Encoding for Flash, Mobile and HTML5

Jan Ozer
www.streaminglearningcenter.com
jozer@mindspring.com/
276-238-9135

Copyright © 2011 Jan Ozer, All Rights Reserved

Agenda

- Terms and terminology
- Producing H.264
- Producing WebM
- Encoding for adaptive streaming
- Comparing H.264 encoders
- If you see red, pay attention
Terms and Terminology

- Codec/compression
- Bandwidth
- Resolution
- Data rate
- Bitrate control
- Container formats, program and transport streams

Compression/Codec

- Compression
  - Used to shrink the size of video/audio still images
  - Common codecs
    - Video - H.264, MPEG-2, WMV
    - Audio - MP3, AAC, WMV
    - Still image - JPG, PNG, GIF
- Codecs - all of the above
  - Any technology that **COMPresses in the studio**, then **DECompresses in the field**
Why is Compression Important?

- To achieve the target data rate, you have to compress.
- Compression is “lossy,” the more you compress, the more you lose.
- This is immutable.

About Bandwidth

- What is bandwidth?
  - Viewer’s connection speed.
- Why is it important?
  - Controls your viewer’s ability to retrieve and play video smoothly.
  - Higher delivery bandwidths mean higher data rates, which means better quality.
Bandwidth

- Viewer’s connection speed to the Internet
  - Increasing dramatically
  - www.speedtest.net
Paradigm Shift

- Modem connection speeds (28.8/56 kbps)
  - Make the video fit
- Broadband
  - Make it good enough
  - Make it affordable
    - CNN/ESPN around 900 kbps combined; not to fit pipes, but because it’s “good enough” and affordable
- Cellular
  - Make it fit

About Resolution

- What is it?
  - Actual pixels in the file
Resolutions That We Know and Love

**Acquisition**
- DV - 720x480
- 720p - 1280x720
- 1080i/p - 1920x1080

**Distribution**
- DVD - 720x480
- Streaming - 720p or smaller
- iPod - 320x240 - 720p

Why is Resolution So Important

- Most video starts life at 720x480 or higher
- Most video is scaled down for streaming
  - 320x240 is the general minimum
  - Ranges up to 720p and sometimes higher
- Resolution is key quality factor
  - At a set bit rate, increasing resolution degrades quality because you must compress more pixels
  - Can’t say that 300 kbps is “adequate” without knowing resolution
What is Data Rate?

- Every time you produce a streaming file, you have to choose a data rate
  - Considerations - quality, cost, viewer bandwidth
- Uncompressed video is very, very large, to bulky to efficiently deliver - so you have to compress a lot!

Why is Data Rate Important?

- Largely determines video quality
  - At static resolution and frame rate, increasing bit rate increases quality
- Because we have to pay for it
Constant vs Variable Bit Rate

- **Constant Bit Rate (CBR)**
  - One bit rate applied to entire video, irrespective of content

- **Pros:**
  - Computationally easy
  - Fast - one pass will do it

- **Cons:** Doesn’t optimize quality
Constant vs Variable Bit Rate

- Variable Bit Rate (VBR)
  - Dynamic bit rate matches motion in video
- Pros: Best quality
- Cons:
  - Need two or more passes
  - Can produce stream with significant variability

How do I Produce the Best Quality CBR?

- Use 2-pass CBR when available
  - Scans file (like VBR), but packs data into a consistent stream
  - Best of both worlds when available
- 1-pass of live or draft work
How Do I Produce the Optimal VBR File?

- 2 passes or more
- Use “Constrained"
  - Constrains to data rate to specified max
- Set Target and Max/Min
  - Overall target – 500 kbps
  - Max/Peak bit rate – how high rate can go when varying
  - Rule of thumb is 1.5 - 2X of target
  - If minimum setting, use .5x

When Should I Use VBR/CBR?

- Constrained VBR (usually 2X) for most streaming applications
  - Broadband has sufficient headroom to handle spikes
- Constrained VBR for virtually all progressive delivery
- Constrained VBR for most cellular connections, though not universally
- CBR for live, particularly when constrained
- CBR or constrained VBR for adaptive
Container Formats, Transport Streams, Program Streams

- Container format:
  - A meta-file format whose specification describes how data and metadata are stored
  - Examples: MPG, MP4, MOV, F4V, WMV, FLV

- Transport stream:
  - Transport stream specifies a **container format** encapsulating packetized elementary streams, with **error correction** and **stream synchronization** features for maintaining transmission integrity when the signal is degraded (e.g. - broadcast)
  - Examples: .ts,

- Program stream:
  - **Container format** with multiplexed audio and video
  - Examples: .ps, VOB, EVO
Why Do We Care?

- When producing for single file delivery or streaming, you have to choose the right container format/transport stream for your application
  - Streaming
    - Flash - MP4, FLV, F4V, MOV - H.264
    - HTML5 - MP4 - H.264, WebM - VP8
    - iDevices - MP4, MOV - H.264
    - Adaptive - HTTP Live Streaming - .ts

H.264 Specific Parameters

- Introduction to H.264
  - The MPEG-4 specification
  - The MPEG-4 codec
  - MPEG-4 spec audio options
  - The H.264 codec
MPEG-4 Specification

- Introduced in 1998 by ISO/IEC Moving Picture Experts Group
- 28 “Parts” within specification
  - Part 2 - MPEG video codec (the MPEG-4 codec)
  - Part 3 - MPEG-4 audio (AAC, etc)
  - Part 10 - Advanced Video Coding (AVC/H.264)
  - Part 14 - container format (MP4)

The MPEG-4 Video Codec

- Used only for low power devices: Two profiles:
  - Simple Profile - very low power, low bandwidth applications
  - Advanced Simple Profile - Simple plus:
    - Support for "MPEG"-style quantization
    - Support for B pictures (a.k.a. B-frames)
    - Motion compensation

- Check specs on target devices
  - Will refer to MPEG-4 encoding only in mobile segments
  - Never use for computer playback
MPEG-4 Audio

- AAC-Low Complexity (AAC-LC)
  - The most basic and most broadly compatible
  - Also called AAC+, aacPlus and
- High Efficiency AACv2 (2006)
  - Also called enhanced aacPlus, Enhanced AAC+, aacPlus v2 and eAAC+

MPEG-4 Audio Presentation

Compressor

Squeeze

Adobe Media Encoder

Episode

Streaming Learning Center

Copyright © 2011 Jan Ozer, All Rights Reserved
MPEG-4 Audio Summary

- Presentation varies by encoding tool
  - Some let you choose audio codecs that are not MPEG-4 compatible
  - Usage dictated by specs of target device
  - When in doubt, choose Plain Jane AAC
    - Most compatible on the playback side
    - Universally available on MPEG-4/H.264 encoders

What is H.264?

- Adapted by ISO and ITU
  - Telephony/portal
  - TV - consumer electronics
  - Computer electronics
- Only codec adopted by top three streaming providers (Apple, Adobe, Microsoft)
What’s MPEG-4/H.264 Cost?

- For free Internet video (e.g. no subscription or pay per view), free in perpetuity
  - Still technically a licensing obligation, but there are no teeth and no motivation to enforce
- For subscription or PPV, there may be a royalty obligation
- Check www.mpeg-la.com

What is an MP4 file (and what are the variants)?

