Microsoft Proposed Extensions to the USB Video Class for H.264

Version 1.0

December 9, 2011

Abstract

This document contains the Microsoft proposal to update the USB Video Class in order to support a USB device that encodes video from a capture source in a manner compliant with the H.264 specification. The proposed specification addresses all aspects of the USB Video Class: discovery, negotiation, control, and streaming.

The proposal described in this document is not fully implemented in Windows 8, and sections that are not currently supported are marked as such.

Future updates to the proposal described in this document will reside at <http://www.usb.org/developers/devclass_docs/>:

* Updates to the H.264 Video Payload Header, H.264 Video Format Descriptor and H.264 Video Frame Descriptor will all reside in a new H.264 Payload document.
* The proposed changes to Probe & Commit and the new Encoder Controls will be applied to the USB\_Video\_Class\_1.1 specification.
* The Examples will be added to the USB\_Video\_Examples\_1.1 document.

This information applies to the following operating systems:  
 Windows Developer Preview

Windows 8 Beta

The current version of this specification is maintained on the web at:   
 [Microsoft Proposed Extensions to the USB Video Class for H.264](http://www.microsoft.com/fwlink/?LinkId=233063)

***Disclaimer****: This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.*

*This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes*

© 2011 Microsoft Corporation. All rights reserved.

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Change |  |  |  |
| December 7, 2011 | First publication | | | |

Contents

[1. Normative References 4](#_Toc309376234)

[2. Terminology 4](#_Toc309376235)

[2.1. Abbreviations 4](#_Toc309376236)

[2.2. Definition 5](#_Toc309376237)

[3. Overview 5](#_Toc309376238)

[4. H.264 Payload 6](#_Toc309376239)

[4.1. H.264 Payload Header 6](#_Toc309376240)

[4.2. H.264 Payload Data 9](#_Toc309376242)

[5. H.264 Video Format Descriptor 10](#_Toc309376243)

[6. H.264 Video Frame Descriptor 15](#_Toc309376244)

[7. H.264 Probe/Commit Control 18](#_Toc309376245)

[8. H.264 Encoding Units 23](#_Toc309376246)

[8.1. H.264 Encoding Unit Descriptor 24](#_Toc309376247)

[8.2. Encoding Units Operational Model 26](#_Toc309376248)

[8.2.1. Device States for Probe, Commit, and Encoding Units 27](#_Toc309376249)

[8.2.2. Encoding Unit Requests 29](#_Toc309376250)

[8.2.3. Stream Negotiatiation Examples 30](#_Toc309376251)

[8.2.4. Setting Resolution and Frame Interval 32](#_Toc309376252)

[8.2.5. Wildcard Masks 33](#_Toc309376253)

[8.3. Select Layer Control 33](#_Toc309376254)

[8.4. Video Resolution Control 34](#_Toc309376255)

[8.5. Profile and Toolset Control [Not Supported in Windows 8] 35](#_Toc309376256)

[8.6. Minimum Frame Interval Control 37](#_Toc309376257)

[8.7. Slice Mode Control [Not Supported in Windows 8] 38](#_Toc309376258)

[8.8. Rate Control Mode Control 39](#_Toc309376259)

[8.8.1. Variable Bit Rate (VBR) 41](#_Toc309376260)

[8.8.2. Constant Bit Rate (CBR) 42](#_Toc309376261)

[8.8.3. Constant QP Mode 42](#_Toc309376262)

[8.8.4. Global VBR 42](#_Toc309376263)

[8.8.5. Low Delay and Non-Low Delay Modes 43](#_Toc309376264)

[8.9. Average Bitrate Control 43](#_Toc309376265)

[8.10. CPB Size Control 44](#_Toc309376266)

[8.11. Peak Bit Rate Control [Not Supported in Windows 8] 45](#_Toc309376267)

[8.12. Quantization Parameter Control 45](#_Toc309376268)

[8.12.1. Quantization Weighting Matrices 47](#_Toc309376269)

[8.13. Synchronization and Long-Term Reference Frame Control 47](#_Toc309376270)

[8.14. Priority ID Control [Not Supported in Windows 8] 49](#_Toc309376271)

[8.15. Start or Stop Layer Control [Not Supported in Windows 8] 50](#_Toc309376272)

[9. Request Error Code Control 51](#_Toc309376273)

[10. SVC and Simulcast Support [Partially Supported in Windows 8] 52](#_Toc309376274)

[10.1. SVC Overview 52](#_Toc309376275)

[10.2. SVC Capability Advertisement 53](#_Toc309376276)

[10.3. SVC Stream/Layer Configuration 54](#_Toc309376277)

[10.3.1. Initialization 54](#_Toc309376278)

[10.3.2. Configuration Constraint 57](#_Toc309376279)

[10.3.3. Initialization and Run-Time Encoding Control 58](#_Toc309376280)

[10.3.4. Sub-Bitstream Definition 58](#_Toc309376281)

[11. MVC and Simulcast Support [Not Supported in Windows 8] 59](#_Toc309376282)

[11.1. MVC Overview 59](#_Toc309376283)

[11.2. MVC Capability Advertisement 59](#_Toc309376284)

[11.3. MVC Stream/View Configuration 60](#_Toc309376285)

[11.3.1. Initialization 60](#_Toc309376286)

[11.3.2. Configuration Constraint 61](#_Toc309376287)

[11.3.3. Initialization and Run-Time Encoding Control 61](#_Toc309376288)

[12. Technical decisions (will be moved to a FAQ at some point) 67](#_Toc309376289)

[13. Driver Behavior Notes 61](#_Toc309376290)

[14. Overview 71](#_Toc309376291)

[15. Payload 71](#_Toc309376292)

[16. Format Descriptor 72](#_Toc309376293)

[17. Frame Descriptor 74](#_Toc309376294)

[17.1. Constrained Baseline 720p 74](#_Toc309376295)

[17.2. UC Constrained High 720p 74](#_Toc309376296)

[17.3. Constrained Baseline 360p 75](#_Toc309376297)

[18. 76](#_Toc309376298)

[19. Probe and Commit 76](#_Toc309376299)

[19.1. Negotiating 360p, 15 fps H.264 for Chat 76](#_Toc309376300)

[19.2. Negotiating 720p, 30 fps H.264 for Sharing 77](#_Toc309376301)

[20. Encoding Units 77](#_Toc309376302)

[20.1. Example 1 78](#_Toc309376303)

[20.1.1. Average Bit Rate and CPB Size 78](#_Toc309376304)

[20.1.2. New Sync Frame 79](#_Toc309376305)

[20.1.3. Constant QP Mode 79](#_Toc309376306)

[20.2. Example 2 80](#_Toc309376307)

[20.2.1. Select Layer 80](#_Toc309376308)

# Normative References

1. The H.264/MPEG-4 AVC standard (referred to hereinafter as H.264) is specified in the following document:
   1. *ITU-T Rec. H.264 | ISO/IEC 14496-10 Advanced video coding for generic audiovisual services*. The standard is available at <http://www.itu.int/rec/T-REC-H.264>. Unless otherwise specified, this document refers to the edition approved by ITU-T in March 2010 (posted at the ITU-T website link above).
   2. The Scalable Video Coding (SVC) extensions to the H.264/MPEG-4 AVC standard (referred to hereinafter as SVC) are specified in Annex G of the above document.
   3. The Multiview Video Coding (MVC) extensions to the H.264/MPEG-4 AVC standard (referred to hereinafter as MVC) are specified in Annex H of the above document.
2. When supported, the use of SVC and simulcast of multiple streams in the context of this specification shall additionally conform to the following specification:
   1. *Unified Communication Specification for H.264/MPEG-4 AVC and SVC Encoder Implementation*. Also known as the “UC H.264 Config Spec”. This specification is available at <http://technet.microsoft.com/lync/gg278176.aspx>. Unless otherwise specified, this document refers to the edition of version 1.1 published in April 2011 (posted at the Microsoft web site link above).
3. USB Video Class (UVC) 1.1 specification, <http://www.usb.org/developers/devclass_docs#approved>.
4. RFC 2119: Key words for use in RFCs to Indicate Requirement Levels   
   http://tools.ietf.org/html/rfc2119

# Terminology

## Abbreviations

For the purposes of this specification, the following abbreviations apply:

AU Access Unit (A single decoded video frame)

CABAC Context- Adaptive Binary Arithmetic Coding

CAVLC Context- Adaptive Variable-Length Coding

CGS Coarse-Grained Scalability

EOS End of Slice

FID Frame Identifier

HRD Hypothetical Reference Decoder

IDR Instantaneous Decoding Refresh

MB Macroblock

MGS Medium-Grained Scalability

MVC Multiview Video Coding

NAL Network Abstraction Layer

POC Picture Order Count

PPS Picture Parameter Set

PTS Presentation Time Stamp

QP Quantization Parameter

SCP Start Code Prefix

SCR Source Clock Reference

SEI Supplemental Enhancement Information

SOF Start of Frame

SPS Sequence Parameter Set

SVC Scalable Video Coding

## Definition

For the purposes of this specification, the following definitions apply:

Bitstream A sequence of bits that forms a representation of a NAL unit stream.

NAL unit (NALU) An H.264/MPEG-4 syntax structure containing a one-byte header and the payload byte string.

Reference frame A frame that may be used for inter prediction in the decoding process of subsequent frame(s) in decoding order.

Simulcast streams Multiple concurrent, independently coded bitstreams from the same source, interleaved according to the UC H.264 Config Spec.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

# Overview

Updates to the current UVC 1.1 specification are required in order to support configuring an H.264/MPEG-4 encoder and transmission of encoded H.264/MPEG-4 streams over USB. Theses updates include:

* New Video Payload Header for H.264
* New Video Format Descriptor for H.264
* New Video Frame Descriptor for H.264
* Updates to the Probe/Commit Control to support temporal encoded streams
* New H.264 Encoding Units

# H.264 Payload

## H.264 Payload Header

Every payload transfer containing H.264 video data must start with a payload header. The format of the payload header is defined as follows.

Table 4-1 Header Format for H.264 Streams

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| HLE | Header Length | | | | | | | |
| BFH[0] | EOH | ERR | STI | EOS | SCR | PTS | EOF | FID |
| PTS | PTS[7:0] | | | | | | | |
|  | PTS[15:8] | | | | | | | |
|  | PTS[23:16] | | | | | | | |
|  | PTS[31:24] | | | | | | | |
| SCR | SCR[7:0] | | | | | | | |
|  | SCR[15:8] | | | | | | | |
|  | SCR[23:16] | | | | | | | |
|  | SCR[31:24] | | | | | | | |
|  | SCR[39:32] | | | | | | | |
|  | SCR[47:40] | | | | | | | |

Table 4-2 Format of the Payload Header

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Offset** | **Field** | **Size** | **Value** | **Description** |
| 0 | **bHeaderLength** | 1 | Number | Header Length (HLE) field. Specifies the length of the payload header in bytes, including this field. |
| 1 | **bmHeaderInfo** | 1 | Bitmap | Bit Field Header (BFH[0]) field. Provides information on the sample data following the header, as well as the availability of optional header fields in this header.  D0: **Frame ID** (FID). This bit toggles at each H.264 Access Unit (AU) start boundary and stays constant for the rest of the AU.  D1: **End of Frame** (EOF). This bit indicates the end of an H.264 Access Unit and must be set to 1 only in the last payload transfer belonging to an Access Unit.  D2: **Presentation Time Stamp** (PTS). This bit must be set to 1 for each payload header that includes **dwPresentationTime** data.  D3: **Source Clock Reference** (SCR). This bit must be set to 1 for each payload transfer that includes **dwSourceClock** data.  D4: **End of Slice** (EOS). An H.264 frame may consist of several slices. This bit, when set, indicates the end of an H.264 Slice NAL Unit and must be set to 1 only in the last payload transfer belonging to a slice.  D5: **Still Image** (STI). This bit, when set, indicates the payload transfer contains data that belongs to an IDR slice.  D6: **Error** (ERR). This bit is set if there was an error in the video or still image transmission for this payload. The Stream Error Code control would reflect the cause of the error.  D7: **End of header** (EOH). This bit, when set, indicates the end of the BFH fields. |
| 2 | **dwPresentationTime** | 4 | Number | **Presentation Time Stamp** (PTS).  The source clock time, in native device clock units, when the raw frame capture begins. This field must be present for every payload transfer. Payload transfers generated from a single capture time must have the same PTS. The PTS is in the same units as specified in the **dwClockFrequency** field of the Video Probe Control response. |
| 6 | **scrSourceClock** | 6 | Number | A two-part **Source Clock Reference** (SCR) value.  This field must be present for each payload transfer and must be the same for all payload transfers within the same video frame.  The use of SCR is redefined in this specification, putting constraints on SCR that are compatible with the UVC 1.1 specification:   * SCR must be captured for SOF when the first video data of a video frame is put on the USB bus. * SCR must remain constant for all payload transfers within a single AU.   D31..D0: Source Time Clock in native device clock units.  D42..D32: 1-KHz SOF token counter.  D47..D43: Reserved. Set to zero.  The least-significant 32 bits (D31..D0) contain clock values sampled from the System Time Clock (STC) at the source. The clock resolution shall be specified by the **dwClockFrequency** field of the Probe and Commit response of the device. This value shall comply with the associated stream payload specification.  The times at which the STC is sampled must be correlated with the USB Bus Clock. To that end, the next most-significant 11 bits of the SCR (D42..D32) contain a 1-KHz SOF counter, representing the frame number at the time the STC was sampled. The STC is sampled when the first video data of a video frame is put on the USB bus. The SOF counter is the same size and frequency as the frame number associated with USB SOF tokens; it is required to match the current frame number.  The most-significant five bits (D47..D43) are reserved, and must be set to zero. |

## H.264 Payload Data

H.264 payload data consists of video that is encoded using the H.264 Annex B byte-stream format and is byte oriented. The payload transfer size is variable and the total payload transfer length (the combined payload header and payload data) for each payload transfer must not exceed the maximum payload transfer size, as specified by the **dwMaxPayloadTransferSize** fieldin the video Probe and Commit Control.

A raw H.264 bitstream, in the Annex B byte-stream format, is a sequence of Start Code Prefix (SCP) plus NALU pairs, possibly with zero-byte padding after the NALU data. The first SCP for a picture is 4 bytes long. Each subsequent SCP for the same picture may be either 3 or 4 bytes long. A NALU has variable size. Each NALU starts with a NALU type indicator. The compressed bits for each slice are contained in a single NALU. A video frame may be represented using multiple NALUs, because a video frame can have multiple slices.

Zero-valued bytes that appear at the end of an H.264 Annex B byte-stream NALU are referred to as "trailing\_zero\_8bits" in the H.264 specification. For purposes of this specification, such zero-valued bytes are considered part of the NALU.

For purposes of this specification a *frame* is either an H.264-coded frame or a complementary pair of H.264-coded fields. Non-paired fields are not supported. For purposes of this specification, an *IDR frame* is defined as either a coded frame that is an IDR picture, or a coded pair of fields in which the first coded field is an IDR picture.

For purposes of this specification, a *random access I frame* is a frame that does not use any other frames as references for inter-picture prediction and does not have any frames that follow it in *both* decoding order (that is, bitstream order) and output order (that is, display order) that use frames that precede it in decoding order as references for inter-picture prediction. If a random access I frame is coded as a pair of fields, the second field may use the first field as a reference for inter-picture prediction.

A NALU can span multiple payload transfers. If a payload transfer contains the last byte of the last Annex B byte-stream NALU of a slice, the EOS flag is set in the payload header. No additional bytes may be contained in the payload transfer beyond the NALU containing this last slice. A new slice must start in a different payload transfer. The slice data will be preceded by an SCP, and may be preceded by other NALUs, for example, SPS/PPS and/or SEI messages. When data from a new capture time begins being transferred, the FID is toggled between 0 and 1, and the PTS/SCR must be set in the payload header. The FID must be the same for subsequent payload transfers belonging to the same capture time. Buffering period (BP) and picture timing (PT) supplemental enhancement information (SEI) NALUs can be used to carry additional timing information in the elementary bitstream. When present, a NALU containing a BP or PT SEI message must contain only one SEI message. A NALU containing a BP SEI message must be the first SEI NALU of the AU. A NALU containing a PT SEI message must be the first SEI NALU of the AU other than a NALU containing a BP SEI message, if present. Decoders should use this timing information, when available, to derive relative frame-capture times when the video comes from a variable-frame-rate source. When such timing information is present, random-access I frames and IDR frames shall have an associated BP SEI message.

