens
- 2.35:1 is used to show letterboxed "cinemascope" movies on 16:9 screens

Radial Plates Usage

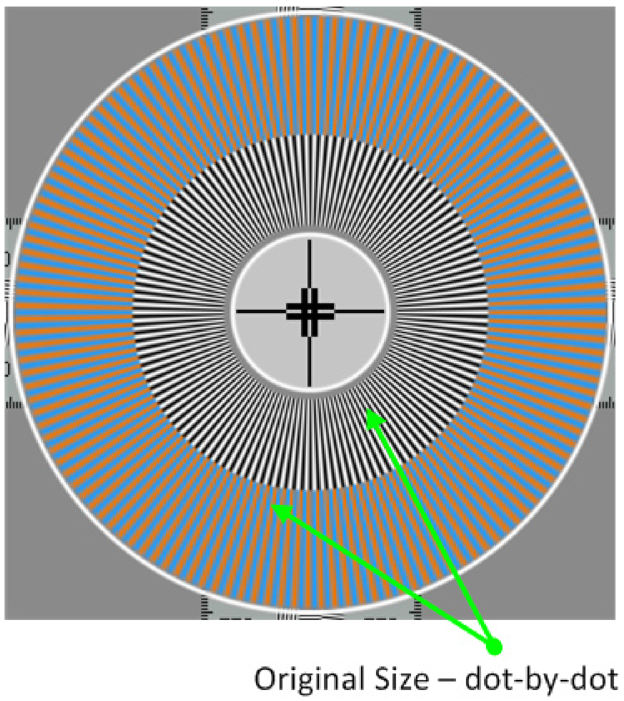


Figure 23

Zoomed in cut-out of sections from ST test depicting the use of radial plates in original size and no scaling is observed. Color lines with arrows and text are not part of the test pattern and are superimposed for explanation purposes only.

Full contrast of fine details in all directions is seen without any issues inhibited by scaling (Figure 23).

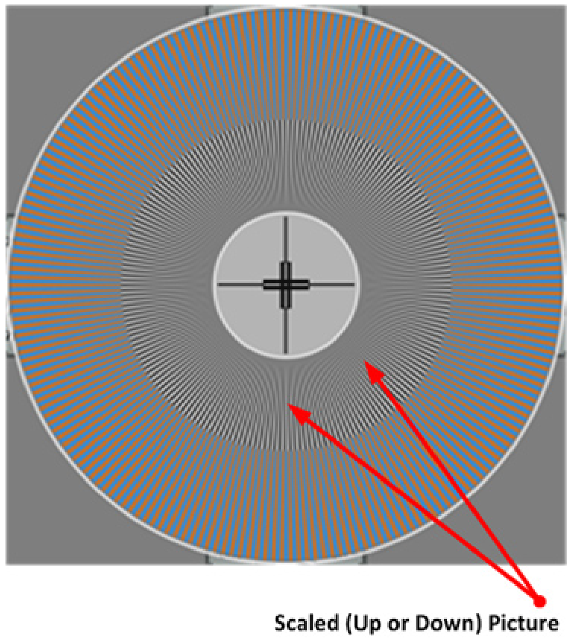


Figure 24

Zoomed in cut-out of sections from ST test depicting the use of radial plates in scaled up/down size and the use of scaling is observed. Color lines with arrows and text are not part of the test pattern and are superimposed for explanation purposes only.

Figure 24 depicts loss and/or distortion of fine details.

Sharpness Test Usage

Optimal Sharpness

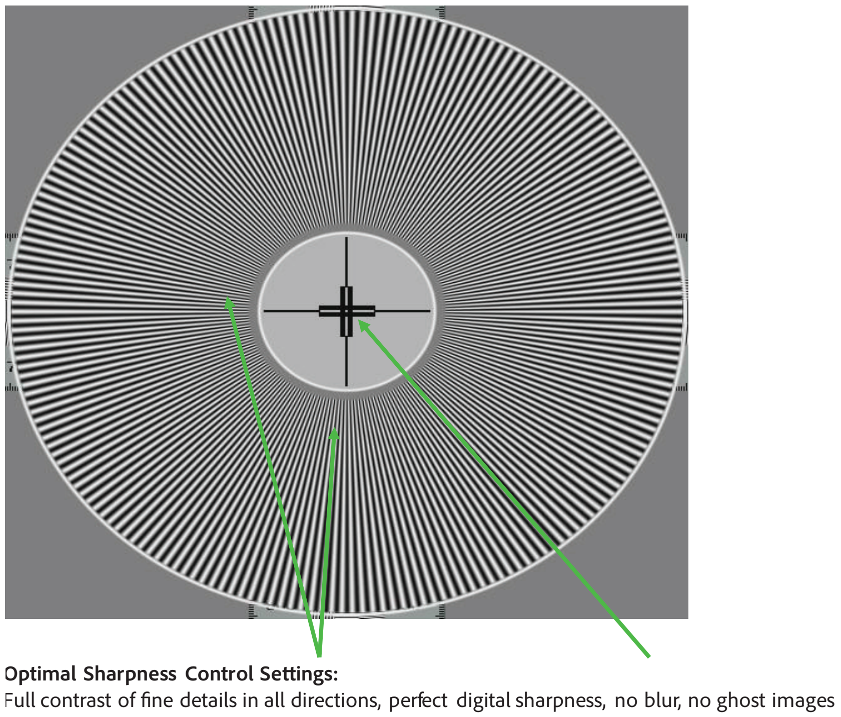

Optimal Sharpness Control Settings:
Full contrast of fine details in all directions, perfect digital sharpness, no blur, no ghost images

Figure 25

Zoomed in cut-out of sections from ST test depicting the sharpness and optimal control of sharpness. Color lines with arrows and text are not part of the test pattern and are superimposed for explanation purposes only.

Sharpness Test Usage

Not Enough Sharpness

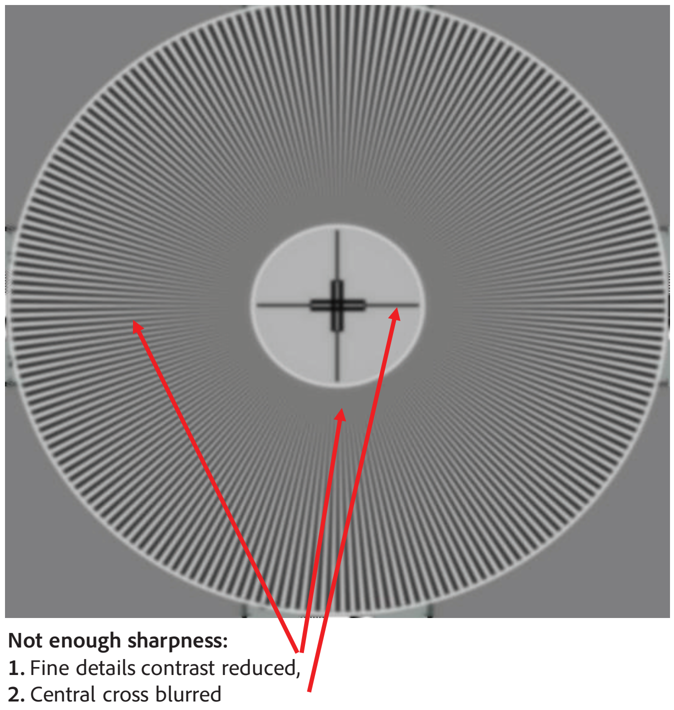

Not enough sharpness:
1. Fine details contrast reduced,
2. Central cross blurred

Figure 26

Zoomed in cut-out of sections from ST test depicting the sharpness and lack of sharpness. Color lines with arrows and text are not part of the test pattern and are superimposed for explanation purposes only.

Too Much Sharpness

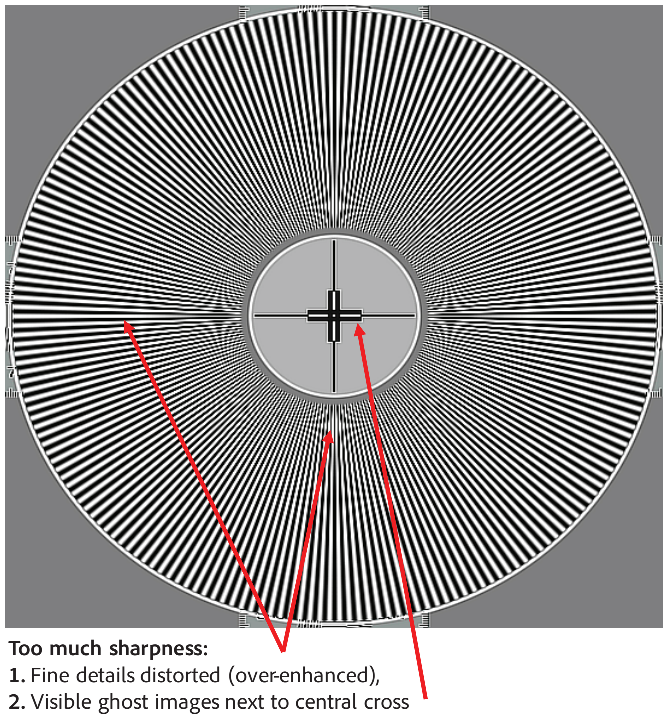

Too much sharpness:
1. Fine details distorted (over-enhanced),
2. Visible ghost images next to central cross

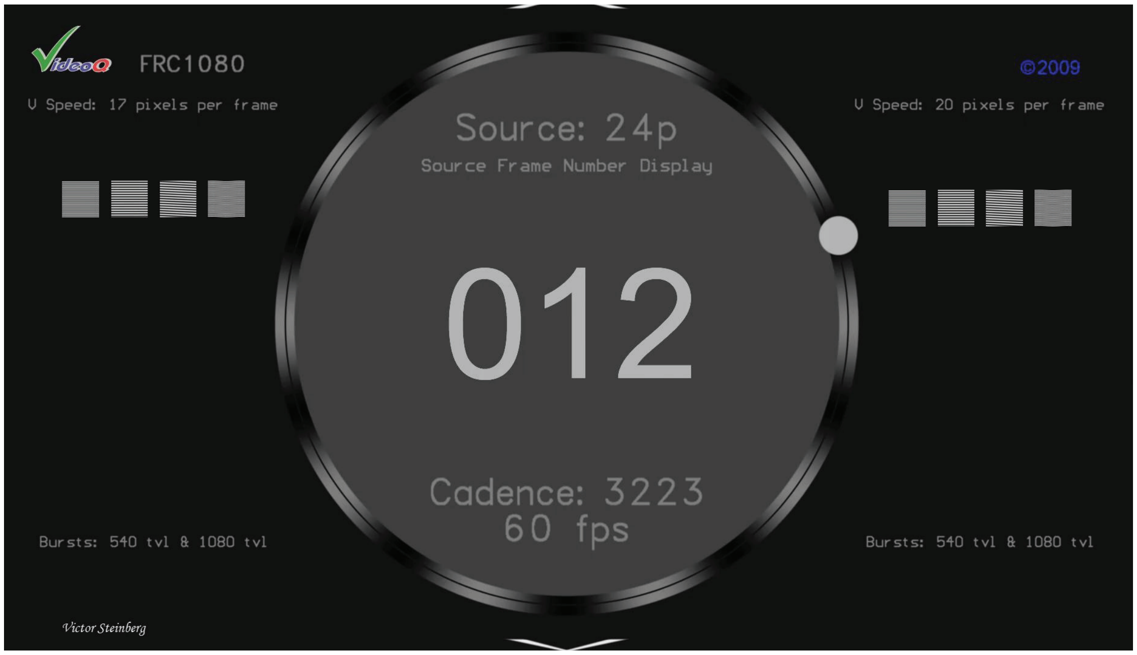
Figure 27

Zoomed in cut-out of sections from ST test depicting the sharpness and excess of sharpness. Color lines with arrows and text are not part of the test pattern and are superimposed for explanation purposes only.

Interlacing, Frame Rate Conversion, and Cadence

Frame Rate Conversion (FRC) — Dynamic Test

De-interlacing, Frame Rate Conversion & Cadence Correction Check

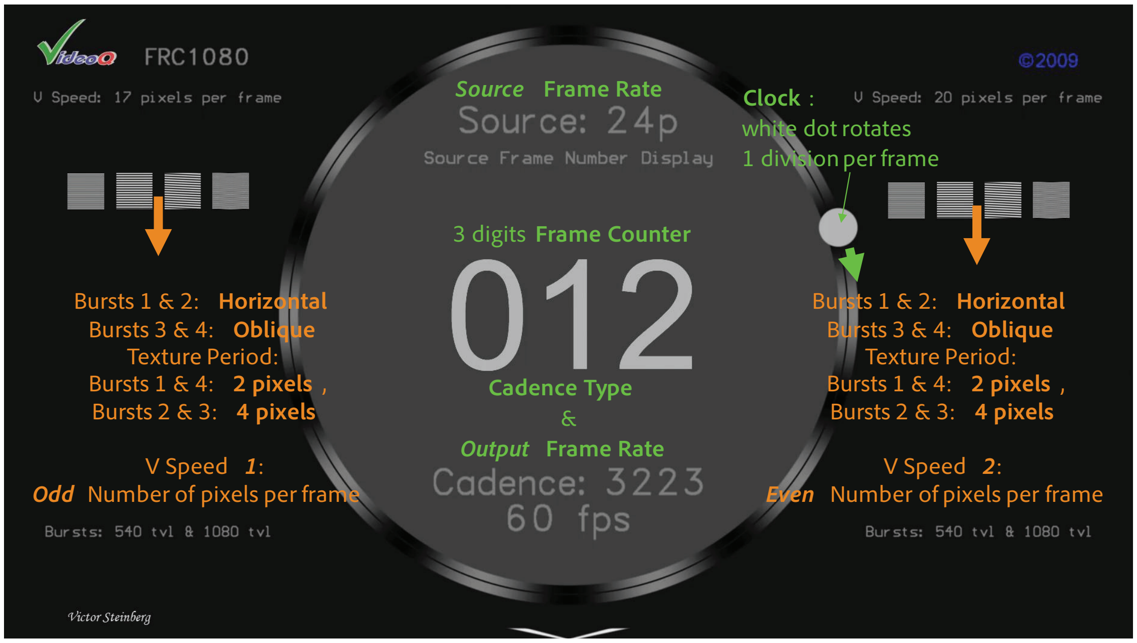


VideoQ
FRC1080
©2009
V Speed: 17 pixels per frame
V Speed: 20 pixels per frame
Source: 24p
Source Frame Number Display
012
Cadence: 3223
60 fps
Bursts: 540 tvl & 1080 tvl
Bursts: 540 tvl & 1080 tvl
Victor Steinberg

Figure 28

Depiction of a dynamic test pattern of de-interlacing, frame rate conversion, and cadence correction.

FRC Test Usage



VideoQ
FRC1080
©2009
V Speed: 17 pixels per frame
V Speed: 20 pixels per frame
Source Frame Rate
Source: 24p
Source Frame Number Display
Clock : white dot rotates 1 division per frame
3 digits Frame Counter
012
Bursts 1 & 2: Horizontal
Bursts 3 & 4: Oblique
Texture Period:
Bursts 1 & 4: 2 pixels ,
Bursts 2 & 3: 4 pixels
Cadence Type & Output Frame Rate
Cadence: 3223
60 fps
V Speed 1: Odd Number of pixels per frame
V Speed 2: Even Number of pixels per frame
Bursts: 540 tvl & 1080 tvl
Victor Steinberg

Figure 29

Depiction of a dynamic test pattern of de-interlacing, frame rate conversion, and cadence correction with usage commentary. Colored lines with arrows and colored text are not part of the test pattern and are superimposed for explanation purposes only.

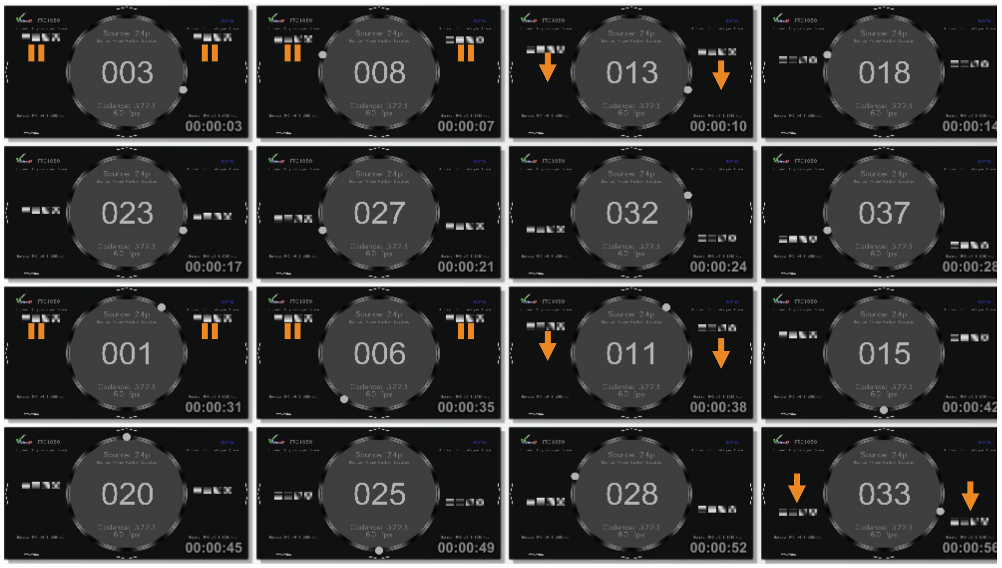


Figure 30

Snapshots of FRC dynamic test pattern in motion of de-interlacing, frame rate conversion, and cadence correction. Colored arrows (depicting direction) and colored lines (depicting paused moment in time) are not part of the test pattern and are superimposed for explanation purposes only.