- .MP4 - official MPEG-4 wrapper
- .M4V - Apple’s variant for iTunes and devices
- .MOV - H.264 file for editing or QuickTime delivery
- .F4V - H.264 for Flash
- .3GP - (not shown) - phone
- .MPG - H.264 in MPEG-2 transport stream
H.264 Encoding Parameters

- The basics
- I, B and P-frames
- Search related options
- Miscellaneous options

H.264 Encoding - Basics

- Profiles and Levels
- Entropy encoding
What are H.264 Profiles?

- “Define a set of coding tools or algorithms that can be used in generating a bitstream”

<table>
<thead>
<tr>
<th>Feature</th>
<th>Baseline</th>
<th>Extended</th>
<th>Main</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>I and P Slices</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>B Slices</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Multiple Reference Frames</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Intra Loop Deblocking Filter</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CAVLC Entropy Coding</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CABAC Entropy Coding</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Interlaced Coding (Pic4x4, MIAF4)</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GPU vs. 2D Transform Adaptivity</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Quantitative Scaling Modeless</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Separate Ch and Cr OP control</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Separate Color Plane Coding</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Predictive Lossless Coding</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Which Profile?

- High Profile
- Baseline
Which Profile?

- Critical to know your target profile before encoding
  - Device
    - iPod/iPhone - always Baseline
    - iPad - Main
  - Computer playback - High for all targets
- Issues to consider
  - iPad/iPhone/iPod Touch – one file for all, use Baseline
  - Computer/iPad - use Main

What are H.264 Levels?

- “Constrains key parameters in the bitstream”

<table>
<thead>
<tr>
<th>Level number</th>
<th>Max video bit rate (Kbps) for Baseline, Extended and Main Profiles</th>
<th>Max video bit rate (Kbps) for High Profile</th>
<th>Examples for high resolution @ frame rate=max allowed frame rate in Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>64 kbps</td>
<td>60 kbps</td>
<td>128x96@30.9 (8) 176x144@15.0 (4)</td>
</tr>
<tr>
<td>1a</td>
<td>128 kbps</td>
<td>160 kbps</td>
<td>128x96@30.9 (8) 176x144@15.0 (4)</td>
</tr>
<tr>
<td>1.1</td>
<td>172 kbps</td>
<td>240 kbps</td>
<td>176x144@30.3 (6) 320x240@10.0 (3)</td>
</tr>
<tr>
<td>1.2</td>
<td>384 kbps</td>
<td>480 kbps</td>
<td>320x240@30.0 (7) 576x320@15.2 (8)</td>
</tr>
</tbody>
</table>
H.264 Levels

- Primarily an issue when encoding for devices
  - Must ensure that encoding parameters are within target level (most templates do this); otherwise video won’t load onto the device
- For computer playback,
  - Flash/QuickTime/SL/HTML5 can play ALL levels of ALL supported profiles – so level isn’t a concern
  - Rather, it’s can the computer play the file as configured
    - A netbook will try to load a 1080p file encoded for Flash
    - Flash Player won’t complain
    - But playback won’t be pretty

H.264 Levels – Bottom Line

- With devices, choosing the right profile and level is critical
- With computers, profiles and levels don’t determine whether the file will load or play well
- Rather, you have to choose a resolution and data rate that will play smoothly on your targets
Apple Compressor - Compression Settings

- Available options and presentation varies by encoding tool
- Apple’s is very simple

Frame Reordering:
- Checked or unchecked produces Main Profile
- Frame reordering is Main with 1 B-frame (always)
- Must use iPod preset to produce Baseline profile for iPod

AME - Compression Settings

- Adobe lets you choose Profile and level directly
- If level too low for selected encoding parameters, you’ll see an error message
  - Just increase the level until the error message goes away
Entropy Encoding

- CABAC (Context-adaptive binary arithmetic coding)
  - More efficient (e.g. better quality), but harder to decode
- CAVLC (Context-adaptive variable-length coding)
  - Less efficient, easier to decode
- Big question - does quality improvement outweigh increase in required CPU horsepower

CABAC vs CAVLC Quality

- In challenging scenes at low data rates, CABAC was noticeably better
- Most authorities place quality advantage at 10-15%
CABAC vs CAVLC

Performance

- Does increase playback requirements slightly on lower power computers
- My recommendation:
  - Always enable CABAC

<table>
<thead>
<tr>
<th>Playback 720p files</th>
<th>CABAC (%)</th>
<th>CAVLC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP 8710w - Core 2 duo (% of both CPUs)</td>
<td>31.1%</td>
<td>30.5%</td>
</tr>
<tr>
<td>PowerMac - Dual 2.7 GHz PPC G5 (% of 1 processor)</td>
<td>71.17</td>
<td>67.34</td>
</tr>
</tbody>
</table>

What Would YouTube Do?

- High Profile
- CABAC
Profile/CABAC in Squeeze and Episode

I-, B- and P-frames

- Caveats:
  - Presented differently by each encoding tool
  - Will only cover most critical and most common parameters
What are I, B and P Frames?

- **I-Frame** - encoded without reference to other frames (also called Key Frames)
- **P** - looks backward to I and P frames (predictive)
- **B** - looks forward and backward to previous I and P frames (Bi-directional interpolated)
  - No frames refer to B-Frame (most of the time)

What do I Need to Know About Key Frames?

- Least efficient – so largest (which is bad)
- But, key frames enhance interactivity
  - All playback starts on a key frame
  - When seeking to random frame, must start playback at key frame
  - Maximum interval should be 5-10 seconds
- Key frames "reset" quality:
  - Useful at scene changes
  - Enable natural key frames or key frames at scene changes
What do I Need to Know About B Frames?

- The most “efficient” frame
  - So improves quality (comparisons to come)
- Hardest to decode
  - Decoder has to have all referenced frames in memory to decode B-frame
  - Frame usually delivered out of order, which also complicates playback

Streaming Learning Center

B-frames - Yes/No
B-frames - Yes/No

- Noticeable quality improvement
- 5-10% increase in decompression CPU load
- Recommend
  - Say “YES” to B-frames
  - 3 is a good number for real world video
  - Experiment with higher numbers with animations

Typical B-Frame Encoding Parameters

- Number is number of B frames between I and P-Frames; (BBBBBPPBBBBBPBBBP)
  - 3 is recommended
- Reference frames
  - Number of frames searched for redundancies
  - 5 is recommended setting
Advanced B-frame Options

- Adaptive B-frames -
  - Will change order if it will improve quality (e.g. insert a key frame at a scene change)
- B-slice as reference
  - Allow B-frames to be reference frames for P-frames
  - Some quality improvement, can cause playback issues on low power devices

More Advanced B-frame Options

- IDR frames
  - No frames look behind IDR frame
  - All key frames s/be IDR
- B-frames
  - Adaptive
  - Reference B-frames
    - P-frames can reference
    - Pyramid-B-frames
    - B-frames can reference
- Rhozet recommends:
  - IDR frequency - 1
  - Reference B-Pictures/ pyramid enabled

http://www.rhozet.com/products_whitepapers.html
Search Related Options

- Searching for redundancies; multiple factors
  - Search shape (8x8/16x16) – size of shape used for searching (smaller is more accurate)
  - Sub-pixel mode – (full/half/quarter pixel) – smaller is more accurate
  - Fast - trades encoding speed for quality

Sorenson Squeeze - Effort Matters

- Fast/Medium/Best settings controls unspecified search and other parameters
  - Substantial difference in time and quality
    - Fast - 8:10
    - Best - 18:37
- Use Fast for draft work
- Use Best for final unless time constrained
Help file: “In general, values over 50 yield very small improvements in visible image quality.”