# H.264 Video Format Descriptor

Table 5-1 H.264 Video Format Descriptor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Offset** | **Field** | **Size** | **Value** | **Description** |
| 0 | **bLength** | 1 | Number | Size of this descriptor in bytes. The value must be 52. |
| 1 | **bDescriptorType** | 1 | Constant | CS\_INTERFACE descriptor type. |
| 2 | **bDescriptorSubtype** | 1 | Constant | VS\_FORMAT\_H264 descriptor subtype (defined as 0x13). |
| 3 | **bFormatIndex** | 1 | Number | Index of this format descriptor. |
| 4 | **bNumFrameDescriptors** | 1 | Number | Number of Frame Descriptors following that correspond to this format |
| 5 | **bDefaultFrameIndex** | 1 | Number | Default frame index. |
| 6 | **bMaxCodecConfigDelay** | 1 | Number | Maximum number of frames the encoder takes to respond to a command. |
| 7 | **bmSupportedSliceModes**  **\*Note\*** EU\_SLICE\_MODE\_CONTROL is not supported in Windows 8. | 1 | Bitmap | Slice mode:  D0: Maximum number of MBs per slice mode.  D1: Target compressed size per slice mode.  D2: Number of slices per frame mode.  D3: Number of Macroblock rows per slice mode.  D4-D7: Reserved; set to 0.  Set everything to 0 if only one slice per frame is supported. |
| 8 | **bmSupportedSyncFrameTypes** | 1 | Bitmap | D0: Reserved; set to 0.  D1: IDR frame with SPS and PPS headers.  D2: IDR frame (with SPS and PPS headers) that is a long-term reference frame.  D3: Non-IDR random-access I frame (with SPS and PPS headers).  D4: Generate a random-access I frame (with SPS and PPS headers) that is not an IDR frame and is a long-term reference frame.  D5: P frame that is a long-term reference frame.  D6: Gradual Decode Refresh frames.  D7: Reserved; set to 0. |
| 9 | **bResolutionScaling** | 1 | Number | Specifies the support for resolution downsizing:  0: Not supported.  1: Limited to x1.5 or x2.0 scaling in both directions, while maintaining the aspect ratio.  2: Limited to x1.0, x1.5 or x2.0 scaling in either direction.  3: Arbitrary scaling.  4 to 255: Reserved.  Resolution scaling is implemented using the Video Resolution Encoding Unit, and cannot set the resolution beyond that specified in the currently selected frame descriptor. |
| 10 | **bSimulcastSupport**  \*Note that Windows 8 does not support Simulcast payloads so this byte should always be set to 0. | 1 | Number | Number of H.264 video streaming endpoints and number of streams this endpoint supports:  0: One endpoint and one stream.  1: One endpoint and multiple streams.  For devices that support simulcast, all Frame Descriptors that support simulcast should be under a Format Descriptor with this **bSimulcastSupport** flag set to 1. Those resolutions that do not support simulcast should be under a different instance of this Format Descriptor with bSimulcastSupport set to 0.  See Section 9.3.2 for details on simulcast streaming. |
| 11 | **bmSupportedRateControlModes** | 1 | Bitmap | Supported rate-control modes:  D0: Variable bit rate (VBR) with low delay. (H.264 low\_delay\_hrd\_flag = 1)  D1: Constant bit rate (CBR). (H.264 low\_delay\_hrd\_flag = 0)  D2: Constant QP.  D3: Global VBR with low delay. (H.264 low\_delay\_hrd\_flag = 1)  D4: VBR with non-low delay. (H.264 low\_delay\_hrd\_flag = 0)  D5: Global VBR with non-low delay. (H.264 low\_delay\_hrd\_flag = 0)  D7-D6: Reserved; set to 0. |
| 12 | **wMaxMBperSecOneResolutionNoScalability** | 2 | Number | Maximum macroblock-processing rate, in units of 1000 MB/s, allowed for a single AVC stream. See Section 9.3.2 for details. |
| 14 | **wMaxMBperSecTwoResolutionsNoScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for two AVC streams of different resolution. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 16 | **wMaxMBperSecThreeResolutionsNoScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for three AVC streams of different resolution. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 18 | **wMaxMBperSecFourResolutionsNoScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for four AVC streams of different resolution. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 20 | **wMaxMBperSecOneResolutionTemporalScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for temporal scalable SVC, summing up across all layers when all layers have the same resolution. See Section 9.3.2 for details. |
| 22 | **wMaxMBperSecTwoResolutionsTemporalScalablility** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for temporal scalable SVC, summing up across all layers when all layers consist of two different resolutions. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 24 | **wMaxMBperSecThreeResolutionsTemporalScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for temporal scalable streams, summing up across all layers when all layers consist of three different resolutions. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 26 | **wMaxMBperSecFourResolutionsTemporalScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for temporal scalable streams, summing up across all layers when all layers consist of four different resolutions. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 28 | **wMaxMBperSecOneResolutionTemporalQualityScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for temporal and quality scalable SVC streams, summing up across all layers when all layers have the same resolution. See Section 9.3.2 for details. |
| 30 | **wMaxMBperSecTwoResolutionsTemporalQualityScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for temporal and quality scalable SVC streams, summing up across all layers when all layers consist of two different resolutions. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 32 | **wMaxMBperSecThreeResolutionsTemporalQualityScalablity** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for temporal and quality scalable SVC streams, summing up across all layers when all layers consist of three different resolutions. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 34 | **wMaxMBperSecFourResolutionsTemporalQualityScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for temporal and quality scalable SVC streams, summing up across all layers when all layers consist of four different resolutions. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 36 | **wMaxMBperSecOneResolutionsTemporalSpatialScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for temporal and spatial scalable SVC streams, summing up across all layers when all layers have the same resolutions. See Section 9.3.2 for details. |
| 38 | **wMaxMBperSecTwoResolutionsTemporalSpatialScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for temporal and spatial scalable SVC streams, summing up across all layers when all layers consist of two different resolutions. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 40 | **wMaxMBperSecThreeResolutionsTemporalSpatialScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for temporal and spatial scalable SVC streams, summing up across all layers when all layers consist of three different resolutions. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 42 | **wMaxMBperSecFourResolutionsTemporalSpatialScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for temporal and spatial scalable SVC streams, summing up across all layers when all layers consist of four different resolutions. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 44 | **wMaxMBperSecOneResolutionFullScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for fully scalable streams, summing up across all layers when all layers have the same resolutions. See Section 9.3.2 for details. |
| 46 | **wMaxMBperSecTwoResolutionsFullScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for fully scalable streams, summing up across all layers when all layers consist of two different resolutions. See Section 9.3.2 for details. Zero for devices that do not support simulcast. |
| 48 | **wMaxMBperSecThreeResolutionsFullScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for fully scalable streams, summing up across all layers when all layers consist of three different resolutions. See Section 10.3.2 for details. Zero for devices that do not support simulcast. |
| 50 | **wMaxMBperSecFourResolutionsFullScalability** | 2 | Number | Maximum macroblock processing rate, in units of 1000 MB/s, allowed for fully scalable streams, summing up across all layers when all layers consist of four different resolutions. See Section 10.3.2 for details. Zero for devices that do not support simulcast. |

# H.264 Video Frame Descriptor

Each Frame Descriptor describes a unique resolution/profile combination. The resolution is defined by wWidth and wHeight. The profile is defined by **wProfile** and **wConstrainedToolset**. Codec usages, capabilities, rate control methods, video frame rates, and so forth are then enumerated for that combination.

Table 6-1 H.264 Payload Video Frame Descriptor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Offset** | **Field** | **Size** | **Value** | **Description** |
| 0 | **bLength** | 1 | Number | Size of this descriptor in bytes. The value must be (44 + (**bNumFrameIntervals** \* 4)). |
| 1 | **bDescriptorType** | 1 | Constant | CS\_INTERFACE descriptor type. |
| 2 | **bDescriptorSubtype** | 1 | Constant | VS\_FRAME\_H264 descriptor subtype (defined as 0x14). |
| 3 | **bFrameIndex** | 1 | Number | Index of this Frame Descriptor. |
| 4 | **wWidth** | 2 | Number | The width, in pixels, of pictures output from the decoding process. Must be a multiple of 2. Does not need to be an integer multiple of 16, and can be specified using a frame cropping rectangle in the active SPS. |
| 6 | **wHeight** | 2 | Number | The height, in pixels, of pictures output from the decoding process. Must be a multiple of 2. When field coding or frame/field adaptive coding is used, shall be a multiple of 4. Does not need to be an integer multiple of 16, and can be specified using a frame-cropping rectangle in the active SPS. |
| 8 | **wSARwidth** | 2 | Number | Sample aspect ratio width (as defined in H.264 Annex E); shall be relatively prime with respect to bSARheight. |
| 10 | **wSARheight** | 2 | Number | Sample aspect ratio height (as defined in H.264 Annex E); shall be relatively prime with respect to bSARwidth. |
| 12 | **wProfile**  \*Note that in Windows 8, regardless of the wProfile specified in this field, the payload MediaType will be H.264/H.264-ES. Refer to the MSDN documentation of eAVEncH264VProfile for the complete list of supported profiles. | 2 | Number | The first two bytes of the sequence parameter set, specified by profile\_idc and constraint flags in the H.264 specification, to indicate the profile and applicable constraints to be used. For example:  0x4240: Constrained Baseline Profile  0x4200: Baseline Profile  0x4D00: Main Profile  0x6400: High Profile  0x5300: Scalable Baseline Profile  0x5600: Scalable High Profile  0x7600: Multiview High Profile  0x8000: Stereo High Profile |
| 14 | **bLevelIDC** | 1 | Number | The level, as specified by the level\_idc flag (9, 10, 11, 12, 13, 20, 21, 22, 30, 31, 32, 40, 41, 42, etc.). For example:  0x1F: Level 3.1.  0x28: Level 4.0.  Note that this should ordinarily indicate the minimum level that supports the resolution and maximum bit rate for this frame descriptor. |
| 15 | **wConstrainedToolset** | 2 | Number | Constrains the features allowed by **wProfile** according to **wConstrainedToolset**:  0: No constraints. All tools defined by the selected wProfile and the bmSetting set are allowed.  1: UC Constrained High Toolset[[1]](#footnote-2).  2: UC Scalable Constrained High1.  3: UC Scalable Constrained Baseline1.  4 to 65535: Reserved. |
| 17 | **bmSupportedUsages**  \*Note Windows 8 only supports modes D0, D1, D16, D17, and D18. | 4 | Bitmap | D0: Real-time/UCConfig mode 0.  D1: Real-time/UCConfig mode 1.  D2: Real-time/UCConfig mode 2Q.  D3: Real-time/UCConfig mode 2S.  D4: Real-time/UCConfig mode 3.  D7-D5: Reserved; set to 0.  D15-D8: Broadcast modes.  D16: File Storage mode with I and P slices (for example, IPPP).  D17: File Storage mode with I, P, and B slices (for example, IB…BP).  D18: File storage all-I-frame mode.  D23-D19: Reserved; set to 0.  D24: MVC Stereo High Mode.  D25 : MVC Multivew Mode.  D31-D26 : Reserved; set to 0. |
| 21 | **bmCapabilities** | 2 | Bitmap | D0: CAVLC only.  D1: CABAC only.  D2: Constant frame rate.  D3: Separate QP for luma/chroma.  D4: Separate QP for Cb/Cr.  D5: No picture reordering.  D6: Long Term Reference frame.  D15-D7: Reserved; set to 0. |
| 23 | **bmMaxSVCCapabilities**  \*Note that only the first three bits, (D0-2), are parsed and used by Windows 8. | 4 | Bitmap | D0-2: Maximum number of temporal layers minus 1.  D3: Rewrite support.  D6-4: Maximum number of CGS layers minus 1.  D9-7: Maximum number of MGS sublayers.  D10: Additional SNR scalability support in spatial enhancement layers.  D13-11: Maximum number of spatial layers minus 1.  D31-14: Reserved.  See Section 9.2 for details. |
| 27 | **bmMaxMVCCapabilities**  [Not supported  in Window 8] | 4 | Bitmap | D2-0: Maximum number of temporal layers minus 1.  D10-3: Maximum number of view components minus 1.  D31-11: Reserved. See Section 10 for details. |
| 31 | **dwMinBitRate**  [Not supported  in Window 8] | 4 | Number | Specifies the minimum bit rate, at maximum compression and longest frame interval, in units of bps, at which the data can be transmitted. |
| 35 | **dwMaxBitRate**  [Not supported  in Window 8] | 4 | Number | Specifies the maximum bit rate, at minimum compression and shortest frame interval, in units of bps, at which the data can be transmitted. |
| 39 | **dwDefaultFrameInterval** | 4 | Number | Specifies the frame interval the device indicates for use as a default, in 100-ns units. |
| 43 | **bNumFrameIntervals** | 1 | Number | Specifies the number of frame intervals supported. |
| 44 | **dwFrameInterval(1)** | 4 | Number | Shortest frame interval supported (at the highest frame rate), in 100-ns units. |
|  | **…** |  |  |  |
| 44 + (**bNumFrameIntervals** \* 4) – 4 | **dwFrameInterval (bNumFrameIntervals)** | 4 | Number | Longest frame interval supported (at lowest frame rate), in 100-ns units. |

# H.264 Probe/Commit Control

This section describes the additional fields to be added to the standard UVC 1.1 Probe/Commit Control in order to support H.264. Note that for simplicity, the required bandwidth shall be estimated using the maximum bit rate for the selected profile/resolution and number of simulcast streams. The USB bandwidth reserved shall be the calculated by the host as the advertised **dwMaxBitRate** from the selected Frame Descriptor multiplied times the number of simulcast streams as defined in the **bmLayoutPerStream** field. The H.264 Interface Descriptor shall have multiple Alternate settings that support each of the bit rates calculated from the Frame Descriptors.

The Probe and Commit operational model is the same as specified in section 4.3.1.1.1 of the “Universal Serial Bus Device Class Definition for Video Devices Revision 1.1” specification.

After the initial Probe and Commit, Encoding Units are used to finalize the configuration of the encoder. This hybrid model of Descriptor plus Encoding Unit was chosen as the best model to navigate the complex space of encoder configuration.

Table 7-1 Video Probe and Commit Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | VS\_PROBE\_CONTROL  VS\_COMMIT\_CONTROL | | |
| Mandatory Requests | | GET\_CUR, GET\_MIN, GET\_MAX, GET\_RES, GET\_DEF, GET\_LEN, GET\_INFO, SET\_CUR | | |
| **wLength** | | 48 | | |
| Offset | Field | Size | Value | Description |
| 0-33 | **…** | … | … | UVC 1.1 Probe/Commit.  For H.264 formats, the following fields are ignored: **bmHint**, **wKeyFrameRate**, **wPFrameRate**, **wCompQuality**, **wCompWindowSize**, **wDelay**, **bPreferedVersion**, **bMinVersion**, **bMaxVersion**.  For H.264 formats, the **dwMaxVideoFrameSize** field is required, and indicates the maximum size of an Access Unit for the negotiated H.264 frame index. This size determines the buffer size that the driver allocates to store an Access Unit. Therefore, during streaming, the device shall never provide an Access Unit larger than **dwMaxVideoFrameSize**. The sender is required to indicate an Access Unit boundary via the FID bit in the payload header. |
| 34 | **bUsage** | 1 | Number | Current bUsage:  1: Real-time/UCConfig mode 0.  2: Real-time/UCConfig mode 1.  3: Real-time/UCConfig mode 2Q.  4: Real-time/UCConfig mode 2S.  5: Real-time/UCConfig mode 3.  6-8: Reserved.  9-16: Broadcast modes.  17: File Storage mode with I and P slices (for example, IPPP)  18: File Storage mode with I, P, and B slices (for example, IB…BP)  19: File storage with all-I-frame mode.  20-24: File storage modes.  25: MVC Stereo High mode.  26 : MVC Multivew mode.  0, 27-255: Reserved; set to 0.  The Real-time/UCConfig mode selected must be the highest of all the UCConfig modes that will be used among all simulcast streams. The specific configuration for each stream shall be specified in the **bmLayoutPerStream** field.  The MVC mode selected must be the highest of all the MVC modes that will be used among all simulcast streams. The specific configuration for each stream shall be specified in the **bmLayoutPerStream** field. |
| 35 | **bBitDepthLuma**  [Not supported  in Window 8] | 1 | Number | Represents bit\_depth\_luma\_minus8 + 8, which must be the same as bit\_depth\_chroma\_minus8 + 8. |
| 36 | **bmSetting**  [Not supported  in Window 8] | 1 | Bitmap | D0: CAVLC only.  D1: CABAC only.  Bits D1-D0 have the following meaning:  00: Let the device choose CAVLC/CABAC.  01: CAVLC only.  10: CABAC only.  11: Reserved.  D2: Constant frame rate.  D3: Separate QP for luma/chroma.  D4: Separate QP for Cb/Cr.  D5: No picture reordering.  D7-D6: Reserved; set to 0. |
| 37 | **bMaxNumberOfRefFramesPlus1**  [Not supported  in Window 8] | 1 | Number | Host indicates the maximum number of frames stored for use as references. (When non-zero, the max\_num\_ref\_frames syntax element in H.264 shall be less than or equal to this value minus 1.) |
| 38 | **bmRateControlModes** | 2 | Number | This field contains four subfields, each of which is a 4-bit number.  Indicates the rate-control mode for each stream. When bUsage is in the range [1, 5] or [25, 26], the number of streams is inferred from the **bmLayoutPerStream** field. For all other bUsage values, the number of streams is 1.  D3-D0: Rate-control mode for the first stream (with stream\_id=0).  D7-D4: Rate-control mode for the second stream (with stream\_id=1).  D11-D8: Rate control mode for the third stream (with stream\_id=2).  D15-D12: Rate control mode for the fourth stream (with stream\_id=3.)  When bmRateControlModes is non-zero, each 4-bit subfield can take one of the following values:  0: Not applicable, because this stream is non-existent.  1: VBR with low delay.  2: CBR.  3: Constant QP.  4: Global VBR with low delay.  5: VBR with non-low delay.  6: Global VBR with non-low delay.  7-15: Reserved. |
| 40 | **bmLayoutPerStream**  \*Note that comply with the limit of a single AVC stream, only the first 2 byte subfield is supported, and of that only the low 3 bits controlling temporal layers are valid. | 8 | Number | This field contains four subfields, each of which is a 2-byte number.  For **bUsage** in the range [1, 5], this field indicates the specific SVC layering structure for each stream.  For **bUsage** in the range [25, 26], this field indicates the specific MVC view structure for each stream, respectively. For all other supported bUsage values, this field shall be ignored.  When **bUsage** is in the range [1, 5] (SVC mode):  D15-0: SVC layering structure for simulcast stream with stream\_id 0.  D31-16: SVC layering structure for simulcast stream with stream\_id 1.  D47-32: SVC layering structure for simulcast stream with stream\_id 2.  D63-48: SVC layering structure for simulcast stream with stream\_id 3. It is recommended to associate streams with lower resolution/lower bit rate with smaller stream\_id.  See Section 9 for details.  When **bmUsage** is in the range [25, 26] (MVC mode):  D15-0: MVC view structure for simulcast stream with stream\_id 4.  D31-16: MVC view structure for simulcast stream with stream\_id 5.  D47-32: MVC view structure for simulcast stream with stream\_id 6.  D63-48: MVC view structure for simulcast stream with stream\_id 7.  See Section 10 for details. |

The following table describes VS\_PROBE\_CONTROL request attributes.

