Universal Serial Bus

Device Class Definition

for

Video Devices:

H.264 Payload

Frequently Asked Questions

(FAQ)

Revision 1.00

April 26, 2011

Contributors

Ross Cutler	Microsoft Corp.
Ming-Chieh Lee	Microsoft Corp.
Stephen Cooper	Microsoft Corp.
Maribel Figuera	Microsoft Corp.
Richard Webb	Microsoft Corp.
Andrei Jefremov	Skype
Remy Zimmermann	Logitech Inc.
Chandrashekhar Rao	Logitech Inc.

USB-IF AND THE AUTHORS OF THIS SPECIFICATION EXPRESSLY DISCLAIM ALL LIABILITY FOR INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. USB-IF AND THE AUTHORS OF THIS SPECIFICATION ALSO DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT IN-FRINGE THE INTELLECTUAL PROPERTY RIGHTS OF OTHERS.

THIS SPECIFICATION IS PROVIDED "AS IS" AND WITH NO WARRANTIES, EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE. ALL WARRANTIES ARE EXPRESSLY DISCLAIMED. NO WARRAN-TY OF MERCHANTABILITY, NO WARRANTY OF NON-INFRINGEMENT, NO WARRANTY OF FIT-NESS FOR ANY PARTICULAR PURPOSE, AND NO WARRANTY ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

IN NO EVENT WILL USB-IF OR USB-IF MEMBERS BE LIABLE TO ANOTHER FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA OR ANY INCIDENTAL, CONSEQUENTIAL, INDIRECT, OR SPECIAL DAMAGES, WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THE USE OF THIS SPECIFICATION, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSI-BILITY OF SUCH DAMAGES.

All product names are trademarks, registered trademarks, or service marks of their respective owners.

AVC/H.264 Disclaimer

Any implementation of the specification described herein would require a MPEG LA AVC/H.264 Patent Portfolio license to essential patent rights for the AVC/H.264 (MPEG-4 Part 10) digital video coding standard. See http://www.MPEGLA.com.

Revision History

Version	Date	Description
0.10	July 12, 2010	Initial Draft
0.80	December 16,2010	Updated
0.81	December 17,2010	Updated
0.82	December 20,2010	Copyright updates
0.83	January 11, 2011	Added more clarifications regarding GET_DEF and GET_MAX
0.84	January 17, 2011	Updated
0.85	February 1, 2011	Updated comments , copyright
0.86	March 8, 2011	Updated with latest 0.88 specifications. Added wLayerID.
0.90	April 4, 2011	Added LTR, Framerate.
0.91	April 12,2011	Updated after Reykjavik F2F meeting
0.92	April 13,2011	Updated with Formatting and cleanup
0.93	April 15,2011	Updated with LTR comments and added bEntropyCABAC
0.94	April 22, 2011	Updated after review
0.95	April 22, 2011	Included Stephen's updates
1.00	Aprril 26, 2011	Updated for XU control change table ref.

Table of Contents

1 Introduction

1.1 Purpose

This document addresses frequently asked questions (FAQ) by developers regarding the implementation of the USB Video Class H.264 Payload specification; it covers implementation specific support for the encoding device interface and provides aid to implementers and, as such, is informative only. Should conflicts arise between this document and the normative specification, the specification shall take precedence.

1.2 Scope

This FAQ document is based on the UVC H.264 payload specification; it provides implementers supplemental background information and examples.

1.3 Related Documents

[1] USB Video Class 1.1 (http://www.usb.org/developers/devclass_docs#approved)
[2] USB_Video_Payload_Frame_Based_1.1
[3] USB_Video_Payload_Stream_Based_1.1
[4] USB_Video_Payload_MJPEG_1.1
[5] RTP Payload for H.264 (http://tools.ietf.org/html/rfc3914)
[6] ITU H.241 (http://www.itu.int/itu-t/recommendations/index.aspx?ser=H)
[7] ITU T.81 (http://www.itu.int/itu-t/recommendations/index.aspx?ser=T)
[8]The H.264/MPEG-4 AVC standard (http://www.itu.int/rec/T-REC-H.264) (referred to hereafter simply as H.264) is specified in the following document:

a. ITU-T Rec. H.264 | ISO/IEC 14496-10 Advanced video coding for generic audiovisual services. The standard is available at. Unless otherwise specified, this document refers to the edition approved by ITU-T in March 2010 (posted at the ITU-T web site link above).
b. The Scalable Video Coding (SVC) extensions to the H.264/MPEG-4 AVC standard (referred to hereafter simply as SVC) are specified in Annex G of the above document.
c. The Multiview Video Coding (MVC) extensions to the H.264/MPEG-4 AVC standard (referred to hereafter simply as MVC) are specified in Annex H of the above document.

