Next Gen Open Video (NGOV) Requirements

Rationale

- Enable innovation in video compression technology at the speed of the web
- The web is built on open, vibrant technologies

Key to Priorities

0 = Critical. Can't launch without it.
1 = Important. If feature is a risk to the target launch date, requires an Eng+PM vote to defer.
2 = Optional. Nice to have but can be deferred to shorten or meet target launch date.

Core Bitstream Requirements

Priority	Area	Description
0	Quality improvement	Reduce video bitrate by 50% with image quality comparable to VP8 (SSIM, PSNR).
0	Theoretical decoding complexity	No more than 40% higher than VP8.
0	Alt-ref frames	In VP8 temporal layers use the Golden and Alt-Ref frames so that they cannot be used for boosting compression efficiency. An easy solution would be to add more alt-refs.
0	Screensharing	Reduce bitrate requirement by 80% for same quality as VP8 in screensharing apps.
0	No profiles	Maintain single bitstream profile for all use cases. Any valid NGOV decoder must be able to decode any NGOV bitstream.
1	No resolution limitations	Support infinite video resolutions.
1	Frame level parallelism	Ability to decode consecutive frames in parallel
1	Encoder latency	Support independently encodable slices (i.e., eliminate need for full-frame latency). This is even more important in resolutions greater than 1080p.

1	Encoded domain stream stitching	Support taking multiple encoded streams and re-format them into one steam without transcoding, think creating a Brady Bunch experience without transcoding.
2	Resolution independence	A single encoded stream can be used to support any resolution and bitrate. Also known as “golden stream.”

Tools Requirements

A mature toolset is essential to building a content ecosystem (post-production, playback, etc.) around the new codec.

Priority	**Area**	**Description**
0	Encoder feature parity w/ libvpx	Keep all encoding features from libvpx.
0	Deliver a separate decoder for ARM	Standalone library, designed and optimized for ARM v7 with Neon.
0	Deliver a separate encoder for ARM	Standalone library, designed and optimized for ARM.
1	Implementation decode performance (desktop)	Software decoder must be able to decode realtime 4K video on lowest-end Intel i5 Ivybridge processor on the market in Q2 2013.
1	Implementation encode performance (desktop)	Software encoder must be able to simulcast (i.e., simultaneous encode and decode) 1080p video on lowest-end Intel i5 Ivybridge processor on market in Q2 2013.
1	Implementation decode performance (mobile)	Software decoder must be able to decode 1080p video top 30% of phones in the market in Q2 2013.
1	Implementation encode performance (mobile)	Software encoder must be able to simulcast 720p video on top 30% of smartphones in the market in Q2 2013.
1	Precise rate control	We must provide encoding settings (quantizers, dropped frames, etc.) so authors can get as close as possible to the requested target bitrate.
1	Encoder autoconfigure	Determine the best encoding settings based on the source material and output use case.

1	Separate decoder for Intel	Standalone library that only does decoding, optimized for x86.
1	More rigorous testing	Encourage commercial testing companies to cover corner use cases.
2	Fast transcoder	Enable "true" transcoding from VP8 and H.264 to NGOV in 50% of time than decoding to raw and recoding.

RTC Requirements

We have identified many techniques that we can implement in NGOV to create a better realtime UX.

Priority	**Area**	**Description**
0	Signal denoising	Improve webcam video denoising in the encoder.
0	Change resolution without having to send a new key frame.	When the network parameters changes we may need to change the resolution dynamically. Today we have to generate a key frame for doing this, which shouldn't be necessary. Also see "keyframes" in Bitstream section above.
0	Better control around the quantizing in a frame	We should be able to get better and more even rate control if we could get better QP adaptation within a frame. This may be solved by additional and more efficient segments.
0	Temporal prediction of motion vectors	Extrapolate motion vectors from previous frame to predict the vectors of the current frame to improve coding efficiency.
1	Webcam sensor profiling	To help denoising effort, create a table of how the ten most popular webcams bring noise the image.

1	Add lossless compression & transmit states	This can be useful for exchanging reference buffers with an encoder at the send-side and a decoder at the receive-side, which may be useful when new participants join a conference or when switching layers to avoid affecting other participants.
1	Motion tracking	Improve the codec's ability to enable motion tracking / face detection. If we could do this using hooks in the encoder (an interface to query the encoder for useful features such as motion vectors, residuals etc,).
1	Denoising and deshaking done in encoding path	Better stabilization of image could be done if done in the encoding path (as opposed to pre- or post-processing). This is very important for mobile use cases.
1	Stream stitching	Can we enable better/faster stream stitching?
1	Split partitions into packet-size pieces	If we would decide to allow decoding with errors in the future, it would be useful to have a way to adapt partition sizes to packet sizes (~1200 bytes).