My tests confirmed those results:
- Encoding time at 50 - 3:33
- Encoding time at 100 - 8:03
- Difference noticeable on only one video within test sequence
Search Related Options

- Other tools
  - Compressor/Adobe Media Encoder – no search related Options
  - High end tools – Inlet Fathom, others
    - Unique controls; check help file
Slices

- Slices (Episode and Squeeze)
  - Divides frames into segments to speed encoding
  - Can't search between slices
  - Can reduce quality
  - Set to lowest value (either 0 or 1)

Other Options

- Deblocking filter - always on
- Rate distortion optimization
  - Dynamic balance between quality and bit rate - as in - is the extra quality worth the bits necessary to achieve it?
  - Typically enable
- Hadamard Transform - enable
H.264 Specific Encoding Tutorials

- Understanding key H.264 Encoding Parameters
  - http://www.vimeo.com/5377029
- Apple Compressor
  - http://vimeo.com/5462108
- Adobe Media Encoder CS4
  - http://www.vimeo.com/5118579
- Sorenson Squeeze
  - http://www.vimeo.com/5279015
- Telestream Episode Pro

Producing H.264 Video for Computer Playback

- Format specific considerations
  - Flash
  - HTML5
  - Mobile
### Flash and H.264

<table>
<thead>
<tr>
<th>Flash Player</th>
<th>Playback Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video codec</td>
<td>H.264 ONLY, not MPEG-4</td>
</tr>
<tr>
<td>Profiles</td>
<td>Baseline, Main, High</td>
</tr>
<tr>
<td>Audio codec</td>
<td>AAC, AAC-LC, HE-AAC</td>
</tr>
<tr>
<td>Container formats</td>
<td>F4V (preferred), MP4, M4V, M4A, MOV, 3GP</td>
</tr>
</tbody>
</table>

### HTML5 and H.264

- No published specs that I could find, but tested playback on Chromium, Safari and IE9
Producing for HTML5

- All browsers have their own players, no specifications (that I could find)
  - Baseline, Main, High all supported up to 1080p
  - Ditto for AAC-LC, HE-AAC, HE-AAC v2
- Use MP4 file since F4V or MOV could trigger Flash or QuickTime players

Producing for Devices

- Levels matter
- Supplied specs vary by vendor
  - Some are all over the map
### iDevice Specification Overview

<table>
<thead>
<tr>
<th>Device spec</th>
<th>Original iPod (pre-5g)</th>
<th>iPod Nano/Classic</th>
<th>iPod touch/ iPhone 4/iPod touch</th>
<th>iPhone 4/iPod touch</th>
<th>iPad 1&amp;2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen rez</td>
<td>320x240</td>
<td>320x240</td>
<td>480x320</td>
<td>960x640</td>
<td>1024x768</td>
</tr>
<tr>
<td>Aspect ratio</td>
<td>4:3</td>
<td>4:3</td>
<td>16:9-ish</td>
<td>16:9-ish</td>
<td>4:3</td>
</tr>
<tr>
<td>Codec spec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video codec</td>
<td>H.264</td>
<td>H.264</td>
<td>H.264</td>
<td>H.264</td>
<td>H.264</td>
</tr>
<tr>
<td>Max data rate</td>
<td>768 kbps</td>
<td>2.5 Mbps</td>
<td>2.5 Mbps</td>
<td>14 Mbps</td>
<td>14 Mbps</td>
</tr>
<tr>
<td>Max video rez</td>
<td>320x240</td>
<td>640x480</td>
<td>640x480</td>
<td>720p</td>
<td>720p</td>
</tr>
<tr>
<td>Frame rate</td>
<td>30 fps</td>
<td>30 fps</td>
<td>30 fps</td>
<td>30 fps</td>
<td>30 fps</td>
</tr>
<tr>
<td>Profile/level</td>
<td>Baseline to Level 1.3</td>
<td>Baseline to Level 3.0</td>
<td>Baseline to Level 3.0</td>
<td>Main to Level 3.1</td>
<td>Main to Level 3.1</td>
</tr>
<tr>
<td>Audio codec</td>
<td>AAC-LC</td>
<td>AAC-LC</td>
<td>AAC-LC</td>
<td>AAC-LC</td>
<td>AAC-LC</td>
</tr>
<tr>
<td>Max data rate</td>
<td>160 kbps</td>
<td>160 kbps</td>
<td>160 kbps</td>
<td>320 kbps</td>
<td>320 kbps</td>
</tr>
<tr>
<td>Audio params</td>
<td>48 kHz, stereo</td>
<td>48 kHz, stereo</td>
<td>48 kHz, stereo</td>
<td>48 kHz, stereo</td>
<td>48 kHz, stereo</td>
</tr>
<tr>
<td>Container formats</td>
<td>m4v/mp4/mov</td>
<td>m4v/mp4/mov</td>
<td>m4v/mp4/mov</td>
<td>m4v/mp4/mov</td>
<td>m4v/mp4/mov</td>
</tr>
</tbody>
</table>

- Limited number of devices, very well defined
- 3 categories, low, medium and high

### Producing for iDevices

- Two scenarios
  - Podcasts
    - Covered next
  - Streaming to iDevices
    - Best done with HTTP Live Streaming, covered later
Producing for iDevices

- Podcasts
  - Downloaded ~ 50 podcasts, mostly grouped into three resolutions
  - More detailed presentation at iDevice seminar on Tuesday

---

### Podcast Overview

<table>
<thead>
<tr>
<th>Video codec</th>
<th>320x180 (or 240)</th>
<th>640x360 (or 480)</th>
<th>720p</th>
</tr>
</thead>
<tbody>
<tr>
<td>H.264 codec, Baseline profile</td>
<td>H.264 codec, Baseline profile</td>
<td>H.264 codec, Main profile</td>
<td></td>
</tr>
<tr>
<td>Data rate</td>
<td>528 kbps</td>
<td>1.319 Mbps</td>
<td>2.845 Mbps</td>
</tr>
<tr>
<td>Key frames</td>
<td>120</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>Frame rate</td>
<td>match source</td>
<td>match source</td>
<td>match source</td>
</tr>
<tr>
<td>Audio</td>
<td>AAC LC</td>
<td>AAC LC</td>
<td>AAC LC</td>
</tr>
<tr>
<td>Data rate</td>
<td>111 kbps/stereo</td>
<td>114 kbps/stereo</td>
<td>134 kbps/stereo</td>
</tr>
<tr>
<td>Extension</td>
<td>.m4v</td>
<td>.m4v</td>
<td>.m4v</td>
</tr>
</tbody>
</table>

- Bitrate control - use VBR constrained to less of max data rate or 2X
- Entropy encoding - CABAC when producing with Main Profile
- Otherwise, check your encoding presets - make sure they reasonably conform to these configurations
Android OS

<table>
<thead>
<tr>
<th>Android OS</th>
<th>Playback Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video codecs supported in OS</td>
<td>H.263, MPEG-4 Simple Profile, H.264 Baseline (starting with Android 3.0)</td>
</tr>
<tr>
<td>Audio codecs</td>
<td>AAC, AAC-LC, HE-AAC up to 160 kbps</td>
</tr>
<tr>
<td>Container formats</td>
<td>3GP or MP4</td>
</tr>
</tbody>
</table>


Google Recommendations

- **Lower quality video**
  - Video codec: H.264 Baseline Profile
  - Video resolution: 176 x 144 px
  - Video frame rate: 12 fps
  - Video bitrate: 56 Kbps
  - Audio codec: AAC-LC
  - Audio channel: 1 (mono)
  - Audio bitrate: 24 Kbps

- **Higher quality video**
  - Video codec: H.264 Baseline Profile
  - Video resolution: 480 x 360 px
  - Video frame rate: 30 fps
  - Video bitrate: 500 Kbps
  - Audio codec: AAC-LC
  - Audio channel: 2 (stereo)
  - Audio bitrate: 128 Kbps

- Low is very conservative
- Check targeted hardware devices for additional capabilities
BlackBerry

- Defined at bit.ly/blackberryvidspecs
- However, there are 34 listed phones, each with different specs.