[9] When supported, the use of SVC and simulcast of multiple streams in the context of this specification shall additionally conform to the following specification:

a. Unified Communication Specification and Interfaces for H.264/MPEG-4 AVC and SVC Encoder Implementation.
b. The specification is available at http://technet.microsoft.com/en-us/lync . Unless otherwise specified, this document refers to the edition of version 1.01 (posted at the Microsoft web site link above).

1.4 Terms and Abbreviations

Term	Definition
AVC	Advanced Video Coding (see H.264)
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAVLC	Context-based Adaptive Variable Length Coding
CBR	Constant Bit Rate
CPB	Coded Picture Buffer
DPB	Decoded Picture Buffer
H.264	ISO/IEC 14496 Part 10
IDR	Instantaneous Decoder Refresh. Intraframe with no past reference.
MB	Macroblock
MJPG	Motion JPEG. See UVC standard reference payload specification.
MPF	Multiplexed Payload Format
MVC	Multiview Video Coding
NAL	Network Abstract Layer
NALU	Network Access Layer Unit
PPS	Picture Parameter Set
QP	Quantization Parameter
SCR	Source Clock Reference
SEI	Supplemental Enhancement Information
SPS	Sequence Parameter Set
SVC	Scalable Video Coding
USB	Universal Serial Bus
UVC	USB Video Class
VBR	Variable Bit Rate
VC	Video Control
VS	Video Streaming
VUI	Video Usability Information
XU	Extension Unit
YUY2	Interleaved 16-bit YUV data. Y, U, Y, V.

2 Frequently Asked Questions (FAQ):

2.1 Reference Documents

Question: What are the documents referred to by the UVC H.264 Payload specification?

Answer: The following documents are referenced and relied upon by this specification:

Category	Document topic	Description
Base specification	Universal Serial Bus Device Class Definition for Video Devices (USB_Video_Class)	Defines the overall USB Video Class framework
USB Payload MJPEG	USB Video Payload MJPEG 1.1	Payload/container used for multiplexed streams.
USB Payload Stream	USB Video Payload Stream Based 1.1	Payload used for single stream
H.264	ITU-T H.264 specification SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS Infrastructure of audiovisual services – Coding of moving Video (T-REC-H.264-201003-I!!PDF-E.pdf)	The ITU-T specification describes the H.264 encoding/decoding standard.

Table 1: Documents

2.2 Host Software Driver

Question: Does this specification implementation require a new host software driver?

Answer: The H.264 payload specification is compatible with UVC 1.0 and 1.1 drivers and relies on proper support of the MJPG and/or Stream Based payload format support. The new controls are defined using standard UVC extension unit controls.

2.3 Multiplexed Format

Question: Why is the Multiplexed Payload Format required?

Answer: The Multiplexed Payload Format is specified to support the preview functionality for the application on a single pin. This format uses the standard MJPEG payload and uses standard way of adding application segment.

2.4 Probe & Commit

Question: How is UVC Probe and Commit different than UVCX_VIDEO_CONFIG_PROBE and UVCX_VIDEO_CONFIG_COMMIT?

Answer: UVCX_VIDEO_CONFIG_PROBE and UVCX_VIDEO_CONFIG_COMMIT are part of extension unit (XU). These are used for negotiations (UVCX_VIDEO_CONFIG_PROBE) and setting (UVCX_VIDEO_CONFIG_COMMIT) the negotiated parameters. The application still has to continue with the UVC probe and commit.

2.5 Multiple Streams

Question: How do you configure multiple streams on a single pin?

Answer: The bStreamMuxOption parameters has bit mask as below to configure the different stream.

Bit 0: Enable/Disable auxiliary stream
0: auxiliary stream disabled. Bits 1-7 ignored.
1: auxiliary stream enabled. PROBE/COMMIT fields apply to streams indicated by bits 1-7.
Bit 1: Embed H.264 auxiliary stream.
bStreamID identifies the simulcast stream to be configured.
Bit 2: Embed YUY2 auxiliary stream.
Bit 3: Embed NV12 auxiliary stream

Bit 4…5: Reserved

Bit 6: Container MJPG, The payload may not contain usable MJPEG data.

