

VDPAU

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Video Decode and Presentation API for Unix</b>	<b>1</b>
1.1	Introduction	1
1.2	API Partitioning	1
1.3	Object Types	1
1.3.1	Device Type	2
1.3.2	Surface Types	2
1.3.3	Transfer Types	2
1.4	Data Flow	2
1.5	Entry Point Retrieval	3
1.5.1	Philosophy	3
1.6	Multi-threading	3
1.7	Surface Endianness	4
1.8	Video Decoder Usage	5
1.8.1	MPEG-1 and MPEG-2	5
1.8.2	H.264	5
1.8.3	VC-1 Simple and Main Profile	6
1.8.4	VC-1 Advanced Profile	6
1.8.5	MPEG-4 Part 2 and DivX	6
1.8.6	H.265/HEVC - High Efficiency Video Codec	6
1.9	Video Mixer Usage	7
1.9.1	VdpVideoSurface Content	7
1.9.2	VdpVideoMixer Surface List	7
1.9.3	Weave De-interlacing	8
1.9.4	Bob De-interlacing	9
1.9.5	Advanced De-interlacing	9
1.9.6	De-interlacing Rate	9
1.9.7	Inverse Telecine	10
1.9.8	Telecine (Pull-Down) Flags	10
1.10	Extending the API	10
1.10.1	Enumerations and Other Constants	10
1.10.2	Structures	11
1.10.3	Functions	11
1.11	Display Preemption	11
1.11.1	Trademarks	11

<b>2</b>	<b>Module Index</b>	<b>13</b>
2.1	Modules . . . . .	13
<b>3</b>	<b>Data Structure Index</b>	<b>15</b>
3.1	Data Structures . . . . .	15
<b>4</b>	<b>File Index</b>	<b>17</b>
4.1	File List . . . . .	17
<b>5</b>	<b>Module Documentation</b>	<b>19</b>
5.1	Core API . . . . .	20
5.1.1	Detailed Description . . . . .	21
5.2	Basic Types . . . . .	22
5.2.1	Detailed Description . . . . .	22
5.2.2	Macro Definition Documentation . . . . .	22
5.2.2.1	VDP_FALSE . . . . .	22
5.2.2.2	VDP_TRUE . . . . .	22
5.2.3	Typedef Documentation . . . . .	22
5.2.3.1	VdpBool . . . . .	22
5.3	Miscellaneous Types . . . . .	23
5.3.1	Detailed Description . . . . .	24
5.3.2	Macro Definition Documentation . . . . .	24
5.3.2.1	VDP_CHROMA_TYPE_420 . . . . .	24
5.3.2.2	VDP_CHROMA_TYPE_422 . . . . .	24
5.3.2.3	VDP_CHROMA_TYPE_444 . . . . .	24
5.3.2.4	VDP_INDEXED_FORMAT_A4I4 . . . . .	24
5.3.2.5	VDP_INDEXED_FORMAT_A8I8 . . . . .	25
5.3.2.6	VDP_INDEXED_FORMAT_I4A4 . . . . .	25
5.3.2.7	VDP_INDEXED_FORMAT_I8A8 . . . . .	25
5.3.2.8	VDP_INVALID_HANDLE . . . . .	25
5.3.2.9	VDP_RGBA_FORMAT_A8 . . . . .	25
5.3.2.10	VDP_RGBA_FORMAT_B10G10R10A2 . . . . .	26