<table>
<thead>
<tr>
<th>BlackBerry Devices</th>
<th>Lower Common Denominator Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video specs</td>
<td>MPEG-4, Simple Profile, 320x240 resolution, 24 fps, @ max rate of 768 kbps.</td>
</tr>
<tr>
<td>Audio codecs</td>
<td>AAC-LC, HE-AAC and HE-AACv2</td>
</tr>
<tr>
<td>Container formats</td>
<td>MP4, M3A, 3GP, MOV</td>
</tr>
</tbody>
</table>

HP webOS - Max Settings

<table>
<thead>
<tr>
<th>By Codec</th>
<th>H.264</th>
<th>MPEG-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile</td>
<td>Baseline</td>
<td>Simple</td>
</tr>
<tr>
<td>Level</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Resolution</td>
<td>640x480</td>
<td>640x480</td>
</tr>
<tr>
<td>Frame rate</td>
<td>30 fps</td>
<td>30 fps</td>
</tr>
<tr>
<td>Bit rate</td>
<td>1.5 Mbps</td>
<td>1.5 Mbps</td>
</tr>
<tr>
<td>Audio codec</td>
<td>AAC-LE, HE-AAC, HE-AACv2</td>
<td>AAC-LE, HE-AAC, HE-AACv2</td>
</tr>
<tr>
<td>Bit rate</td>
<td>1.5 Mbps</td>
<td>1.5 Mbps</td>
</tr>
<tr>
<td>Container formats</td>
<td>MP4, M4A, M4V, MOV, 3GP, 3G2</td>
<td>MP4, M4A, M4V, MOV, 3GP, 3G2</td>
</tr>
</tbody>
</table>
**HP webOS - Recommended**

- Defined at bit.ly/webosvideospec

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Bandwidth</th>
<th>Recommended</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP progressive download</td>
<td>High</td>
<td>512 Kbps, H.264 Baseline Profile, 480 x 320 pixels, 30 fps. 64 Kbps, AAC+, 44 KHz, stereo.</td>
<td>All local formats are supported.</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>128 Kbps, H.264 Baseline Profile, 320 x 240 pixels, 20 fps. 24 Kbps, eAAC+, 44 KHz, stereo.</td>
<td>All local formats are supported.</td>
</tr>
<tr>
<td>Real time streaming protocol (RTSP)</td>
<td>Low</td>
<td>128 Kbps, H.264 Baseline Profile, 320 x 240 pixels, 20 fps. 24 Kbps, eAAC+, 44 KHz, stereo.</td>
<td>Video: H.264, MPEG-4, and H.263 Audio: AAC and AMR</td>
</tr>
</tbody>
</table>

**Windows Phone 7**

<table>
<thead>
<tr>
<th></th>
<th>H.264</th>
<th>MPEG-4</th>
<th>MPEG-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile</td>
<td>Baseline/Main/High</td>
<td>Simple</td>
<td>Advanced Simple</td>
</tr>
<tr>
<td>Resolution/frame rate</td>
<td>720x480 @ 30 fps, 720x576 @ 25 fps</td>
<td>800x600 (720p for HD capture devices) @ 30 fps</td>
<td>800x600 @ 30 fps</td>
</tr>
<tr>
<td>Average data rate</td>
<td>2 Mbps</td>
<td>2 Mbps</td>
<td>2 Mbps</td>
</tr>
<tr>
<td>Peak data rate</td>
<td>22 Mbps</td>
<td>22 Mbps</td>
<td>27 Mbps</td>
</tr>
<tr>
<td>Bit rate control</td>
<td>CBR/VBR</td>
<td>CBR/VBR</td>
<td>CBR/VBR</td>
</tr>
<tr>
<td>Audio codec</td>
<td>AAC-LC, HE-AAC v1, HE-AAC v2</td>
<td>AAC-LC</td>
<td>AAC-LC</td>
</tr>
<tr>
<td>Audio channels/samples</td>
<td>stereo/48 kHz</td>
<td>stereo/48 kHz</td>
<td>stereo/48 kHz</td>
</tr>
<tr>
<td>Audio bit rate</td>
<td>320 kbps</td>
<td>320 kbps</td>
<td>320 kbps</td>
</tr>
<tr>
<td>Container formats</td>
<td>.mp4, .m4v, .3gp, .3g2</td>
<td>.mp4, .m4v, .3gp, .3g2</td>
<td>.mp4, .m4v</td>
</tr>
</tbody>
</table>

- Some other codecs supported (like WMV)
- Details at bit.ly/windowsphonevidspecs

---

Copyright © 2011 Jan Ozar, All Rights Reserved
Producing WebM

- Overview
- Survey of encoding tools
- Live WebM encoding options

WebM Overview

- What is WebM
  - VP8 video codec (purchased from On2)
  - Open source Vorbis audio codec
  - WebM container format based on Matroska container
- Open source and royalty free
- Browser support:
  - Native - current versions of Chrome, Opera, FireFox
  - Via plug-in - Internet Explorer 9, Safari
Producing WebM

- Many tools, few are worthwhile
- *Streamingmedia* review - bit.ly/webmencoderreview
  - Performed 12/2010
  - Updated 4/2011 for book
- Basic workflow
  - Google encoded files for a presentation at StreamingMedia West (11/2010)
  - I encoded and compared results to Google output

Miro Video Converter

- Verdict
  - Good for experimentation but not production
- Issues
  - Can’t configure presets
  - Presets use 160 kbps/audio, which is too high
  - Serious dropped frames during encoding
Firefogg

- **Verdict**
  - Best free alternative, but has issues

- **Issues**
  - Output didn't match configuration input

Wildform Flix WebM

- **Verdict**
  - Awful (and withdrawn after the review)
Telestream Episode

- **Verdict**
  - Easy to use, fast, very good quality, but few VP8 config options

- **Issues**
  - Originally had issues playing back in Chrome; resolved in version 6.1
  - Minimal VP8 config options
    - Not a bad thing if you don’t like to tinker

Sorenson Squeeze 7

- **Verdict**
  - Fast, very good quality, highly configurable

- **Issues**
  - About 18% slower than Episode on same system for single file encoding, but can encode in parallel
Producing WebM

- Common parameters addressed by some encoders, like Squeeze, or via command line
  - Will explore those exposed by Squeeze
  - Will discuss most relevant parameters