Bit 7: Reserved

The Default bStreamMuxOption shall be set to 0. The configuration shall be reset to 0 with the stream reset.

2.6 AV Synchronization

Question: How does this specification help the AV synchronization issue?

Answer: The specification defines the AV synchronization requirement from the driver implementation and defines it in Appendix-C.

2.7 Dynamic Controls

Question: Why are the dynamic XU controls required?

Answer: The dynamic tables are used to configure or get the stream parameters, while streaming.

The XU controls parameters are following:

Control Selector	Value	Comments
UVCX_RATE_CONTROL_MODE	0x03	Configuration of the encoder in bitrate/quality mode.
UVCX_TEMPORAL_SCALE_MODE	0x04	Number of layers
UVCX_SPATIAL_SCALE_MODE	0x05	Setting the spatial mode
UVCX_SNR_SCALE_MODE	0x06	Setting quality mode
UVCX_LTR_BUFFER_SIZE_CONTROL	0x07	LTR Buffer usage
UVCX_LTR_PICTURE_CONTROL	0x08	LTR Control
UVCX_PICTURE_TYPE_CONTROL	0x09	I , IDR frame requests
UVCX_VERSION	0x0A	Spec. version supported from the device
UVCX_FRAMERATE_CONFIG	0x0C	Dynamic frame rate configuration
UVCX_VIDEO_ADVANCE_CONFIG	0x0D	Configuration for level_idc
UVCX_BITRATE_LAYERS	0x0E	Bitrate per layer
UVCX_QP_STEPS_LAYERS	0x0F	Minimum/Maximum QP Configuration per layers

Table 2: Dynamic XU control

2.8 Delay Parameters

Question: What do the two delay parameters wEstimatedVideoDelay and wEstimatedMaxConfigDelay indicate?

Answer:

wEstimatedVideoDelay:

The device provides estimated delay pertaining to complete data pipeline, which is from end of exposure to transmit from the device.

wEstimatedMaxConfigDelay:

The device provides estimated time taken to change the configuration.

2.9 Time Stamp

Question: What is the function of bTimestamp?

Answer: The bTimestamp parameter enables buffering period and picture timing SEI messages.

2.10 View

Question: What is the function of bView?

Answer: The bView parameter is used for Multiview and Stereo Profiles. This parameter allows the host to configure each view. The host can send UVCX_VIDEO_CONFIG_PROBE GET_MAX to get the max number of views supported by the device.

2.11 MJPG Container

Question: What is the function Bit 6 Container MJPG of bStreamMuxOption?

Answer: Bit 6 indicates that the device will use MJPG as a container only; no useful imaging data will be present in the MJPG payload and shall be ignored.

2.12 Bitrate layer control

Question: How does UVCX_BITRATE_LAYERS control works?

Answer: The host can get and change the parameters as described in following steps:

- To get current values of dwPeakBitrate and dwAverageBitrate parameters setting of any specific wLayerID.
  SET_CUR: The host sets the wLayerID as required and dwPeakBitrate = dwAverageBitrate = 0. This indicates to the device that the next GET_CUR is for this specific wLayerID.
  GET_CUR: This will provide the currently set values for dwPeakBitrate and dwAverage Bitrate values for the specific wLayerID Used in previous SET_CUR.
- To set the values of dwPeakBitrate and dwAverageBitrate.
  SET_CUR: The host sets the wLayerID as required and dwPeakBitrate = desired valid value and dwAverageBitrate = desired valid value.

2.13 Simulcast

Question: How is Simulcast configured?

Answer: The host shall set bit 1 of the parameter bStreamMuxOption to access the bStreamID. The StreamID has 3 bits (bits 2-0) to support 7 streams (0-6). A value of 7 shall be used to simultaneously refer to all streams.

2.14 Negotiation Failure

Question: How can a device report a Probe/Commit negotiation failure?

Answer: In case a device cannot generate a valid configuration in response to Probe/Commit negotiation, it must set the wWidth and wHeight fields to zero.

2.15 Device Configuration

Question: Which is faster and the easier way to configure the device?