The appearance of the bursts may change significantly after sub-optimal spatial and/or temporal scaling. Snapshots above illustrate possible variations. They are strongly dependent on Motion Speed and Motion Type. Slightly different vertical position increments of odd numbers of frame lines per frame, on the left side (or even, on the right side), demonstrates differences in performance between various de-interlacers.

Moving Frequency Bursts Usage

De-Interlacing Quality Check Methodology

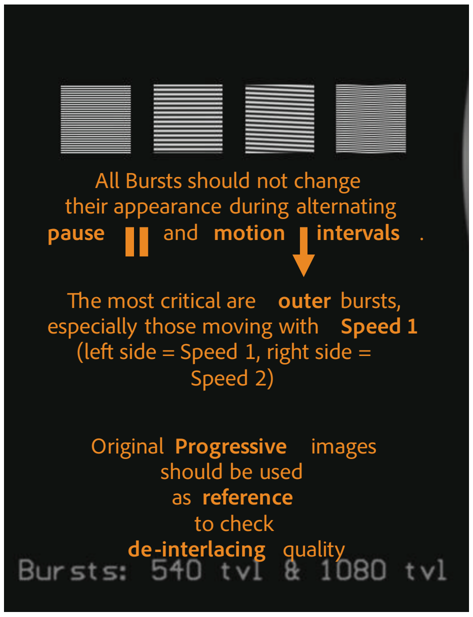

All Bursts should not change their appearance during alternating **pause** and **motion** **intervals** .
The most critical are **outer** bursts, especially those moving with **Speed 1** (left side = Speed 1, right side = Speed 2)
Original **Progressive** images should be used as **reference** to check **de-interlacing** quality
Bursts: 540 tvl & 1080 tvl

Figure 31

Zoomed in cut-out of moving frequency bursts section from FRC dynamic test pattern. Colored arrows (depicting direction) and colored lines (depicting paused moment in time) are not part of the test pattern and are superimposed for explanation purposes only.

Frame Rate Conversion and Cadence Correction Check Usage

Frame Counter Continuity (Drop/Freeze) Check:

- Observe 3 digit *Frame Numbers* in Video Editor or similar software tool

•Check the appearance of *white Clock Dots* — there should not be *frozen, double, or missing* images (no noticeable irregularities)

Cadence Correction Check:

- Observe 3 digit frame numbers in video editor or similar software tool — for example adequate 60 FPS 3232 to 24p 111 correction results in regular increment of visible frame number by one.
- By checking the appearance of white Clock Dots — correct results in regular motion pattern, without frozen, double, or missing images.

Audio Video Synchronization

CNT — Dynamic Test

AV Sync Test

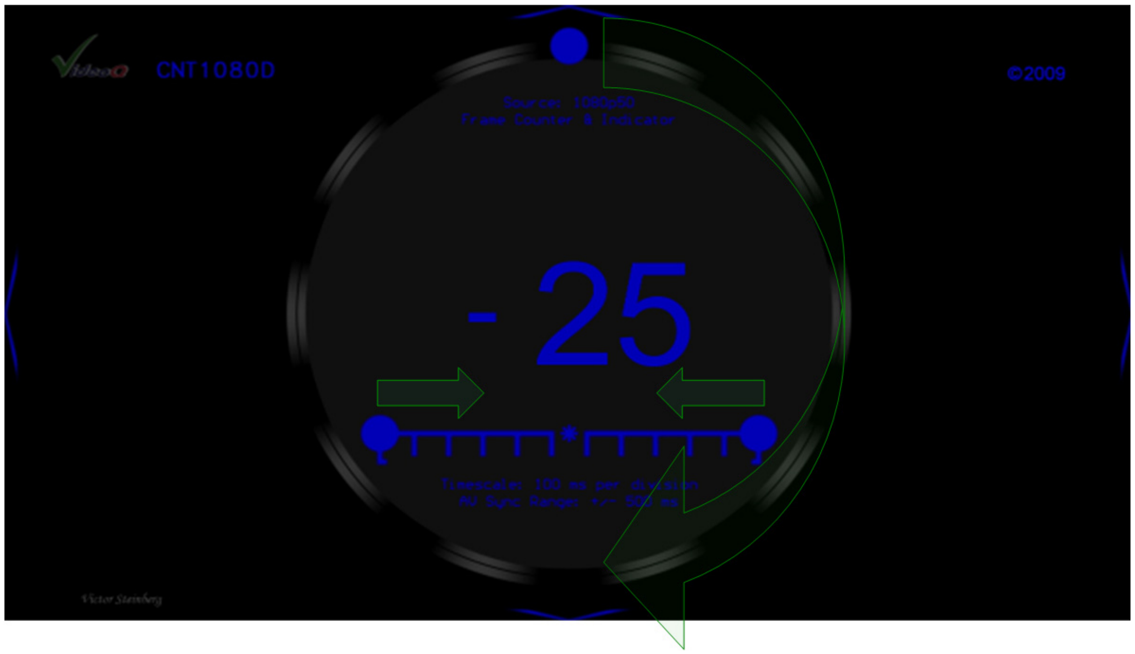

CNT1080D
©2009
Source: 1080p50
Frame Counter & Indicator
-25
Timescale: 100 ms per division
AV Sync Range: +/- 500 ms
Victor Steinberg

Figure 32

Depiction of video component of dynamic test pattern for audio/video synchronization verification. Colored arrows (depicting direction) are not part of the test pattern and are superimposed for explanation purposes only.

Technical Specification:

Measurable AV Sync Range:	+/- 30s
AV Sync Measurement Accuracy:	+/-1 ms
Timeline Grid Display:	2 x 5 x 100 ms
Clock Rotation:	10 frames, 36 degree per frame

Clock (rotating blue dot) serves to check motion continuity and smoothness, (i.e. lack of any drop/freeze disturbances, which may affect the measurement results). See “CNT Test Audio Element” for use case details.

CNT Test Audio Element

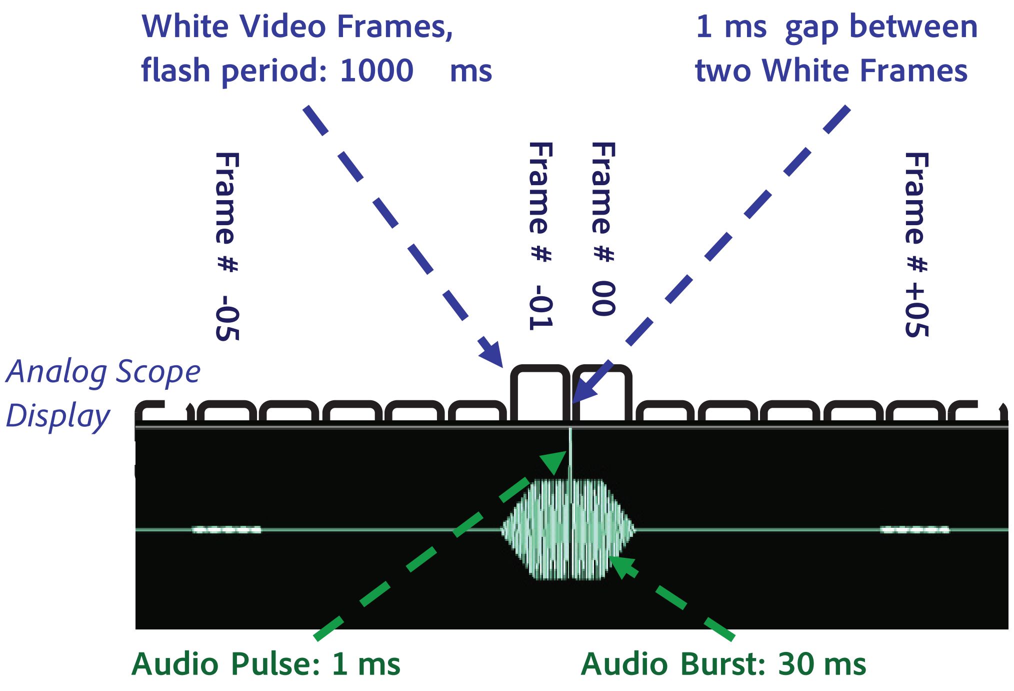


Figure 33

Audio component of AV synchronization check test with corresponding White Frame reference. What is shown here is a timeline snippet of AV sync test with overlaying frame information to help explain the underlying audio component of the test.

	Peak Level	Width	Period
Pulses:	0 dBfs	1 ms	1000 ms
Loud Bursts:	-6 dBfs	30 ms	1000 ms
Low Bursts:	-40 dBfs	20 ms	100 ms

CNT in Motion — Snapshots in time

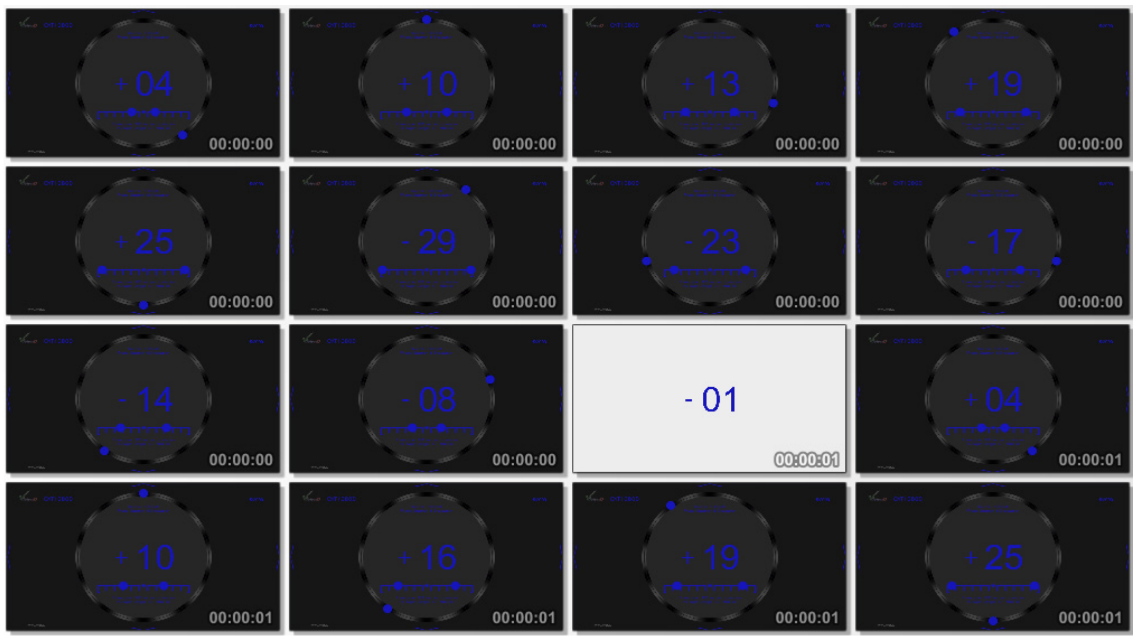


Figure 34

Depiction of snapshots in time of CNT dynamic test pattern in motion.

Video APL (Average Picture Level) changes from very low to almost maximum 100% white for two frames, flashing white every one second. Two special White Frames are numbered -01 and 00. Audio Pulse hits exactly the middle point of this group of two frames (i.e. the boundary between two White Frames) (see Figure 34).

AV Sync Check Methodology

Visual-Aural Check:

- Listen to beeping low level 1KHz bursts, then much louder 1KHz Loud Burst starting 15ms before main Clap Pulse, *and* observe negative 2 digits Frame Numbers approaching "-01" whilst blue Clock Dots are coming closer.
- AV Sync errors can be *estimated* with about 30ms accuracy

Instrumental Measurement:

Step 1:

- Display CNT 2 digits Frame Numbers in Video Editor and find the timeline position of two White Frames numbered "-01" and "00".

Step 2:

- Find the timeline positions of the *Loud Burst* and *Clap Pulse in the* Audio Track.
- AV Sync Errors can be *measured* with about 1ms accuracy.
- For large errors (more than 500ms) also check the relative timeline positions of the AV components for a longer period of time(5 sec to 60 sec) in the AV Sync Test Sequence.

Improving Overall Video Quality

Achieving the highest possible video quality within set target parameters is the goal of any encoding project. This section will begin by grouping key elements of the encoding process and explain their impact on production of high quality video assets. Then, the impact of these key elements on the playback experience will be discussed.

Key considerations and affected areas:

Video

- Video bitrate allocation
- Denoising, degraining, deblocking, and similar preprocessing operations
- Deinterlacing, including inverse telecine
- Frame rate conversion
- Video frame size scaling, including aspect ratio conversion
- Color space and levels scheme transformation

Audio

- Audio bitrate allocation
- Levels normalization
- Channel remapping and downmixing
- Sampling rate and bitrate rate conversion

Stream packaging

- Stream bitrate budget allocation
- Audio Video synchronization
- Keyframe alignment
- Playback capability of a client system

Video

Market trends in general: According to data published by Akamai in the report for Quarter 3 of 2009, the US average broadband connection was 3.8Mbps with broadband adoption of 57%. These details and other relevant information can be found in these documents: Adoption Trends Report: *http://www.akamai.com/dv5* and State of the Internet White paper: *http://www.akamai.com/stateoftheinternet/.*

Video Bitrate Allocation

Bitrate calculation is one of the critical parts of the encoding process. It determines the how much data can be packed into the specific stream and what kind of quality factor can be created as a result.

Denoising, Degraining, Deblocking, Preprocessing Operations

Film grain can add a lot of unnecessary artifacts during the encoding process. Grain is seen as part of the picture, so the encoder will have a hard time distinguishing it from the picture itself. Film grain and noise can eat up a large amount of your bitrate in the encoding process, as the grain pattern changes completely from frame to frame. Degrain and denoise filters can be applied to eliminate graininess in a video.

Deinterlacing Content

When content is produced for broadcast, interlacing is used to improve motion fidelity and reduce required bandwidth to push through the highest possible picture quality to the display device. This is a standard technique in broadcast; in web video it is not — deinterlaced or progressive video is used for web delivery. Computer monitors are not designed to display interlaced content, so interlacing is unnecessary and can create artifacts, thus it is not advisable to have content remain interlaced for web delivery. If your source content is interlaced, it is advised that a method of motion compensated or motion adaptive deinterlacing be applied prior to any other operation.

Progressively segmented Frames (PsF) are full frames of video. The PsF format was developed to transmit progressive images through an interlace environment using legacy hardware connectivity. A PsF signal could often be mistakenly treated as interlaced source. The essence of PsF content is exactly the same as progressive content. So an important part in PsF processing, therefore, it is that it should not be treated the same as interlaced source. Any professional ingest systems should detect the existence of the PsF format and output the frames as progressive video, not as interlaced. If you are confident that the incoming signal is PsF and the ingest system is treating it as interlaced, then check the input profile settings to make sure that the output is not interlaced (i.e. 30 PsF signal in -> 30 FPS progressive file out).

Inverse Telecine
When it is known that content has been shot at 24 or (23.976) progressive frames per second, but it was later telecined for broadcast to 29.97 interlaced fields per second using 3:2 or similar pulldown process, it is strongly recommended that the content be reversed back from 29.97 interlaced fields per second to its original state of 24 (or 23.976) progressive frames per second for motion fidelity and higher quality compression. This process of inverse telecine is always performed as part of deinterlacing process. The most critical part of the process is determining that telecine was applied, using 3:2 pulldown automatic detection. If such a detector mistakenly detected video mode as film mode or vice versa, then the resulting output will demonstrate catastrophic jerkiness, jaggies, etc.

Frame Rate Conversion
In some cases, source is shot at one frame rate, and then converted to higher frame rate. For instance, the content was shot on film at 24 progressive frames per second (FPS), but at some point the decision was made to convert it up to 60 frames per second. Technically this should result in smoother motion; but if poorly converted, the video will stutter and not achieve the intended playback motion at 60 FPS. If high quality, motion-compensated and motion adaptive frame rate conversion was done, then the video should not perceptually differ from originally shot content at 24 FPS. However, some negligible differences should be expected.

Understanding this, it is important to also have the hardware capacity to play back video at its current frame rate to assess whether it actually behaves as expected, with proper motion. If any hiccups are noted, play back the file at the original slower frame rate first, and inspect your hardware playback performance. Once it is confirmed that hardware capacity is adequate for smooth playback, the content can then be transcoded to the faster frame rate.

Video Frame Size and Picture Scaling
When scaling down frame size, it is recommended that the original aspect ratio be maintained, and that the resulting frame dimension be an even divisor of 16. Add black bars if necessary to compensate for difference between frame size and player display dimensions.