Table 7-2 VS\_PROBE\_CONTROL Requests

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| GET\_CUR | Returns the current values of the streaming interface. All supported fields set to zero will be returned with an acceptable negotiated value. Prior to the initial SET\_CUR operation, the GET\_CUR state is undefined. This request shall stall in case of negotiation failure. |
| GET\_MIN | Returns the minimum value for all negotiated fields. |
| GET\_MAX | Returns the maximum value for all negotiated fields. |
| GET\_RES | Return the resolution of each supported field in the Probe/Commit data structure. |
| GET\_LEN | Returns the length of the Probe data structure in bytes. |
| GET\_DEF | Returns the default value for all negotiated fields. |
| GET\_INFO | See UVC 1.1 specification. |
| SET\_CUR | Sets the streaming interface Probe state. This is the attribute used for stream parameter negotiation.  This request shall stall in the case where the device would be placed into an unsupported state or the case where the value for a negotiated field is out of range. See the definition in section 8 of the request error code controls. |

The following table describes VS\_COMMIT\_CONTROL request attributes.

Table 7-3 VS\_COMMIT\_CONTROL Requests

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| GET\_CUR | Returns the current values of the streaming interface. Prior to initial SET\_CUR operation, the GET\_CUR state is undefined.. |
| GET\_MIN | Not specified. |
| GET\_MAX | Not specified. |
| GET\_RES | Not specified. |
| GET\_LEN | Returns the length in bytes of the Commit data structure. |
| GET\_DEF | Not specified. |
| GET\_INFO | See UVC 1.1 specification. |
| SET\_CUR | Sets the streaming interface Probe state. This is the attribute used for stream parameter negotiation.  This request shall stall in the case where the device would be place into an unsupported state or the case where value for a negotiated field is out of range. See the definition in section 8 of the request error code controls. |

# H.264 Encoding Units

H.264 Encoding Unit requests are used to set or read the attributes of a video Control inside an Encoding Unit of the video function. The following paragraphs present a detailed description of all possible Controls an Encoding Unit can incorporate. For each Control, the layout of the parameter block together with the appropriate Control Selector is listed for all forms of the Get/Set Encoding Unit Control request. All values are interpreted as unsigned unless otherwise specified.

Encoding Units can be called before streaming starts as well as during streaming. When called before streaming, Encoding Units must not be called until a successful COMMIT(SET\_CUR) request has been accepted by the device. Encoding Units should use the default commit state, as returned by COMMIT(GET\_CUR), to understand the boundaries of what is currently possible.

This specification defines the following Encoding Unit controls for use before and during streaming:

* Profile and Toolset [Not supported in Window 8]
* Slice Mode [Not supported in Window 8]
* Peak Bit Rate [Not supported in Window 8]
* Priority ID [Not supported in Window 8]
* Select Layer
* Synchronization and Long Term Reference Frame
* Video Resolution
* Minimum Frame Interval
* Rate Control Mode
* Average Bitrate Control
* CPB Size Control
* Quantization Parameter
* Start or Stop Layer [Not supported in Window 8]

## H.264 Encoding Unit Descriptor

The Encoding Unit is uniquely identified by the value in the **bUnitID** field of the Encoding Unit descriptor (EUD). No other Unit or Terminal within the same video function may have the same ID. This value must be passed with each request that is directed to the Encoding Unit.

The **bSourceID** field is used to describe the connectivity for this Encoding Unit. It contains the ID of the Unit or Terminal to which this Encoding Unit is connected via its Input Pin. The **bmControls** field is a bit-map, indicating the availability of certain encoding Controls for the video stream. For future expandability, the number of bytes occupied by the **bmControls** field is indicated in the **bControlSize** field. The **bControlSize** field is permitted to specify a value less than the value needed to cover all the control bits (including zero), in which case the unspecified **bmControls** bytes will not be present and the corresponding control bits are assumed to be zero.

An index to a string descriptor is provided to further describe the Encoding Unit.

The layout of the Encoding Unit descriptor is detailed in the following table.

Table 8-1 Encoding Unit Descriptor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Offset** | **Field** | **Size** | **Value** | **Description** |
| 0 | **bLength** | 1 | Number | Size of this descriptor, in bytes. The value must be 7+2*n*, where *n* is the size in bytes of bmControls and bmControlsRuntime. |
| 1 | **bDescriptorType** | 1 | Constant | CS\_INTERFACE descriptor type. |
| 2 | **bDescriptorSubtype** | 1 | Constant | VC\_H264\_ENCODING\_UNIT descriptor subtype (defined as 0x07). |
| 3 | **bUnitID** | 1 | Number | A non-zero constant that uniquely identifies the Unit within the video function. This value is used in all requests to address this Unit. |
| 4 | **bSourceID** | 1 | Constant | ID of the Unit or Terminal to which this Unit is connected. |
| 5 | **iH264Encoding** | 1 | Index | Index of a string descriptor that describes this H.264 encoding unit. |
| 6 | **bControlSize** | 1 | Number | Size, in bytes, of the **bmControls** and **bmControlsRuntime** fields. The value must be *n*. |
| 7 | **bmControls** | *n* | Bitmap | A bit set to 1 indicates that the specified  control is supported for initialization:  D0: Select Layer.  D1: Profile and Toolset.  D2: Video Resolution.  D3: Minimum Frame Interval.  D4: Slice Mode.  D5: Rate Control Mode.  D6: Average Bitrate.  D7: CPB Size.  D8: Peak Bit Rate.  D9: Quantization Parameter.  D10: Synchronization and Long Term Reference Frame.  D11: Priority ID.  D12: Start or Stop Layer/View.  D13 ..(n\*8-1): Reserved; set to zero. |
| 7+n | **bmControlsRuntime** | *n* | Bitmap | A bit set to 1 indicates that the mentioned  control is supported during runtime:  D0: Select Layer.  D1: Profile and Toolset.  D2: Video Resolution.  D3: Minimum Frame Interval.  D4: Slice Mode.  D5: Rate Control Mode.  D6: Average Bitrate.  D7: CPB Size.  D8: Peak Bit Rate.  D9: Quantization Parameter.  D10: Synchronization and Long-Term Reference Frame.  D11: Priority ID.  D12: Start or Stop Layer/View.  D13 ..(n\*8-1): Reserved, set to zero. |

Table 8-2 H.264 Encoding Unit Control Selectors

|  |  |
| --- | --- |
| **Control Selector** | **VALUE** |
| EU\_CONTROL\_UNDEFINED | 0x00 |
| EU\_SELECT\_LAYER\_CONTROL | 0x01 |
| EU\_PROFILE\_TOOLSET\_CONTROL | 0x02 |
| EU\_VIDEO\_RESOLUTION\_CONTROL | 0x03 |
| EU\_ MIN\_FRAME\_INTERVAL\_CONTROL | 0x04 |
| EU\_ SLICE\_MODE\_CONTROL | 0x05 |
| EU\_RATE\_CONTROL\_MODE\_CONTROL | 0x06 |
| EU\_AVERAGE\_BITRATE\_CONTROL | 0x07 |
| EU\_CPB\_SIZE\_CONTROL | 0x08 |
| EU\_PEAK\_BIT\_RATE\_CONTROL | 0x09 |
| EU\_QUANTIZATION\_PARAMS\_CONTROL | 0x0A |
| EU\_SYNC\_REF\_FRAME\_CONTROL | 0x0B |
| EU\_PRIORITY\_ID\_CONTROL | 0x0C |
| EU\_START\_OR\_STOP\_LAYER\_CONTROL | 0x0D |

## Encoding Units Operational Model

Support of Encoding Unit controls by the device is optional. Encoding Units can be used to configure the codec before streaming or while streaming. Requests to Encoding Units controls can be issued only after a successful COMMIT\_CONTROL(SET\_CUR), which sets the device *default commit state*. Use COMMIT\_CONTROL(GET\_CUR) to get the *default commit state* at any time. The host must make all subsequent EU requests based on the *default commit state*. Similarly, the device shall use *default commit state* to validate any subsequent Encoding Unit control request.

Upon a successful COMMIT\_CONTROL(SET\_CUR), there will be fields that have not yet been initialized such as parameters for rate control or resolutions for simulcast stream. The device must establish default values for these unspecified fields such that the device can successfully stream with no further control from the host. The rules the device must follow to establish defaults are defined in Table 8-3. When a *default commit state* is established by the host (using COMMIT\_CONTROL(SET\_CUR)) the device must reestablish the additional default values using the rules defined in Table 8-3. In this way the host can reset the device to default behavior at any time.

Table 8-3. Default Encoding Unit state after VS\_COMMIT\_CONTROL SET\_CUR request.

| **EU parameter** | **First stream** | **Additional streams** |
| --- | --- | --- |
| **wLayerOrViewID** | 0 | 0 |
| **Width, height** | COMMIT (bFrameIndex) | DD ≤ Resolution of the first stream |
| **Minimum frame interval** | COMMIT (dwFrameInterval) | DD ≥ Minimum frame interval of the first stream |
| **Average bit rate** | DD | DD |
| **CPB size** | DD | DD |
| **Rate control mode** | COMMIT (bmRateControlModes) | COMMIT (bmRateControlModes) |
| **Quantization parameter** | DD | DD |
| **Priority ID** | COMMIT (bUsage & bmLayoutPerStream) | COMMIT (bUsage & bmLayoutPerStream) |
| **Slice mode** | DD | DD |
| **Profile** | COMMIT (bFrameIndex) | Same as first stream |
| **UC constrained toolset** | COMMIT (bFrameIndex) | Same as first stream |
| **bmSettings** | COMMIT (bFrameIndex)   * Entropy: CABAC if reported as supported in bFrameIndex. * No picture reordering: 1. * Others: DD. | Same as first stream |
| **Peak bit rate** | Default dwAverageBitRate x 64 | Default dwAverageBitRate x 64 |

* COMMIT(*x*): Indicates that the parameter is given by the COMMIT\_CONTROL(GET\_CUR) structure field indicated within parentheses.
* DD: Indicates the parameter is decided by the device maker.

In this specification, all Encoding Units Controls except EU\_SELECT\_LAYER\_CONTROL- and EU\_START\_OR\_STOP\_LAYER\_CONTROL are referred to as Encoder Configuration Units. The host may issue SET\_CUR requests to Encoder Configuration Units to configure encoder attributes for one or multiple layers.

After a default active state has been established, the host may issue one or more Encoder Configuration Unit control requests before or during streaming. Each Encoder Configuration Unit operation is atomic.

### Device States for Probe, Commit, and Encoding Units

Figure 8-1 illustrates five high-level device states and the USB requests that can trigger transitions between these device states when the request succeeds. USB requests not shown in Figure 1, such as GET\_CUR, GET\_MIN, GET\_MAX, GET\_DEF among others, do not trigger a state transition. At state 0, the device state is undefined after powering up. State 1 represents the *default commit state* of the device after a successful SET\_CUR(COMMIT). Any additional initialization of the device through Encoding Units moves the device from State 2 to State 3.

Table 8-4 describes how Encoding Units interact with each these five states. This includes possible errors that may be logged by the device in response to USB Requests issued to Encoding Unit controls. These errors may be retrieved by the host using the Request Error Code Control.

Figure 8-1. Device State Transitions Diagram

**SET\_CUR(COMMIT)**

**SET\_ALT\_INTERFACE 1**

**SET\_CUR(EU)**

**SET\_ALT\_INTERFACE 1**

**SET\_ALT\_INTERFACE 0**

**SET\_ALT\_INTERFACE 0**

**STREAMING**

**NOT STREAMING**

**SET\_CUR(EU)**

If the controlling application wishes to return to state 1 from states 2 or 3, they have the option of either undoing all Encoding Unit changes that have been made, or stopping the stream and renegotiating for state 1 using Probe/Commit.

Table 8-4 Encoding Units, Devices States and Error Code Control Responses

|  |  |  |  |
| --- | --- | --- | --- |
| **State** | **Streaming State** | **Encoding Unit State** | **Error Code Control Response to USB Requests issued to Encoding Unit Controls** |
| 0 | not streaming | undefined | Stall “Invalid Control” if EU is not supported or “Wrong State” if EU is supported after initial SET\_CUR(COMMIT). |
| 1 | not streaming | *default commit state* | Stall with error code:   * “Invalid Control” if EU is not supported. Else, * “Invalid Request” if USB request is not supported. Else, * “Wrong State” if EU is supported only while streaming or if the active wLayerOrView is not valid. Else, * “Out of Range” if any of the input arguments to the EU is invalid. |
| 2 | not streaming | customized by host | Same as in state 1 |
| 3 | streaming | customized by host | Stall with error code:   * “Invalid Control” if EU is not supported. Else, * “Invalid Request” if USB request is not supported. Else, * “Wrong State” if EU is supported only before streaming or if the active wLayerOrView is not valid. Else, * “Out of Range” if any of the input arguments to the EU is invalid. |
| 4 | streaming | *default commit state* | Same as in state 3. |

### Encoding Unit Requests

Table 8-5 describes Encoding Unit request attributes. Each Encoding Unit supports one or more of these attributes as specified in sections 8.3 through 8.15.

Table 8-5 Encoding Unit Requests

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| GET\_CUR | Returns the current values of all fields. Prior to the initial SET\_CUR and after Commit operations, the GET\_CUR state shall return the GET\_DEF state. Prior to the initial COMMIT\_CONTROL(SET\_CUR), the GET\_CUR state is undefined. |
| GET\_MIN | Returns the minimum value for all fields. |
| GET\_MAX | Returns the maximum value for all fields. |
| GET\_DEF | Returns the default value for all fields. |
| SET\_CUR | Sets the current state. For encoder configuration units, this is the attribute used for layer configuration and sets all the fields.  This request shall stall in case an unsupported state or field is specified. See section 9 for the definition of the request error code controls. |

### Stream Negotiation Examples

A successful USB isochronous bandwidth negotiation for a two layer stream.



Figure 8-2. Successful USB Isochronous Negotiation Example in a Simulcast Scenario

Figure 8-2 illustrates stream configuration and successful USB isochronous bandwidth negotiation in a simulcast scenario. In this example, during Probe/Commit, stream 1 is set to QP mode and stream 2 is set to VBR mode. Then, before streaming, the host configures the resolution of each stream using the EU\_VIDEO\_RESOLUTION\_CONTROL control. Rate control mode parameters are configured via the EU\_QUANTIZATION\_PARAMS\_CONTROL for stream 1, and via the EU\_ AVERAGE\_BITRATE\_CONTROL and EU\_CPB\_SIZE\_CONTROL for stream 2. After streaming starts, the host changes the average bit rate for the second stream.



Figure 8-3 Failed Request to an Encoding Unit Control Example

Figure 8-3 demonstrates a failed Encoding Unit request. In this case the host attempts to set rate control parameters to an invalid setting. After failure, the host resets the device by calling Commit with the same settings as used before. The host then sends valid rate control parameters to the device and starts streaming.