Answer: The configuration of the device can be done using following examples.

a. The host can start with GET_DEF probe. The device will return the optimized device supported parameters. The Host will update/negotiate the parameters required to be changed.

b. The host can start with GET_MAX probe. The device will return the maximum supported capabilities parameters. The Host will update required parameters during subsequent Probe/Commit negotiation.

c. The host can configure minimum parameter and send it to device. The device will provide the closest possible configuration based on its capabilities and the host provided parameters.

2.16 Multiplexed Streams and FrameInterval

Question: How are different frame intervals handled for multiplexed streams?

Answer: Frame intervals for multiplexed streams may be different. The host shall check the device's supported and returned Probe/Commit configurations. The UVCX Probe/Commit FrameInterval (dwFrameInterval) will be dependent on the exposed UVC Probe/Commit dwFrameInterval.

2.17 Parameters Reset

Question: How can host reset the configuration of static and dynamic parameters?

Answer: The UVCX_ENCODER_RESET control is used to reset the default configuration of the device; it can be used for specific StreamID.

2.18 LTR Buffer Size control use cases

Question: What are the use cases for the UVCX_LTR_BUFFER_SIZE_CONTROL XU control?

Answer: Use cases:

1) A single LTR frame can be useful to perform periodic long term refreshes to erase drift caused by decoder-side unreported error concealment in non-LTR frames. But this case is not tolerant to errors in the new LTR frame (update (A) using A).
2) Two LTR frames are useful to further protect against transmission errors in one LTR frame(update(A) using B, if the update fails, then B is available to use and host can request a NEW LTR update of A using B).
3) More LTR frames may be useful to robustly transmit an INTRA frame and then late-binding that into an IDR by invalidating preceding LTRs which precede it.

2.19 LTR Buffer Size control behavior

Question: What should be the UVCX_LTR_BUFFER_SIZE_CONTROL XU control device behavior?

Answer: The Device should support following's to get UVCX_LTR_BUFFER_SIZE_CONTROL functionalities.

1. This Command must support GET_MAX in addition to GET_CUR, and SET_CUR. GET_MAX returns the maximum values for bLTRBufferSize and bLTREncoderControl. A targeted decoder might not be able to support these maximum values, and so the host will request smaller values.
2. In case encoder doesn't support host controlled memory operations GET_MAX for bLTRBufferSize should return 0.
3. The host shall do GET_MAX after each change in resolution or level parameter change. The device shall provide updated information. i.e. it could happen that due to memory limitations the encoder would no longer be able to provide any LTR frames after a change to a higher resolution. Yet would be able to provide such functionality again after a change to a lower one.
4. If the device allows the host to manage the LTR buffers, it shall assign continuous index space starting from 0. For example if the host is allowed to control 2 positions in long term buffer that means those are LTR frames 0 and 1. The number of buffers available for host control is bTRLBufferSize - bLTREncoderControl.
5. The encoder is responsible for signaling appropriate Decoder picture buffer parameters in SPS. It shall make sure that buffer size stays within the limits given the assigned level. The device may generate IDR if necessary.
6. If bLTRBufferSize>0 then an IDR frame shall update index 0 in LTR buffers. It is recommended for both host initiated and encoder initiated updates.
7. A subset of the LTR buffers may be allocated to the encoder to boost coding efficiency. The number of buffers available to the encoder is bLTREncoderControl.
8. SET_CUR should specify values of the parameters honoring the MAX limitations, with an additional limitation that bLTREncoderControl must be less or equal to than bLTRBufferSize.

2.20 LTR Picture Control Behavior

Question: What should be the device behavior for XU control UVCX_ LTR_PICTURE_CONTROL?

Answer: The device behavior for UVCX_ LTR_PICTURE_CONTROL should be as follows:

a. The encoder is not required to utilize all (or any) the frames in the LTR buffer unless explicitly asked to (using bEncodeUsingLTR bitmap). The encoder processing power limitation could force encoder to use only one frame as a reference.
b. Free Choice Mode: mode of initial operation of the encoder between the first IDR frame and when the first UVCX_LTR_PICTURE_CONTROL with bEncodeUsingLTR>0 is received by encoder. Encoder may use one, some or all frames from decoded picture buffer in Free Choice Mode other than the ones under host control (0 through bLTRBufferSize – bLTREncoderControl -1).
c. Limited Choice Mode: mode of operation of the encoder after reception of a UVCX_LTR_PICTURE_CONTROL with bEncodeUsingLTR>0. Note, once Encoder has entered a Limited Choice Mode it expected to remain in such mode until a new IDR frame is generated.

d. Once a command with bEncodeUsingLTR > 0 is executed at frame N. Encoder shall not have a free choice of frames to use as references (Limited Choice Mode). For encoding frames N+1 and future the following rules apply

   I. It shall NOT use frames from short term reference buffer older than N (N, N+1 etc are usable. N-1, N-2 etc are not usable)

   II. It shall NOT use any frames from LTR buffer other than the set described by most recent bEncodeUsingLTR and it applies to the encoder controlled portion of LTR buffer as well.