### WebM Options - 1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Recommended</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile</td>
<td>0</td>
<td>Non-zero values makes content easier to decode. Use only with HD content targeted towards low power playback</td>
</tr>
<tr>
<td>Encoding Threads</td>
<td>Number of cores - 1</td>
<td>Lets you efficiently use multi-core system</td>
</tr>
<tr>
<td>Quality vs Speed</td>
<td>Good</td>
<td>Good/O configuration should match Best quality but will encode twice as fast</td>
</tr>
<tr>
<td>Quality/speed slider</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
### WebM Options - 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Recommended</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum quality</td>
<td>0-4</td>
<td>Allow user to set minimum and maximum quality limits. Will override data rate setting, though, so be careful</td>
</tr>
<tr>
<td>Maximum quality</td>
<td>50-63</td>
<td></td>
</tr>
<tr>
<td>Reference frames</td>
<td>Enabled</td>
<td>Use of --auto-alt-ref can substantially improve quality in many situations (though there are still a few where it may hurt).</td>
</tr>
</tbody>
</table>

### WebM Options - 3

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Recommended</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lag in frames</td>
<td>16 (Squeeze only goes to 15)</td>
<td>Relates to alt-reference frame</td>
</tr>
<tr>
<td>Buffer sizes</td>
<td>As shown</td>
<td>Recommend buffer settings</td>
</tr>
<tr>
<td>Rate Control Resizing/Drop Frames</td>
<td>Disabled</td>
<td>“These are specialized parameters and are not generally recommended”</td>
</tr>
</tbody>
</table>
WebM Options - 4

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Recommended</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering</td>
<td>As shown</td>
<td>“There are better filtering options available in specialist pre-processing products”</td>
</tr>
<tr>
<td>Static region threshold</td>
<td>As shown</td>
<td>“Any non zero value runs the risk of introducing artifacts caused by regions of the image not being updated”</td>
</tr>
<tr>
<td>Expected CPU cores for playback</td>
<td>2</td>
<td>Squeeze specific option, not WebM</td>
</tr>
<tr>
<td>Error Resilient</td>
<td>On when Alt-Ref-Frames enabled</td>
<td></td>
</tr>
</tbody>
</table>

Other WebM Encoding

- Command line options - multiple are available
  - Check [www.webmproject.org](http://www.webmproject.org) for links and command line arguments
- Live Options
  - Broadcast International - Head End
  - Entropy Wave E1000
  - Flumotion WebTV
  - AFAIK, none of the majors have live WebM encoding yet
Encoding for Adaptive Streaming

- Introduction
- Transmux strategies
- Flash
  - RTMP
  - HTTP
- HTTP Live Streaming to iOS/Android

Adaptive Streaming - Introduction

- Concept
  - Customize experience for viewer device and bandwidth
    - High power/high bandwidth – great experience
    - Lower power/low bandwidth – lesser experience, but it plays
  - Adapt to changing conditions
  - All transparent to the viewer
Major League Baseball

- Example
  - MLB offers 11 streams in subscription service
  - Intelligent player
    - Monitors CPU
    - Monitors buffer level
  - System adjusts speed to ensure optimal quality stream

Advantages of Adaptive Streaming

- Enables highest quality viewing experience
  - Can create very high quality streams because the system will shift to lower quality if required
  - Rewards high performance/high bitrate consumers while still serving those at the other end of the spectrum
Technology Overview

- Streams switched to adapt to factors like:
  - Changing delivery bandwidths (avoid hard stops)
  - CPU utilization at client (avoid frame drops)
- Information is gathered by player
  - Server-based systems (RTMP Flash) deliver a different stream when change is required, switching at key frame
  - HTTP-based systems (HTTP Flash, iOS) use 2-10 second file chunks
    - Player retrieves chunk from different source file to effectuate stream switch (more later)

Encoding for Adaptive Streaming

- How to configure streams to:
  - Optimize playback experience across all served devices and bandwidths
    - How many streams, what resolutions, what data rates
  - Work within requirements of adaptive streaming technology
    - Key frame interval, VBR vs. CBR, audio parameters
Transmuxing Technologies

- Most producers must serve two clients
  - Flash (or Silverlight)
  - iOS (and now Android)
- In the past, that meant two separate encoding and delivery workflows
- Now, multiple technologies for:
  - “Transmuxing” H.264 stream
  - Using correct protocol to distribute to target

Options
- Technology providers - Wowza, Microsoft, Adobe
- Service providers - Akamai (in the network repackaging)
- Key point:
  - If serving multiple targets, you must produce using lowest common denominator H.264 encoding parameters
Inputs for Analysis

- Multiple case studies (Harvard, Turner, MTV, NBC, Indiana University)
  - *StreamingMedia* here: bit.ly/ozeradaptive
- White papers and other guides as identified throughout
- Several consulting projects

RTMP Flash - Overview

- Server driven system
  - Uses files as encoded (no chunking)
  - Server switches as necessary at key frame
Sources

- Maxim Levkov, “Video encoding and transcoding recommendations for HTTP Dynamic Streaming on the Flash Platform,” (adobe.ly/Levkovhttp)
- “Encoding Guidelines Dynamic Streaming for Flash over HTTP,” Akamai website (bit.ly/akamaiwhitepaper)

RTMP Flash: How Many Streams?

- Considerations
  - Sufficient to cover a broad spectrum of bandwidths
  - Not so many that changing is frequent
  - Number of window sizes served
    - At least one for each relevant window size
RTMP Flash: How Many Streams?

- **Considerations**
  - Number of window sizes served

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Format</th>
<th>Frame Size</th>
<th>Total Bitrate</th>
<th>Audio Bitrate</th>
<th>bits/pixel @30 fps</th>
<th>bits/pixel @24 fps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile &amp; constrained (low)</td>
<td>baseline, mono, 10 fps</td>
<td>448x242</td>
<td>130</td>
<td>48</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>Mobile &amp; constrained (high)</td>
<td>baseline, mono</td>
<td>448x253</td>
<td>450</td>
<td>48</td>
<td>0.12</td>
<td>0.15</td>
</tr>
<tr>
<td>Sidebar placements</td>
<td>main profile, stereo</td>
<td>384x216</td>
<td>400</td>
<td>96</td>
<td>0.12</td>
<td>0.15</td>
</tr>
<tr>
<td>Small in-page</td>
<td>main profile, stereo</td>
<td>512x288</td>
<td>750</td>
<td>96</td>
<td>0.16</td>
<td>0.18</td>
</tr>
<tr>
<td>Medium in-page</td>
<td>main profile, stereo</td>
<td>640x384</td>
<td>1200</td>
<td>96</td>
<td>0.18</td>
<td>0.20</td>
</tr>
<tr>
<td>Large in-page</td>
<td>main profile, stereo</td>
<td>768x432</td>
<td>1700</td>
<td>96</td>
<td>0.18</td>
<td>0.20</td>
</tr>
<tr>
<td>Full size in-page</td>
<td>main profile, stereo</td>
<td>960x540</td>
<td>2200</td>
<td>96</td>
<td>0.14</td>
<td>0.17</td>
</tr>
<tr>
<td>HD 720p (full screen)</td>
<td>high profile, stereo</td>
<td>1280x720</td>
<td>2000</td>
<td>96</td>
<td>0.12</td>
<td>0.15</td>
</tr>
</tbody>
</table>

- **MTV Schema**
  - Have at least one for each window size served (optimum quality when stream size=window size)
  - **Never switch to quality higher than current viewing size**

- **Other considerations**
  - Most if subscription service (MLB with 11)
  - More if entertainment
    - MTV with 8, though Turner (PGA golf) with 4
  - Fewer if education or corporate
    - Indiana - 3, Harvard, 5
  - **Overall - 3 - 8 for non-subscription**
RTMP Flash: What Resolution?