5.3.2.11	VDP_RGBA_FORMAT_B8G8R8A8 . . . . .	26
5.3.2.12	VDP_RGBA_FORMAT_R10G10B10A2 . . . . .	26
5.3.2.13	VDP_RGBA_FORMAT_R8G8B8A8 . . . . .	26
5.3.2.14	VDP_YCBCR_FORMAT_NV12 . . . . .	26
5.3.2.15	VDP_YCBCR_FORMAT_UYVY . . . . .	27
5.3.2.16	VDP_YCBCR_FORMAT_V8U8Y8A8 . . . . .	27
5.3.2.17	VDP_YCBCR_FORMAT_Y8U8V8A8 . . . . .	27
5.3.2.18	VDP_YCBCR_FORMAT_YUYV . . . . .	27
5.3.2.19	VDP_YCBCR_FORMAT_YV12 . . . . .	27
5.3.3	Typedef Documentation . . . . .	28
5.3.3.1	VdpChromaType . . . . .	28
5.3.3.2	VdpIndexedFormat . . . . .	28
5.3.3.3	VdpRGBAFormat . . . . .	28
5.3.3.4	VdpYCbCrFormat . . . . .	28
5.4	Error Handling . . . . .	29
5.4.1	Detailed Description . . . . .	29
5.4.2	Typedef Documentation . . . . .	29
5.4.2.1	VdpGetErrorString . . . . .	29
5.4.3	Enumeration Type Documentation . . . . .	30
5.4.3.1	VdpStatus . . . . .	30
5.5	Versioning . . . . .	32
5.5.1	Detailed Description . . . . .	32
5.5.2	Macro Definition Documentation . . . . .	32
5.5.2.1	VDPAU_INTERFACE_VERSION . . . . .	32
5.5.2.2	VDPAU_VERSION . . . . .	33
5.5.3	Typedef Documentation . . . . .	33
5.5.3.1	VdpGetApiVersion . . . . .	33
5.5.3.2	VdpGetInformationString . . . . .	33
5.6	VdpDevice; Primary API object . . . . .	34
5.6.1	Detailed Description . . . . .	34

5.6.2	Typedef Documentation . . . . .	34
5.6.2.1	VdpDevice . . . . .	34
5.6.2.2	VdpDeviceDestroy . . . . .	34
5.7	VdpCSCMatrix; CSC Matrix Manipulation . . . . .	35
5.7.1	Detailed Description . . . . .	35
5.7.2	Macro Definition Documentation . . . . .	36
5.7.2.1	VDP_COLOR_STANDARD_ITUR_BT_601 . . . . .	36
5.7.2.2	VDP_COLOR_STANDARD_ITUR_BT_709 . . . . .	36
5.7.2.3	VDP_COLOR_STANDARD_SMPTE_240M . . . . .	36
5.7.2.4	VDP_PROCOMP_VERSION . . . . .	36
5.7.3	Typedef Documentation . . . . .	36
5.7.3.1	VdpColorStandard . . . . .	36
5.7.3.2	VdpCSCMatrix . . . . .	36
5.7.3.3	VdpGenerateCSCMatrix . . . . .	36
5.8	VdpVideoSurface; Video Surface object . . . . .	38
5.8.1	Detailed Description . . . . .	38
5.8.2	Typedef Documentation . . . . .	39
5.8.2.1	VdpVideoSurface . . . . .	39
5.8.2.2	VdpVideoSurfaceCreate . . . . .	39
5.8.2.3	VdpVideoSurfaceDestroy . . . . .	40
5.8.2.4	VdpVideoSurfaceGetBitsYCbCr . . . . .	40
5.8.2.5	VdpVideoSurfaceGetParameters . . . . .	40
5.8.2.6	VdpVideoSurfacePutBitsYCbCr . . . . .	41
5.8.2.7	VdpVideoSurfaceQueryCapabilities . . . . .	41
5.8.2.8	VdpVideoSurfaceQueryGetPutBitsYCbCrCapabilities . . . . .	42
5.9	VdpOutputSurface; Output Surfaceobject . . . . .	43
5.9.1	Detailed Description . . . . .	44
5.9.2	Macro Definition Documentation . . . . .	44
5.9.2.1	VDP_COLOR_TABLE_FORMAT_B8G8R8X8 . . . . .	44
5.9.3	Typedef Documentation . . . . .	44