   III. LTR frames updated after frame N was encoded can be used as reference (similar to #I case)

   IV. Encoder is free to update own portion of LTR buffer with newer frames and use those in future encoding.

e. Graphical representation of #d

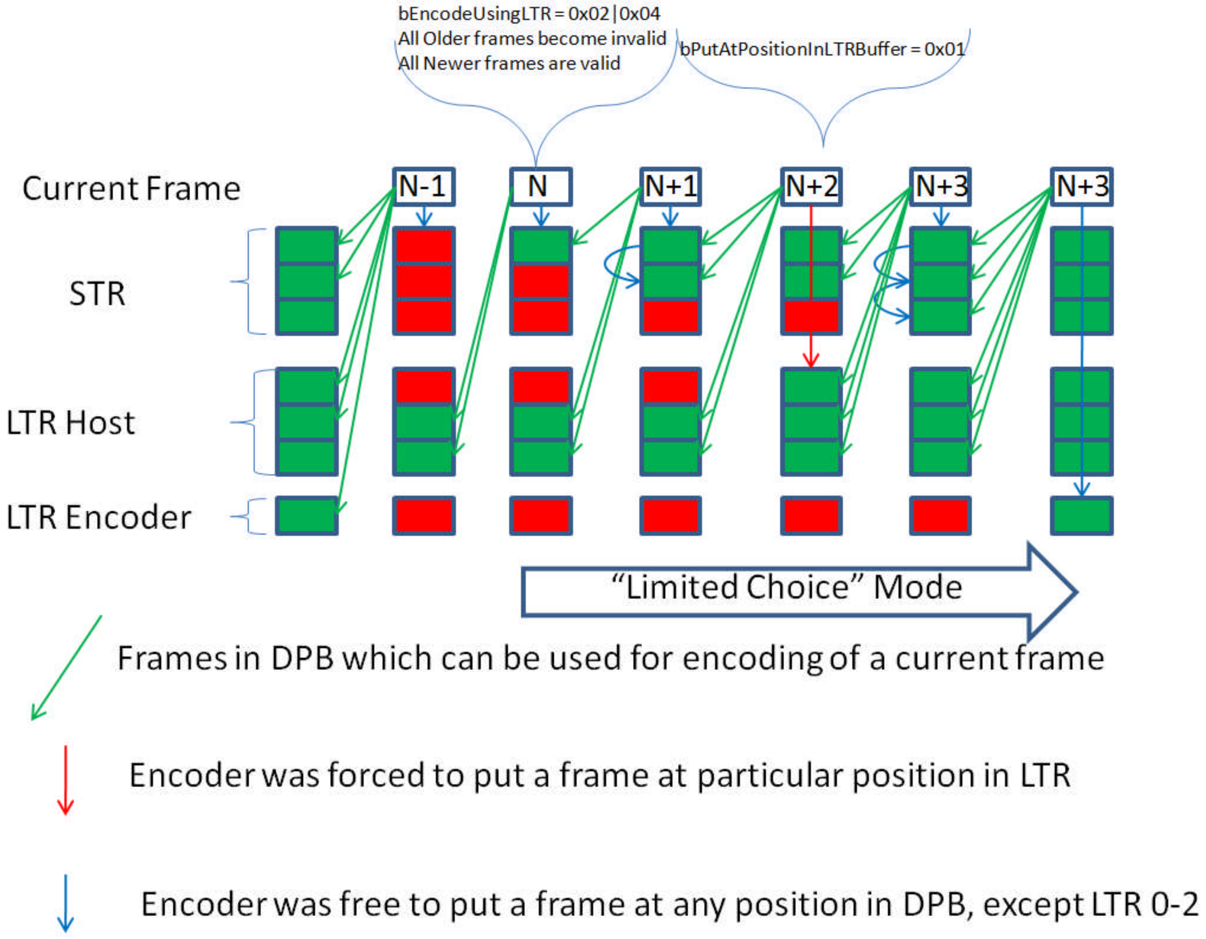
bEncodeUsingLTR = 0x02|0x04
All Older frames become invalid
All Newer frames are valid
bPutAtPositionInLTRBuffer = 0x01
Current Frame
N-1
N
N+1
N+2
N+3
N+3
STR
LTR Host
LTR Encoder
"Limited Choice" Mode
Frames in DPB which can be used for encoding of a current frame
Encoder was forced to put a frame at particular position in LTR
Encoder was free to put a frame at any position in DPB, except LTR 0-2