- MTV - At least one for each window size
  - Don’t shift to higher stream unless window size changes
  - Never send 720p stream to 768x432 window
- Adobe white paper (right)
  - Cluster around window sizes
  - Switches least apparent w/in same window size

<table>
<thead>
<tr>
<th>Stream</th>
<th>Picture Size</th>
<th>V</th>
<th>A</th>
<th>AV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>768x432</td>
<td>250</td>
<td>64</td>
<td>214</td>
</tr>
<tr>
<td>2</td>
<td>768x432</td>
<td>250</td>
<td>64</td>
<td>214</td>
</tr>
<tr>
<td>3</td>
<td>512x288</td>
<td>450</td>
<td>64</td>
<td>514</td>
</tr>
<tr>
<td>4</td>
<td>512x288</td>
<td>450</td>
<td>64</td>
<td>514</td>
</tr>
<tr>
<td>5</td>
<td>512x288</td>
<td>450</td>
<td>64</td>
<td>514</td>
</tr>
<tr>
<td>6</td>
<td>512x288</td>
<td>1200</td>
<td>64</td>
<td>1200</td>
</tr>
<tr>
<td>7</td>
<td>768x432</td>
<td>1400</td>
<td>64</td>
<td>164</td>
</tr>
<tr>
<td>8</td>
<td>1280x720</td>
<td>1700</td>
<td>64</td>
<td>1764</td>
</tr>
<tr>
<td>9</td>
<td>1280x720</td>
<td>2500</td>
<td>64</td>
<td>2564</td>
</tr>
<tr>
<td>10</td>
<td>1280x720</td>
<td>3500</td>
<td>64</td>
<td>3564</td>
</tr>
<tr>
<td>11</td>
<td>1920x1080</td>
<td>4200</td>
<td>64</td>
<td>4264</td>
</tr>
<tr>
<td>12</td>
<td>1920x1080</td>
<td>5800</td>
<td>64</td>
<td>5864</td>
</tr>
</tbody>
</table>

bit.ly/Levkovhttp

RTMP Flash: What Resolution?

- Adobe Media Encoder Presets
  - Where the rubber meets the road for Adobe producers

<table>
<thead>
<tr>
<th>Preset</th>
<th>Resolution</th>
<th>Video bitrate (kbps)</th>
<th>Profile</th>
<th>Keyframe</th>
<th>Audio bitrate (kbps)</th>
<th>Channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone and Tablet, 3G, 16x9</td>
<td>512x288</td>
<td>300</td>
<td>Baseline</td>
<td>72</td>
<td>48</td>
<td>Stereo</td>
</tr>
<tr>
<td>Phone and Tablet, 3G, 16x9</td>
<td>512x288</td>
<td>450</td>
<td>Baseline</td>
<td>72</td>
<td>48</td>
<td>Stereo</td>
</tr>
<tr>
<td>Phone and Tablet, 3G, 16x9</td>
<td>512x288</td>
<td>650</td>
<td>Baseline</td>
<td>72</td>
<td>48</td>
<td>Stereo</td>
</tr>
<tr>
<td>PC &amp; TV, SD, Med, 16x9</td>
<td>768x432</td>
<td>1,140</td>
<td>Main</td>
<td>72</td>
<td>64</td>
<td>Stereo</td>
</tr>
<tr>
<td>PC &amp; TV, SD, High 16x9</td>
<td>768x432</td>
<td>1,740</td>
<td>Main</td>
<td>72</td>
<td>64</td>
<td>Stereo</td>
</tr>
<tr>
<td>PC &amp; TV, HD, Med, 16x9</td>
<td>1280x720</td>
<td>2,440</td>
<td>High</td>
<td>72</td>
<td>64</td>
<td>Stereo</td>
</tr>
<tr>
<td>PC &amp; TV, SD, High 16x9</td>
<td>1280x720</td>
<td>3,440</td>
<td>High</td>
<td>72</td>
<td>64</td>
<td>Stereo</td>
</tr>
</tbody>
</table>

RTMP Flash: What Resolution?

- Other considerations
  - Never encode at larger than source
    - Scaling upwards degrades quality
  - Match size to display window
    - If you have to scale during display, scale up, not down
    - Scaling down uses 40% more CPU
  - Very few producers adhere to mod-16 (where height/width are divisible by 16)
    - But all favor at least mod-8, all at least mod-4

RTMP Flash: What Data Rates?

- Considerations
  - Must be sufficiently far apart to avoid constant switching
  - Will vary with stream size
    - Smaller gaps at lower bandwidths
    - Larger gaps at higher bandwidths
RTMP Flash: What Data Rates?

- MTV Schema

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Format</th>
<th>Frame Size</th>
<th>Total Bitrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile &amp; constrained (low)</td>
<td>baseline, mono, 10 fps</td>
<td>448x256</td>
<td>150</td>
</tr>
<tr>
<td>Mobile &amp; constrained (high)</td>
<td>baseline, mono</td>
<td>448x256</td>
<td>450</td>
</tr>
<tr>
<td>Small in-page</td>
<td>main profile, stereo</td>
<td>394x216</td>
<td>400</td>
</tr>
<tr>
<td>Medium in-page</td>
<td>main profile, stereo</td>
<td>512x288</td>
<td>750</td>
</tr>
<tr>
<td>Large in-page</td>
<td>main profile, stereo</td>
<td>640x360</td>
<td>1200</td>
</tr>
<tr>
<td>Full-size in-page</td>
<td>main profile, stereo</td>
<td>768x432</td>
<td>1700</td>
</tr>
<tr>
<td>HD 720p (full screen)</td>
<td>high profile, stereo</td>
<td>960x540</td>
<td>2200</td>
</tr>
</tbody>
</table>

For Smooth Streaming, but good starting point
- Enter parameters, calculates rez/data rate for you
- Attempts to keep quality consistent at all data rates
RTMP Flash: How should H.264 encoding parameters change?

- Computer playback
  - High profile/CABAC Enabled
- Mixed computer and device
  - Streams must match lowest common denominator targets

In General - Key Frame Parameters

- Key frame interval must be consistent across all files
  - no scene change detection
- All key frames s/b IDR frames
- Intervals shorter (2-5 seconds) so stream switching is more nimble
- For chunk-based technologies, should divide equally into chunk duration
  - If eight second chunks, use 1, 2 or 4 seconds
In General - VBR vs CBR

- Live - use CBR
- On-Demand
  - CBR more conservative but can cost you quality
  - Unconstrained VBR - can cause unnecessary stream switches
    - Large packet of high bandwidth VBR that takes too long to deliver can make player think that buffer is low
  - Constrained VBR
    - Highly constrained at low data rates (1.2x)
    - Less constrained at high data rates, where difference in bandwidths is great (MTV uses 2X constrained)
    - But, CBR is safer and more conservative