Selecting the right target frame size for video is a key component of the player and encoding process. When selecting a frame size try to consider the following best practices:

- Choose width and height dimensions that are divisible by 16. If the dimensions are not divisible by 16, encoders will have different solutions to achieve 16x16 macroblock division. Highly advanced H.264 core codecs add padding to reach the nearest 16-divisible width and height. Less advanced H.264 codecs will clip, resize, or splice the picture in order to adhere to 16x16 macroblock divisibility. Although, at high resolutions (HD and above) such divisibility may not be as critical, as close-to-16-divisibility is padded to the nearest 16x16 divisible block, thus increasing pixel and bitrate count. However, at lower resolutions, such divisibility is critical because every pixel and bit describing it needs to be accounted for. It is generally recommended to avoid resolutions that are not evenly divisible by 16. If the target size does not conform to 16-divisibility, it is suggested to add black bars (horizontal) or pillar bars (vertical), also known as blanking space, to conform.
- Maintain the original aspect ratio.
- If switching the display between frame sizes to accommodate different target audiences (i.e. enabling full screen playback, adjusting video size to enable smoother playback for those with poor network connections), avoid scaling the video down, instead scale up.

For example, if you decide that encoding one video size (e.g. 1280x720) and then use this same file for smaller display sizes (e.g. 256x144 or 768x432) you would do your viewer a disservice by engaging unnecessary bandwidth and system resources while causing deterioration of video quality.

When using video with a larger frame size than the intended frame size, several performance issues can occur:

- The player will engage more CPU resources to maintain scaling of the content into the designated frame size
- The player will maintain the higher bitrate of the larger frame size while trying to scale down, thus unnecessarily burdening the client CPU and bandwidth.
- Even though scaling down may seem like a reasonable approach to accommodating different clients while using the single highest frame size for all lower screen sizes, the general client performance impact can be devastating, taking up as much as 40% of processing resources that could otherwise be used for some other task.
- As scaling down takes place the picture quality will suffer as well. The image will be softer, jaggier edges will occur, and finer details will be less visible.

Instead of using a single video file to service different target audiences, encode for separate audiences and conditions. Whenever possible, work from original source and not from subsequently encoded video. If at the time of content creation there is an understanding what audience is being targeted, it is recommended that designated content for that size be used. For example, if video is shot specifically for high-definition broadcast, that same content simply encoded at a smaller dimension may lose detail, resulting in a suboptimal viewing experience. However, if the same content has been specifically shot for web delivery, then use of that content is advisable. When video is shot at 1920x1080, for instance, and it was shot progressively and then processed for interlaced broadcast, it is advisable that such source not be used if there is a possibility to create content at a smaller dimension. If no other source is available, then retaining frame size proportions is advised, while producing pre-scaled versions of playable files at the time of encoding or transcoding.

One of the simplistic techniques that is often used for producing downscaled (reduced) quality video from an interlaced source is to use frame sizes that are exactly half of the original size. It should be noted that due to the lack of filtering not only vertical sharpness is lost but also 100% aliasing may be introduced. For example, 1920x1080 interlaced source, with halving technique, produces a 960x540 progressive source.

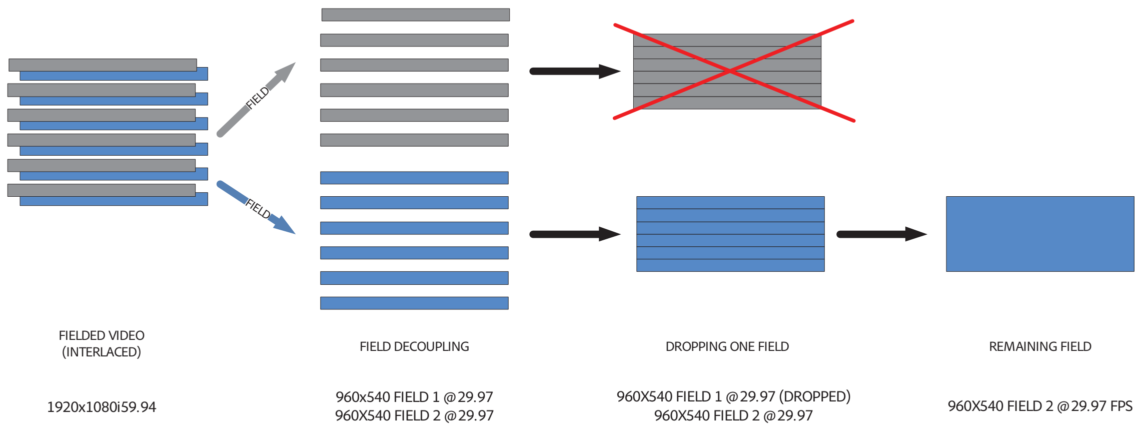


Figure 35

Field drop technique process diagram.

This technique is used because interlaced content is based on having two distinct moments in time captured in one half frame in the form of a field, where each field is exact half of the picture. This interlacing technique was established in the broadcast industry and by various standard associations as a compression mechanism to retain high quality picture data while passing through existing legacy equipment. The technique of halving the interlaced picture through decoupling and dropping of one of the fields is not always optimal in all cases. It is a selective process of trial and error. If there is a lot of motion and panning, this technique may produce a stuttering effect, where each picture in sequence is not an actual picture in sequence, but rather one picture less because of the dropped field. However, if there are slower moving pans or not as much motion, the video may look fine. This technique is mentioned is because it is quick and not as resource intensive for transcoding, but it has to be chosen with caution and weighed against other techniques, such as motion adaptive and motion compensated deinterlacing.

If you have to scale frame size and your source is interlaced, for best results deinterlace video first then apply scaling, as illustrated in Figure 36. If video is scaled first and then deinterlaced, the interlacing consequently will lose fidelity and introduce blocky artifacts or blurry images once actual encoding takes place.

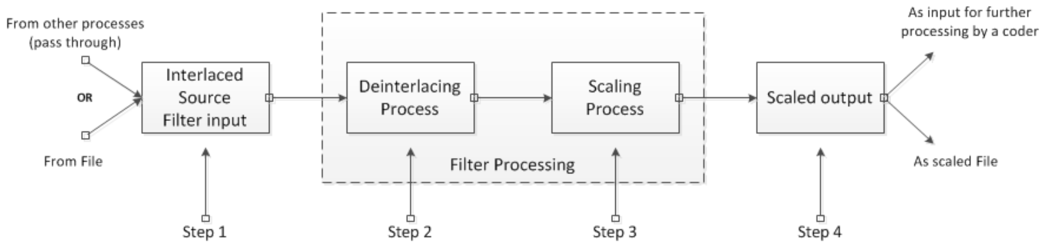


Figure 36

Depiction of the process of scaling interlaced content. Step 1 shows ingestion of interlaced source, whether it is an input from another process or an independent process via file. Step 2 is a preceding deinterlacing process to Step 3. Step 3 is a scaling process. Step 4 is a final step in this scaling procedure. The final step is then used either as an input to any other subsequent processes or as a final file.

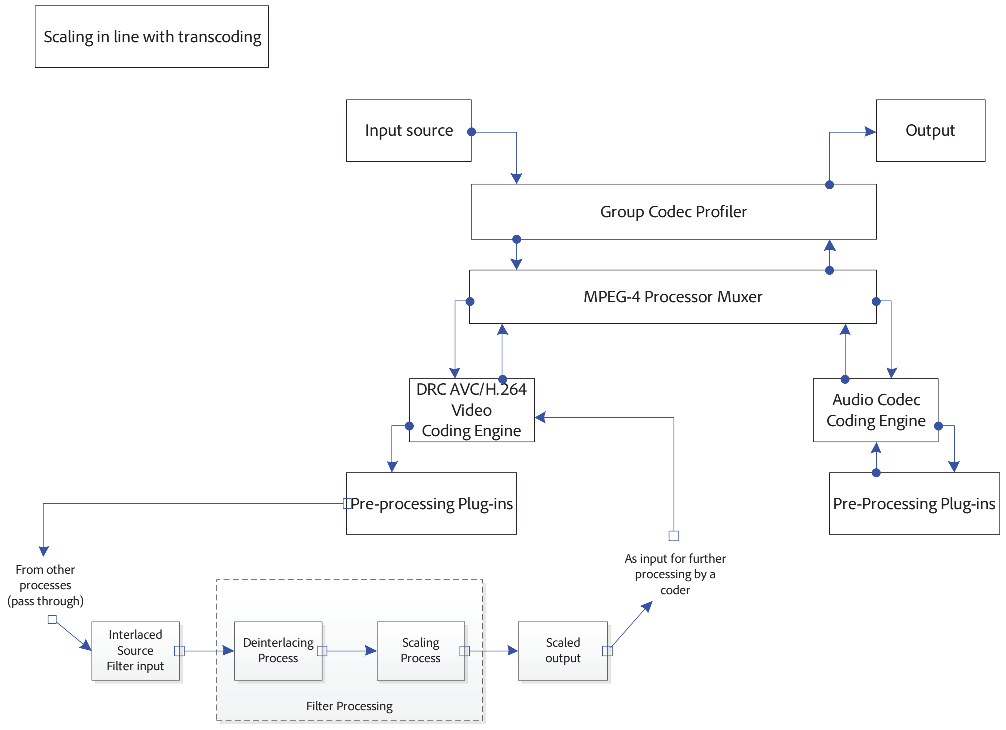

Scaling in line with transcoding
Input source
Output
Group Codec Profiler
MPEG-4 Processor Muxer
DRC AVC/H.264 Video Coding Engine
Audio Codec Coding Engine
Pre-processing Plug-ins
Pre-Processing Plug-ins
From other processes (pass through)
As input for further processing by a coder
Interlaced Source Filter input
Deinterlacing Process
Scaling Process
Scaled output
Filter Processing

Figure 37

An example of in-line scaling with transcoding (as implemented in Digital Rapids StreamZ software) .

Charts on the following pages list commonly used aspect ratios with both width (pixels) and height (lines) divisible by 16, by 8, and by 4.

1.33:1 (known as 4:3)

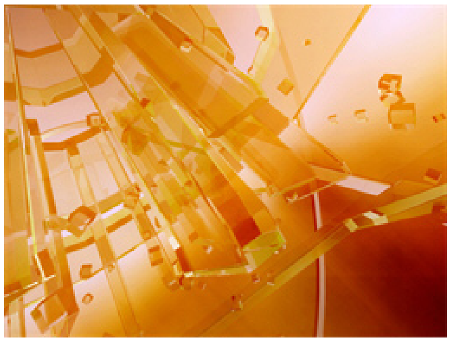


Figure 38

Picture above depicts video frame adhering to 1.33:1 (4:3) picture aspect ratio, where picture aspect ratio does not change from source state to encode state to display state. (source, Artbeats)

The following is the table of various frame sizes that are adhering to 4:3 (1.333) picture aspect ratio that is divisible by 16, 8, and 4.

Divisible by 16 (width x height)	Divisible by 8 (width x height)		Divisible by 4 (width x height)		
64 x 48	32 x 24	832 x 624	112 x 84	624 x 468	1184 x 888
128 x 96	64 x 48	864 x 648	128 x 96	640 x 480	1104 x 828
192 x 144	96 x 72	896 x 672	144 x 108	656 x 492	1120 x 840
256 x 192	128 x 96	928 x 696	160 x 120	672 x 504	1136 x 852
320 x 240	160 x 120	960 x 720	176 x 132	688 x 516	1200 x 900
384 x 288	192 x 144	992 x 744	192 x 144	704 x 528	1216 x 912
448 x 336	224 x 168	1024 x 768	208 x 156	720 x 540	1232 x 924
512 x 384	256 x 192	1056 x 792	224 x 168	736 x 552	1248 x 936
576 x 432	288 x 216	1088 x 816	240 x 180	752 x 564	1264 x 948
640 x 480	320 x 240	1120 x 840	256 x 192	768 x 576	1280 x 960
704 x 528	352 x 264	1152 x 864	272 x 204	784 x 588	1296 x 972
768 x 576	384 x 288	1184 x 888	288 x 216	800 x 600	1312 x 984
832 x 624	416 x 312	1216 x 912	304 x 228	816 x 612	1328 x 996
896 x 672	448 x 336	1248 x 936	320 x 240	832 x 624	1344 x 1008
960 x 720	480 x 360	1280 x 960	336 x 252	848 x 636	1360 x 1020
1024 x 768	512 x 384	1312 x 984	352 x 264	864 x 648	1376 x 1032
1088 x 816	544 x 408	1344 x 1008	368 x 276	880 x 660	1392 x 1044
1152 x 864	576 x 432	1376 x 1032	384 x 288	896 x 672	1408 x 1056
1216 x 912	608 x 456	1408 x 1056	400 x 300	912 x 684	1424 x 1068
1280 x 960	640 x 480	1440 x 1080	416 x 312	928 x 696	1440 x 1080
1344 x 1008	672 x 504	1472 x 1104	432 x 324	944 x 708	1456 x 1092
1408 x 1056	704 x 528	1504 x 1128	448 x 336	960 x 720	1472 x 1104
1472 x 1104	736 x 552	1536 x 1152	464 x 348	976 x 732	1488 x 1116
1536 x 1152	768 x 576	1568 x 1176	480 x 360	992 x 744	1504 x 1128
1600 x 1200	800 x 600	1600 x 1200	496 x 372	1008 x 756	1520 x 1140
			512 x 384	1024 x 768	1536 x 1152
			528 x 396	1040 x 780	1552 x 1164
			544 x 408	1056 x 792	1568 x 1176
			560 x 420	1072 x 804	1584 x 1188
			576 x 432	1088 x 816	1600 x 1200
			592 x 444	1152 x 864	
			608 x 456	1168 x 876	

Best performance	🟩
Good performance	🟨
Worst performance	🟥

1.78:1 (known as 16:9)

Figure 39

Picture above depicts video frame adhering to 1.78:1 (16:9) picture aspect ratio, where picture aspect ratio does not change from source state to encode state to display state. (source Artbeats)

The following is the table of various frame sizes that are adhering to 1.78:1 picture aspect ratio that is divisible by 16, 8, and 4.

Divisible by 16 (width x height)	Divisible by 8 (width x height)	Divisible by 4 (width x height)	
256 x 144	128 x 72	64x36	1152x648
512 x 288	256 x 144	128x72	1216x684
768 x 432	384 x 216	192x108	1280x720
1024 x 576	512 x 288	256x144	1344x756
1280 x 720	640 x 360	320x180	1408x792
1536 x 864	768 x 432	384x216	1472x828
1792 x 1008	896 x 504	448x252	1536x864
2048 x 1152	1024 x 576	512x288	1600x900
	1152 x 648	576x324	1664x936
	1280 x 720	640x360	1728x972
	1408 x 792	704x396	1792x1008
	1536 x 864	768x432	1856x1044
	1664 x 936	832x468	1920x1080
	1792 x 1008	896x504	1984x1116
	1920 x 1080	960x540	2048x1152
	2048 x 1152	1024x576	2112x1188
		1088x612	

Best performance	🟩
Good performance	🟨
Worst performance	🟥

1.85:1

Figure 40

Picture above depicts video frame adhering to 1.85:1 picture aspect ratio, where picture aspect ratio does not change from source state to encode state to display state. (source Artbeats)

The following is the table of various frame sizes that are adhering to 1.85:1 picture aspect ratio that is divisible by 16, 8, and 4.

Best performance 🟩
Good performance 🟨
Worst performance 🟥

Divisible by 16 (width x height)	Divisible by 8 (width x height)	Divisible by 4 (width x height)	
740 x 400	370 x 200	185 x 100	1295 x 700
1480 x 800	740 x 400	370 x 200	1480 x 800
2220 x 1200	1110 x 600	555 x 300	1665 x 900
	1480 x 800	740 x 400	1850 x 1000
	1850 x 1000	925 x 500	2035 x 1100
	2220 x 1200	1110 x 600	2220 x 1200

2.0:1 (common to Red One, formerly used as SuperScope)

Figure 41

Picture above depicts video frame adhering to 2.0:1 picture aspect ratio, where picture aspect ratio does not change from source state to encode state to display state. (source Artbeats)

The following is the table of various frame sizes that are adhering to 2.0:1 picture aspect ratio that is divisible by 16, 8, and 4.

Best performance 🟩
Good performance 🟨
Worst performance 🟥

Divisible by 16 (width x height)	Divisible by 8 (width x height)	Divisible by 4 (width x height)	
800 x 400	400 x 200	200 x 100	1400 x 700
1440 x 720	800 x 400	400 x 200	1600 x 800
1600 x 800	1200 x 600	600 x 300	1800 x 900
2400 x 1200	1600 x 800	800 x 400	2000 x 1000
	2000 x 1000	1000 x 500	2200 x 1100
	2400 x 1200	1200 x 600	2400 x 1200

2.35:1 (also known as 2.39:1)

Figure 42

Picture above depicts video frame adhering to 2.35:1 picture aspect ratio, where picture aspect ratio does not change from source state to encode state to display state. (source Artbeats)

The following is the table of various frame sizes that are adhering to 2.35:1 picture aspect ratio that is divisible by 16, 8, and 4.