Figure 1: Device LTR Behavior

f. It is expected that in case UVCX_ LTR_PICTURE_ CONTROL with bEncodeUsingLTR>0 is executed then in order to improve coding efficiency and network control logic:
   I. Reference Picture Re-Ordering command is inserted to slice header by the encoder with frames actively used for encoding moved at beginning of the list. The semantics of a command is described in “7.4.3.1 Reference picture list modification semantics“ in H.264 standard.
   II. The actual number of active reference frames signaled via num_ref_idx_l0_active_minus1 as described in “7.4.3 Slice header semantics” in H.264 standard.

g. bPutAtPositionInLTRBuffer is in range 1 to N are assigned to be controlled by user.

h. bPutAtPositionInLTRBuffer = 0x0 means that encoder has freedom to where to save the frame (save in short term buffer, its own section of LTR i.e. with index N through bLTRBufferSize-1).

i. bEncodeUsingLTR specifies that only a specific subset of long term reference frames of all possible frames in decoded picture buffer can be used for encoding a next frame. Short term reference frames shall not be used by encoder for encoding the current frame.

j. Encoder has full freedom which subset of full set of frames from bEncodeUsingLTR to use as reference. Processing limitations might force encoder to use only 1 frame out of the set. That is a perfectly legal way of operation. It is also legal for the encoder to produce an IDR frame which references no other frames.

k. Both bPutAtPositionInLTRBuffer and bEncodeUsingLTR may be non zero simultaneously. That indicates that frame must be encoded using a specific subset of LTR frames (or as IDR) AND the result should be put at LTR buffer location bPutAtPositionInLTRBuffer (which must be 0 if IDR).

l. IDR frame shall always update position 0 in LTR Buffer.

m. Only one command should be executed per frame. If more than 1 command is received by encoder, it should execute only the first one received on the current frame. Any second command should be executed on the next frame. However the following optimizations are possible:
   I. Command Merge. Due to independent nature of bPutAtPositionInLTRBuffer and bEncodeUsingLTR if encoder receives two commands where in one it is asked to put the result into LTR buffer only (bPutAtPositionInLTRBuffer>0 and bEncodeUsingLTR=0) but in another it is asked to limit the set of possible references but not save frame in LTR buffer (bPutAtPositionInLTRBuffer = 0 but bEncodeUsingLTR>0). Both can be executed for the same frame.
   II. Command Merge. Two (or more) commands with bPutAtPositionInLTRBuffer with identical values can be merged in case of
      a. bEncodeUsingLTR are identical. Then only one command needs to be executed. All others can be skipped
      b. If bEncodeUsingLTR are all zero except in one case. Only the command with non zero bEncodeUsingLTR and accompanied bPutAtPositionInLTRBuffer needs to be executed

2.21 Layer ID

Question: What is wLayerID? How do I use it?

Answer: wLayerID combines the H.264 SVC parameters of quality_id, dependency_id, and temporal_id with the Simulcast parameter of stream_id into a single value. In this way wLayerID allows the host to configure individual SVC layers inside specific Simulcast streams. The host can configure individual layers or all SVC layers at the same time. The details of wLayerID are described below:

wLayerID structure:

wLayerID				
Reserved (3 bits)	Stream ID (3 bits)	Quality ID (3 bits)	Dependency ID (4 bits)	Temporal ID (3 bits)
15 13	12 10	9 7	6 3	2 0

StreamID:

The StreamID provides specification of a specific H.264 stream in the case of a simulcast sequence. The StreamID has 3 bits (bits 12-10 in wLayerID) to support 7 streams (0-6). A value of 7 shall be used to simultaneously refer to all streams. In the case of a single H.264 stream, stream_id is always 0. Non-zero StreamID only appears in cases of simulcast of two or more H.264 streams.