### Setting Resolution and Frame Interval

When changing resolution and/or minimum frame interval, whether during initialization or streaming, the following restrictions apply:

* The resolution must be equal to or less than the value established during Probe and Commit.
* The frame interval must be equal to or greater than the frame interval established during Probe and Commit.
* Resolution should be set prior to frame interval, which should be set prior all other Encoding Unit controls for that layer.

### Wildcard Masks

Wildcard masks may be used in association with the **wLayerOrViewID** field for SET\_CUR, GET\_CUR, GET\_MIN, GET\_MAX, and GET\_DEF requests for all EUs, with the limitations shown in the following table.

Table 8-6 Wildcard Usage for Encoding Unit Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EU Control** | **stream\_id** | **temporal\_id** | **quality\_id** | **dependency\_id** |
| Select Layer | Yes | Yes | Yes | Yes |
| Video Resolution | No | Yes | Yes | No |
| Profile and Toolset | No | No | No | No |
| Minimum Frame Interval | No | No | Yes | Yes |
| Slice Mode | No | Yes | Yes | Yes |
| Rate Control Mode | No | Yes | Yes | Yes |
| Average Bitrate | No | No | No | No |
| CPB Size | No | No | No | No |
| Peak Bit Rate | No | No | No | No |
| Quantization Parameter | No | No | No | No |
| Sync and Long-Term Reference Frame | No | No | No | No |
| Priority ID | No | No | No | No |
| Start or Stop Layer | No | No | No | No |

Note that grayed out Encoding Units are not supported in Windows 8.

## Select Layer Control

This control is used to specify the active layer/view for subsequent EU control operations. The default value for **wLayerOrViewID** is 0. Multiple layers/views can be specified in **wLayerOrViewID** by using a wildcard mask. Wildcard masks for SVC streams are defined in section 10.3.3, and wildcard masks for MVC streams are defined in section 11.3.3.

Table 8-7 Select Layer Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_SELECT\_LAYER\_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR, GET\_INFO | | |
| **wLength** | | 2 | | |
| Offset | Field | Size | Value | Description |
| 0 | **wLayerOrViewID** | 2 | Number | **For SVC streams**, a combination of dependency\_id, quality\_id, temporal\_id and stream\_id. Bits:  0-2: dependency\_id.  3-6: quality\_id.  7-9: temporal\_id.  10-12: stream\_id.  13-15: Reserved; set to 0.  When SVC is not supported, bits 0-9 are zero. When simulcast of multiple H.264 streams is not supported, bits 10-12 are zero.  **For MVC streams**, a combination of view\_id, temporal\_id, and stream\_id. Bits:  0-6: view\_id.  7-9: temporal\_id.  10-12: stream\_id.  13-15: Reserved; set to 0. |

The dependency\_id, quality\_id, temporal\_id, and view\_id of each layer of a stream are determined by the SVC/MVC capability mode negotiated in Probe and Commit. The exact matching between each SVC layer and these values is specified in the UCConfig specification and the H.264 standard. The stream\_id parameter is used to differentiate between H.264 streams, when simulcast of two or more H.264 streams is enabled. In the case of a single H.264 stream, stream\_id is always zero. Each additional H.264 stream is given a unique stream\_id by incrementing the stream\_id by 1. Section 10.1 describes how stream\_id is determined for AVC and SVC streams. Section 11.1 describes how stream\_id is determined for AVC and MVC streams.

## Video Resolution Control

The Video Resolution control is used to independently initialize the resolution of each layer in a multi-layer stream. If **bResolutionScaling** is true, this control can be used to change resolution of individual layers during streaming. The **wWidth** and **wHeight** fields are used to set the width and height of the decoded bitmap frame of a stream. **wWidth** and **wHeight** must each be a multiple of two.

When using this control to change video resolution, the new resolution is restricted to ratios as described in the bResoluitionScaling field in the Video Format Descriptor.

The bit rate limits described by **dwMinBitRate** and **dwMaxBitRate** may change when video resolution is changed. The application should use the GET\_MIN and GET\_MAX request on the Average Bit Rate control and on the CPB size control to determine the new values for minimum and maximum bit rate, and minimum and maximum CPB size after requesting a resolution change.

Supporting dynamic resolution changes means that changes to **wWidth** and **wHeight** do not stop the stream.

Table 8-8 Video Resolution Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_VIDEO\_RESOLUTION\_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR, ~~GET\_MIN, GET\_MAX, GET\_DEF, GET\_INFO~~ | | |
| **wLength** | | 4 | | |
| Offset | Field | Size | Value | Description |
| 0 | **wWidth** | 2 | Number | The width, in pixels, of pictures output from the decoding process. |
| 2 | **wHeight** | 2 | Number | The height, in pixels, of pictures output from the decoding process. |

## Profile and Toolset Control [Not Supported in Windows 8]

The Profile and Toolset control is used to specify the profile\_idc and constraint flags for the current layer(s) and to further constrain features within the profile. There are restrictions on using this control to change profile. Specifically, this control cannot be used to increase the number of layers beyond the number negotiated in Probe and Commit. Also, this control cannot be used to move the codec between AVC, SVC, and MVC profiles. The wConstrainedToolset field contains reserved bits for future extensibility.

Table 8-9 Profile Toolset Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_PROFILE\_TOOLSET\_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR, ~~GET\_DEF, GET\_INFO~~ | | |
| **wLength** | | 6 | | |
| Offset | Field | Size | Value | Description |
| 0 | **wProfile** | 2 | Number | The first two bytes of the sequence parameter set, specified by profile\_idc and constraint flags in the H.264 spec to indicate the profile and applicable constraints to be used. For example:  0x4240: Constrained Baseline Profile  0x4200: Baseline Profile  0x4D00: Main Profile  0x6400: High Profile  0x5300: Scalable Baseline Profile  0x5600: Scalable High Profile  0x7600: Multiview High Profile  0x8000: Stereo High Profile |
| 2 | **wConstrainedToolset** | 2 | Number | Constrains the features allowed by wProfile according to wConstrainedToolset:  0: No constraints. All tools defined by the selected **wProfile** and the **bmSetting** set are allowed.  1: UC Constrained High Toolset[[2]](#footnote-3).  2: UC Scalable Constrained High Toolset.  3: UC Scalable Constrained Baseline 4 to 65535: Reserved |
| 4 | **bmSetting** | 2 | Bitmap | D0: CAVLC only.  D1: CABAC only.  Bits D0-D1have the following meaning:  00: Let the device choose CAVLC/CABAC.  10: CAVLC only.  01: CABAC only.  11: Reserved.  D2: Constant frame rate.  D3: Separate QP for luma/chroma.  D4: Separate QP for Cb/Cr.  D5: No picture reordering.  D15-D6: Reserved; set to 0. |

## Minimum Frame Interval Control

The Minimum Frame Interval control is used to specify the minimum frame interval of the stream associated with the current **wLayerOrViewID**. This value reflects the maximum frame rate when combining the base layer and all the enhancement temporal layers. The new **dwFrameInterval** must be a frame interval that is advertised in the currently negotiated frame descriptor and greater than or equal to the frame interval negotiated in Probe and Commit. When **dwFrameInterval** is applied to a temporal scalable stream, this control shall be applied to the highest temporal enhancement layer, and the dyadic constraint between successive temporal layers must be fulfilled.

Note that the bit rate limits described by **dwMinBitRate** and **dwMaxBitRate** may change when the frame interval is changed. The application should use the GET\_MIN and GET\_MAX request on the Average Bitrate control to determine the new values for minimum bit rate and maximum bit rate.

In this control, the video frame interval, dwFrameInterval, is specified in 100-ns units and is derived from the frame rate as follows.

For a frame rate *x*, the video frame interval is (10,000,000/*x*) truncated to an integer value.

For example:

15 fps: Frame interval = (10000000/15) = 666666

30 fps: Frame interval = (10000000/30) = 333333

25 fps (PAL): Frame interval = (10000000/25) = 400000

29.97 fps (NTSC): Frame interval = (10000000/29.97) = 333667

Table 8-10 Minimum Frame Interval Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_MIN\_FRAME\_INTERVAL\_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR ~~GET\_MIN, GET\_MAX, GET\_DEF, GET\_INFO~~ | | |
| **wLength** | | 4 | | |
| Offset | Field | Size | Value | Description |
| 0 | **dwFrameInterval** | 4 | Number | Frame interval in 100-ns units. Cannot go below the **dwFrameInterval** set in Probe/Commit. Must be an interval advertised in the associated Frame Descriptor that has the same resolution, profile, and toolset of the current layer.  This control is undefined if applied to any layer that is not the highest layer. |

## 

## Slice Mode Control [Not Supported in Windows 8]

The Slice Mode control is used to specify the slice mode of current layer/view. This control is supported only if the codec supports using more than one slice per frame. If the device supports this control it must support one or more of the slice modes described below.

All GET requests apply to the current slice mode.

Table 8-11 Slice Mode Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_SLICE\_MODE\_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR, ~~GET\_DEF, GET\_MIN, GET\_MAX, GET\_INFO~~ | | |
| **wLength** | | 4 | | |
| Offset | Field | Size | Value | Description |
| 0 | **wSliceMode** | 2 | Number | Most significant byte: **Mode**.  0: Maximum number of MBs per slice mode.  1: Target compressed size per slice mode.  2: Number of slices per frame mode.  3: Number of Macroblock rows per slice mode.  4-255: Reserved. |
| 2 | **wSliceConfigSetting** |  | Number | The meaning of this field depends on **wSliceMode** mode:  Mode 0: Maximum number of MBs per slice. An integer.  Mode 1: Target size for each slice NALU in bytes. Applies to all layers. When indicated, the encoder should stay within 10% of the target value, except for the last slice of a picture, which can be smaller. An integer.  Mode 2: Number of slices per frame. Applies to all layers. An integer.  Mode 3: Number of macroblock rows per slice. An integer. |

## Rate Control Mode Control

The Rate Control Mode control is used to specify the rate control mode of the current layer. After the mode is established, rate control parameters are set using the Peak Bit Rate control, Average BitRate control, CPB Size control, or Quantization control.

For variable and constant bit-rate buffer modeling, this document specifies rate control operation in terms of a leaky bucket model. The bits used to encode each picture are analogous to cups of water being dumped into the top of the bucket when each picture is encoded (after re-ordering the pictures as necessary for bitstream orders that differ from display order); the level of water in the bucket indicates the number of bits waiting to be sent to the decoder; and the water leaking out of a hole in the bottom of the bucket corresponds to bits flowing into the decoder through a transmission channel. The leaky bucket is a traffic meter that contains two parameters:

* RP = **dwPeakBitRate** (bits per second), which is the peak bit rate at which bit can flow out from the bottom of the bucket
* B = **dwCPBsize** (bits), which is the coded picture buffer (CPB) capacity



The buffer serves to smooth out local bit rate fluctuations while limiting the total bit usage that is possible over longer durations and limiting the buffering capacity necessary for a decoder to be able to decode the video content.

The leaky bucket model at the encoder has a corresponding mirror-image model that operates from the decoder perspective. As bits leak out of the encoder buffer, they conceptually enter into a corresponding decoder input buffer, which continues to fill up until the decoding time of a picture arrives – at which time the bits for that picture are removed from the decoder’s CPB.

If too many bits are dumped into the bucket too quickly, the buffer capacity B is exceeded before enough bits have time to drain out of the hole in the bottom of the bucket, and the buffer is said to “overflow” from the encoder perspective. From the decoder perspective, an overflow could occur if the removal of pictures from the decoder CPB at the decoding times of those pictures is not fast enough to keep up with the amount of bits that have been flowing into it from the encoder. (The conformance to the H.264 HRD model is specified in terms of the decoder perspective, although the encoder perspective tends to be easier to describe when expressing the intent.)

The encoder shall ensure that the leaky bucket never overflows. The application must specify the rate-control parameters in the Average Bit Rate and CPB Size EU Controls in an HRD-conformant manner with respect to the profile and level combination.

Table 8-12 Rate Control Mode Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_RATE\_CONTROL\_MODE\_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR, ~~GET\_DEF, GET\_INFO~~ | | |
| **wLength** | | 1 | | |
| Offset | Field | Size | Value | Description |
| 0 | **bRateControlMode** | 1 | Number | 0: Reserved  1: Variable Bit Rate low delay (VBR)  2: Constant bit rate (CBR)  3: Constant QP  4: Global VBR low delay (GVBR)  5: Variable bit rate non-low delay (VBRN)  6: Global VBR non-low delay (GVBRN)  7-255: Reserved |

### Variable Bit Rate (VBR)

The VBR control mode limits the long-term maximum bit rate of transmission, but allows the bit rate to vary substantially on a short-term basis. In particular, in the VBR mode the encoder is not required to continue to produce a significant bit rate in cases where there is little or no motion activity in the coded video scene content.

In the variable bit rate (VBR) mode of bit rate control operation, it is allowed for the encoder’s leaky bucket to sometimes “run dry”, that is, for all bits to leave the encoder CPB and for there to be some period of time during which no bits are flowing. The presence of such periods of time during which no bits are flowing implies that the average bit rate over a long duration of video content can be less than the peak bit rate RP.

When bits are flowing, the H.264 HRD operates at the specified peak bit rate RP = **dwPeakBitRate** bps rather than the “average” bit rate **dwAverageBitRate** bps. The average bit rate parameter **dwAverageBitRate** serves only as a guideline, as follows:

* The average bit rate parameter **dwAverageBitRate** represents only a “target” or “guideline”, indicating the average bit rate expected to be produced by the video encoder when operating under normal lighting conditions with a normal degree of video scene activity. To calculate dwAverageBitRate, the encoder should sample over the period T = CPB size / Peak bit rate.
* It is allowed for the bit rate produced by the video encoder to exceed dwAverageBitRate (for example, when there is an exceptionally high degree of activity in the video scene).
* It is also allowed for the bit rate produced by the video encoder to be less than dwAverageBitRate (for example, when there is very little activity in the video scene or when lighting conditions are poor).

In contrast, the peak bit rate RP = **dwPeakBitRate** bps and the total buffer capacity B, which are the operating parameters of the leaky bucket model, correspond to a mandatory maximum not to be exceeded by the encoder on a long-term basis (that is, the leaky bucket model shall not overflow).

The average bit rate parameter **dwAverageBitRate** shall be set to a value less than or equal to peak bit rate = dwPeakBitRate x 64.

For SVC multi-layer bitstreams, the VBR-control model applies to the currently selected substream as defined in section 10.3.4. For AVC bitstreams, this rate control model applies to the entire bitstream, because there is only one layer in the bitstream.

### Constant Bit Rate (CBR)

CBR control mode is similar to the VBR control mode, except that dummy bits are generated when necessary in order to ensure an exactly constant bit rate flowing from the encoder leaky bucket (there are always bits flowing to the decoder). For CBR operation the average bit rate is equal to the peak bit rate, because there is no variability in the bit rate in this case. In the CBR mode, the average bit rate parameter **dwAverageBitRate** shall be set equal to the peak bit rate = dwPeakBitRate x 64.

For SVC multi-layer bitstreams, this rate-control model applies to the currently selected substream as defined in section 10.3.4. For AVC multi-layer bitstreams, this rate-control model applies to the entire bitstream, because there is only one layer in the bitstream.

The encoder shall ensure that the leaky bucket never overflows nor underflows in the CBR case. The application must specify the rate-control parameters in the Average Bitrate and CPB size controls in an HRD-conformant manner with respect to the profile and level combination.

### Constant QP Mode

In Constant QP mode, the encoder shall use the **dwQpPrime** fields to derive a constant QP for I, P, and B slices (or EI, EP, and EB slices if quality or spatial scalability is employed).

### Global VBR

In Global VBR mode, the rate-control model applies to the entire bitstream. For single-layer AVC bitstreams, this mode is identical to VBR mode. For multi-layer SVC bitstreams, this mode implies that no explicit rate control is required for the sub-bitstreams, as long as the entire bitstream obeys the rate-control model specified in the Average BitRate and CPB size Encoding Unit controls.

### Low Delay and Non-Low Delay Modes

Within the VBR and Global VBR modes of operation, there are two variants that correspond to values of the H.264 low\_delay\_hrd\_flag syntax element. These variants concern the timely availability of the coded bits for decoding purposes. If the decoding time of a picture arrives but not all of the bits that represent that picture have yet drained out of the encoders leaky bucket model (and therefore those bits are not yet available in the decoder’s input buffer when the decoding time of the picture arrives), the leaky bucket model is said to “underflow”[[3]](#footnote-4):

* When low\_delay\_hrd\_flag is equal to 1, underflow is allowed for the leaky bucket model. Operating in this manner can help reduce the average end-to-end delay through the system (although it may sometimes cause the decoder to not exactly reproduce the correct timing of the pictures for its output/display purposes).
* When operating with the H.264 low\_delay\_hrd\_flag equal to 0, the HRD shall not underflow when operating at the peak bit rate. Although the leaky bucket is allowed to sometimes “run dry”, the decoding time of a picture is not allowed to arrive before all of the bits for that picture have yet departed from the encoder’s leaky bucket (which corresponds to arrival into the decoder’s corresponding input buffer).