Divisible by 16 (width x height)	Divisible by 8 (width x height)	Divisible by 4 (width x height)
956 x 400	478 x 200	239 x 100
1912 x 800	956 x 400	478 x 200
2868 x 1200	1434 x 600	717 x 300
	1912 x 800	956 x 400
	2390 x 1000	1195 x 500
	2868 x 1200	1434 x 600
		1673 x 700
		1912 x 800
		2151 x 900
		2390 x 1000
		2629 x 1100
		2868 x 1200

Best performance	🟩
Good performance	🟨
Worst performance	🟥

Combined table of aspect ratio (i.e. stretch coefficient) and corresponding size dimensions of width and height divisible by 16

Height	Width				
	1.33 (4x3)	1.78 (16x9)	1.85	2.0:1	2.39:1
	1.3333	1.7777	1.85	2.0	2.35 (2.39)
96	128				
144	192	256			
192	256				
240	320				
288	384	512			
336	448				
384	512				
400			740	800	956
432	576	768			
480	640				
528	704				
576	768	1024			
624	832				
672	896				
720	960	1280		1440	
768	1024				
800			1480	1600	1912
816	1088				
864	1152	1536			
912	1216				
960	1280				
1008	1344	1792			
1056	1408				
1104	1472				
1152	1536	2048			
1200	1600		2220	2400	2868

Note: 720x576 & 720x480 (not square pixels). Pixel dimensions shown in the table are square pixel dimensions, except for 720x576 (PAL), 720x480(NTSC), and anamorphic sizes.

Format Conversion to 16:9 Aspect Ratio

The ongoing trend in picture formats is to use a wide aspect ratio. One of the common selections for online delivery is 16:9. The following guides were designed to help conform various common picture aspect ratios to 16:9 aspect ratio. The table that follows allows for black bar allocation for proper format retention and corresponding pixel count for black bar padding and active video frame size, should the coding processing software require manual entry.

The process of adding black bars and conforming the source media to 16:9 video frame sizes can be done either in the coding process end or on the player end. However, it may not be an efficient use of resources to apply the padding on the player side. Depending on the process and the quality of scaler involved in the transcoding process, adding black bars on the encoding end may produce better results.

Whichever process you choose, the following guides will reduce your time and effort in deciding on sizes and their corresponding parameters for conforming media to 16:9 picture aspect ratio.

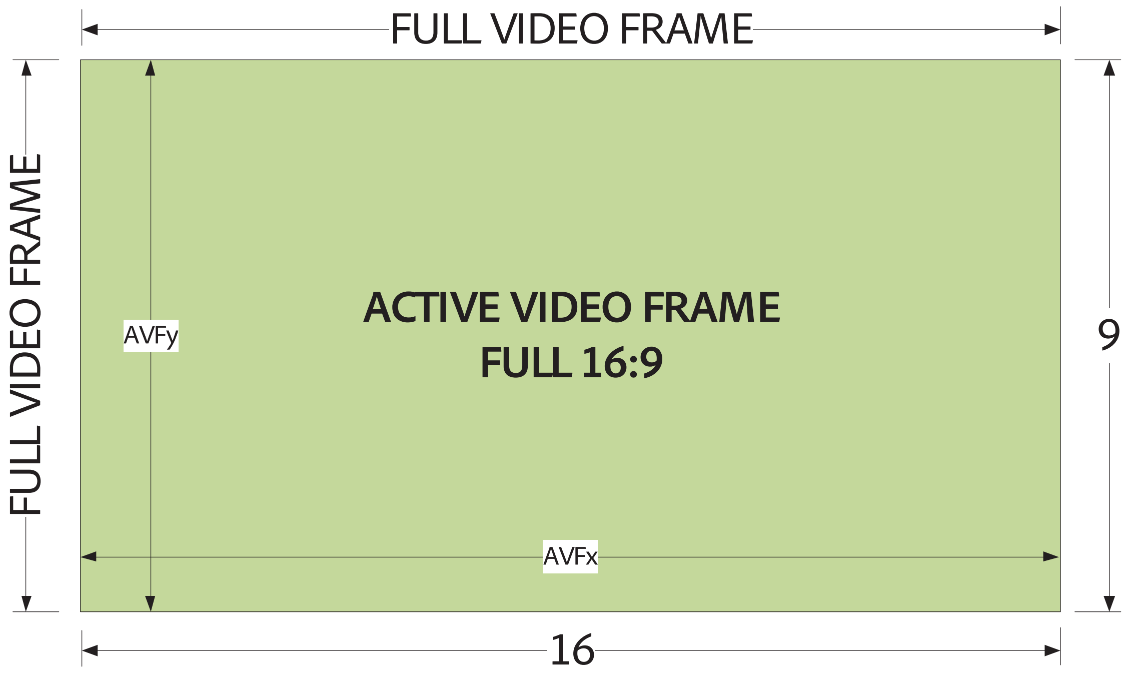


Figure 43

Unconverted full active video frame, 16:9 aspect ratio full frame video.

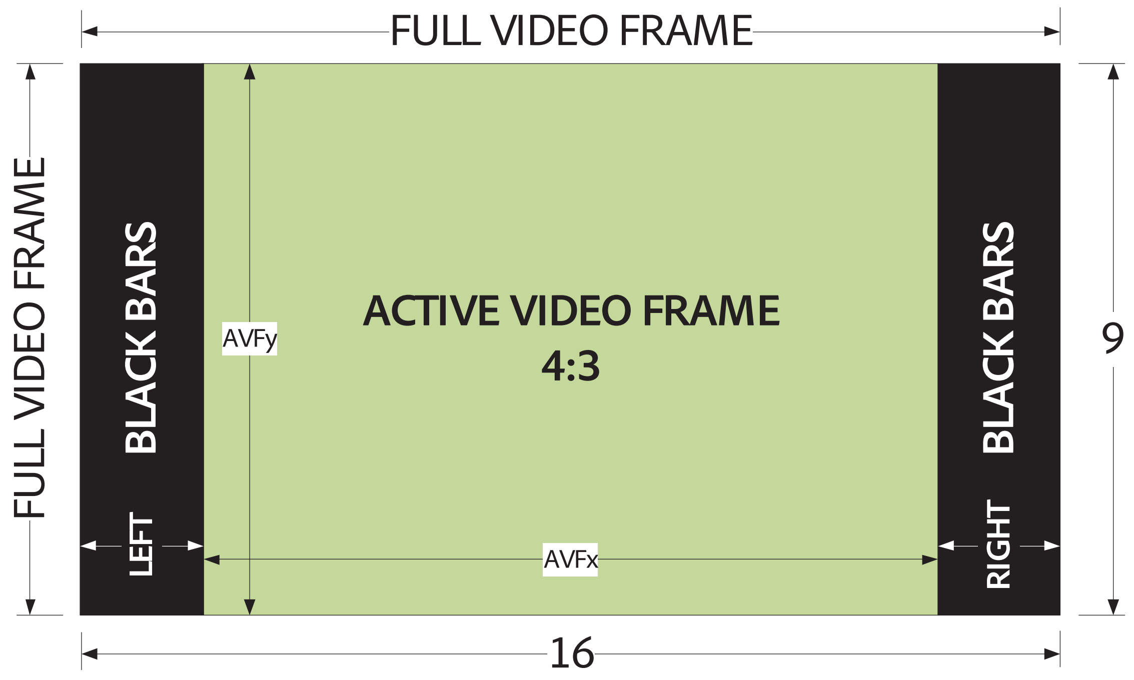


Figure 44
4:3 aspect ratio active video frame conversion to 16:9 aspect ratio full frame video.

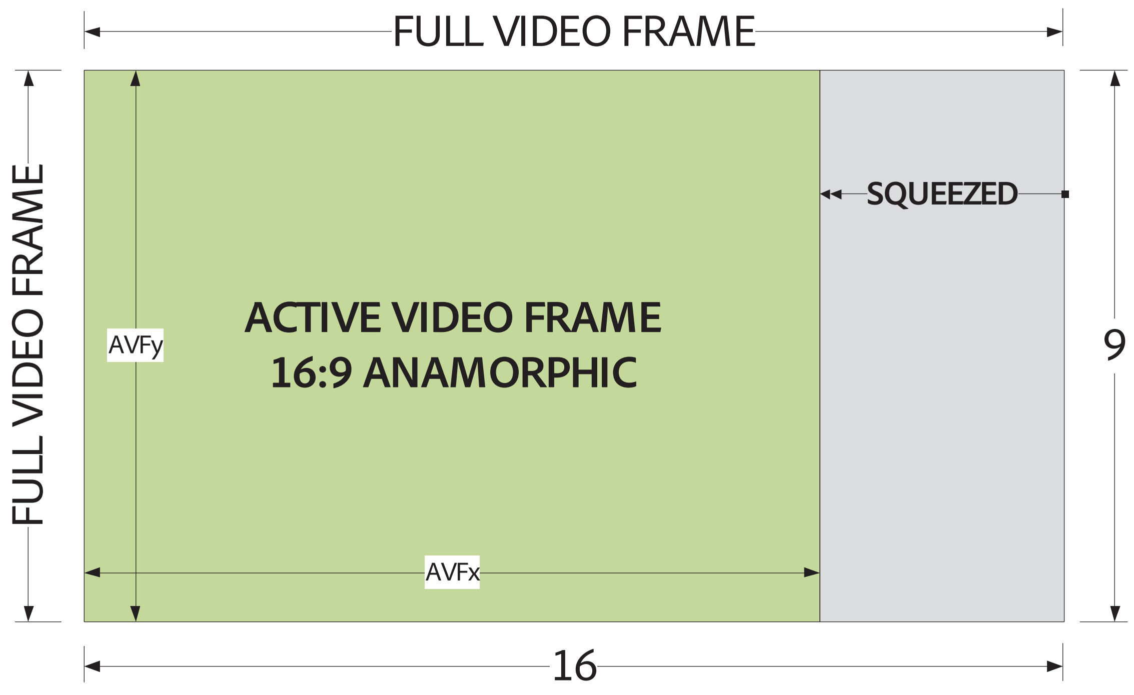


Figure 45
Anamorphed 16:9 aspect ratio active video frame conversion to 16:9 aspect ratio full frame video.

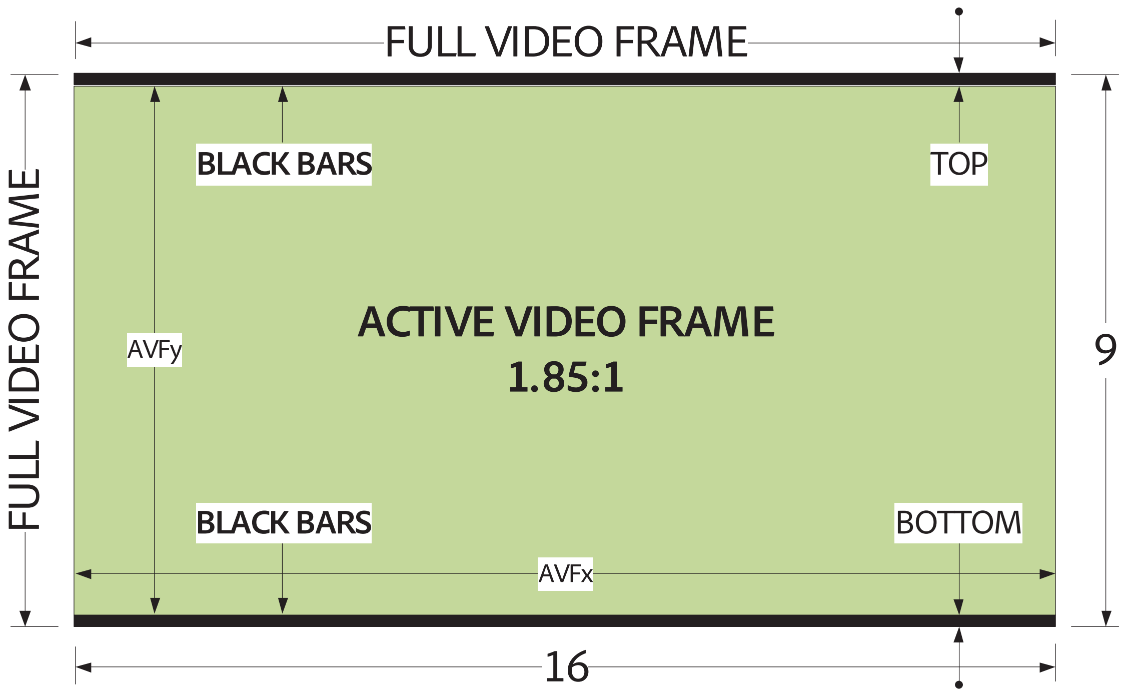


Figure 46

1.85:1 aspect ratio active video frame conversion to 16:9 aspect ratio full frame video.

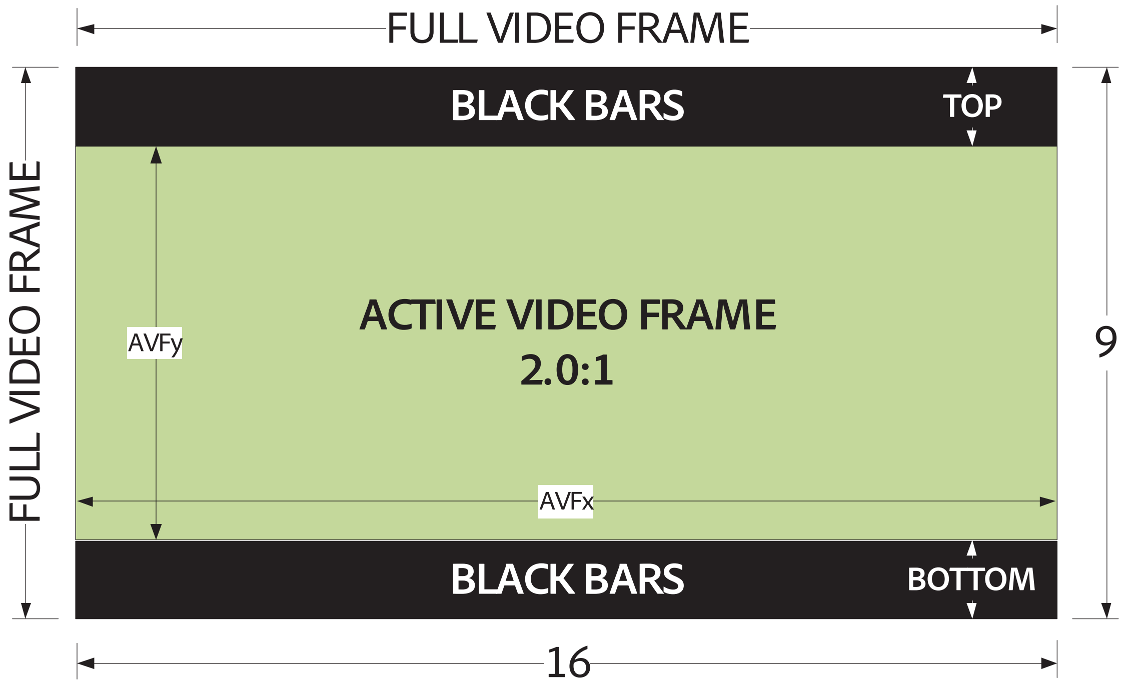


Figure 47

2.0:1 aspect ratio active video frame conversion to 16:9 aspect ratio full frame video.

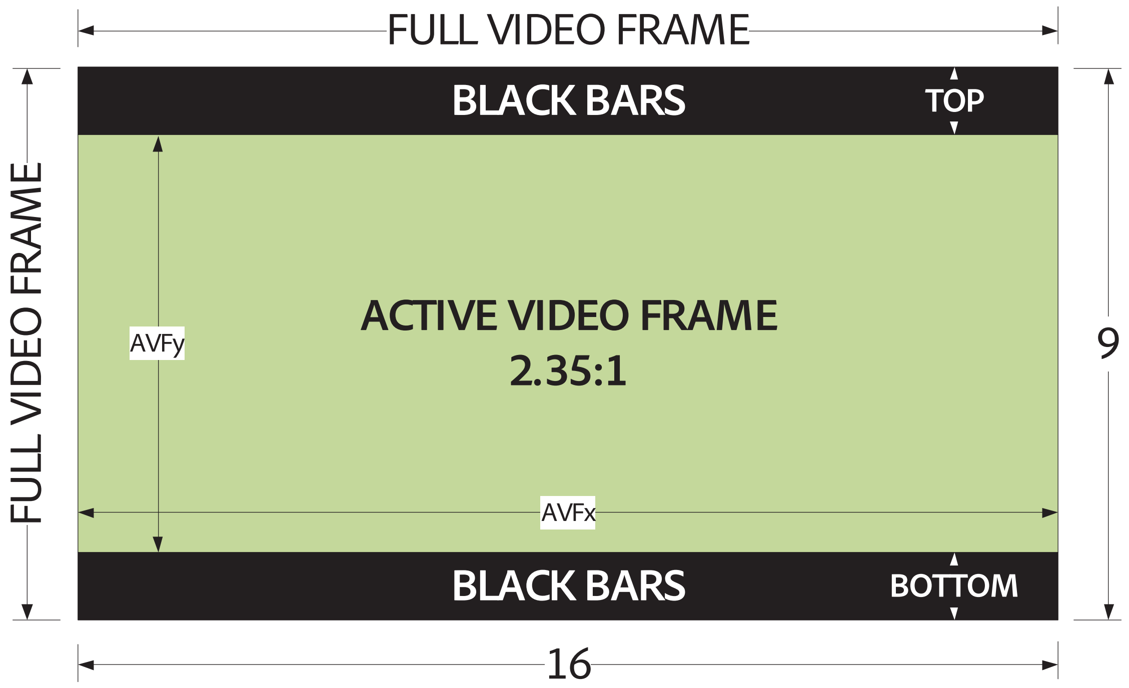


Figure 48

2.35:1 aspect ratio active video frame conversion to 16:9 aspect ratio full frame video.