VBR vs CBR

**Video bitrate**

- A
  - med chunk
- B
  - large chunk
- C
  - Small chunk

**Player buffer**

- A
  - Good delivery
- B
  - Slow delivery
- C
  - Very fast delivery

Buffer OK
- Status quo
- No change

Ruh, roh!
- Buffer too low
- Switch to lower bitrate

Live is good!
- Switch to higher bitrate
In General - Audio Parameters

- Most conservative - use same parameters for all files
  - Popping can occur if audio parameters change
  - But, doesn’t optimize experience at higher bit rates
- If you do switch audio parameters
  - Switch from stereo to mono at same per channel sampling rate and bit rate
    - From 128 kbps/44 kHz/16-bit/stereo to 64 kbps/44 kHz/16-bit/mono
  - Test to ensure no artifacts when switching streams

HTTP Flash

- How it works
- How encoding is different from RTMP
HTTP Flash - How it Works

To Boil it Down

MP4 → Packager

Media file chunks

Manifest files that tell player where to find file chunks
HTTP Flash - How it Works

- Encoding
  - Encode files as normal
  - Insert into file packager, which:
    - Creates manifest file that tells the player where to look for files at different bit rates (F4M)
    - Divides encoded files into chunks (F4F files)
    - All files uploaded to HTTP server

- Meanwhile, at the player
  - Player retrieves manifest file first
  - Player monitors bandwidth/CPU/other heuristics
  - Retrieves chunks from different locations as necessary

HTTP Flash: How Encoding is Different

- Number of files/rez/data rate – as discussed
- Key frame, bitrate control, audio, make modifications previously discussed
  - 2-5 second key frame, evenly divisible into chunk size
  - CBR or highly constrained VBR
  - Audio – same or test to make sure no artifacts

- Best sources:
  - Maxim Levkov - Adobe - very detailed recommendations
    - adobe.ly/Levkovhttp
  - Akamai - White paper: “Encoding Guidelines for Dynamic Streaming for Flash over HTTP”
    - bit.ly/akamaiwhitepaper

Streaming Learning Center
Apple HTTP Live Streaming

- Primary technology for iOS devices
  - Also supported in Android 3.0
- How it works
- How to customize your encoding

Apple HTTP Live Streaming: How it works

- Encoding
  - Encode as normal, send to segmenter
  - Files chunked, inserted into transport stream (.ts extension)
  - Manifest file (M3U8) created
  - Uploaded to server
- Client
  - Monitors heuristics
  - Changes retrieved file as necessary
Encoding for HTTP Live Streaming

- Sources

HTTP Live Streaming

<table>
<thead>
<tr>
<th>iPad</th>
<th>16:9 Aspect Ratio</th>
<th>4:3 Aspect Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions</td>
<td>Frame Rate *</td>
<td>Total Bit Rate</td>
</tr>
<tr>
<td>CELL</td>
<td>480x320</td>
<td>na</td>
</tr>
<tr>
<td>CELL</td>
<td>480x320</td>
<td>16</td>
</tr>
<tr>
<td>CELL</td>
<td>480x320</td>
<td>12 to 15</td>
</tr>
<tr>
<td>CELL</td>
<td>480x320</td>
<td>24.97</td>
</tr>
<tr>
<td>WIFI</td>
<td>640x360</td>
<td>29.97</td>
</tr>
<tr>
<td>WIFI</td>
<td>640x360</td>
<td>29.97</td>
</tr>
<tr>
<td>WIFI</td>
<td>640x360</td>
<td>29.97</td>
</tr>
<tr>
<td>WIFI</td>
<td>640x360</td>
<td>29.97</td>
</tr>
<tr>
<td>WIFI</td>
<td>480x320</td>
<td>na</td>
</tr>
<tr>
<td>WIFI</td>
<td>480x320</td>
<td>16</td>
</tr>
<tr>
<td>WIFI</td>
<td>480x320</td>
<td>12 to 15</td>
</tr>
<tr>
<td>WIFI</td>
<td>480x320</td>
<td>24.97</td>
</tr>
<tr>
<td>WIFI</td>
<td>480x320</td>
<td>24.97</td>
</tr>
<tr>
<td>WIFI</td>
<td>480x320</td>
<td>24.97</td>
</tr>
</tbody>
</table>

Bit.ly/bestpracticehttplive
HTTP Live - Encoding Parameters

- Apple gives very good guidance
- Filling in the blanks
  - Profile/level - optimize for target
    - M3U8 can be device sensitive - won’t send older iPods to iPad/iPhone 4 stream
  - VBR/CBR
    - As discussed
  - Key frame - chunked technology, optimal if key frame divides evenly into chunk duration

WebM-based Adaptive Streaming

- Nascent market;
  - Some technology providers
    - Anevia
    - Quavlive
  - HTML5 standard coming in a DASH
    - Dynamic Adaptive Streaming over HTTP (DASH)
    - [http://mpeg.chiariglione.org/working_documents.htm](http://mpeg.chiariglione.org/working_documents.htm)
  - No encoding recommendations at this point
Comparing H.264 Encoding Tools

- Test Description
- Mac, then Windows. In each:
  - Meet the participants
  - Quality
  - Performance
  - Features

Test Description

- Quality
  - SD - 640x480x29.97 @ 468 kbps video/32 kbps audio - highest quality encoding available
  - HD - 720p @ 29.97 @ 800 kbps/video/128 kbps audio - highest
  - Compared still frame and motion quality
- Encoding speed
  - Encode SD/HD Files
  - One to many (produce for adaptive)
  - Many to one (high volume encoding shop)
Mac H.264 Encoders

- Meet the participants

<table>
<thead>
<tr>
<th>Company</th>
<th>Platforms</th>
<th>Adobe Media Encoder (Mac/Windows)</th>
<th>Apple Compressor Mac Only</th>
<th>Apple Compressor with x264Encoder</th>
<th>Sorenson Squeeze</th>
<th>Telestream Episode Pro 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platforms</td>
<td>Mac/Windows</td>
<td>Adobe</td>
<td>Apple Mc only</td>
<td>My Comet 3G</td>
<td>Sorenson Media</td>
<td>Telestream</td>
</tr>
<tr>
<td>Price as tested</td>
<td>Bundle-only</td>
<td>Free</td>
<td>Free</td>
<td>$799</td>
<td>$995</td>
<td>$995</td>
</tr>
<tr>
<td>H.264 Codec</td>
<td>MainConcept</td>
<td>Apple</td>
<td>x264</td>
<td>MainConcept</td>
<td>MainConcept</td>
<td>MainConcept</td>
</tr>
<tr>
<td>Other Streaming Codecs Supported</td>
<td>VP6</td>
<td>None</td>
<td>None</td>
<td>VP6, WMV</td>
<td>VP6, WMV</td>
<td></td>
</tr>
</tbody>
</table>

Mac H.264 Encoders

- Quality in a nutshell:
  - Don’t encode in QT/Compressor using the Apple codec; Instead try the x264Encoder
Mac Quality Summary

<table>
<thead>
<tr>
<th></th>
<th>Adobe Media Encoder</th>
<th>Apple Compressor</th>
<th>Apple Compressor with x264Encoder</th>
<th>Sorenson Squeeze</th>
<th>Telestream Episode Pro 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD frame quality</td>
<td>Excellent</td>
<td>Fair</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>SD motion quality</td>
<td>Excellent</td>
<td>Good</td>
<td>Very Good</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>HD frame quality</td>
<td>Very Good</td>
<td>Poor</td>
<td>Very Good</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>HD motion quality</td>
<td>Excellent</td>
<td>Fair</td>
<td>Very Good</td>
<td>The standard by which others are judged</td>
<td>Excellent</td>
</tr>
<tr>
<td>Bottom line</td>
<td>Just a hair behind the leaders in HD quality</td>
<td>Avoid if possible</td>
<td>Very, very good, but verify compatibility</td>
<td>Version 6.1.1 brings parity with Squeeze</td>
<td>Excellent</td>
</tr>
</tbody>
</table>