The low delay and non-low delay variants apply to the VBR and Global VBR modes. This is reflected in the following four rate control modes specified above:

1: Variable bit rate low delay (VBR)

4: Global VBR low delay (GVBR)

5: Variable bit rate non-low delay (VBRN)

6: Global VBR non-low delay (GVBRN)

If no VUI information (Annex E) is included in the SPS, then fixed\_frame\_rate\_flag is assumed to be 0 and thereby low\_delay\_hrd\_flag is assumed to be 1. These settings correspond to a low-delay, variable-frame-rate stream.

## Average Bitrate Control

The Average Bitrate control is used to specify the average bit rate of the current layer(s). This control is used in all bRateControlMode modes except Constant QP. SET\_CUR shall stall with a “Wrong State” error code if the specified stream is not set to CBR, VBR, VBRN, Global VBR or Global VBRN.

For AVC bitstreams or when in Global VBR rate-control mode, this EU applies to the entire bitstream. For SVC streams, **dwAverageBitRate** is set per sub-bitstream, as defined in section 10.3.4.

This EU has no effect when the device operates in Constant QP rate-control mode.

The **dwAverageBitRate** value returned by the device upon a GET\_MIN request specifies the minimum average bit rate for the sub-bitstream at the sub-bitstream frame interval. For GET\_MAX, the device returns the maximum capability for the overall stream at the current frame interval for that stream, summing up bit rates across all layers in the sub-bitstream. These values are for the active resolution as specified by EU\_VIDEO\_RESOLUTION\_CONTROL GET\_CUR().

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_AVERAGE\_BITRATE\_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR, ~~GET\_MIN, GET\_MAX, GET\_DEF, GET\_INFO~~ | | |
| **wLength** | | 4 | | |
| Offset | Field | Size | Value | Description |
| 0 | **dwAverageBitRate** | 4 | Number | Average bit rate, in bits per second. Must be less than or equal to **dwPeakBitRate** . Applies for all rate control modes except Constant QP. |

## CPB Size Control

The CPB Size control is used to specify the CPB size of the current layer(s). This control is used in all **bRateControlMode** modes except Constant QP. SET\_CUR shall stall with a “Wrong State” error code if the specified stream is not set to CBR, VBR, VBRN, Global VBR or Global VBRN.

**dwCPBsize** is set per sub-bitstream, as defined in section 10.3.4.

This EU has no effect when the device operates in Constant QP rate-control mode.

The **dwCPBsize** value returned by the device upon a GET\_MIN request specifies the minimum CPB size for use with the average bit rate for the current sub-bitstream. For GET\_MAX the device shall return the maximum CPB size for the current sub-bitstream supported by the device for the given **bLevelIDC** specified in the frame descriptor. The host must set the current dwCPBsize appropriately for the **dwPeakBitRate**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_CPB\_SIZE\_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR, ~~GET\_MIN, GET\_MAX, GET\_DEF, GET\_INFO~~ | | |
| **wLength** | | 4 | | |
| Offset | Field | Size | Value | Description |
| 0 | **dwCPBsize** | 4 | Number | CPB size, in units of 16 bits, corresponding to the NAL CPB size. Must be a size supported by the profile and level combination. Applies for all rate control modes except Constant QP. |

## Peak Bit Rate Control [Not Supported in Windows 8]

The Peak Bit Rate control is used to specify the peak bit rate of the current layer. This control applies only when **bRateControlMode** is set to *VBR*, *VBRN*, *Global VBR*, or *Global VBRN*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_PEAK\_BIT\_RATE\_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR, ~~GET\_MIN, GET\_MAX, GET\_DEF~~ | | |
| **wLength** | | 4 | | |
| Offset | Field | Size | Value | Description |
| 0 | **dwPeakBitRate** | 4 | Number | Peak bit rate, in units of 64 bits per second. Must be a rate supported by the profile and level combination. Used only for VBR, VBRN, Global VBR, and Global VBRN mode. |

## Quantization Parameter Control

The Quantization Parameter control is used to specify quantization parameters for the current layer. This control applies only when **bRateControlMode** is set to *Constant QP*. SET\_CUR shall stall if the **bRateControlMode** mode for the active layer is not set to Constant QP.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_QUANTIZATION\_PARAMS\_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR, ~~GET\_MAX, GET\_MIN, GET\_DEF, GET\_INFO~~ | | |
| **wLength** | | 6 | | |
| Offset | Field | Size | Value | Description |
| 0 | **wQpPrime\_I** | 2 | Number | Applicable only in constant QP rate-control mode.  Use this parameter to set/get QP for I frames. Up to three values can be passed to configure the picture quantization parameters:  D7-D0: QP'Y  D11-D8: chroma\_qp\_index\_offset as a signed 4-bit number in two’s complement representation. The value shall be in the range −8 to +7, inclusive.  D15-D12: second\_chroma\_qp\_index\_offset as a signed 4-bit number in two’s complement representation. The value shall be in the range −8 to +7, inclusive. |
| 2 | **wQpPrime\_P** | 2 | Number | Applicable only in constant QP rate-control mode.  Use this parameter to set/get QP for P frames. Up to three values can be passed to configure the picture quantization parameters:  D7-D0: QP'Y  D11-D8: chroma\_qp\_index\_offset as a signed 4-bit number in two’s complement representation. The value shall be in the range −8 to +7, inclusive.  D15-D12: second\_chroma\_qp\_index\_offset as a signed 4-bit number in two’s complement representation. The value shall be in the range −8 to +7. |
| 4 | **wQpPrime\_B** | 2 | Number | Applicable only in constant QP rate control mode.  Use this parameter to set/get QP for B frames. Up to three values can be passed to configure the picture quantization parameters:  D7-D0: QP'Y  D11-D8: chroma\_qp\_index\_offset as a signed 4-bit number in two’s complement representation. The value shall be in the range −8 to +7, inclusive.  D15-D12: second\_chroma\_qp\_index\_offset as a signed 4-bit number in two’s complement representation. The value shall be in the range −8 to +7. |

Note: The H.264 standard specifies that QP'Y = QPY + 6 \* bit\_depth\_luma\_minus8, where bit\_depth\_luma\_minus8 corresponds to bBitDepthLuma – 8, negotiated in Probe/Commit.

### Quantization Weighting Matrices

This specification does not support control of quantization weighting matrices. For profiles that support the use of quantization weighting matrices, when a QP value control is specified, the quantization weighting matrix entries must either be flat matrices with all entries equal to 16, or must have rate-distortion behavior that is approximately similar to the use of flat matrices with entries equal to 16.

## Synchronization and Long-Term Reference Frame Control

This control is used to manage insertion of synchronization frames and long-term reference frames into the current layer. When the host requests the generation of a sync frame, the encoder shall insert the specified **bSyncFrameType** into all the dependency representations associated with the current stream as identified by the current **wLayerOrViewID**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_SYNC\_REF\_FRAME\_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR, ~~GET\_MIN, GET\_MAX, GET\_INFO~~ | | |
| **wLength** | | 4 | | |
| Offset | Field | Size | Value | Description |
| 0 | **bSyncFrameType** | 1 | Number | 0: Reset. Allow the encoder to determine the timing of synchronization frames.  1: Generate an IDR frame (with SPS and PPS headers) for all of the dependency representations of the current stream, if all of the layer representations of the current stream have quality\_id equal to 0. Otherwise, generate key frames for all the dependency representations.  2: Generate an IDR frame (with SPS and PPS headers) that is a long-term reference frame for all of the dependency representations of the current stream, if all of the layer representations of the current stream have quality\_id equal to 0. Otherwise, generate a key frame for the associated dependency representation of the current **wLayerOrViewID**.  3: Generate a non-IDR random-access I frame (with SPS and PPS headers ) for the associated dependency representation of the current **wLayerOrViewID**.  4: Generate a non-IDR random-access I frame (with SPS and PPS headers) that is a long-term reference frame for the associated dependency representation of the current **wLayerOrViewID**.  5: Generate a P frame that is a long-term reference frame for the associated dependency representation of the current **wLayerOrViewID**.  6: Gradual Decoder Refresh (GDR).  7-255: Reserved. |
| 1 | **wSyncFrameInterval** | 2 | Number | In milliseconds. This field indicates the periodic recurrences of the selected **bSyncFrameType**. A value of **wSyncFrameInterval**=0 indicates a single **bSyncFrameType** with no requirement for periodic recurrence. |
| 3 | **bGradualDecoderRefresh** | 1 | Number | Indicates a count of frames over which the gradual decoder refresh occurs. Valid only when **bSyncFrameType** = 5 (GDR). When **bSyncFrameType** is not 5, this field must be 0. From a recovery point of view, (**bGradualDecoderRefresh**+1) represents the number of frames in decoding order required to completely refresh the picture.  Bits:  0-6: recovery\_frame\_cnt  7: Reserved  Use **wSyncFrameInterval** to establish the interval between Gradual Decoder Refresh (GDR) periods.  When using GDR, the encoder should use recovery point SEI messages to indicate gradual decoder refresh access points. |

GET\_MIN and GET\_MAX can be used to determine the minimum and maximum recovery\_frame\_cnt over which the encoder can implement GRD. GET\_MIN and GET\_MAX can also be used to determine the minimum and maximum recurrence of all **bSyncFrameType**.

GET\_MIN and GET\_MAX may be used to determine whether the device supports changes to **wSyncFrameInterval**. If GET\_MIN and GET\_MAX return the same **wSyncFrameInterval** value as GET\_CUR, then the device does not support changes to this value.

GET\_CUR can be used to check whether the last SET\_CUR request issued by the host was set correctly on the device. Prior to a successful SET\_CUR, GET\_CUR is undefined.

## Priority ID Control [Not Supported in Windows 8]

The Priority ID control is used to set SVC syntax element prority\_id for all of the NALUs in a specific layer. The host may pass the priority ID value for each layer for the encoder to add to prefix NALUs with nal\_unit\_type equal to 14 or NALUs with nal\_unit\_type equal to 20.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_PRIORITY\_ID \_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR, ~~GET\_INFO~~ | | |
| **wLength** | | 1 | | |
| Offset | Field | Size | Value | Description |
| 0 | **bPriority** | 1 | Number | The value of priority\_id for the scalable layer specified by the current **wLayerOrViewID**. |

On a GET\_MIN request, the device shall return 0. On a GET\_MAX request, the device shall return 63.

On a GET\_DEF request, the device shall return values based on **bUsage**. When **bUsage** is between 1 and 5 (Real-time/UCConfig modes), the device shall return defaults according to the rules specified in the UCConfig specification, governed by the following algorithm:

1: Initialize *pid\_value* to 0.

2: Iterate active streams from largest to smallest *stream\_id*.

3: Iterate layers in the stream from small to large *TDQId*[[4]](#footnote-5)

4: The default value of ***priority\_id*** is equal to *pid\_value*.

5: Increment *pid\_value* by one.

6: End

7: End

When **bUsage** is outside the 1 to 5 range, GET\_DEF shall return **bPriorityID** = 0.

## Start or Stop Layer Control [Not Supported in Windows 8]

This control is used to start or stop streaming of the current layer(s). By default all layers will be streamed. The host may stop individual layers (and their dependents) from streaming by using this control. When individual layers (and their dependents) are stopped, this control may be used to restart them.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Selector | | EU\_START\_OR\_STOP\_LAYER\_CONTROL | | |
| Mandatory Requests | | SET\_CUR, GET\_CUR, ~~GET\_INFO~~ | | |
| **wLength** | | 1 | | |
| Offset | Field | Size | Value | Description |
| 0 | bUpdate | 1 | Number | 0: Stop streaming the current layer and all layers depending on it.  1: Start streaming the current layer (and all layers it depends on) when the device starts streaming. If this control is not issued before streaming is enabled (before SET\_INTERFACE), the encoder shall stream every layer. When streaming is enabled, GET\_CUR returns the current state of the layer (0 if stopped, or 1 if started). |

# Request Error Code Control

This read-only control indicates the status of each host-initiated request to a Terminal, Unit, interface, or endpoint of the video function. If the device is unable to fulfill the request, it will indicate a stall on the control pipe and update this control with the appropriate code to indicate the cause. This control will be reset to 0 (no error) upon the successful completion of any control request, including requests to this control. The table below specifies the **bRequestErrorCode** error codes that the device must return from a VC\_REQUEST\_ERROR\_CODE\_CONTROL request. Asynchronous control requests are a special case, where the initial request will update this control, but the final result is delivered via the Status Interrupt Endpoint. (See sections 2.4.2.2, “Status Interrupt Endpoint” and 2.4.4, “Control Transfer and Request Processing, ” in the UVC 1.1 specification).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Offset** | **Field** | **Size** | **Value** | **Description** |
| 0 | bRequestErrorCode | 1 | Number | 0x00: No error.  0x01: Not ready.  0x02: Wrong state.  0x03: Power.  0x04: Out of range.  0x05: Invalid unit.  0x06: Invalid control.  0x07: Invalid request.  0x08-0xFE: Reserved for future use.  0xFF: Unknown. |

Note that grayed out Error Codes are not supported in Windows 8.

**No error**: The request succeeded. Returns STATUS\_SUCCESSFUL.

**Not ready**: The device has not completed a previous operation. The device will recover from this state as soon as the previous operation has completed. Returns STATUS\_DEVICE\_NOT\_READY.

**Wrong State**: The device is in a state that disallows the specific request. The device will remain in this state until a specific action from the host or the user is completed. For example, if the host issues an Encoding Unit request while streaming to an Encoding Unit Control that is not supported while streaming, the device shall stall with “Wrong State.” [Not supported in Windows 8]

**Power**: The actual Power Mode of the device is not sufficient to complete the request.

**Out of Range**: Result of a SET\_CUR request when attempting to set a value outside of the MIN and MAX range, or a value that does not satisfy the constraint on the selected layers/views. Returns STATUS\_INVALID\_PARAMETER.

**Invalid Unit**: The Unit ID addressed in this request is not assigned.

**Invalid Control**: The Control addressed by this request is not supported. For control requests issued to Encoding Units, this error indicates this Encoding Unit is not supported at any time. Returns STATUS\_INVALID\_DEVICE\_REQUEST.

**Invalid Request**: This request is not supported by the Control.

# SVC and Simulcast Support [Partially Supported in Windows 8]

This section provides technical background, detailed descriptions, and examples to illustrate how to support SVC and/or simulcast in this specification. Developers may skip this section if the encoder does not support the generation of SVC bitstreams. In that case, both **bmSVCCapabilities** and **bmLayoutPerStream** shall be set to 0.

**Limited Windows 8 Support for SVC**

Windows 8 supports only temporal scaling. Further, even if the stream contains temporal layers, it will not be marked as SVC.I Instead, it will be marked as an H.264 AVC elementary stream (KSDATAFORMAT\_SUBTYPE\_H264\_ES). Limiting support to only temporal scaling affects several fields, as follows:

* **bmMaxSVCCapabilities**: Only the first 3 bits of this bitmap are valid.
* **bmLayoutPerStream**: Only the first 3 bits of this bitmap are valid.

## SVC Overview

Scalable Video Coding (SVC) is primarily specified in Annex G of the H.264/MPEG-4 Advanced Video Coding (AVC) standard. Within an access unit (AU), there is one “base layer” that is formatted as an H.264/AVC coded picture, and one or more additional scalable layer representations, each of which represents an additional “enhancement layer” of an SVC-encoded bitstream for the same instant in time. SVC supports three main types of classes of scalability: temporal, quality (or SNR), and spatial scalability. Quality scalability can be further classified into Coarse-Grained Scalability (CGS) and Medium-Grained Scalability (MGS). An SVC bitstream may contain arbitrary combinations of these three classes of scalability. To simplify the design, this specification considers only the most commonly used layering structures as defined in the UCConfig Specification, summarized as follows:

* *Temporal scalability*, if it is used, is applied first in layering a SVC bitstream. A temporal layer is identified by the syntax element temporal\_id for an H.264 NALU. The value of temporal\_id must start from 0 and increase continuously.
* [Not supported in Windows 8] *Quality scalability*, if it is used, is applied next in layering a SVC bitstream. A quality layer is identified by the syntax element dependency\_id in CGS mode and quality\_id in MGS mode for an H.264 NALU. The values of quality\_id and dependency\_id must start from 0 and increase continuously. When MGS is used, an MGS layer is split into multiple sublayers by means of transformed coefficient partitioning. CGS is effectively a special case of spatial scalability when two successive spatial layers have identical spatial resolutions.
* [Not supported in Windows 8] *Spatial scalability*, if it is used, is applied last in layering a SVC bitstream. A spatial layer is identified by the syntax element dependency\_id in an H.264 NALU. Additional quality scalable layers may be applied in a spatial enhancement layer.

With these constraints, for a particular layering structure the values of temporal\_id, dependency\_id, and quality\_id associated with a layer can be determined without ambiguity and used as a unique identifier for that layer.

## SVC Capability Advertisement

The encoder notifies the SVC capabilities in **bmSVCCapabilities** in the Video Frame Descriptor. The following table shows the format description.