Aspect Ratio Conversion									
Active Video Frame			Black Bars				→	16:9 Full Video Frame	
Aspect Ratio	AVFx	AVFy	Right	Left	Top	Bottom		Width	Height
1920x1080 (1088) (FullHD)									
1.333:1 (4:3)	1440 px	1080 px	224 px	224 px	0	0	→	1920 px	1080 px
1.333:1 (16:9 AN)	1440 px	1080 px	0	0	0	0	→	1920 px	1080 px
1.7777:1 (16:9)	1920 px	1080 px	0	0	0	0	→	1920 px	1080 px
1.85:1	1920 px	1038 px	0	0	21 px	21 px	→	1920 px	1080 px
2.0:1	1920 px	960 px	0	0	60 px	60 px	→	1920 px	1080 px
2.35:1	1920 px	818 px	0	0	131 px	131 px	→	1920 px	1080 px
1792x1008									
1.333:1 (4:3)	1344 px	1008 px	224 px	224 px	0	0	→	1792 px	1008 px
1.333:1 (16:9 AN)	1344 px	1008 px	0	0	0	0	→	1792 px	1008 px
1.7777:1 (16:9)	1792 px	1008 px	0	0	0	0	→	1792 px	1008 px
1.85:1	1792 px	968 px	0	0	20 px	20 px	→	1792 px	1008 px
2.0:1	1792 px	896 px	0	0	56 px	56 px	→	1792 px	1008 px
2.35:1	1792 px	762 px	0	0	123 px	123 px	→	1792 px	1008 px
1536x864									
1.333:1 (4:3)	1152 px	864 px	192 px	192 px	0	0	→	1536 px	864 px
1.333:1 (16:9 AN)	1152 px	864 px	0	0	0	0	→	1536 px	864 px
1.7777:1 (16:9)	1536 px	864 px	0	0	0	0	→	1536 px	864 px
1.85:1	1536 px	830 px	0	0	17 px	17 px	→	1536 px	864 px
2.0:1	1536 px	768 px	0	0	48 px	48 px	→	1536 px	864 px
2.35:1	1536px	654 px	0	0	105 px	105 px	→	1536 px	864 px
1280x720 (720p)									
1.333:1 (4:3)	960 px	720 px	160 px	160 px	0	0	→	1280 px	720 px
1.333:1 (16:9 AN)	960 px	720 px	0	0	0	0	→	1280 px	720 px
1.7777:1 (16:9)	1280 px	720 px	0	0	0	0	→	1280 px	720 px
1.85:1	1280 px	692 px	0	0	14 px	14 px	→	1280 px	720 px
2.0:1	1280 px	640 px	0	0	40 px	40 px	→	1280 px	720 px
2.35:1	1280 px	544 px	0	0	88 px	88 px	→	1280 px	720 px
1024x576									
1.333:1 (4:3)	768 px	576 px	128 px	128 px	0	0	→	1024 px	576 px
1.333:1 (16:9 AN)	768 px	576 px	0	0	0	0	→	1024 px	576 px
1.7777:1 (16:9)	1024 px	576 px	0	0	0	0	→	1024 px	576 px
1.85:1	1024 px	554 px	0	0	22 px	22 px	→	1024 px	576 px
2.0:1	1024 px	512 px	0	0	32 px	32 px	→	1024 px	576 px
2.35:1	1024 px	436 px	0	0	70 px	70 px	→	1024 px	576 px

Legend	
🟦 Pillarbox (Partial Width, Full Height)	🟨 Full Width, Full Height
⬜ Letterbox (Full Width, Partial Height)	🟧 Anamorphic Width, Full Height

Aspect Ratio Conversion									
Active Video Frame			Black Bars				→	16:9 Full Video Frame	
Aspect Ratio	AVFx	AVFy	Right	Left	Top	Bottom		Width	Height
768x432									
1.333:1 (4:3)	576 px	432 px	96 px	96 px	0	0	→	768 px	432 px
1.333:1 (16:9 AN)	576 px	432 px	0	0	0	0	→	768 px	432 px
1.7777:1 (16:9)	768 px	432 px	0	0	0	0	→	768 px	432 px
1.85:1	768 px	416 px	0	0	8 px	8 px	→	768 px	432 px
2.0:1	768 px	384 px	0	0	24 px	24 px	→	768 px	432 px
2.35:1	768 px	326 px	0	0	63 px	63 px	→	768 px	432 px
512x288									
1.333:1 (4:3)	384 px	288 px	64 px	64 px	0	0	→	512 px	288 px
1.333:1 (16:9 AN)	384 px	288 px	0	0	0	0	→	512 px	288 px
1.7777:1 (16:9)	512 px	288 px	0	0	0	0	→	512 px	288 px
1.85:1	512 px	276 px	0	0	6 px	6 px	→	512 px	288 px
2.0:1	512 px	256 px	0	0	16 px	16 px	→	512 px	288 px
2.35:1	512 px	218 px	0	0	35 px	35 px	→	512 px	288 px
256x144									
1.333:1 (4:3)	192 px	144 px	32 px	32 px	0	0	→	256 px	144 px
1.333:1 (16:9 AN)	192 px	144 px	0	0	0	0	→	256 px	144 px
1.7777:1 (16:9)	256 px	144 px	0	0	0	0	→	256 px	144 px
1.85:1	256 px	138 px	0	0	3 px	3 px	→	256 px	144 px
2.0:1	256 px	128 px	0	0	8 px	8 px	→	256 px	144 px
2.35:1	256 px	108 px	0	0	18 px	18 px	→	256 px	144 px

Legend			
🟦	Pillarbox (Partial Width, Full Height)	🟨	Full Width, Full Height
⬜	Letterbox (Full Width, Partial Height)	🟧	Anamorphic Width, Full Height

Anamorphic Video Transformation

In an effort to optimize various resources to produce the best experience, one of the decisions that comes up often is to use anamorphic video resizing in order to gain additional bits per pixel. Frequently, this anamorphic sizing technique is used to reduce bandwidth requirements. Typically, the savings in such cases is as much as 25%. However, when making a decision whether or not to use anamorphic size, the considerations must be kept in mind:

- Maintain constant height (ex. 1920x1080 source → 1440x1080 encode → 1920x1080 display).
- Use highest quality scaler.
- Do not do anamorphing for interlaced content (progressive only).
- Applicable only to 16:9 content.
- Scaling needs to be exact encode and decode (same stretch coefficient as source).
- 16:9 anamorphing width works only with 4:3 sizes that have the same height (see item 1 above).
- Decoder performance could introduce higher processing consumption.
- Technical metadata needs to explicitly stamp that it is 16:9 content at encode time, even though the sizes will correspond to the exact 16:9 dimensions.

- The transformation does not come for free. The cost will be in picture quality. Re-transformed video, at player/display time, will be a slightly softer than the equivalent picture without anamorphic transformation.

The following anamorphic width transformation diagram and table was put together to assist in the decision making process.

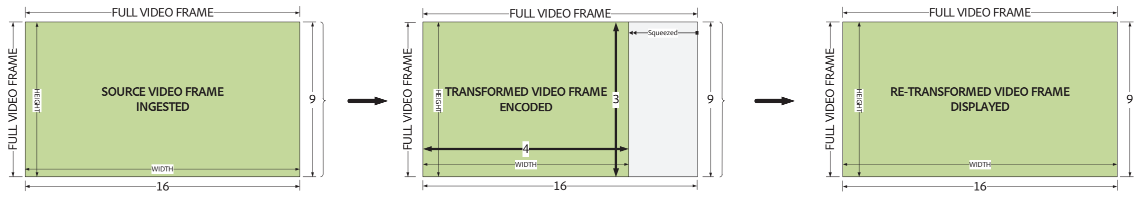

FULL VIDEO FRAME
SOURCE VIDEO FRAME INGESTED
HEIGHT
WIDTH
16
9
TRANSFORMED VIDEO FRAME ENCODED
Squeezed
3
4
RE-TRANSFORMED VIDEO FRAME DISPLAYED

Figure 49

Anamorphic width transformation diagram.

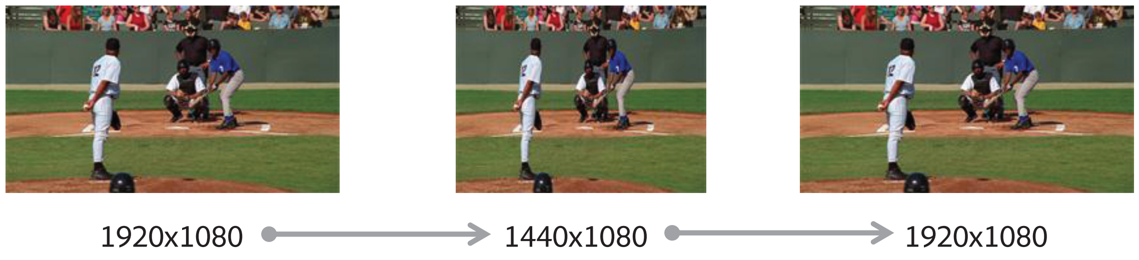


Figure 50

Anamorphic width transformation video snapshots.

Source Video Frame (Ingested)			Transformed (Anamorphed) Video Frame (Encoded)					Target Re-Transformed Video Frame (Displayed)			
Aspect Ratio Stretch Factor	Width	Height	Aspect Ratio	Width Squeeze (Factor)	Source Width (Delta)	Width	Height	Aspect Ratio	Aspect Ratio Stretch Factor	Width	Height
16:9 = 1.7777	1920	1080	4:3 = 1.333 (16:9 AN)	3:4 = 0.75	25%	1440	1080	16:9 = 1.7777	4:3 = 1.333	1920	1080
16:9 = 1.7777	1792	1008	4:3 = 1.333 (16:9 AN)	3:4 = 0.75	25%	1344	1008	16:9 = 1.7777	4:3 = 1.333	1792	1008
16:9 = 1.7777	1536	864	4:3 = 1.333 (16:9 AN)	3:4 = 0.75	25%	1152	864	16:9 = 1.7777	4:3 = 1.333	1536	864
16:9 = 1.7777	1280	720	4:3 = 1.333 (16:9 AN)	3:4 = 0.75	25%	960	720	16:9 = 1.7777	4:3 = 1.333	1280	720
16:9 = 1.7777	1024	576	4:3 = 1.333 (16:9 AN)	3:4 = 0.75	25%	768	576	16:9 = 1.7777	4:3 = 1.333	1024	576
16:9 = 1.7777	1024	432	4:3 = 1.333 (16:9 AN)	3:4 = 0.75	25%	576	432	16:9 = 1.7777	4:3 = 1.333	768	432
16:9 = 1.7777	512	288	4:3 = 1.333 (16:9 AN)	3:4 = 0.75	25%	384	288	16:9 = 1.7777	4:3 = 1.333	512	288
16:9 = 1.7777	256	144	4:3 = 1.333 (16:9 AN)	3:4 = 0.75	25%	192	144	16:9 = 1.7777	4:3 = 1.333	256	144

Figure 51

Anamorphic width conversion table.

Color Space and Levels Scheme Transformation

Consider an example, if your input is in the YUV color matrix domain and your output is in the YUV color matrix domain, the transcoder can achieve the output through conversion to the RGB color matrix domain (YUV in -> RGB at processing time -> YUV out) and then back to the YUV color matrix domain, while modifying the video levels scheme (16-235 -> 0–255 -> 16-235). By doing so, the video changes its color matrix and video levels scheme unnecessarily and, therefore, loses its integrity of color matrix and video scheme levels. If this video undergoes several of these transformational stages, the picture at the final stage will end up looking entirely different from the original.

The following are examples of such transformations in YUV and RGB domains:

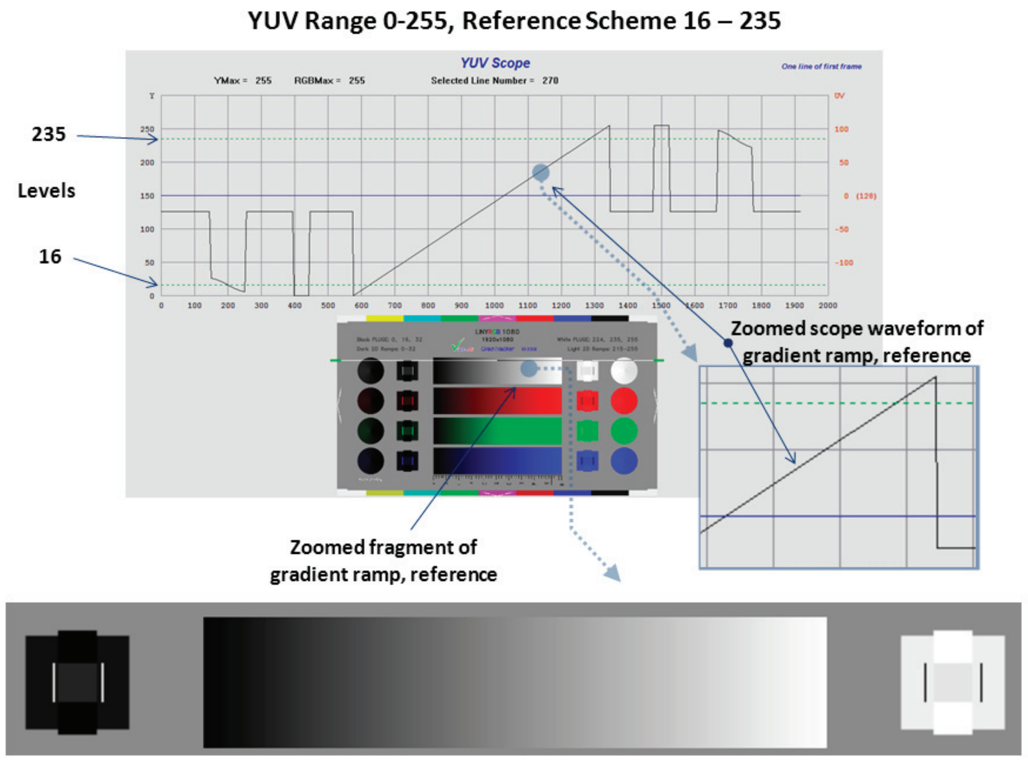


Figure 52

Depiction of linear YRGB reference test pattern: reference scheme of 16-235, ramp test range 0-255, inclusive sub range levels below black (16) and white (235). Notice clean gradient and no banding of color gradient ramp and clean waveform on the ramp.

When video originated in the video domain (i.e. broadcast television), and is maintained in the video domain of broadcast television, then the chance of it transforming to RGB 0-255 color scheme is minimal. Thus, the picture in Figure 52 is an indication of what the video on the YUV scope will look like, as far as the color level scheme is concerned.

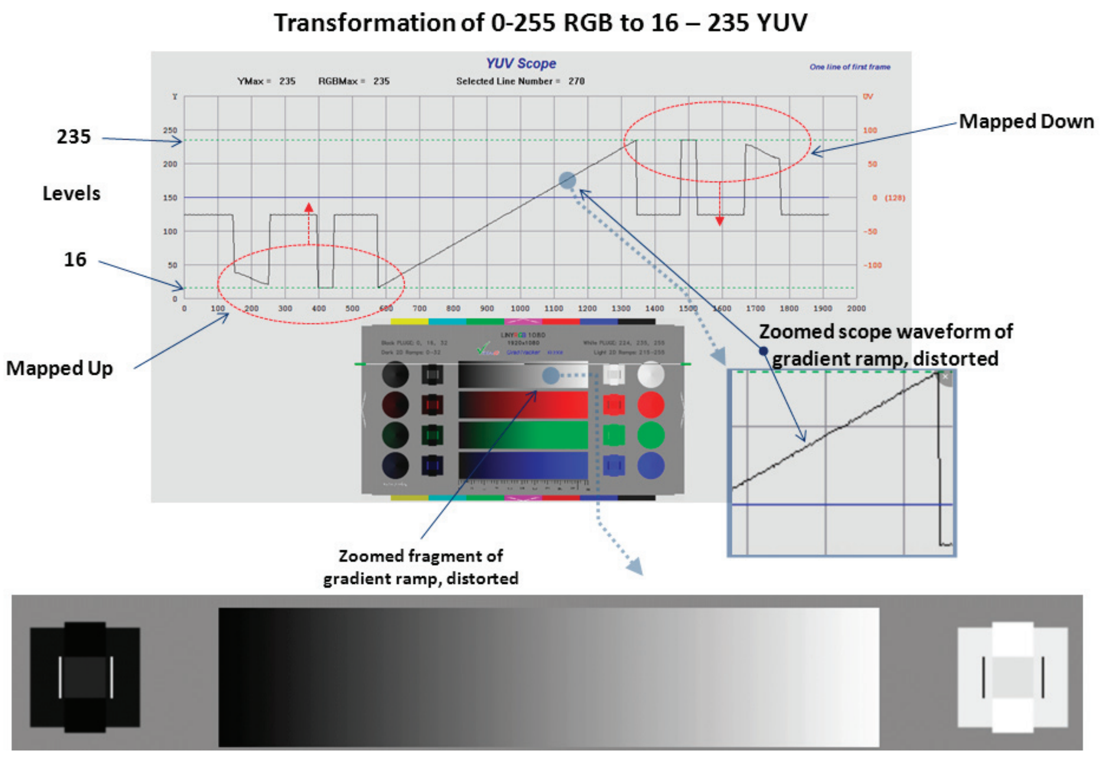


Figure 53

Depiction of Linear YRGB test pattern transformation from 0-255 RGB to 16-235 YUV, with color matrixing and range mapping. Notice banding of color gradient ramp and slightly distorted waveform on the ramp, no potential danger of clipping (i.e. safe mode, boost proof). Player will do final transformation of level mapping from 16 down to 0 and 235 up to 255.