5.9.3.1	<a href="#">VdpColorTableFormat</a>	44
5.9.3.2	<a href="#">VdpOutputSurface</a>	45
5.9.3.3	<a href="#">VdpOutputSurfaceCreate</a>	45
5.9.3.4	<a href="#">VdpOutputSurfaceDestroy</a>	45
5.9.3.5	<a href="#">VdpOutputSurfaceGetBitsNative</a>	45
5.9.3.6	<a href="#">VdpOutputSurfaceGetParameters</a>	46
5.9.3.7	<a href="#">VdpOutputSurfacePutBitsIndexed</a>	46
5.9.3.8	<a href="#">VdpOutputSurfacePutBitsNative</a>	47
5.9.3.9	<a href="#">VdpOutputSurfacePutBitsYCbCr</a>	47
5.9.3.10	<a href="#">VdpOutputSurfaceQueryCapabilities</a>	48
5.9.3.11	<a href="#">VdpOutputSurfaceQueryGetPutBitsNativeCapabilities</a>	48
5.9.3.12	<a href="#">VdpOutputSurfaceQueryPutBitsIndexedCapabilities</a>	48
5.9.3.13	<a href="#">VdpOutputSurfaceQueryPutBitsYCbCrCapabilities</a>	49
5.10	<a href="#">VdpBitmapSurface; Bitmap Surfaceobject</a>	50
5.10.1	<a href="#">Detailed Description</a>	50
5.10.2	<a href="#">Typedef Documentation</a>	51
5.10.2.1	<a href="#">VdpBitmapSurface</a>	51
5.10.2.2	<a href="#">VdpBitmapSurfaceCreate</a>	51
5.10.2.3	<a href="#">VdpBitmapSurfaceDestroy</a>	51
5.10.2.4	<a href="#">VdpBitmapSurfaceGetParameters</a>	51
5.10.2.5	<a href="#">VdpBitmapSurfacePutBitsNative</a>	52
5.10.2.6	<a href="#">VdpBitmapSurfaceQueryCapabilities</a>	52
5.11	<a href="#">VdpOutputSurface Rendering Functionality</a>	54
5.11.1	<a href="#">Detailed Description</a>	55
5.11.2	<a href="#">Macro Definition Documentation</a>	55
5.11.2.1	<a href="#">VDP_OUTPUT_SURFACE_RENDER_BLEND_STATE_VERSION</a>	55
5.11.2.2	<a href="#">VDP_OUTPUT_SURFACE_RENDER_COLOR_PER_VERTEX</a>	55
5.11.2.3	<a href="#">VDP_OUTPUT_SURFACE_RENDER_ROTATE_0</a>	55
5.11.2.4	<a href="#">VDP_OUTPUT_SURFACE_RENDER_ROTATE_180</a>	55
5.11.2.5	<a href="#">VDP_OUTPUT_SURFACE_RENDER_ROTATE_270</a>	55

5.11.2.6	VDP_OUTPUT_SURFACE_RENDER_ROTATE_90	56
5.11.3	Typedef Documentation	56
5.11.3.1	VdpOutputSurfaceRenderBitmapSurface	56
5.11.3.2	VdpOutputSurfaceRenderOutputSurface	57
5.11.4	Enumeration Type Documentation	58
5.11.4.1	VdpOutputSurfaceRenderBlendEquation	58
5.11.4.2	VdpOutputSurfaceRenderBlendFactor	59
5.12	VdpDecoder; Video Decoding object	60
5.12.1	Detailed Description	63
5.12.2	Macro Definition Documentation	63
5.12.2.1	VDP_BITSTREAM_BUFFER_VERSION	63
5.12.2.2	VDP_DECODER_LEVEL_DIVX_NA	63
5.12.2.3	VDP_DECODER_LEVEL_H264_1	63
5.12.2.4	VDP_DECODER_LEVEL_H264_1_1	63
5.12.2.5	VDP_DECODER_LEVEL_H264_1_2	63
5.12.2.6	VDP_DECODER_LEVEL_H264_1_3	63
5.12.2.7	VDP_DECODER_LEVEL_H264_1b	63
5.12.2.8	VDP_DECODER_LEVEL_H264_2	63
5.12.2.9	VDP_DECODER_LEVEL_H264_2_1	63
5.12.2.10	VDP_DECODER_LEVEL_H264_2_2	63
5.12.2.11	VDP_DECODER_LEVEL_H264_3	63
5.12.2.12	VDP_DECODER_LEVEL_H264_3_1	63
5.12.2.13	VDP_DECODER_LEVEL_H264_3_2	63
5.12.2.14	VDP_DECODER_LEVEL_H264_4	63
5.12.2.15	VDP_DECODER_LEVEL_H264_4_1	63
5.12.2.16	VDP_DECODER_LEVEL_H264_4_2	63
5.12.2.17	VDP_DECODER_LEVEL_H264_5	63
5.12.2.18	VDP_DECODER_LEVEL_H264_5_1	63
5.12.2.19	VDP_DECODER_LEVEL_HEVC_1	63
5.12.2.20	VDP_DECODER_LEVEL_HEVC_2	64