Table 10-1 Bit Values of bmSVCCapabilities Field

|  |  |
| --- | --- |
| Bitfields | Name |
| [2-0] | MaxNumOfTemporalLayersMinus1 |
| 3 | RewriteSupport [Must be 0 for Windows 8, set to 0] |
| [6-4] | MaxNumOfCGSLayersMinus1 [Must be 0 for Windows 8, set to 0] |
| [9-7] | MaxNumOfMGSSublayers [Must be 0 for Windows 8, set to 0] |
| 10 | AdditionalSNRScalabilitySupport [Not supported in Windows 8, set to 0] |
| [13-11] | MaxNumOfSpatialLayersMinus1 [Not supported in Windows 8, set to 0] |
| [31-14] | Reserved to 0 |

**MaxNumOfTemporalLayersMinus1:** Indicates the maximum number of temporal layers in a bitstream. A non-zero value indicates that the encoder supports the creation of temporal scalable bitstreams. This specification allows and supports only values between 0 and 3.

**RewriteSupport:** Indicates whether the encoder supports the creation of quality scalable bitstreams that can be converted into bitstreams that conform to one of the non-scalable H.264/AVC profiles, by using a low-complexity rewriting process.

**MaxNumOfCGSLayersMinus1:** Indicates the maximum number of CGS quality layers in a bitstream. A non-zero value indicates that the encoder supports the creation of CGS quality scalable bitstreams. This specification allows and supports only values between 0 and 2, which corresponds to a maximum of three sublayers.

[Not supported in Windows 8]**MaxNumOfMGSSublayers:** Indicates the maximum number of MGS sub-layers allowed in an MGS layer in a bitstream. This specification requires that if supported, only two MGS layers (one base layer and one MGS enhancement layer with multiple sublayers) are present in a spatial layer. A non-zero value indicates that the encoder supports the creation of MGS quality scalable bitstreams. When supported, this specification allows only a value between 2 and 4, which corresponds to a minimum of two and a maximum of four sublayers. Key frame generation shall be supported in MGS.

[Not supported in Windows 8]**AdditionalSNRScalabilitySupport:** Indicates whether additional quality (or SNR) layers are allowed to be present in a spatial enhancement layer. When this field is 1, additional SNR scalability may be introduced based on the capability of quality scalability specified for the base spatial layer. That is, the introduction of quality layers in a spatial enhancement layer is constrained by RewriteSupport, MaxNumOfCGSLayersMinus1, KeyFrameSupport, and MaxNumOfMGSSublayers.

[Not supported in Windows 8]**MaxNumOfSpatialLayersMinus1:** Indicates the maximum number of spatial layers in a bitstream. A non-zero field indicates that the encoder supports the creation of spatial scalable bitstreams. This specification allows and supports only values between 0 and 2, which corresponds to a maximum of three spatial layers.

For encoders that support the generation of only AVC single-layer streams, bmSVCCapabilities shall be set to 0.

## SVC Stream/Layer Configuration

**No Windows 8 Support for Simulcast**

Windows 8 supports only a single stream on each streaming interface. Windows 8 supports only the first stream, stream 0, of the potential four simulcast streams defined in this section.

### Initialization

The encoder indicates the number of simulcast streams and the layering structure associated with each stream in the **bmLayoutPerStream** field in the Probe/Commit Control. These simulcast streams may be AVC single-layer streams, SVC multi-layer streams, or a combination of both. In this specification, at most four simulcast streams are allowed. They are indexed with stream\_id 0, 1, 2 and 3. The following table shows the format description.

Table 10-2 Byte Layout of bmLayoutPerStream Field

|  |  |  |  |
| --- | --- | --- | --- |
| SVC\_STR3[63:48] | SVC\_STR2[47:32] | SVC\_STR1[31:16] | SVC\_STR0[15:0] |

Each 16-bit subfield in **bmLayoutPerStream** describes the layering structure of one simulcast stream. To identify an individual stream, this specification uses the terminology of SVC\_STR*x* (*x* = 0, 1, 2 and 3) where *x* is the stream\_id of the stream.:

SVC\_STR0[15:0], SVC\_STR1[31:16], SVC\_STR2[47:32], SVC\_STR3[63:48]:

|  |  |
| --- | --- |
| Bitfields | Name |
| [2-0] | NumOfTemporalLayers |
| 3 | SNRModeBase [Not supported in Windows 8, set to 0] |
| 4 | SNRModeAttributeBase [Not supported in Windows 8, set to 0] |
| [6-5] | NumberOfSNRLayersMinus1Base [Not supported in Windows 8, set to 0] |
| 7 | SNRMode1st [Not supported in Windows 8, set to 0] |
| 8 | SNRModeAttribute1st [Not supported in Windows 8, set to 0] |
| [11-9] | NumberOfSNRLayers1st [Not supported in Windows 8, set to 0] |
| 12 | SNRMode2nd [Not supported in Windows 8, set to 0] |
| 13 | SNRAttribute2nd [Not supported in Windows 8, set to 0] |
| [15-14] | NumberOfSNRLayers2nd [Not supported in Windows 8, set to 0] |

**NumOfTemporalLayers:**

Indicates the number of temporal layers in the bitstream. This value is effectively the maximum value of the syntax element temporal\_id in the H.264 SVC specification plus 1. For example, if this field is 3, three temporal layers are present in the bitstream, corresponding to temporal\_id 0, 1, and 2. The value 0 indicates that the corresponding stream is not present. The value of this field must not exceed the maximum number of temporal layers specified in **bmSVCCapabilities**.

[Not supported in Windows 8]**SNRModeBase:**

Indicates whether CGS or MGS is used to generate quality layers in the base spatial layer. The value 0 means CGS is used, and 1 means MGS is used. When CGS is used, **SNRModeAttributeBase** indicates whether the rewriting process is enabled. The value 0 means rewriting is disabled, and 1 means rewriting is enabled. When MGS is used, **SNRModeAttributeBase** indicates whether key frame generation is enabled. The value 0 means key frame generation is disabled, and 1 means it is enabled. The use of CGS or MGS mode must follow what was advised in **bmMaxSVCCapabilities**.

[Not supported in Windows 8]**NumberOfSNRLayersMinus1Base:**

Indicates the number of CGS quality layers or MGS sublayers, depending on the value of SNRModeBase, in the base spatial layer. When CGS is used (SNRModeBase is 0), this field effectively corresponds to the values of the syntax element dependency\_id in the H.264 SVC specification. For example, if this field is 2, three CGS layers are present in the base spatial layer in the bitstream, corresponding to dependency\_id 0, 1, and 2. When MGS is used (SNRMode is 1), this field effectively corresponds to the values of the syntax element quality\_id in the H.264 SVC specification. For example, if this field is 2, three MGS sublayers are present in the base spatial layer in the bitstream, corresponding to quality\_id 0, 1, and 2. The value 0 implies that no SNR scalability is introduced in the base spatial layer. The value of this field must not exceed the maximum number of quality layers specified in **bmMaxSVCCapabilities**.

[Not supported in Windows 8]**SNRMode1st:**

Indicates whether CGS or MGS is used to generate additional quality layers in the first spatial enhancement layer (if present). The value 0 means CGS is used, and 1 means MGS is used. When CGS is used, SNRModeAttribute1st indicates whether the rewriting process is enabled. The value 0 means rewriting is disabled, and 1 means rewriting is enabled. When MGS is used, **SNRModeAttribute1st** indicates whether key frame generation is enabled. The value 0 means key frame generation is disabled, and 1 means it is enabled. The use of CGS or MGS mode must be constrained by **bmMaxSVCCapabilities**.

[Not supported in Windows 8]**NumberOfSNRLayers1st:**

Indicates whether spatial scalability is introduced in the bitstream and how additional SNR scalability is used. If the value is 0, spatial scalability is not introduced in the bitstream. If the value is 1, spatial scalability is used but no additional SNR scalability is introduced in the first spatial enhancement layer. If the value is 2 or larger, spatial scalability is used and additional SNR scalability is introduced in the first spatial enhancement layer. In that case, this field indicates the number of CGS quality layers or MGS sublayers, depending on the value of **SNRMode1st**, used in the first spatial enhancement layer. When this field is non-zero, the maximum number of spatial layers advised in **bmSVCCapabilities** must be at least 2. When this field is larger than 1, the use of additional SNR scalability must not exceed the capability of quality scalability specified in **bmMaxSVCCapabilities**.

When CGS is used (**SNRMode1st** is 0), the value of this field effectively corresponds to the values of the syntax element dependency\_id in the H.264 SVC specification. For example, if this field is 3, three CGS layers are present in the first spatial enhancement layer in the bitstream, corresponding to dependency\_id *K*+1, *K*+2, and *K*+3, where *K* is 0 if **SNRModeBase** is 1 and *K* is **NumberOfSNRLayersMinus1Base** if **SNRModeBase** is 0. When MGS is used (**SNRMode1st** is 1), this value effectively corresponds to the values of the syntax element quality\_id in the H.264 SVC specification. For example, if this field is 3, three MGS sublayers are present in the first spatial enhancement layer in the bitstream, corresponding to quality\_id 0, 1, and 2. The value of this field must not exceed the maximum number of quality layers specified in **bmMaxSVCCapabilities**.

[Not supported in Windows 8]**SNRMode2nd**:

Indicates whether CGS or MGS is used to generate additional quality layers in the second spatial enhancement layer (if present). The value 0 means CGS is used, and 1 means MGS is used. When CGS is used, **SNRModeAttribute2nd** indicates whether the rewriting process is enabled. The value 0 means rewriting is not used, and 1 means rewriting is used. When MGS is used, **SNRModeAttribute2nd** indicates whether key frame generation is enabled. The value 0 means key frame generation is disabled, and 1 means key frame generation is enabled. The use of CGS or MGS mode must be constrained by **bmMaxSVCCapabilities**.

[Not supported in Windows 8]**NumberOfSNRLayers2nd:**

Indicates whether the second spatial enhancement layer is introduced in the bitstream and if so, whether additional SNR scalability is used. If the value is 0, the second spatial enhancement layer is not present in the bitstream. If the value is 1, the second spatial enhancement layer exists but no additional SNR scalability is introduced. If the value is 2 or larger, additional SNR scalability is introduced in the second spatial enhancement layer. In that case, the value of this field indicates the number of CGS quality layers or MGS sublayers, depending on the value of **SNRMode2nd**, used in the second spatial enhancement layer. When this field is non-zero, **NumberOfSNRLayers1st** must also be non-zero, and the maximum number of spatial layers advised in **bmSVCCapabilities** must be at least 3. When this field is larger than 1, the use of additional SNR scalability must not exceed the capability of quality scalability specified in **bmMaxSVCCapabilities**.

When CGS is used (**SNRMode2nd** is 0), the value of this field effectively corresponds to the values of the syntax element dependency\_id in the H.264 SVC specification. For example, if this field is 3, three CGS layers are present in the second spatial enhancement layer in the bitstream, corresponding to dependency\_id *K*+1, *K*+2, and *K*+3, where *K* equals 1 if both **SNRModeBase** and **SNRMode1st** are 1; *K* equals (**NumberOfSNRLayersMinus1Base** + 1) if **SNRModeBase** is 0 but **SNRMode1st** is 1, and *K* equals **NumberOfSNRLayers1st** if **SNRModeBase** is 1 but **SNRMode1st** is 0.

When MGS is used (**SNRMode2nd** is 1), this value effectively corresponds to the values of the syntax element quality\_id in the H.264 SVC specification. For example, if this field is 3, three MGS sublayers are present in the second spatial enhancement layer in the bitstream, corresponding to quality\_id 0, 1, and 2. The value of this field must not exceed the maximum number of quality layers specified in **bmMaxSVCCapabilities**.

For each 16-bit subfield, a non-zero value indicates the presence of the corresponding simulcast stream. For encoders that support only one AVC single-layer stream, at most one of the subfields may be set, and it must be set to 1. For encoders that supports SVC but do not support simulcast, at most one of the subfields may be set, and it may be set to a value greater than or equal to 1.

The host should configure simulcast streams starting from the subfield corresponding to the lowest stream\_id. However, the device must be able to handle configurations that result in a gap in stream\_id.

Configuration of SVC and simulcast streams involve a single Probe/Commit Control followed by multiple encoding unit (EU) controls, as specified in the rest of this section.

### Configuration Constraint

In the Probe/Commit Control, the encoder specifies the number of active simulcast streams and the layering layout of each simulcast stream in **bmLayoutPerStream**. The following fields in the Video Frame Descriptor are employed as an indicator for available hardware resources given the degree of SVC scalability and the number of resolution rescalings:

* **dwMaxMBperSecOneResolutionNoScalability** represents the maximum macroblock processing rate when only AVC non-scalable streams are used and when only one resolution is requested (no spatial rescaling).
* **dwMaxMBperSecTwoResolutionsNoScalability**, **dwMaxMBperSecThreeResolutionsNoScalability**, and **dwMaxMBperSecFourResolutionsNoScalability** represent the maximum macroblock processing rate when only AVC non-scalable streams are used and when two (one spatial rescaling), three (two spatial rescaling) and four resolutions (three spatial rescaling) across all layers in all streams are requested.
* **dwMaxMBperSecOneResolutionTemporalScalability**, **dwMaxMBperSecTworResolutionsTemporalScalability**, **dwMaxMBperSecThreeResolutionsTemporalScalability** and **dwMaxMBperSecFourResolutionsTemporalScalability** are used when temporal scalability is employed in streams across which all layers consist of one, two, three, and four resolutions, respectively.
* **dwMaxMBperSecOneResolutionTemporalQualityScalability**, **dwMaxMBperSecTwoResolutionsTemporalQualityScalability**, **dwMaxMBperSecThreeResolutionsTemporalQualityScalability**, and **dwMaxMBperSecFourResolutionsTemporalQualityScalability** are used when both temporal and quality scalability are employed in streams across which all layers consist of one, two, three, and four resolutions.
* **dwMaxMBperSecOneResolutionFullScalability**, **dwMaxMBperSecTwoResolutionsFullScalability**, **dwMaxMBperSecThreeResolutionsFullScalability**, and **dwMaxMBperSecFourResolutionsFullScalability** are used when full SVC scalability is requested.

### Initialization and Run-Time Encoding Control

After the probe is done, the caller uses encoding unit control to deliver detailed per-layer configuration before issuing the Commit control. A layer in a stream is uniquely identified by **wLayerOrViewID**, a combination of four subfields: dependency\_id (bits 0-2), quality\_id (bits 3-6), temporal\_id (bits 7-9) as defined in the H.264 specification, and stream\_id (bits 10-12) as defined in Section 10.3.1. **wLayerOrViewID** defines the scope of EU controls, which allows different configurations for different layers in different streams. When a control is issued for a specific **wLayerOrViewID**, the control must be applied to all layers that have dependency\_id, quality\_id, temporal\_id and stream\_id as given by the subfields in **wLayerOrViewID**.

In Windows 8, where only temporal layers are supported, only bits 0-2 of **wLayerOrViewID** are supported.

To reduce the number of calls, applications may use wildcard masks. A wildcard mask is a **wLayerOrViewID** where one or more of the subfields has all bits set to 1. For example, **wLayerOrViewID** can be set to 0x0007 to indicate that the scope of configuration applies to all temporal layers of the stream.

### Sub-Bitstream Definition

Several Encoding Units apply to sub-bitstreams instead of individual SVC layers. The sub-bitstream is determined as follows. Let **wLayerOrViewID** be the result from a GET\_CUR request issued to the EU\_SELECT\_LAYER\_CONTROL control. Then, the sub-bitstream is given by all the NAL units that meet all the following conditions:

1. stream\_id is equal to the stream\_id indicated by **wLayerOrViewID**.
2. temporal\_id is less or equal than temporal\_id indicated by **wLayerOrViewID**.
3. dependency\_id is less than dependency\_id indicated by **wLayerOrViewID**, or dependency\_id is equal to dependency\_id indicated by **wLayerOrViewID** and quality\_id is less or equal than quality\_id indicated by **wLayerOrViewID**.

For SVC, a **wLayerOrViewID** wildcard mask is not allowed.

# MVC and Simulcast Support [Not Supported in Windows 8]

This section provides technical background, detailed descriptions, and examples to illustrate how to support MVC and/or simulcast in this specification. Developers may skip this section if the encoder does not support the generation of MVC bitstreams. In that case both **bmMaxMVCCapabilities** and **bmLayoutPerStream** shall be set to 0.

## MVC Overview

The MVC design bears certain similarity as SVC. Within an access unit, there is one "base view component" that is formatted as an ordinary H.264/AVC coded picture. Within the same access unit, there are one or more additional view components that each represents an additional "non-base view" of a multiview encoding for the same instant in time. Within a stream, each view is uniquely identified by the syntax\_element view\_id, starting from 0 (corresponding to the base view) to the number of views minus 1. Similar to the constraint posed in SVC streams, the value of temporal\_id (when temporal scalability is supported) must be assigned starting from 0 and increased continuously. Thus for a particular layering structure, the values of view\_id and temporal\_id associated with a view can be determined without ambiguity and used as an unique identifier for that view.