Typically when computer graphics originated using the RGB 0-255 scheme and later converted to YUV 16-235, light distortions are introduced in addition to banding. The choice of LINYRGB synthetic pattern from VideoQ technical test collection was made for better demonstration of transformations that take place in such a process (Figure 53).

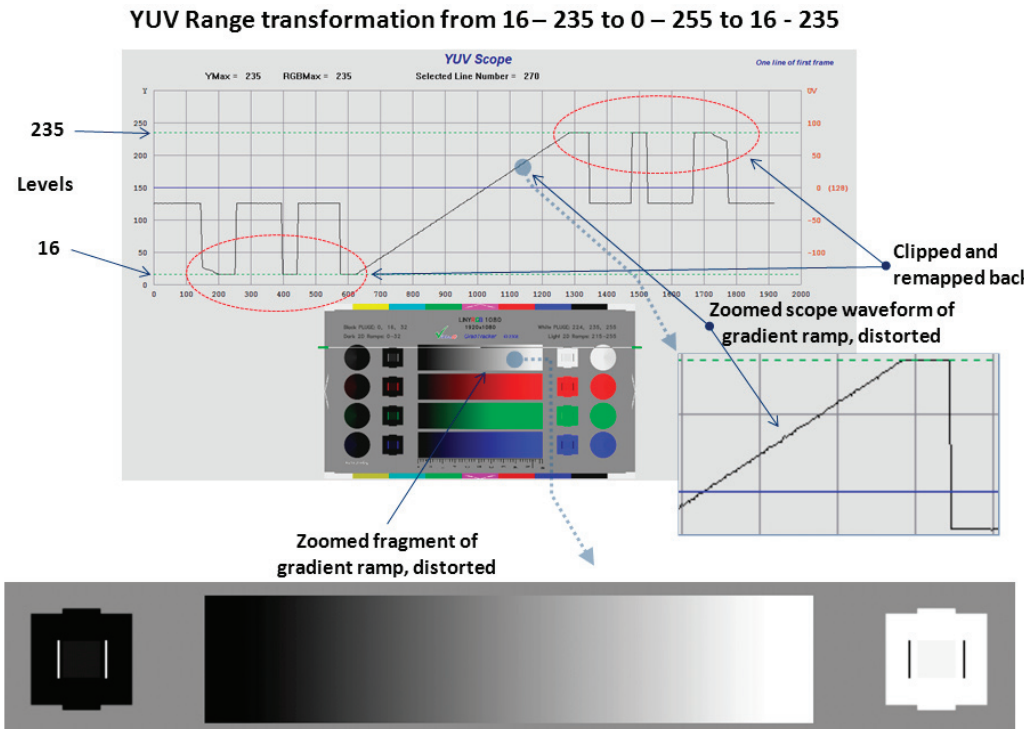

YUV Range transformation from 16 – 235 to 0 – 255 to 16 - 235
YUV Scope
YMax = 235 RGBMax = 235
Selected Line Number = 270
One line of first frame
235
Levels
16
Clipped and remapped back
Zoomed scope waveform of gradient ramp, distorted
Zoomed fragment of gradient ramp, distorted

Figure 54

Depiction of result of two cascaded level mapping processes. Notice distorted gradient and banding of color gradient ramp and distorted waveform on the ramp, also visible gray limits of white on the right of Luma gradient. Black square on the left is no longer showing perpendicular rectangular shape, clearly visible in reference Figure 52. White square on the right is no longer showing white perpendicular rectangular shape, clearly visible in reference Figure 52. Hence, the clipping of white and black occurs. If further transformations take place, entire color scheme will deteriorate significantly, resulting in irreversible reduction of picture quality.

For example, during the process of selecting a video source for transcoding for the web, a choice was made to select video that was intended for broadcast. Then during the transcoding transformation process the guidelines were not specified to maintain video in the YUV 16-235 level scheme and through some interim process video was transformed to the RGB 0-255 level scheme. Then transformed video was used as a source for an editing operation while being transformed back to the YUV 16-235 level scheme. In this transformational phase of video going from YUV to RGB to YUV, the picture loses its fidelity and any substantial details are washed out. Any further transformations between YUV and RGB will worsen the picture even further. The choice of LINYRGB synthetic pattern from VideoQ technical test collection was made for better demonstration of transformations that take place in such a process (Figure 54).

Considering that the transformation of YUV 16-235 to RGB 0-255 needs to happen one way or the other, it is important to understand that it should happen at the end of the chain and not earlier, because at the display side (RGB 0-255) transformation will happen anyway. By making certain that this transformation does not happen earlier than at the end, you are avoiding chance of further clipping and irreparable damage of color scheme transformations, earlier than necessary. Also, similar damage can be done by boosting gain. However, at the end of the day, anything outside of 16-235 will be lost or clipped, so don't place any valuable picture data outside of 16-235 scheme. Figures 55 and 56 show how such valuable picture data may get lost in the transformational process if excessive boost is applied.

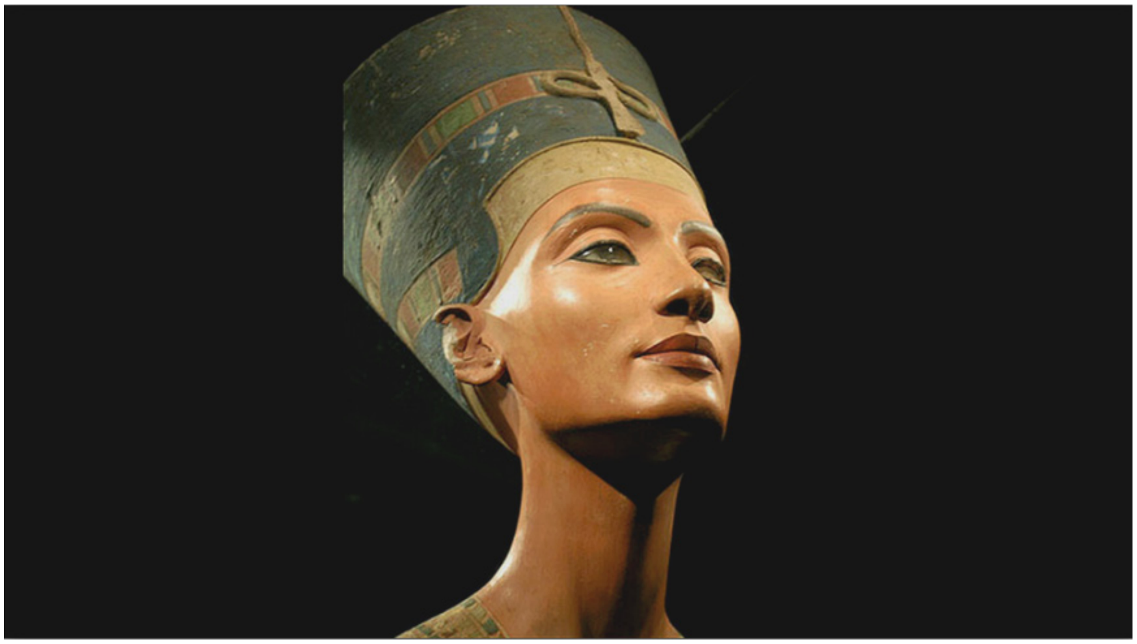


Figure 55

Original source, 0-255 scheme. Notice black and white levels, excellent picture details in darker and lighter areas. [Image of Nefertiti, source Wikipedia, under GNU, author unknown.]

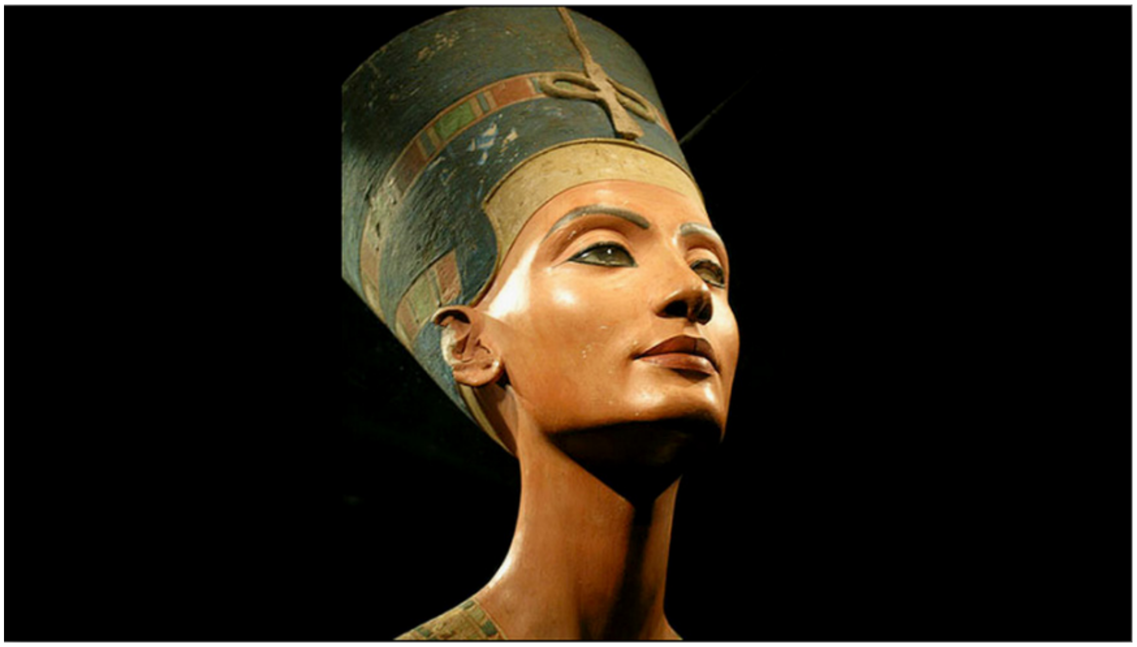


Figure 56

Example of wrongful assumption, erroneously expected 16-235 scheme and transformed to 0-255 scheme, notice clipping in black and white levels, and loss of picture detail in darker and whiter areas.

RGB scheme of 0–255 does not foresee levels below black and above white, which exist in YUV scheme (16-235). Because of this, transformation of 16-235 to 0-255 done once, simply clips levels with no major distortions introduced, except banding. However, if 0-255 gets interpreted erroneously as 16–235 then black and white crash happens (16 levels of close to black and 20 levels of close to white will be clipped). This issue is irrecoverable, unfortunately, even though one would try to reverse it by converting it to 16-235, this still will not help. It will make the picture worse and add banding. This can happen not in the situation where picture metadata was bad or wrong, but when it does not exist at all. Keep in mind that when you are viewing the picture on a PC monitor, there are a number of permutations of transformations that may be beyond the player's control, as shown in the following table.

Color Space & Level Scheme	Rendered as	Display Interface Type	Comments
RGB 0-255 4:4:4	RGB 0-255 4:4:4	VGA/DVI	Legacy, currently not in use practically, requires about 50% higher bitrate
YUV 16-235 4:2:2	RGB 0-255 4:4:4	VGA/DVI	Most widely used
YUV 16-235 4:2:2	RGB 0-255 4:4:4	HDMI/DisplayLink	Increasingly popular and gaining momentum and marketshare, replacing DVI
YUV 16-235 4:2:2	YUV 16-235 4:4:4 or 4:2:2	HDMI/DisplayLink	Transformed inside the display device into RGB 0-255 scheme

Even if your image was originally born as 0-255, keeping it as such may result in a chance of severe clipping because it may be erroneously interpreted as 16-235 scheme. Therefore it makes sense to transform it to the 16-235 scheme and maintain that scheme throughout the process, thus minimizing the chance of clipping during chain of transformation processes.

Audio

The quality of sound produced during transformational encoding processes plays a significant part not only in the finished part of the output, but also in the quality of the playback experience.

Audio Bitrate Allocation

The bitrate allocation for audio is a straightforward process — much less complex than video bitrate selection.

The elements that comprise of the final bitrate for audio are:

- Sampling rate
- Amount of channels
- Bit depth
- Encoding mode

Sampling Rate

Flash Player supports a 44.1 kHz sampling rate. Everything below 44.1 gets up-sampled at play time. So, down-sampling audio to anything below 44.1 kHz, such as 11 kHz or 22 kHz is not advisable, unless you have specific bandwidth considerations that cannot be covered by a high efficiency codec such as HE-AAC v2.

Channel Remapping

An audio track typically carries one or more channels of audio. Each channel of audio discretely identifies its designated speaker location (left, right, etc.) in the stereo mode, or unidentified channel designation in mono mode. Flash Player downmixes all multichannel audio tracks into two stereo channels or single mono channel lineup.

Bit Depth

For bit depth, if possible always use 16 bit. It has a wide range of values (65,536). The sampling rate and value range of bit depth has a direct relationship; the lower the sampling rate, the lower the bit depth, the lower the bitrate, thereby lowering audible quality. The higher the sampling rate, the higher the bit depth and the higher the bitrate, therefore producing higher audible quality.

Levels Normalization

During the audio production process, sound engineers typically set audio normalization levels below 0 dB to allow for flexibility within subsequent processing. Audio level normalization is a subjective element, with each organization setting its own standards. The levels could typically range from -6 dBfs to -18 dBfs. When processing audio, it is imperative to know what the normalization level is to avoid oversaturation and clipping.

Encoding Mode Selection
When selecting target bitrate, one of the main elements in that process is audio encoding mode. Deciding which encoding mode to concentrate on is based on target device capability and bandwidth availability. Presently, there are three audio coding modes of AAC types: AAC (AAC LC), HE AAC v1, and HE AAC v2. AAC (Advanced Audio Coding) or AAC LC (Low Complexity) was adapted early on and it is frequently used in low powered devices and offers the least amount of compression of the three. HE AAC v1 (High Efficiency) includes Spectral Band Replication (SBR) and it is able to encode at a lower data rates than AAC LC. HE AAC v2 includes SBR and PS (Parametric Stereo), and it compresses data rates even further than HE AAC v1.

Spectral Band Replication is a method for highly efficient coding of the high frequencies in audio compression algorithms that enables audio codecs to deliver the same quality at half the bitrate.

Parametric Stereo works by encoding a stereo signal into one monaural signal and transmitting information about the stereo image (parametric information). Parametric Stereo reduces the data rate needed for equivalent quality. It is only available with HE AAC v2 encoding mode.

The drawback of HE AAC v1 and HE AAC v2 may be resource usage in lower powered devices which have no hardware decoding support. However, in multibitrate environments, where the audio bitrate should be the same across all segments, the selection of HE AAC v2 will prevail.

CODEC	BITRATE							
AAC		48	56	64	80	96	112	128	160
HEAAC v1	40	48	56	64	80	96	112	128	
HEAAC v2	40	48	56	64	80	96	112	128	

Figure 57

Table of typically selected bitrates for AAC, HEAAC v1, and HEAAC v2 CODECs.

Important Note About Video/Audio Synchronization

Although video and audio can be multiplexed into a multiprogram stream within the MPEG-4 format, due to the need for legacy support, video in the Flash Player supports only a single audio and video stream within a file. Synchronization of audio and video is done through the audio reference clock. So, if your audio is experiencing issues such as subpar quality, noise or pauses or there is no audio track included, then video will either go out of sync or will drop frames. Each frame of audio and video have a timestamp associated with them. These timestamps are created during the encoding process and act as reference points during decoding and playback. When audio and video timestamps are misaligned or unable to synchronize, either the audio or the video will start to drift apart unless a reliable resynchronization or realignment mechanism is triggered to bring them back into alignment. If the realignment is unsuccessful, such as when audio is garbled or cannot be decoded, the video and audio will continue to drift apart and eventually the video will start to skip frames all together. An example of such behavior can be seen in live video encoding when the audio is turned off and on frequently for some reason, or the volume is too low for a quality decode to take place. Another example would be a case where the video has paused due to audio silence or drop-out. In this case, the viewer would observe a fast forward effect to the next logical stop or keyframe.

The desktop audio clock is the most stable, inexpensive, consistent, and reliable clock to reference time scale against. Most modern desktop systems have hardware audio decoders built-in, and therefore are able to decode audio reliably, which is why Adobe chose audio to be the reference clock carrier for video and audio. However, when audio cannot be reliably decoded by the built-in audio decoder, the video suffers as well. So, it is vital that the audio source and encoded output is good quality regardless of the intended bitrate.