QualityID:

The QualityID provides specification of a specific Quality layer in a multi-layer SVC stream. The QualityID has 3 bits (bits 9-7 in wLayerID) to support 7 Quality layers (0 enhancements – 6 enhancements layers). A value of 7 shall be used to simultaneously refer to all quality layers. In the case of a single-layer H.264 stream, QualityID shall always be 0. In the case of a SVC stream not using MGS mode SNR scalability, QualityID shall always be 0. A non-zero QualityID shall only appear in SVC streams using MGS mode SNR scalability where 1 indicates the first quality enhancement layer, up to the maximum quality Enhancement layer. The MSG mode of SNR scalability partitions transform coefficients into separate Quality layers.

DependencyID:

The DependencyID provides specification of a specific Dependency Layer in a multi-layer SVC stream. The DependencyID has 4 bits (bits 6-3 in wLayerID) to support 15 dependency layers (0 enhancements – 14 enhancements layers). A value of 15 shall be used to simultaneously refer to all Dependency layers. In the case of a single-layer H.264 stream, DependencyID shall always be 0. In the case of a SVC stream not using either CGS mode SNR scalability or Spatial scalability mode, DependencyID shall always be 0. A non-zero DependencyID shall only appear in SVC streams using either CGS mode SNR scalability or Spatial scalability where 1 indicates the first SNR or spatial enhancement layer, up to the maximum SNR or spatial Enhancement layer defined as the sum of bSpatialScaleMode and the number of CGS mode SNR scalable enhancement layers identified in table 8 of the specification.

TemporalID:

The TemporalID provides specification of a specific Temporal Layer in a multi-layer SVC stream. The TemporalID has 3 bits (bits 2-0 in wLayerID) to support 7 temporal layers (0 enhancements – 6 enhancements layers). A value of 7 shall be used to simultaneously refer to all temporal layers. In the case of a single-layer H.264 stream, TemporalID shall always be 0. In the case of a SVC stream not using temporal scalability, TemporalID shall always be 0. A non-zero TemporalID shall only appear in SVC streams using temporal scalability where 1 indicates the first temporal enhancement layer, up to the maximum temporal Enhancement layer bTemporalScaleMode set in the UVCX_TEMPORAL_SCALE_MODE control.

Reserved:

The Reserved field has 3 bits (bits 15-13 in wLayerID) and shall always be 0.

2.22 Hints

Question: How do the host and device use bmHints during configuration?

Answer: The bmHints field indicates the host application's preference to "lock" some of the parameters in the PROBE/COMMIT structure. Those parameters with their corresponding bmHints bit set to 0 are considered free for the device to set. If the device cannot generate a valid configuration after considering adjusting just these parameters, then it shall consider adjusting the remaining parameters (those with the bmHins bit set to 1) in the order from lowest priority (I FramePeriod) to highest priority (Resolution). If the device still cannot generate a valid configuration after considering adjusting any of these parameters, then it must set wWidth and wHeight to zero.

The bmHints are defined as follows with Resolution being the highest priority, and priority decreasing as the associated value increases.

0x0001:	Resolution (wHeight and wWidth)
0x0002:	Profile (wProfile)
0x0004:	Rate Control Mode (bRateControlMode)
0x0008:	Usage Type (bUsageType)
0x0010:	Slice Mode (wSliceMode)
0x0020:	Slice Unit (wSliceUnits)
0x0040:	MVC View (bView)
0x0080:	Temporal (bTemporalScaleMode)
0x0100:	SNR (bSNRScaleMode)
0x0200:	Spatial (bSpatialScaleMode)
0x0400:	Spatial Layer Ratio (bSpatialLayerRatio)
0x0800:	Frame interval (dwFrameInterval)
0x1000:	Leaky Bucket Size (wLeakyBucketSize)
0x2000:	Bit Rate (dwBitRate)

0x4000: Entropy CABAC (bEntropyCABAC)
0x8000: I FramePeriod (wIFramePeriod)

2.23 bEntropyCABAC

Question: What is baseline constrained plus CABAC?

Answer: The UVCX_VIDEO_CONFIG_PROBE/COMMIT control allows the host to specify the profile as Baseline constrained while also setting bEntropyCABAC to true. Since CABAC is not a feature in baseline this format is not compliant with the H.264 specification. In this case the device should mark the stream as Main profile while staying within the feature set of Baseline constrained for all features other than CABAC.