Aside from minor differences in high-level syntax, the encoding format for a non-base view is basically the same as that of the base view. To take advantage of correlation between adjacent view components, the view components of other views within the same access unit and the preceding view components for the same non-base view (in decoding order) can be used as reference pictures. Thus, both temporal prediction within a single view (across different access units) and inter-view prediction across different views (within the same access unit) are supported in MVC.

## MVC Capability Advertisement

The encoder notifies the MVC capabilities in **bmMaxMVCCapabilities** in the Video Frame Descriptor. The following is the format description:

|  |  |
| --- | --- |
| Bitfields | Name |
| [2-0] | MaxNumOfTemporalLayersMinus1 |
| [10-3] | MaxNumOfViewsMinus1 |
| [31-11] | Reserved |

**MaxNumOfTemporalLayersMinus1:** Indicates the maximum number of temporal layers in a bitstream. A non-zero value of this field indicates that the encoder supports the creation of temporal scalable bitstreams. This specification allows and supports only values between 0 and 3.

**MaxNumOfViewsMinus1:** Indicates the maximum number of view components in a bitstream. A non-zero value of this field indicates the encoder supports the generation of MVC bitstreams. This specification allows and supports only values between 0 and 127. For encoders that support only the Stero High profile, this field is equal to 1 (that is, the number of supported views is limited to two).

For encoders that support only the generation of ordinary AVC single-layer streams, **bmMaxMVCCapabilities** shall be set to 0.

## MVC Stream/View Configuration

### Initialization

The encoder indicates the number of simulcast streams and the structure associated with each stream in **bmLayoutPerStream** in the Probe/Commit Control. The simulcast streams can be AVC single-layer streams, MVC multi-view streams, or a combination of both. In this specification, at most four MVC simulcast streams are allowed and supported, and they are indexed with stream\_id 4, 5, 6, and 7. The following is the format description.

|  |  |  |  |
| --- | --- | --- | --- |
| MVC\_STR7[63:48] | MVC\_STR6[47:32] | MVC\_STR5[31:16] | MVC\_STR4[15:0] |

Each 16-bit field (MVC\_STR*x,* where *x* = 4, 5, 6 and 7) in **bmLayoutPerStream** describes the structure of one simulcast stream. Each simulcast stream is identified by *x*, or referred to as stream\_id hereinafter.

MVC\_STR4[15:0], MVC\_STR5[31:16], MVC\_STR6[47:32], MVC\_STR7[63:48]:

|  |  |
| --- | --- |
| Bitfields | Name |
| [0-2] | NumOfTemporalLayers |
| [3-10] | NumberOfViewsMinus1 |
| [11-15] | Reserved |

**NumOfTemporalLayers:** Indicates the number of temporal layers in the bitstream. This value effectively corresponds to the values of syntax element temporal\_id in the H.264 MVC specification. For example, if this field is 3, three temporal layers, corresponding to temporal\_id 0, 1, and 2, are present in the bitstream. The value 0 indicates the corresponding stream is not present. The value of this field must not exceed the maximum number of temporal layers specified in **bmMaxMVCCapabilities**.

**NumOfViewsMinus1:** Indicates the number of views in the bitstream. This value effectively corresponds to the values of syntax element view\_id in the H.264 MVC specification. For example, if this field is 1, two view components, corresponding to view\_id 0 and 1, are present in the bitstream. The value of this field must not exceed the maximum number of view components specified in **bmMaxMVCCapabilities**.

Remark: For each 16-bit subfield, a non-zero value indicates the presence of the corresponding simulcast stream. For encoders that support only the generation of single ordinary AVC single-layer streams, one and only one of the subfields may be set to 1. For encoders that support MVC but not simulcast capability, one and only one of the subfields may be greater than or equal to 1.

The host should configure simulcast streams starting from the subfield corresponding to the lowest stream\_id. However, the device must be able to handle configurations that result in a gap in stream\_id.

### Configuration Constraint

In the Probe/Commit Control, the encoder specifies the number of active simulcast streams and the MVC configuration corresponding to each simulcast stream in **bmLayoutPerStream**. The valid configurations are constrained by the maximum number of macroblocks per second similar to the SVC case specified in Section 10.3.2. When the encoder is constrained to the MVC Stereo High profile or the MVC Multiview profile, the maximum number of macroblocks per second for temporal scalability is considered.

### Initialization and Run-Time Encoding Control

After the probe is done, the caller uses encoding unit control to deliver detailed per-layer or per-view configuration before issuing the Commit control. A layer/view in a stream is uniquely identified by **wLayerOrViewId**, a combination of three subfields: view\_id (bits 0-6) and temporal\_id (bits 7-9) as defined in the H.264 MVC specification, and stream\_id (bits 10-12) as defined in Section 10.3.1. **wLayerOrViewID** defines the scope of EU controls, which allows different configurations for different layers/views in different streams. When a control is issued for a specific **wLayerOrViewID**, the control must be applied to all layers/views that have view\_id, temporal\_id, and stream\_id as given by the subfields in **wLayerOrViewID**.

To reduce the number of calls, applications may use wildcard masks. A wildcard mask is a **wLayerOrViewID** where one or more of the subfields has all bits set to 1. For example, **wLayerOrViewID** can be set to 0x1C07, 0x1FF8, and 0x1FFF to indicate that the scope of configuration applies to all temporal layers, all view components, and all layers and views across all simulcast streams, respectively.

After streaming starts, the application can also use EU controls to perform run-time configuration changes.

# Driver Behavior Notes

In Windows 8, the class driver buffers data for an entire Access Unit , using a buffer allocated by the application. The class driver time-stamps the data and sends it up as one sample. To get the maximum buffer size that is needed, the application queries the value of **dwMaxVideoFrameSize** in the Probe control.

Appendix A

Summary of Supported Features

in Windows 8\*

\*Disclaimer: The support listed in the table below is an indication of intent for inclusion in Windows 8, not a guarantee.

|  |  |  |
| --- | --- | --- |
| H.264 Feature | Supported in Windows 8 | Notes |
| H.264 Format Descriptor |  |  |
| bMaxCodecConfigDelay |  |  |
| bmSupportedSliceModes |  |  |
| bmSupportedSyncFrameTypes |  |  |
| bResolutionScaling  (nee bDynamicResolutionScaling) |  |  |
| bSimulcastSupport |  |  |
| bmSupportedRateControlModes |  |  |
| wMaxMBPerSec… |  |  |
| H.264 Frame Descriptor |  |  |
| wWidth, wHeight |  |  |
| wSARwidth, wSARheight |  |  |
| wProfile |  | Windows 8 provides support for the following H.264 profiles: 0x4240: Constrained Baseline Profile 0x4200: Baseline Profile 0x4D00: Main Profile 0x6400: High Profile  **See** [**eAVEncH264VProfile enumeration**](http://msdn.microsoft.com/en-us/library/windows/desktop/dd318776(v=vs.85).aspx) **for details.** |
| wConstrainedToolset |  |  |
| bLevelIDC |  |  |
| bmSupportedUsages |  | 1: Real-time/UCConfig mode 0.  2: Real-time/UCConfig mode 1.  3: Real-time/UCConfig mode 2Q.  4: Real-time/UCConfig mode 2S.  5: Real-time/UCConfig mode 3.  6-8: Reserved.  9-16: Broadcast modes.  17: File Storage mode with I and P slices (e.g., IPPP)  18: File Storage mode with I, P, and B slices (e.g., IB…BP)  19: File storage with all-I-frame mode.  20-24: File storage modes.  25: MVC Stereo High mode.  26 : MVC Multivew mode.  0, 27-255: Reserved; set to 0. |
| bmCapabilities |  |  |
| bmMaxSVCCapabilities |  |  |
| bmMaxMVCCapabilities |  |  |
| dwMinBitRate |  |  |
| dwMaxBitRate |  |  |
| bNumFrameIntervals, etc. |  | All existing UVC 1.0 frame interval fields are supported (no continuous frame interval support) |
| Probe & Commit |  |  |
| bUsage |  |  |
| bBitDepthLuma |  |  |
| bmSetting |  |  |
| bMaxNumberOfRefFramesPlus1 |  |  |
| bmRateControlModes |  |  |
| bmLayoutPerStream |  |  |
| Encoding Unit controls |  |  |
| Encoding Unit Descriptor |  |  |
| Select Layer |  |  |
| Video Resolution |  |  |
| Profile and Toolset |  |  |
| Minimum Frame Interval |  |  |
| Slice Mode Control |  |  |
| Rate Control Mode |  |  |
| Average Bitrate |  |  |
| CPB Size |  |  |
| Peak Bit Rate |  |  |
| Quantization Parameter |  |  |
| Synchronization and Long Term Reference |  |  |
| Priority ID |  |  |
| Start or Stop Layer |  |  |
| Other Controls |  |  |
| Request Error Code |  | Not fully supported.  See [Request Error Code Control](#_Request_Error_Code) for details. |

Appendix B

Technical Decisions

1. Why UVC 1.5 instead of UVC 1.1 for H.264 support?
   1. UVC 1.1 only adds support for a generic streaming payload. We still need discovery of the cameras capabilities (resolutions, frame rates, codec features, etc.) as well as Probe and Commit negotiation to confirm that the current request from the application is supported by the camera. Finally, run-time control of the encoder is required.
2. SEI picture timing. Is this used?
   1. The application would need to handle this by parsing the NALU and put it to the file container if necessary for A/V sync.
3. Why 2-pin solution?
   1. This is simpler for both the camera and the consuming application. The consuming application can use the preview PIN to render the preview without having to decode the H.264 stream. Finally, keeping the uncompressed preview stream separate from the encoded stream allows current solutions that rely on unencoded frames to continue to work as they currently work
4. Why no loop filter mode?
   1. Loop filter mode is used to aid software decoding but it is not deemed necessary for this spec with the proliferation of hardware-assisted H.264 decoders, which will be available on most PCs in the future.
5. Why no still capture from the H.264 pin?
   1. Still capture should happen from the preview pin using one of the three methods specified in the USB Video Class 1.0 specification.
6. Note on bmRateControlMethods = bConstantQP
   1. It is expected that for certain use cases the application may wish to frequently change the target QP. How frequently this change can occur depends on the host driver and camera implementations and is not governed by this specification.
7. Why no FMO and ASO?
   1. These features are assumed to be present for the baseline profile. Although the codec may use these features, no specific control for FMO or ASO is offered in this specification.
8. Why no control of in-loop deblocking filter?
   1. Adding this control would require three more parameters, and it is expected that client applications will not typically need/want to control the in-loop deblocking filter.
9. Why are the values I set in COMMIT not sticking?
   1. During the COMMIT phase it is advisable to do a GET\_CUR after each SET\_CUR for the new H.264 attributes to make sure they stick. Because of the complexity of device behavior, the device may not always configure itself as asked.
10. What does bLevelIDC represent?

bLevelIDC indicates the minimum Level that a decoder must support to decode a stream generated from a specific frame descriptor using the maximum resolution and maximum bitrate supported. The codec should tag the stream using a Level according to the maximum bit rate and frame rate negotiated during Probe/Commit to avoid unnecessary requirements for decoding.

1. My codec does not support long-term reference(LTR). How does the device handle this?

If the device does not support long-term reference frame, the device must set bits 2 and 4 of the bmSyncFrameTypes Bitmap in the H.264 Video Frame Format Descriptor to 0 to indicate LTR is not supported. Moreover, if the host uses the Synchronization and Long-Term Reference Frame Control EU (section 8.13) with **bSyncFrameType** set to 2 or 4, the device must return a protocol STALL to indicate it is unsupported.

1. What is Resolution Scaling?

Resolution scaling allows changes to the resolution being streamed without interrupting video streaming. In other words, changing the resolution shall not result in a pause in streaming, and does not require executing a new Probe/Commit. The source data frame is the same but it should be rescaled prior to H.264 encoding so that the decoded H.264 resolution matches what was requested by the host.

The SPS header in the H.264 stream contains the resolution information. When the resolution is changed, the codec will insert a new SPS, PPS, and IDR frame. At the receiving end, the application can decide whether to rescale the resolution in order to display what was originally set in Probe/Commit, or to simply display the new resolution (which will never exceed what was set in Probe/Commit). **bResolutionScaling** in the H.264 Video Format Descriptor indicates whether the device supports resolution scaling during run time (streaming). Resolution scaling is an alternative way for the host to deal with bandwidth fluctuations. During run time the host may use the Video Resolution EU to request that the resolution of the substream given by **wLayerOrViewID** be changed to a different resolution than what is currently being streamed. The new resolution can never exceed the resolution that was set at Probe/Commit. The requirement is that a change in resolution does not interrupt video streaming nor cause glitches in the video. If the device can support this, then **bResolutionScaling** should be set to 0x01. If it cannot, then **bResolutionScaling** should be set to 0x00.

1. What is **wLayerOrViewID** when there is only a single-layer AVC stream?

When streaming only one layer (one AVC stream), **wLayerOrViewID** is 0. In this case, **wLayerOrViewID** is simply used for specifying the layerID in the Encoding Unit. The H.264 stream itself should not contain a prefix NAL unit to avoid this NAL overhead.

1. Why are the chromal offsets for QPPrime limited to -8 to 7 instead of -12 to 12 as specified in the H.264 spec?

These offsets are represented by only 4 bits each to fit nicely with QP'Y into 2 bytes. It is not expected that applications will desire extreme chromal offsets from the base QP'Y value so -8 to +7 is sufficient.

Appendix C

Examples

UVC 1.5 H.264 Implementation Example

# Overview

This section provides specific examples on how to populate transactions between the host and device in order to discover, negotiate, and control an H.264 streaming device over USB.

# Payload

This specification adds a new EOS flag in the Payload Header and modifies the meaning of the STI flag when streaming H.264. The following table indicates how to populate the H.264 Payload Header when streaming H.264 with multiple slices per access unit. The example below demonstrates 8 slices per frame, which is the maximum number of slices per frame recommended for Window 8. The STI bit indicates that Frame# 1 is an IDR frame. The EOS bit is set for every payload because, in this example, each slice fits in a single payload transfer. Note that this table does not show values for PTS and SCR, which are required for every payload transfer.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Frame # | Slice # | FID (D0) | EOF (D1) | EOS (D4) | STI (D5) |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 2 | 0 | 0 | 1 | 1 |
| 1 | 3 | 0 | 0 | 1 | 1 |
| 1 | 4 | 0 | 0 | 1 | 1 |
| 1 | 5 | 0 | 0 | 1 | 1 |
| 1 | 6 | 0 | 0 | 1 | 1 |
| 1 | 7 | 0 | 0 | 1 | 1 |
| 1 | 8 | 0 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 | 1 | 0 |
| 2 | 2 | 1 | 0 | 1 | 0 |
| 2 | 3 | 1 | 0 | 1 | 0 |
| 2 | 4 | 1 | 0 | 1 | 0 |
| 2 | 5 | 1 | 0 | 1 | 0 |
| 2 | 6 | 1 | 0 | 1 | 0 |
| 2 | 7 | 1 | 0 | 1 | 0 |
| 2 | 8 | 1 | 1 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 | 0 |
| 3 | 2 | 0 | 0 | 1 | 0 |
| 3 | 3 | 0 | 0 | 1 | 0 |
| 3 | … | … | … | … | … |

# Format Descriptor

The Format Descriptor defines properties that apply to the specified format regardless of the resolution, frame rate, or sub-profile being streamed. There are 20 fields at the end of the Format Descriptor, used to indicate the overall capabilities of the encoder when streaming a mix of simulcast and SVC settings. Windows 8 does not support SVC or simulcast, so most of the fields are zero. The two exceptions are the single-stream single-layer stream, which all cameras must support, and **wMaxMBperSecOneResolutionTemporalScalability**, which is used to provide throughput information when streaming two or more temporal layers as noted in Section 10.3.2 of this specification.

|  |  |  |
| --- | --- | --- |
| Descriptor Field | Value | Notes |
| bLength | 52 | Length in bytes |
| bDescriptorType | 0x24 | CS\_INTERFACE |
| bDescriptorSubtype | 0x13 | VS\_FORMAT\_H264 descriptor subtype |
| bFormatIndex | **1** |  |
| bNumFrameDescriptors | 0x02 | Example has 2 Frame Descriptors |
| bDefaultFrameIndex | 0x01 | 1 |
| bMaxCodecConfigDelay | 0x01 | 1 frame |
| bmSupportedSliceModes | 0x00 | None |
| bmSupportedSyncFrameTypes | 0x21 | IDR Frame; Gradual Decoder Refresh Frames |
| bResolutionScaling | 0x03 | Arbitrary Scaling |
| bSimulcastSupport | 0x00 | 1 endpoint, 1 stream |
| bmSupportedRateControlModes | 0x0D | VBR low delay; Constant QP; Global VBR low delay |
| wMaxMBperSecOneResolutionNoScalability | 0x006C | 108,000 MB/sec. 720p 30fps |
| wMaxMBperSecTwoResolutionsNoScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecThreeResolutionsNoScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecFourResolutionsNoScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecOneResolutionTemporalScalability | 0x006C | More than one temporal layer |
| wMaxMBperSecTwoResolutionsTemporalScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecThreeResolutionsTemporalScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecFourResolutionsTemporalScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecOneResolutionTemporalQualityScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecTwoResolutionsTemporalQualityScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecThreeResolutionsTemporalQualityScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecFourResolutionsTemporalQualityScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecOneResolutionTemporalSpatialScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecTwoResolutionsTemporalSpatialScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecThreeResolutionsTemporalSpatialScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecFourResolutionsTemporalSpatialScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecOneResolutionFullScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecTwoResolutionsFullScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecThreeResolutionsFullScalability | 0x0000 | [Must be 0 in Windows 8] |
| wMaxMBperSecFourResolutionsFullScalability | 0x0000 | [Must be 0 in Windows 8] |

# Frame Descriptor

Frame Descriptors are required for each unique combination of format, resolution, and H.264 profile (defined by **wProfile** plus **wConstrainedToolset**). Given these three constraints, additional capabilities such as **bmSupportedUsages**, and **bmMaxSVCCapabilities**, are enumerated.