Stream Packaging

Delivering Video

You can use two different methods to deliver video through Adobe Media Player: progressive download or streaming. Regardless of the method you choose, the experience for viewers is very similar. Your choice depends on your budget and the level of protection your content requires.

Progressive Download
Progressive download downloads and stores video on the viewer's computer. A short period of time is required to buffer and cache the beginning of the media file before it starts playing. Adobe Media Player calculates an appropriate buffer time based on the rate the data is being received and the total length of the video. Once the video has been downloaded, subsequent viewing does not require any buffering. Progressively downloaded files are usually delivered through a content delivery network (CDN) using the standard HTTP protocol. In this case, Adobe Media Player establishes a direct connection with the CDN's servers to retrieve the content.

Streaming
Streaming video delivers the video bits as they are requested, in real time. The bits are viewed and then discarded. The user experience is virtually the same as with downloaded content but has a few key differences. Viewers do not have to wait for video to download before seeking throughout the video. Video is not cached on the viewer's computer, so it cannot be viewed offline. Adobe Media Player can deliver streaming video via the RTMP or RTMPE protocols supported by Adobe Flash Media Server.

The stream packaging (i.e. making of multiplexed file) procedure typically consists of many elements that enable control of the distribution delivery process. Depending on the type of delivery method selected, progressive download or streaming, MOOV Atom location will move either to the end or to the beginning of the file. When progressive delivery is used, the MOOV Atom location should be at the beginning of the file. Typically this is enabled via checking the fast start option at the muxing stage of the transcoding process. Placing the MOOV Atom at the beginning of the file will enable immediate streaming download and playback; whereas, placement of the MOOV Atom at the end of the file will force it to download in full first before it will start playing the file. If progressive download is a desired delivery method, then enabling "progressive download" option in the muxing or transcoding software will place the MOOV Atom at the beginning of the file. Streaming playback will not be affected by the location of the MOOV Atom.

Stream Bitrate Budget Allocation
The stream bitrate budget is comprised of all of the audio-visual elements, in addition to target audience capability, network conditions, distribution capacity, and general economics of delivery. When deciding on a bitrate budget to adhere to, add about 10-20% percent for network spikes.

Audio Video Synchronization
Keeping audio and video in sync is vital for making certain that a viewing experience is watchable. Flash player synchronizes its clock to the audio clock on the system because it is the oldest, most reliable, and least expensive clock to adhere to. Therefore it is important to make sure that the audio source is of the best quality and has the least amount of noise, is not clipped, and is not muted. If audio is not in order, the video will pause or play sporadically because it will try to re-sync itself constantly. Use audio test patterns to ascertain confidence in audio processing systems and audio-video synchronization tests to make sure that neither audio nor video drifts apart. Usually, if audio errors are not severe, audio-video synchronization errors do not occur immediately. Because they may be subtle, their effects may occur over time making audio-video synchronization errors cumulative in nature. These errors can occur either in several repeat cycles of 10-30 minutes each or past a 10-20 minute continuous play period; therefore, making this type of issue difficult to catch unless persistently tested for and observed.

Keyframe Alignment
Keyframe alignment is related to the continuity of keyframes across all of the segments in multibitrate encoding for Dynamic Streaming. See Appendix A " Video Encoding and Transcoding Recommendations for Adobe Flash HTTP Dynamic Streaming" for expanded explanation on keyframe alignment.

Playback Capability of a Client System
When considering the encoding parameters for a desired audience, it is vital to understand what kind of systems the video is intended to work with. Whether there is a support for GPU acceleration or whether there is enough contingency scenarios built in to the player's logic to provide a high-quality viewing experience. Careful planning around the prospective target audience systems will provide a good foundation for encoding parameters in synergy with design of the player experience.

Video Dimension Choices: A Use Case

A video engineering team working on a video player for their company's website has proposed dimensions of 640x480 to follow 4:3 aspect ratio, and 854x480 to follow 16:9 aspect ratio.

These sizes do not follow best practices, which advise a divisibility of 16 (width & height). Although it is technically possible to use the stretch coefficient of 1.7777 to derive a width that corresponds to their height, the height and width in this case is not guaranteed to fall within divisibility of 16. Earlier VP6 and more recent H.264 codecs use 16x16 macroblock subdivisions. If they would like to achieve best client performance, 16x divisible sizes should be used.

Of course, if they need to scale, scaling width should be done first, before height, in order to maintain picture sharpness. It is suggested that they keep the horizontal stretch at a maximum of 30% of the height. For instance, if looking at 720(w) x 480(h), then scaling it to 640x480 will still provide the necessary bit reduction and will maintain picture sharpness. When resizing the horizontal dimension beyond 30% of the scaling threshold you must then consider resizing vertical dimension as well, to maintain aspect ratio.

Therefore, if the original dimensions are 640x480, then the 16x divisible dimension would have to be either 768x432 for best performance results or at least divisible by 8x for good performance results (640x360, 768x432, 896x504).

Performance

CPU Processing Limitation

The CPU performance at the end stations is a significant determinant of their capabilities. CPU consumption depends almost linearly on image size. The Encoder require 2-4x CPU cycles than the Decoder. CPU performance and display size limit the maximum number of participants (decoders) and their respective transmissions that the end station can decode and display. Moreover, the graphics chipset or GPU may offload from the processor tasks such as YUVRGB conversion, 4:2:0 to 4:2:2 conversion, video scaling, postprocessing and even certain stages of the decoding process.

Video Player Design Considerations

The Flash Platform offers a comprehensive set of features, ranging from real-time interaction to complex 3D transformation. It is possible to have an application that supports video playback that also contains a wide range of these sophisticated features and effects — each of them having varying impact on overall performance. Therefore it is important to understand how to optimize certain Flash Player features in video playback applications, especially when delivering high-definition content.

One efficient way to avoid many pitfalls in the design of video playback applications is to utilize Open Source Media Framework (OSMF). OSMF is an open software framework for building robust, feature-rich video players and applications based on the Adobe® Flash® Platform. It enables developers to easily assemble pluggable components to create high-quality, full-featured playback experiences.

Whether you are designing an application from scratch or building it using OSMF, the following best practices should be followed in order to deliver the best overall playback experience.

General Best Practices

Some specific guidelines to follow for the best playback performance in Flash Player include:

- Avoid scaling the video object or playback component; match the video source size
- Make sure that video size is divisible by 16 for best perceptual and playback performance
- Conform player's aspect ratio to the video aspect ratio and pad it with black bars if necessary
- Turn off smoothing and deblocking when using the video object
- Position the video object on full coordinates (e.g. x=0, y=0, width=320, height=240)
- Avoid using alpha overlays. Use visible=false, not alpha=0
- Avoid placing overlays above video object if possible
- Don't apply 3D transformations to video, unless it is very low bitrate

In addition to these performance guidelines, some general rules for interactivity should also be considered when designing a player. Of course, video players will have a specific set of expected behaviors, such as play/pause, stop, fast forward, rewind, mute, and so on. Users also appreciate having a thumbnail preview of the video, and generally don't want a video that starts automatically when they go to a web page. Pausing a video on application start will also save bandwidth costs, as the video will only play when specifically requested.

Bitmap Transformations

Flash Player allows access to the bitmap data of each frame of a video, allowing you to apply a wide variety of effects in realtime, such as filters and other color transformations. This is quite processor-intensive, however, so it is not recommended for large or high bitrate video.

It is also important to note that access to bitmap data of video streamed with Flash Media Server 3 (FMS 3.0) or later is restricted by default. This is a security measure to avoid "stream ripping" or screen recording of the content. To access the bitmap data of a stream, you must modify the application.xml for your FMS application to explicitly permit this, or set it in server-side ActionScript. If you don't have server-side access to your FMS machine to change this setting, you won't be able to access the pixel data in the video.

Full Screen Support

Full screen playback has become an expected feature in professional video player applications. Too often, however, performance is compromised by a lack of optimization in the application code. Recommendations differ depending on the Flash Player version being targeted.

Flash Player 9 or Later

Full screen playback is supported in Flash Player 9 or later by using the fullScreenSourceRect to define the source area, and switching the displayState to "fullScreen." Hardware scaling will be used in full screen mode if supported by the user's video card; if it is not supported (or disabled by the user), the original software-rendered full screen mode will be displayed.

Best practices for optimum performance using this approach include:

- Match the video object/movie size when setting *fullScreenSourceRect*
- Avoid scaling the video (if unavoidable, use an even divisor of original size, e.g. ½, ¼, or 2x)
- Disable smoothing and deblocking
- No overlays or objects below the video object

Version 9,0,28,0 or later of Flash Player must be installed to support full screen mode and 9,0,115,0 or later for hardware-scaled full screen mode. For more details about implementing full screen playback in Flash Player 9 or later, refer to "Exploring Full Screen Mode" in Flash Player 9 at the Adobe Developer Connection: *http://www.adobe.com/devnet/flashplayer/articles/full_screen_mode.html.*

Scaling Video Controls

In most cases, the video controls will also be scaled when switching to full screen mode, which can result in fuzzy or pixelated controls. There are three approaches to full screen scaling that, depending on the use case, can minimize this issue. These are presented in order, from slowest to fastest playback performance.

Software-scaled video with smoothing is a common approach that keeps the controls and UI crisp, but is processor-intensive to render. This solution may not be acceptable for high definition content for this reason.

Software-scaled video with no smoothing offers better playback performance, while keeping the controls crisp. However, depending on the resolution of the original video and the scaled size, the video may be noticeably pixelated.

Hardware-scaled video with smoothing provides the best performance and video quality, and is supported in Flash Player 9 and later. In this case, the fullScreenSourceRect matches the video area (1:1 pixel mapping), and although the controls might appear slightly blurrier, the GPU will take care of the video scaling as well as smoothing.

Some best practices for handling video controls in full screen mode include:

- **Flash Player 9:** Use the method outlined previously with care and only if performance impact is acceptable and fidelity of the user interface is a priority (i.e. standard definition content/low screen resolution)
- **Flash Player 9:** Since the video resolution of high definition content is closer to screen resolution, evaluate if a minor pixelation of controls is acceptable. If so, use the video object or movie size as *fullScreenSourceRect* value, and perform minor resizing of the video controls as needed.

For more information about these full screen scaling techniques, see Jen's Loeffler's blog: *http://www.flashstreamworks.com/archive.php?post_id=1253805570*.

For more details about full screen support, see: *http://www.flashstreamworks.com/archive.php?post_id=1241799250*.

Conclusion

As anyone who has been tasked with media encoding can attest, it is an art form as well as a science. In this paper we have tried to address both the scientific aspects as well as the nuanced tailoring of settings to achieve the highest possible quality for video on demand delivery on the Flash Platform. As the requirements and expectations of media delivery evolve, this paper will be expanded with additional content and updated recommendations.

Glossary

The following table defines acronyms and abbreviations used in this document.

Abbreviation	Meaning
608B	Refers to closed captioning in the EIA 608B (formerly known as CEA 608B) format.
708B	Refers to closed captioning in the EIA 708B Section Nine format. However, the US FCC July 2000 Report and Order is actually the governing document. The Report and Order goes beyond the recommendations of EIA 708B Section Nine.
ASF	Advanced Systems Format. A container file format for holding audio and video elementary streams.
AU	Access Unit. A set of NAL units in a specified form is referred to as an access unit. The decoding of each access unit results in one decoded picture. Each access unit contains a set of VCL NAL units that together compose a primary coded picture. It may also be prefixed with an access unit delimiter to aid in locating the start of the access unit. Some supplemental enhancement information containing data such as picture timing information may also precede the primary coded picture. For additional details, see *http://en.wikipedia.org/wiki/Network_Abstraction_Layer#Access_Units*.
CBR	Constant Bit Rate.
B-frame	A bidirectionally predicted picture, or a picture created by reference to preceding and subsequent pictures.
Block	A set of 8x8 pixels used during Discrete Cosine Transform.
BER	Bit Error Rate
CEA	Consumer Electronics Association.
Compression	Reduction of the number of bits needed to represent an item of data.
CPB	CPB (Coded Picture Buffer) is a first-in, first-out buffer containing access units in decoding order. CPB affects the amount of time it takes to rejoin the primary stream when returning from a trick stream. Typically the CPB value is established in milliseconds.
CPB Delay	CPB delay is the time difference between the arrival of an access unit into the Coded Picture Buffer and when the Decoder starts decoding the access unit. Refer to Section C.1.2 in ISO/IEC14496-10 for additional information.
Discreet Cosine Transform (DCT)	Temporal-to-frequency transform used during spacial encoding of MPEG video.
DV	Digital video.
Elementary Stream (ES)	A bit stream that includes video, audio, or data. It represents the preliminary stage of the Packetized Elementary Stream (PES)
EIA	Electronic Industries Alliance.
Entry point	A point in the bit stream that offers random access.
Enthropy Coding	The process by which DCT cooefficiencients are resized according to the number of times they appear in the bit stream. The most frequently repeated coefficients are expressed in the smallest word length, decreasing the total number of bits need to represent a single frame.
FourCC	Four-character code used to identify data types. For the purposes of this document, this term is used for video and audio codecs.
Frame	Lines of spacial information for a video signal.
GOP	Group of Pictures. A set of pictures used for temporal encoding of MPEG video.
HD	High-definition.
IDR	Instantaneous decoding refresh (ISO/IEC 14496-10).

Abbreviation	Meaning
I-frame	An intra-coded frame, or a frame encoded without reference to any other frame. I-frame acts as a reference for predicted (P-frame) and Bidirectionally predicted (B-frame) pictures in a compressed video stream.
Inter-frame prediction	A compression technique that periodically encodes a complete reference frame and then uses that frame to predict the preceding and following frames.
ITU	International Telecommunications Union (UIT).
Level	A range of picture parameters and combinations of picture parameters specified by H.264. The level of a video signal generally indicates the number of pixels per frame, along with other device-specific parameters such as required memory and buffering.
Mbps	Megabits per second.
Macroblock	A group of 16x16 pixels used for motion estimation in temporal encoding of H.264 video.
Motion Prediction	The process that reduces redundancy in a video signal by measuring an object's motion at the encoder and sending a motion vector to the decoder in place of the encoded object.
MC	Motion Compensation.
Motion Vector	A pair of numbers that represents the vertical and horizontal displacement of a region from one frame to another.
ME	Motion Estimation.
MPEG	Moving Picture Experts Group.
PAT	Program Association Table.
Packetized Elementary Stream (PES)	Packetized elementary stream (ISO/IEC 13818-1). A stream containing variable-length packets of video, audio or data.
Payload	All the bytes in a packet that follow the packet header. For packets of audio or video, the payload contains audio or video data from the PES packets.
Packet	see PES Packet.
PES Packet	The structure used to carry a single frame of audio or video data. It consists of a header and a payload.
PES Packet Header	The leading bytes of a PES packet, which contain ancillary data for the packet.
PID	Packet Identifier (ISO/IEC 13818-1).
Profile	A defined subset of the syntax specified in the H.264 video coding specification. Different profiles indicate different levels of coding complexity for a picture.
P-frame	A predicted frame, or a picture coded using references to a previous I- or P-frame.
PMT	Program Map Table (ISO/IEC 13818-1).
PRX	Extension given to profile .xml files used by Windows Media Encoder.
PSI	Program-Specific Information.
PTS	Presentation Time Stamp (ISO/IEC 13818-1).
SD	Standard-definition.
SDI	Serial Digital Interface.
SEI	Supplemental Enhancement Information (ISO/IEC 14496-10).
SPTS	MPEG-2 Single Program Transport Stream (ISO/IEC 13818-1).
Spacial Encoding	The process of compressing a video signal by eliminating redundancy between adjacent pixels in a frame.
SPS/PPS	Sequence Parameter Set/Picture Parameter Set.
Temporal Encoding	The process that compresses a video signal by eliminating redundancy between sequential frames.

Abbreviation	Meaning
Temporal Masking	The phenomenon that happens when a loud sound drowns out a softer sound that occurs immediately before or after it.
Quantization	Part of the spacial encoding process, it re-orders DCT coefficients according to their visual importance.
Video Compression	The process used to reduce the number of bits needed to repsent a video frame.
VOD	Video on Demand.
VUI	Video Usability Information (ISO/IEC 14496-10).
Weighting	During video compression, it is the process by which degradation, or noise, is strategically placed in more detailed or complex picture areas where the viewer is least likely to notice it.

Reference Materials

"Advanced video coding for generic audiovisual services." ITU-T Recommendation H.264, March 2005.

"MPEG-4 AVC/H.264 video codec comparison." CS MSU Graphics & Media Lab Video Group. December 2005. Web. (http://www.compression.ru/video/index.htm).

Bovik A. *Handbook of Image and Video Processing.* San Diego CA: Academic Press, 2000.

Footen J. and Faust J. *The Service-Oriented Media Enterprise.* Burlington MA: Elseiver Inc., , 2008.

Hanzo L., P. Cherriman P., and J. Streit. *Video Compression and Communications.* 2nd Ed. West Sussex UK: John Wiley & Sons Ltd., 2007.

Keith J. *Video Demystified.* Burlington MA: Elseiver Inc., 2009.

Kovalick A. *Video Systems in an IT Environment.* 2nd Ed. Burlington MA: Elseiver Inc., 2009.

Larson L., R. Costantini. *Flash Video for Professionals.* Indianapolis IN: Wiley Publishing, 2007.