- Episode 6.1 resolved all issues with H.264 encoding quality; now on par with Squeeze
- AME is slightly behind in one category - s/be fine for most jobs
- Seems like bad vintage of x264 encoder - usually on par with others
  - Use that if locked into Compressor workflow

Encoding Schemas

- Serial - one at a time
  - Adobe Media Encoder
- Parallel - multiple simultaneous encodes, usually up to number of cores in system
  - Compressor with Qmaster
- Hybrid parallel
  - Episode Pro - 2 at a time
  - Episode Engine - to number of cores
- Squeeze
  - Many to many - to number of cores (shown on right)
  - But! One to many - serial
### Mac Encoding Speed

<table>
<thead>
<tr>
<th>Encoding/Serial or Parallel</th>
<th>Adobe Media Encoder</th>
<th>Apple Compressor</th>
<th>Apple Compressor with x264Encoder</th>
<th>Sorenson Squeeze</th>
<th>Telesystem Episode Pro 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single SD file to 500 kbps</td>
<td>Serial</td>
<td>2:47</td>
<td>8:21</td>
<td>5:30</td>
<td>7:28</td>
</tr>
<tr>
<td>Single HD file to 800 kbps</td>
<td>2:10</td>
<td>4:30</td>
<td>3:00</td>
<td>3:47</td>
<td>2:14</td>
</tr>
<tr>
<td>Single HD test file to 8 files</td>
<td>15:55</td>
<td>23:36</td>
<td>34:30</td>
<td>24:30</td>
<td>15:17</td>
</tr>
<tr>
<td>8 DV files to 500 kbps</td>
<td>2:48</td>
<td>3:29</td>
<td>3:02</td>
<td>5:20</td>
<td>2:27</td>
</tr>
</tbody>
</table>

- Overall; differences relevant only to very high volume shops
- AME very fast for serial encoder
- Episode - very fast

#### Squeeze
- Slow single file time
- Encodes *multiple files* in parallel, not one
- Compressor - middle of the pack

### Mac Encoding Features

<table>
<thead>
<tr>
<th></th>
<th>Adobe Media Encoder</th>
<th>Apple Compressor</th>
<th>Apple Compressor with x264Encoder</th>
<th>Sorenson Squeeze</th>
<th>Telesystem Episode Pro 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline/Main profile selection?</td>
<td>Yes</td>
<td>Baseline/Main</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CABAC entropy encoding</td>
<td>Automatic</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>B-frame/reference frame controls</td>
<td>Automatic</td>
<td>Automatic</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Other H.264 controls</td>
<td>None</td>
<td>None</td>
<td><strong>Very extensive</strong></td>
<td>Minimal</td>
<td>Minimal</td>
</tr>
<tr>
<td>Adaptive streaming protocols</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

- **Who cares?:**
  - Those who need to customize
  - Those who like to tinker
Windows H.264 Encoders

- Meet the participants

<table>
<thead>
<tr>
<th></th>
<th>Adobe Media Encoder</th>
<th>Microsoft Expression Encoder</th>
<th>Sorenson Squeeze</th>
<th>Telestream Episode Pro 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>Adobe</td>
<td>Microsoft</td>
<td>Sorenson Media</td>
<td>Telemaster</td>
</tr>
<tr>
<td>Platforms</td>
<td>Mac/Windows</td>
<td>Windows</td>
<td>Mac/Windows</td>
<td>Mac/Windows</td>
</tr>
<tr>
<td>Price as tested</td>
<td>Bundle-only</td>
<td>$199.95</td>
<td>$799</td>
<td>$995</td>
</tr>
<tr>
<td>H.264 codec</td>
<td>MainConcept</td>
<td>MainConcept</td>
<td>MainConcept</td>
<td>MainConcept</td>
</tr>
<tr>
<td>Other streaming codecs supported</td>
<td>VP6, WMV</td>
<td>WMV</td>
<td>VP6, WMV</td>
<td>VP6, WMV</td>
</tr>
</tbody>
</table>
Windows Quality Summary

- All encoding tools use MainConcept codec, so no surprise quality is very similar
  - Episode 6.1 resolved all issues with H.264 encoding quality; now on par with Squeeze
  - AME and Expression Encoder perform well across the board
  - No meaningful quality differentiation

<table>
<thead>
<tr>
<th></th>
<th>Adobe Media Encoder</th>
<th>Microsoft Expression Encoder</th>
<th>Sorenson Squeeze</th>
<th>Telestream Episode 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD frame quality</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>SD motion quality</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>HD frame quality</td>
<td>Very Good</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>HD motion quality</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>Bottom line</td>
<td>Just a touch behind in some HD frames</td>
<td>A solid choice, particularly if you like to tinker</td>
<td>The standard by which others are judged</td>
<td>Version 6.1 brings parity with Squeeze</td>
</tr>
</tbody>
</table>

Windows Encoding Speed

- Some encoders let you open multiple instances
  - Yes - Expression Encoder, Squeeze
  - No - Adobe Media Encoder, Episode

- Useful when encoding tool encodes serially
  - Expression Encoder
Windows Encoding Speed

<table>
<thead>
<tr>
<th>Encoding: serial or parallel</th>
<th>Adobe Media Encoder</th>
<th>Microsoft Expression Encoder</th>
<th>Sorenson Squeeze</th>
<th>Telastream Episode Pro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single SD file to 500 kbps</td>
<td>1:30</td>
<td>2:34</td>
<td>3:46</td>
<td>1:21</td>
</tr>
<tr>
<td>Single HD file to 800 kbps</td>
<td>1:33</td>
<td>1:19</td>
<td>2:42</td>
<td>3:26</td>
</tr>
<tr>
<td>Single HD test file to 8 files</td>
<td>12:15</td>
<td>7:40</td>
<td>16:59</td>
<td>10:53</td>
</tr>
<tr>
<td>8 DV files to 500 kbps</td>
<td>2:16</td>
<td>6:04</td>
<td>3:23</td>
<td>2:34</td>
</tr>
<tr>
<td>8 DV files - multiple instances</td>
<td>NA</td>
<td>5:55</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

- Overall; differences relevant only to very high volume shops
- AME very fast for serial encoder
- Expression Encoder was fastest in one to many encodes, which is adaptive streaming scenario

Squeeze
- Slowest in adaptive streaming scenario
- Episode Pro
- Competitive in all tests

Windows Encoding Features

<table>
<thead>
<tr>
<th>H.264 Encoding Features</th>
<th>Adobe Media Encoder</th>
<th>Microsoft Expression Encoder</th>
<th>Sorenson Squeeze</th>
<th>Telastream Episode Pro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline/Main/High profile selection?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CABAC entropy encoding</td>
<td>Automatic</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>B-frame/reference frame controls</td>
<td>Automatic</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Other H.264 controls</td>
<td>None</td>
<td>Very extensive</td>
<td>Minimal</td>
<td>Minimal</td>
</tr>
<tr>
<td>Adaptive streaming presets</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

- Who cares?:
  - Those who need to customize
  - Those who like to tinker
Questions?

- For more information, check out the book
  - Available on Amazon
  - Some copies available today