5.12.2.21 VDP_DECODER_LEVEL_HEVC_2_1 . . . . .	64
5.12.2.22 VDP_DECODER_LEVEL_HEVC_3 . . . . .	64
5.12.2.23 VDP_DECODER_LEVEL_HEVC_3_1 . . . . .	64
5.12.2.24 VDP_DECODER_LEVEL_HEVC_4 . . . . .	64
5.12.2.25 VDP_DECODER_LEVEL_HEVC_4_1 . . . . .	64
5.12.2.26 VDP_DECODER_LEVEL_HEVC_5 . . . . .	64
5.12.2.27 VDP_DECODER_LEVEL_HEVC_5_1 . . . . .	64
5.12.2.28 VDP_DECODER_LEVEL_HEVC_5_2 . . . . .	64
5.12.2.29 VDP_DECODER_LEVEL_HEVC_6 . . . . .	64
5.12.2.30 VDP_DECODER_LEVEL_HEVC_6_1 . . . . .	64
5.12.2.31 VDP_DECODER_LEVEL_HEVC_6_2 . . . . .	64
5.12.2.32 VDP_DECODER_LEVEL_MPEG1_NA . . . . .	64
5.12.2.33 VDP_DECODER_LEVEL_MPEG2_HL . . . . .	64
5.12.2.34 VDP_DECODER_LEVEL_MPEG2_HL14 . . . . .	64
5.12.2.35 VDP_DECODER_LEVEL_MPEG2_LL . . . . .	64
5.12.2.36 VDP_DECODER_LEVEL_MPEG2_ML . . . . .	64
5.12.2.37 VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L0 . . . . .	64
5.12.2.38 VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L1 . . . . .	64
5.12.2.39 VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L2 . . . . .	64
5.12.2.40 VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L3 . . . . .	64
5.12.2.41 VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L4 . . . . .	64
5.12.2.42 VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L5 . . . . .	64
5.12.2.43 VDP_DECODER_LEVEL_MPEG4_PART2_SP_L0 . . . . .	65
5.12.2.44 VDP_DECODER_LEVEL_MPEG4_PART2_SP_L1 . . . . .	65
5.12.2.45 VDP_DECODER_LEVEL_MPEG4_PART2_SP_L2 . . . . .	65
5.12.2.46 VDP_DECODER_LEVEL_MPEG4_PART2_SP_L3 . . . . .	65
5.12.2.47 VDP_DECODER_LEVEL_VC1_ADVANCED_L0 . . . . .	65
5.12.2.48 VDP_DECODER_LEVEL_VC1_ADVANCED_L1 . . . . .	65
5.12.2.49 VDP_DECODER_LEVEL_VC1_ADVANCED_L2 . . . . .	65
5.12.2.50 VDP_DECODER_LEVEL_VC1_ADVANCED_L3 . . . . .	65

5.12.2.51 VDP_DECODER_LEVEL_VC1_ADVANCED_L4 . . . . .	65
5.12.2.52 VDP_DECODER_LEVEL_VC1_MAIN_HIGH . . . . .	65
5.12.2.53 VDP_DECODER_LEVEL_VC1_MAIN_LOW . . . . .	65
5.12.2.54 VDP_DECODER_LEVEL_VC1_MAIN_MEDIUM . . . . .	65
5.12.2.55 VDP_DECODER_LEVEL_VC1_SIMPLE_LOW . . . . .	65
5.12.2.56 VDP_DECODER_LEVEL_VC1_SIMPLE_MEDIUM . . . . .	65
5.12.2.57 VDP_DECODER_PROFILE_DIVX4_HD_1080P . . . . .	65
5.12.2.58 VDP_DECODER_PROFILE_DIVX4_HOME_THEATER . . . . .	65
5.12.2.59 VDP_DECODER_PROFILE_DIVX4_MOBILE . . . . .	65
5.12.2.60 VDP_DECODER_PROFILE_DIVX4_QMOBILE . . . . .	65
5.12.2.61 VDP_DECODER_PROFILE_DIVX5_HD_1080P . . . . .	65
5.12.2.62 VDP_DECODER_PROFILE_DIVX5_HOME_THEATER . . . . .	65
5.12.2.63 VDP_DECODER_PROFILE_DIVX5_MOBILE . . . . .	65
5.12.2.64 VDP_DECODER_PROFILE_DIVX5_QMOBILE . . . . .	65
5.12.2.65 VDP_DECODER_PROFILE_H264_BASELINE . . . . .	65
5.12.2.66 VDP_DECODER_PROFILE_H264_CONSTRAINED_BASELINE . . . . .	66
5.12.2.67 VDP_DECODER_PROFILE_H264_CONSTRAINED_HIGH . . . . .	66
5.12.2.68 VDP_DECODER_PROFILE_H264_EXTENDED . . . . .	66
5.12.2.69 VDP_DECODER_PROFILE_H264_HIGH . . . . .	66
5.12.2.70 VDP_DECODER_PROFILE_H264_HIGH_444_PREDICTIVE . . . . .	66
5.12.2.71 VDP_DECODER_PROFILE_H264_MAIN . . . . .	66
5.12.2.72 VDP_DECODER_PROFILE_H264_PROGRESSIVE_HIGH . . . . .	66
5.12.2.73 VDP_DECODER_PROFILE_HEVC_MAIN . . . . .	66
5.12.2.74 VDP_DECODER_PROFILE_HEVC_MAIN_10 . . . . .	66
5.12.2.75 VDP_DECODER_PROFILE_HEVC_MAIN_12 . . . . .	66
5.12.2.76 VDP_DECODER_PROFILE_HEVC_MAIN_444 . . . . .	66
5.12.2.77 VDP_DECODER_PROFILE_HEVC_MAIN_STILL . . . . .	66
5.12.2.78 VDP_DECODER_PROFILE_MPEG1 . . . . .	66
5.12.2.79 VDP_DECODER_PROFILE_MPEG2_MAIN . . . . .	66
5.12.2.80 VDP_DECODER_PROFILE_MPEG2_SIMPLE . . . . .	66