Lee J-B & Kalva H. *The VC-1 and H.264 Video Compression Standards for Broadband Video Services.* New York NY: Springer, 2009.

Poynton C. *Digital Video and HDTV, Algorithms and Interfaces.* San Francisco CA: Morgan Kaufman Publishers, 2003.

Rahul, Vanam, Eve A. Riskin and Richard E. Ladner. *H.264/MPEG-4 AVC Encoder Parameter Selection Algorithms for Complexity Distortion Tradeoff.* University of Washington, Department of Electrical Engineering, Department of Computer Sciences and Engineering [No Date Published].

Reinhardt R. *Video with Adobe Flash CS4 Professional.* Berkley CA: Adobe Press, Peachpit, 2009.

Richardson I. E. G. *Video Codec Design.* West Sussex UK: John Wiley & Sons, Ltd., 2002.

Simpson W. *Video Over IP.* 2nd Ed. Burlington MA: Elseiver Inc., 2008.

Smart M. and B. Keepence. *Understanding MPEG-4 Video.* IndigoVision, 2008.

Steinberg V. *Video Standards, Signals, Formats, and Interfaces.* Hampshire UK: Snell & Wilcox, 1997.

T. Wiegand, G. J. Sullivan, G. Bjwntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, 13.7 (July 2003): 560–576.

Video Encoding and Transcoding Recommendations for Adobe® Flash HTTP Dynamic Streaming

Preliminary Recommendations for Video on Demand

By Maxim Levkov, Adobe Systems Inc.

Addendum
Table of Contents

Scope

Video Encoding and Transcoding Recommendations for Adobe® Flash HTTP Dynamic Streaming describes techniques, best practices, and recommended settings for encoding video on demand (VOD) files for use with Flash Media Server.

The recommendations presented in this guide are targeted specifically at encoding of video on demand (VOD) (e.g. non-live) content. This document is written for personnel who encode VOD assets for distribution on the Flash Media Server ecosystem.

While similar to recommendations for encoding live content, live is not specifically targeted in this release of the guide. Encoding recommendations for live hardware encoders, used primarily for live broadcasting, will be covered in live content encoding releases of this guide.

Video Settings

The following are recommended video encoding settings for use in targeting HTTP and RTMP multibitrate delivery on the Flash platform, along with behavioral considerations and best practice use cases.

H.264 Settings

GOP Settings

- **GOP IDR control: Closed GOP**
- **Maximum B frames: 2 (Adaptive mode: on) or (Adaptive number of B frames: on)** (depending on the software used)
- **(IDR) Keyframe interval**
  - Content Specific
    - Shorter form content: (30 second commercials with many changing scenes)
      - General range 2 – 5 seconds
      - Typical range 2 - 3 seconds
      - No less than 2 seconds
    - Long form content: (1 minute or longer)
      - General range 5 – 8 seconds
      - Typical range 3 - 5 seconds
      - No less than 3 seconds
  - Static across all bitrates

Behavioral Considerations

Keyframes

In H.264, IDR and non-IDR I-frames have different meaning. IDRs (Instantaneous Decoding Refresh) are 'traditional' I-frames the way they were known in previous standards. An IDR Iframe allows subsequent frames reference itself and the frames after it, thus marking a closed GOP (Group of Pictures). This behavior resets the decoder completely and thus guarantees reliable seeking.

A non-IDR I-frame can be understood as an intra-coded P-frame. They can be referenced by preceding B-frames, which improves picture quality (especially in high motion scenes) and reduces I-frame flicker effect by smoothing the P-to-I frame transition. The side effect of non-IDR frames that they can reduce seeking precision or increase decoder start-up time.

Flash can only change bitrates at IDR keyframe intervals, (referred to as "keyframe" from here forward). Thus, the selection of keyframe distance must be carefully considered. Keyframe interval affects seeking performance (in file playback), decoder playback start-up time (for network streaming), recovery time from network errors, and overall picture quality.

It is suggested to keep the keyframe interval at a static distance of no less than 2 seconds for short form content and no less than 3 seconds for long form content, across all bitrates. See above for suggested ranges.

If keyframes are spaced closer than 2 seconds, resulting response is going to be better for the heuristic bitrate changes, but video quality will suffer. Keyframes are much larger than intermediate frames. If the keyframe rate is taken from one every 2 seconds to one every 1 second, the data rate is nearly doubled if everything else is held constant. (Calculated on the basis of 2x the number of large keyframes less the number of intermediate frames no longer needed, a negligible amount, resulting in almost 2x the rate.) Generally, the goal is to provide a file at a given download bitrate, thus, not all elements can be held constant. Instead, the video quality must be reduced proportionately, in order to get the data rate back down to [a little less than] one-half the new higher rate. So increasing the keyframe rate reduces the quality due to scarcity of bit resources for a given bitrate. For instance, at 100 kbps, doubling the keyframe rate, or reducing keyframe interval by half (for example, if taking from interval of 10 seconds to5 seconds) amounts to reducing the quality by half.

In other words, going from 1 to 2 second keyframe interval will mean that almost 2x the bitrate is available for quality improvement, and going from a 2 to a 3 second keyframe interval gives another 50%, and so on.

Switching

Network Conditions

If the reason for bitrate switching is network conditions, which is typically the case, this increases the risk of under run, causing video to pause or stutter. However, with bigger segments, averaging can actually improve across the download to find the actual data rate, which improves the operation of client-driven HTTP Dynamic Streaming. Ultimately, less network bandwidth and time is spent on overhead (such as HTTP requests, returned headers, and the round-trip latency of waiting for a download to start), and more resources are spent on downloading the actual video.

Processing Resources

If the reason for bitrate switching is lack of CPU processing resources to play back video (which is not as common) it only takes one initial measurement per playback to realize where to cap the rate. Also, if CPU processing resources run out, there are all sorts of other unsolvable problems that take place, trying to instantly detect the failure and then drop no more frames. It is easier to turn off smoothing than it is to downshift the bitrate fast enough so that the user doesn't notice.

The details of this are different for the Flash Media Server multibitrate case, of course, because the server is controlling pushing the data down, rather than the client pulling. However, in the FMS case, fast switching is possible in a way that it isn't for HTTP, which negates almost all the concern about long keyframe intervals if implemented properly.

Generally, keyframe interval and fast switching depends on the content and the users. For long-form content, starting the video at a shorter keyframe interval still results in good quality and acceptable efficiency (2-3 seconds), and switching to a longer interval, with much improved network efficiency and reduced server transaction load in the HTTP case (3-5 seconds or more), after the first minute or two will provide the best user experience to network efficiency ratio.

Note: When **GOP Size** or **GOP Interval** is set to **FIXED GOP size**, then **IDR Indexing/Scene Change Sensitivity** (depending on the software) should will be set to disabled, since setting **Max GOP size with scene detection** or **GOP size determined by scene detection** not active.

Also, when **GOP Size** or **GOP Interval** is set to **FIXED GOP,** the **IDR Index Sensitivity** (determines the sensitivity frequency response to scene change detection) or S**cene Detection Threshold** is normally disabled. It is typically set by sensitivity percentage, (0-100, lowest – least sensitive, highest-most sensitive). In various software packages, these settings are available to the user regardless of interdependence relationships that these settings have — consequently creating an invalid file or simply omitting the settings altogether.

For example:
If encoder is issued a command based on the following settings:
gop.idr_period = 2
gop.keyframes = 30
rc.scene_detect = 50

Then it means that *gop.idr_period* is a period of keyframes frames where Nth I-frame is a keyframe, where every 2nd period is a keyframe. For instance, if your period is a keyframe period set to 2 seconds, then your keyframe period is equal to 60 frames, based on 30 frames per second timeline, because *gop.keyframes* sets a period of I-frames in number of frames. In this case there are two regular periods of 30 frames each. This is set by **GOP Size** or **GOP Interval** to **FIXED GOP.** This also means that *gop.keyframes* are non-IDR I frames.

If your timeline or timescale is other than 30 frames per second, such as 24, 25, 50, or 60 then *gop.keyframes* settings need to be adjusted appropriately.

Caveat: When *rc.scene_detect* is activated (in this case it is set to sensitivity of 50;) it dynamically places the keyframe interval (e.g. whenever a scene change is detected) with an accuracy sensitivity of 50. Essentially, this invalidates the regularity of the keyframe interval set by *gop.idr_period* and *gop.keyframes*. Turning on *rc.scene_detect* mode by **Max GOP size with scene detection** setting is not recommended since it results in no perfect alignment of frames across all of the bitrates.

Additional Behavioral Observations
Generally, allowing the encoder to generate a keyframe at a scene cut produces better quality video than requiring a scene cut to wait for a (later) fixed keyframe interval to occur. This is because it is simply not possible to describe the video image post-cut sufficiently using intermediate frames, and so the quality of the video between the scene cut and the next fixed keyframe point suffers.

However, allowing the encoder to generate a keyframe at a scene cut frequently results in every bitrate selecting exactly at the same keyframe point, because the scene cut detector is typically triggered at exactly the same point. When this happens, keyframe alignment takes place. Since it is an unpredictable behavior that is triggered by quality and consistency of the logic that drives the encoder, it sometimes does not place keyframes in same place, but in our experience it usually does. Even if there isn't an alignment for every keyframe, the video quality is better. The greater risk, of course, is if something happens with scene cut detection that creates future misalignment or if the file format has problems with it, such as in the following scenarios:

- Some encoders allow setting a fixed keyframe interval and allow scene cut detection. This means that even if a scene cut fires a keyframe just before that interval, yet another keyframe is generated right at that point. This ensures that you'll have alignment at least at the fixed keyframe interval point, which is sufficient (but not best).

  a. GOP Size: **GOP size determined by scene detection**

  b. IDR Control: **Variable IDR placement** enabled with *gop.idr_period = { period# }* (see above)

- Some encoders allow setting a maximum keyframe interval and allowing scene cut detection. This means that if a scene cut fires a keyframe, then the next keyframe will happen at the next scene cut or the next maximum interval, whichever happens first. This has the possibility of causing misalignment of all future keyframes after a scene cut, the scene cuts often match up, and if they don't at that one then the next one they do. Of course, there is a possibility of long runs (10 seconds or more) of misalignment, however, and therefore, it is not recommended.

a. See Max **GOP size determined by scene detection** on page 4 and 5.

- The best case would be an encoder which allows encoding one bitrate (i.e., the highest) with scene cut detection plus either a fixed or maximum keyframe interval, writing out a list of where those keyframes are. This list would then be used to encode the rest of the bitrates with forced keyframe placement at exact points matching the first encoded file.
- Another best case would be the encoder toolset that allows for placement of intra coded P frames or non IDR keyframes in between IDR keyframes within closed GOP at scene change.

Keyframes generated at scene cuts and forced at points should always be in alignment. When choosing a keyframe interval selection method, consider:

a. Best case is scene cut detection and maximum-interval-based keyframe point selection done in a first pass for one rate, then used by all other bitrates.

b. As a fallback, use scene cut detection and keyframes at fixed intervals from start-ofcontent (NOT at maximum intervals since last keyframe)

c. Worst case, keyframes at fixed intervals with no scene cut detection

d. Keyframes at scene cuts with maximum intervals and hoping for alignment is not recommended.

Reference frames: 2 - 4, closer to 2

Frame Rate: Must remain constant across all bitrates. This avoids multibitrate synchronization issues and maintains seamless transition during changes in the end user's network conditions.

Rate Control Settings

Mode:	1 or 2 Pass VBR (2 Pass VBR suggested)	
Buffer Length:	1 – 2 seconds (1000ms or 2000ms)	
VBV Buffer: (not available in all software packages)	**Initial Buffer fullness:** 70% (quicker response at seek times or trick modes)	**Final Buffer fullness:** 100% (keeps it full once fullness is reached)

Note about encoder recommendations: It is advised to use same encoder for encoding multibitrate files for all of the encoded content, thus avoiding issues related to predictability of keyframe alignment logic; which may be different in different encoders.

Frame Sizes and Bitrate Switching

In order to maintain the smoothest switching performance across all of the bitrates, thus keeping SPS (Sequence Parameter Set)/PPS (Picture Parameter Set) NAL (Network Abstraction Level) units sent to AVCC the same, the following aspects should be taken into consideration:

- Fixed frame size across all switching bitrates
- Bitrate as a variable component across all switching bitrates
- Same video duration for all switching bitrates

Sample use cases:

- Fixed Page Size
  a. Regular In-page Size Mode (ex., 480p)
  b. Full Screen Size Mode (ex., 720p)

- Fixed Screen Size
  a. Mobile Devices
  b. Netbooks
  c. Set-Top Boxes / IPTV
  d. Full Screen Mode

Avoid scaling down from a larger screen size to a smaller designated frame size. (e.g. , 1280x720 scaled down to 768x432 frame size). This will hinder the performance of the player by as much as 40%, depending on the client's processing capability. Instead, upscale or encode at the final fixed size and use ActionScript player logic to switch among targeted bitrates. Keep frame sizes, with respective aspect ratio, in even divisors of 16 for most optimal performance. Try not to scale the video object in Flash, but use *fullscreenSourceRect* to zoom to fullscreen mode; this improves the smoothness of fullscreen playback significantly. (For more details, visit http://www.flashstreamworks.com/archive.php?post_id=1253805570.)

To clarify and exemplify our recommendations, here is a standard use case of 480p video window in regular website mode, and 720p video window in fullscreen mode (bitrates are just examples).

Without modification of the standard player components – beginner	
480p, regular mode	720p, fullscreen mode
480p – 400 kbps	480p – 400 kbps
480p – 800 kbps	480p – 800 kbps
480p – 1200 kbps	480p – 1200 kbps
720p – 1800 kbps	720p – 1800 kbps
720p – 2400 kbps	720p – 2400 kbps
720p – 3500 kbps	720p – 3500 kbps

Note: Same videos for regular and fullscreen mode due to developer narrow knowledge of how to limit the bitrates during playback. Not typically recommended, because using bitrates # 4 thru 6 in regular mode adds performance and bandwidth overhead, and doesn't increase the quality.

Limit the bitrate in regular mode	
480p, regular mode	720p, fullscreen mode
480p – 400 kbps	480p – 400 kbps
480p – 800 kbps	480p – 800 kbps
480p – 1200 kbps	480p – 1200 kbps
	720p – 1800 kbps
	720p – 2400 kbps
	720p – 3500 kbps

Note: Video will never upgrade to 720p resolution in regular mode, and full screen will have no limit. Repurposing of 480p resolution in fullscreen mode avoids double encodes of same bitrates, in the hope that transitions between bitrate 3 and 4 will be limited. The only drawback of this selection is frame drops between bitrates #3 and #4. This is a more commonly used example.

Dedicated bitrates for each video window view	
480p, regular mode	720p, fullscreen mode
480p – 400 kbps	720p – 400 kbps
480p – 800 kbps	720p – 800 kbps
480p – 1200 kbps	720p – 1200 kbps
	720p – 1800 kbps
	720p – 2400 kbps
	720p – 3500 kbps

Note: 720p fullscreen mode will have an additional version of bitrates #1 thru #3 in 720p, adding 50% more encoding. This is the ideal configuration, with the best possible transition and performance characteristics.

Although it's possible to switch between different video resolutions in FP 10.0, however, you might experience a small video pause or audio pop. FP 10.1 removed this limitation; therefore, approach C is the recommended approach for FP 10.0, if the quality of the transition is most important.

Audio Settings

For optimal performance and quality of service, the following recommendations should be heeded when encoding content for multibitrate switching:

- Encoded audio must maintain a fixed bitrate across all encoded files
- HE-AAC v2 is the recommended codec

Acknowledgements

The author would like to thank the following people for their valuable assistance in the development of this document:

Ashley Still for taking a chance with me on this unprecedented endeavor, for her leadership, and for the opportunity itself.

Pritham Shetty for entrusting me to do this project.

David Devisser for active participation and help resolving day to day issues.

Desiree Motamedi for paving the way for the endearing presentations and helping to acquire great content.

Jens Loeffler for unprecedented day-to-day cooperation on the many tasks this project has entailed.

Victor Steinberg of VideoQ for meticulous technical oversight, lots of time and extraordinary cooperation, excellent ideas, and great engineering in creation of specialized dynamic test patterns.

Matthew Kaufman for straightforward feedback and unparalleled insight into the inner workings of Flash, and for help in putting together the Encoding HTTP Delivery for Flash addendum section of the cookbook.

Lisa Larson-Kelley for excellent editing, creative insight, and for outstanding day to day cooperation on every aspect of this cookbook.

Kevin Towes for keeping me in-tune with latest developments, and then some.

Srinivas Manapragada for helping to put forth the Encoding HTTP Delivery for Flash addendum section of the cookbook.

Abhinav Kapoor, Vishy Swaminathan, Kevin Streeter, and Slavik Lozben for insight on the inner workings of Flash and for their feedback on various sections of the cookbook.

Special appreciation to:

ArtBeats for providing excellent content and timely delivery.

VideoQ for making variety of special test patterns, timely availability, and unquestionable general technical oversight.

Hillman Curtis for providing creative content.

Adobe

Adobe Systems Incorporated
345 Park Avenue
San Jose, CA 95110-2704
USA
www.adobe.com

4/10