## Constrained Baseline 720p

|  |  |  |
| --- | --- | --- |
| Descriptor Field | Value | Notes |
| bLength | 52 | Length in bytes |
| bDescriptorType | 0x24 | CS\_INTERFACE |
| bDescriptorSubtype | 0x14 | VS\_FRAME\_H264 descriptor subtype |
| bFrameIndex | **1** |  |
| wWidth | 1280 |  |
| wHeight | 720 |  |
| wSARwidth | 1 | 1-to-1 aspect ratio |
| wSARheight | 1 |  |
| wProfile | 0x4240 | Constrained Baseline Profile |
| bLevelIDC | 0x20 | Level 3.2 |
| wConstrainedToolset | 0 | Constraints are part of Baseline |
| bmSupportedUsages | 0x00010003 | Real-Time modes 0 and 1, File Storage I-P mode |
| bmCapabilities | 0x0029 | CAVLC; separate QP for luma/chroma; no picture reordering |
| bmMaxSVCCapabilities | 0x00000001 | 2 temporal layers |
| bmMaxMVCCapabilities | 0 | [Must be 0 in Windows 8] |
| dwMinBitRate | 500,000 | Used to calculated AverageBitRate |
| dwMaxBitRate | 20,000,000 | Used to calculated AverageBitRate |
| dwDefaultFrameInterval | 333,333 | 30 fps |
| bNumFrameIntervals | 2 |  |
| dwFrameInterval[0] | 333,333 | 30 fps |
| dwFrameInterval[1] | 666,666 | 15 fps |

## UC Constrained High 720p

|  |  |  |
| --- | --- | --- |
| Descriptor Field | Value | Notes |
| bLength | 52 | Length in Bytes |
| bDescriptorType | 0x24 | CS\_INTERFACE |
| bDescriptorSubtype | 0x14 | VS\_FRAME\_H264 descriptor subtype |
| bFrameIndex | **2** |  |
| wWidth | 1280 |  |
| wHeight | 720 |  |
| wSARwidth | 1 | 1-to-1 aspect ratio |
| wSARheight | 1 |  |
| wProfile | 0x6400 | High Profile |
| bLevelIDC | 0x20 | Level 3.2 |
| wConstrainedToolset | 1 | UC Constrained High Toolset |
| bmSupportedUsages | 0x00020003 | Real-Time modes 0 and 1, File Storage IB….BP mode |
| bmCapabilities | 0x002A | CABAC; separate QP for luma/chroma; No picture reordering |
| bmMaxSVCCapabilities | 0x00000001 | 2 temporal layers |
| bmMaxMVCCapabilities | 0 | [Must be 0 in Windows 8] |
| dwMinBitRate | 500,000 |  |
| dwMaxBitRate | 20,000,000 |  |
| dwDefaultFrameInterval | 333,333 | 30 fps |
| bNumFrameIntervals | 2 |  |
| dwFrameInterval[0] | 333,333 | 30 fps |
| dwFrameInterval[1] | 666,666 | 15 fps |

## Constrained Baseline 360p

|  |  |  |
| --- | --- | --- |
| Descriptor Field | Value | Notes |
| bLength | 52 | Length in bytes |
| bDescriptorType | 0x24 | CS\_INTERFACE |
| bDescriptorSubtype | 0x14 | VS\_FRAME\_H264 descriptor subtype |
| bFrameIndex | **3** |  |
| wWidth | 640 |  |
| wHeight | 360 |  |
| wSARwidth | 1 | 1-to-1 aspect ratio |
| wSARheight | 1 |  |
| wProfile | 0x4240 | Constrained Baseline Profile |
| bLevelIDC | 0x20 | Level 3.2 |
| wConstrainedToolset | 0 | Constraints are part of Baseline |
| bmSupportedUsages | 0x00010003 | Real-Time modes 0 and 1, File Storage I-P mode |
| bmCapabilities | 0x0029 | CAVLC; separate QP for luma/chroma; no picture reordering |
| bmMaxSVCCapabilities | 0x00000001 | 2 temporal layers |
| bmMaxMVCCapabilities | 0 | [Must be 0 in Windows 8] |
| dwMinBitRate | 300,000 |  |
| dwMaxBitRate | 12,000,000 |  |
| dwDefaultFrameInterval | 333,333 | 30 fps |
| bNumFrameIntervals | 2 |  |
| dwFrameInterval[0] | 333,333 | 30 fps |
| dwFrameInterval[1] | 666,666 | 15 fps |

# 

# Probe and Commit

The examples below show a subset of the fields in the Probe/Commit structure for two different Usages. Note that all legacy fields from the UVC 1.1 specification must be present; however, specific legacy fields that may appear relevant to H.264 are not actually used. These unused fields are: **bmHint**, **wKeyFrameRate**, **wPFrameRate**, **wCompQuality**, **wCompWindowSize**, **wDelay**, **bPreferedVersion**, **bMinVersion**, **bMaxVersion**.

## Negotiating 360p, 15 fps H.264 for Chat

**bUsage = UCConfig mode 0**

|  |  |  |
| --- | --- | --- |
| PROBE/COMMIT Field | Value | Notes |
| bmFramingInfo | 0x03 | FID and EOS are always present in Payload Header. |
| dwMaxPayloadTransferSize | 1024 | For H.264 streaming endpoint. |
| dwMaxVideoFrameSize | 200,000 | Max frame size allowed for the negotiated resolution specified by bFrameIndex at minimum dwFrameInterval (in bytes). |
| dwClockFrequency | 150,000,000 | Device clock frequency. |
| bFormatIndex | 1 | H.264 AVC. (Any enumerated H.264 bFormatIndex). |
| bFrameIndex | 3 | 360p. (Any enumerated bFrameIndex for the selected Format). |
| dwFrameInterval | 666,666 | 15 fps. (Any enumerated dwFrameInterval for the selected bFrameIndex). |
| bUsage | 1 | UCConfig mode 0. Must be supported by the Format and Frame index selected. |
| bBitDepthLuma | 0 | [Must be 0 in Windows 8.] |
| bmSetting | 0 | [Must be 0 in Windows 8.] |
| bMaxNumberofRefFramesPlus1 | 0 | [Must be 0 in Windows 8] |
| bmRateControlModes | 1 | VBR with low delay. Note that Windows 8 supports only a single stream, therefore only bits D3-D0 should be set. |
| bmLayoutPerStream | 1 | One additional temporal layer on a single stream. Note that Windows 8 does not support SVC or Simulcast so only the first three bits are used. |

## Negotiating 720p, 30 fps H.264 for Sharing

**bUsage = File Storage mode with I,P, and B**

|  |  |  |
| --- | --- | --- |
| PROBE/COMMIT Field | Value | Notes |
| bmFramingInfo | 0x03 |  |
| dwMaxPayloadTransferSize | 1024 |  |
| dwMaxVideoFrameSize | 260,000 |  |
| dwClockFrequency | 150,000,000 |  |
| bFormatIndex | 1 | H.264 SVC |
| bFrameIndex | 1 | 720p |
| dwFrameInterval | 333,333 | 30 fps |
| bUsage | 17 | File storage IPP…P |
| bBitDepthLuma | 0 | [Must be 0 in Windows 8] |
| bmSetting | 0 | [Must be 0 in Windows 8] |
| bMaxNumberofRefFramesPlus1 | 0 | [Must be 0 in Windows 8]] |
| bmRateControlModes | 2 | CBR |
| bmLayoutPerStream | 1 | One additional temporal layer on a single stream |

# Encoding Units

Windows 8 does not support the full set of Encoding Units (EUs) defined in this specification. The table below indicates which EU Controls are supported, including which Requests.

|  |  |
| --- | --- |
| **EU Control** | **USB Mandatory Requests** |
| **Implemented in Windows 8** | |
| Synch & Long Term Reference | SET\_CUR, GET\_CUR, ~~GET\_MIN, GET\_MAX~~ |
| Average Bit Rate | SET\_CUR, GET\_CUR, ~~GET\_MIN, GET\_MAX, GET\_DEF~~ |
| CPB Size | SET\_CUR, GET\_CUR, ~~GET\_MIN, GET\_MAX, GET\_DEF~~ |
| Select Layer | SET\_CUR, GET\_CUR |
| Video Resolution | SET\_CUR, GET\_CUR, ~~GET\_MIN, GET\_MAX, GET\_DEF~~ |
| Min Frame Interval | SET\_CUR, GET\_CUR, ~~GET\_MIN, GET\_MAX, GET\_DEF~~ |
| Quantization Parameter | SET\_CUR, GET\_CUR, ~~GET\_MAX, GET\_MIN, GET\_DEF~~ |
| **Not implemented in Windows 8 (Device should STALL)** | |
| Priority ID | SET\_CUR, GET\_CUR |
| Slice Mode | SET\_CUR, GET\_CUR, ~~GET\_DEF~~ |
| Profile & Toolset | SET\_CUR, GET\_CUR, ~~GET\_DEF~~ |
| Peak Bit Rate | SET\_CUR, GET\_CUR, ~~GET\_MIN, GET\_MAX, GET\_DEF~~ |
| Start/Stop | SET\_CUR, GET\_CUR |
| Rate Control Mode | SET\_CUR, GET\_CUR, ~~GET\_DEF~~ |

## Example 1

This example presumes that a single-layer 360p @ 30 fps H.264 AVC stream has been negotiated during Probe and Commit, and that the application wants to lower the bit rate due to variable network conditions during a web chat. During the chat, the application inserts an IDR frame in the stream as a new party joins. Later, the network conditions improve dramatically and the application moves to a constant quality mode.

The sample transactions in this example are representative of all Encoding Unit Controls defined in the specification. They all require that a successful SET\_CUR(COMMIT) operation has already been executed. If GET\_CUR is called on any Control before the initial SET\_CUR(COMMIT), the GET\_CUR state is undefined and the request shall stall with a “Wrong State” error code.

### Average Bit Rate and CPB Size

Average Bit Rate and CPB size are frequently set before streaming starts. They also may be set subsequent to streaming, as in this example. Because they are coupled, care should be taken in how these controls are used.

#### Set the Average Bit Rate

The Average Bit Rate control is used to specify the average bit rate of the current layer(s). For this example, there is only one layer so it applies to the entire stream. The Average Bit Rate control is used in all supported **bRateControlModes** modes except Constant QP. For this example we will assume that the encoder is configured to use variable bit rate (VBR) low delay.

SET\_CUR(EU\_AVERAGE\_BITRATE\_CONTROL) to 400,000 bps.

|  |  |
| --- | --- |
| Field | Value |
| dwAverageBitRate | **0x0190** |

#### Set the CPB Size

The CPB Size control is used to specify the CPB size of the current layer(s). As with the Average Bit Rate control, this control is used in all supported **bRateControlModes** modes except Constant QP. For AVC streams, as is the case in this example, or when in Global VBR rate-control mode, **dwCPBSize** is SET/GET for the overall stream.

SET\_CUR (EU\_CPB\_SIZE\_CONTROL) to 500 milliseconds. Note that:

**dwCPBsize** = ( **dwAverageBitRate** (bps) \* leaky bucket size (seconds)) / 16  
 = (400000 \* 0.5) / 16 = 12

|  |  |
| --- | --- |
| Field | Value |
| dwCPBsize | **0x30D4** |

### New Sync Frame

The Synchronization and Long-Term Reference Frame Control is used to insert a specific frame type into the H.264 stream. In this example, we will instruct the encoder to insert a single IDR frame to support the addition of a new party to a chat session.

SET\_CUR (EU\_SYNC\_REF\_FRAME\_CONTROL)

|  |  |
| --- | --- |
| Field | Value |
| bSyncFrameType | **1** |
| wSyncFrameInterval | **0** |
| bGradualDecoderRefresh | **0** |

### Constant QP Mode

This example includes the case where bandwidth is not the determining limit for the encoded stream. In this case the application chooses to provide constant quality.

#### Change the Rate Control Mode

Before changing the rate control mode, the application should confirm that the desired mode is supported by the encoder as declared in **bmSupportedRateControlModes** in the Video Format Descriptor. In this example, Constant QP is supported so the application is free to select this mode.

SET\_CUR(EU\_RATE\_CONTROL\_MODE\_CONTROL)

|  |  |
| --- | --- |
| Field | Value |
| bRateControlMode | **0x03** |

#### Set the Quantization Parameters

The rate control mode is now set to Constant QP with device default values for QP. At this point the application may want to GET the current QP values, but for this example we will simply go ahead and change them to what we want. Specifically:

* QP'Y = 32
* chroma\_qp\_index\_offset = 5
* second\_chroma\_qp\_index\_offset = 5

Each of the three fields in the Quantization Parameter control consists of 3 values: QP for the luminance, and two offset indexes to control the QP for the chroma channels. This aligns with QP control in the H.264 specification with the exception that the ranges for chroma\_qp\_index\_offset and second\_chroma\_qp\_index\_offset are less than those in the H.264 specification. Support for these two offset indexes is declared by **bmCapabilities** in the Video Frame Descriptor.

SET\_CUR (EU\_QUANTIZATION\_PARAMS\_CONTROL)

|  |  |
| --- | --- |
| Field | Value |
| wQpPrime\_I | **0x5520** |
| wQpPrime\_P | **0x5520** |
| wQpPrime\_B | **0x0000** |

The QP values for B frames is not included because the current streaming format does not support B frames.

## Example 2

The second example presumes a 720p @ 30 fps H.264 SVC stream with two temporal layers. As in the previous example, we will change the bit rate of the stream, but in this case we need to change the bit rate for the base layer separate from the entire stream. For this example we assume the following initial bitrate:

* Base layer = 1 Mbps,
* Base layer + Enhancement layer = 1.8 Mbps

And want to reduce to the following bitrates:

* Base layer = 800 kbps
* Base layer + Enhancement layer = 1.2 Mbps

### Select Layer

As with all Encoding Units, in order to have a control apply to a particular sub-bitstream (for example, an enhancement layer and its dependencies), the application must first select that layer using the Select Layer control.

1. Select the base layer:

SET\_CUR (EU\_SELECT\_LAYER\_CONTROL)

|  |  |
| --- | --- |
| Field | Value |
| wLayerOrViewID | **0x0000** |

With the base layer selected, the application can now change the bit rate for that layer.

1. Set dwAverageBitRate to 800 kbps

SET\_CUR (EU\_AVERAGE\_BITRATE\_CONTROL)

|  |  |
| --- | --- |
| Field | Value |
| dwAverageBitRate | **0xC3500** |

Note that at this point the overall bitrate is still 1.8 Mbps.

1. Next we select the first enhancement layer…

SET\_CUR (EU\_SELECT\_LAYER\_CONTROL)

|  |  |
| --- | --- |
| Field | Value |
| wLayerOrViewID | **0x0001** |

1. …and set the bit rate of the associated sub-bitstream to 1.2 Mbps

SET\_CUR (EU\_AVERAGE\_BITRATE\_CONTROL) to 1200 kbps.

|  |  |
| --- | --- |
| Field | Value |
| dwAverageBitRate | **0x124F80** |

1. See the specification “Unified Communication Specification and Interfaces for H.264/MPEG-4 AVC and SVC Encoder Implementation,” referenced in Section 6.1. [↑](#footnote-ref-2)
2. See the specification “Unified Communication Specification and Interfaces for H.264/MPEG-4 AVC and SVC Encoder Implementation,” referenced in Section 1. [↑](#footnote-ref-3)
3. Note that this condition is described from the perspective of a hypothetical reference decoder (HRD), and the time to move the bits from the encoder’s leaky bucket to the decoder’s input buffer is not accounted for. Also, the means by which the decoding time for a picture is determined is not covered here, although it may be indicated by picture timing SEI messages. [↑](#footnote-ref-4)
4. TDQId = ((dependency\_id << 4) + quality\_id) << 3 + temporal\_id [↑](#footnote-ref-5)