5.12.2.81 VDP_DECODER_PROFILE_MPEG4_PART2_ASP . . . . .	66
5.12.2.82 VDP_DECODER_PROFILE_MPEG4_PART2_SP . . . . .	66
5.12.2.83 VDP_DECODER_PROFILE_VC1_ADVANCED . . . . .	66
5.12.2.84 VDP_DECODER_PROFILE_VC1_MAIN . . . . .	66
5.12.2.85 VDP_DECODER_PROFILE_VC1_SIMPLE . . . . .	66
5.12.3 Typedef Documentation . . . . .	66
5.12.3.1 VdpDecoder . . . . .	66
5.12.3.2 VdpDecoderCreate . . . . .	66
5.12.3.3 VdpDecoderDestroy . . . . .	67
5.12.3.4 VdpDecoderGetParameters . . . . .	67
5.12.3.5 VdpDecoderProfile . . . . .	67
5.12.3.6 VdpDecoderQueryCapabilities . . . . .	68
5.12.3.7 VdpDecoderRender . . . . .	68
5.12.3.8 VdpPictureInfo . . . . .	68
5.12.3.9 VdpPictureInfoDivX4 . . . . .	69
5.12.3.10 VdpPictureInfoDivX5 . . . . .	69
5.13 VdpVideoMixer; Video Post-processing and Compositing object . . . . .	70
5.13.1 Detailed Description . . . . .	72
5.13.2 Macro Definition Documentation . . . . .	73
5.13.2.1 VDP_LAYER_VERSION . . . . .	73
5.13.2.2 VDP_VIDEO_MIXER_ATTRIBUTE_BACKGROUND_COLOR . . . . .	73
5.13.2.3 VDP_VIDEO_MIXER_ATTRIBUTE_CSC_MATRIX . . . . .	73
5.13.2.4 VDP_VIDEO_MIXER_ATTRIBUTE_LUMA_KEY_MAX_LUMA . . . . .	74
5.13.2.5 VDP_VIDEO_MIXER_ATTRIBUTE_LUMA_KEY_MIN_LUMA . . . . .	74
5.13.2.6 VDP_VIDEO_MIXER_ATTRIBUTE_NOISE_REDUCTION_LEVEL . . . . .	74
5.13.2.7 VDP_VIDEO_MIXER_ATTRIBUTE_SHARPNESS_LEVEL . . . . .	74
5.13.2.8 VDP_VIDEO_MIXER_ATTRIBUTE_SKIP_CHROMA_DEINTERLACE . . . . .	74
5.13.2.9 VDP_VIDEO_MIXER_FEATURE_DEINTERLACE_TEMPORAL . . . . .	75
5.13.2.10 VDP_VIDEO_MIXER_FEATURE_DEINTERLACE_TEMPORAL_SPATIAL . . . . .	75
5.13.2.11 VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L1 . . . . .	75

5.13.2.12	VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L2 . . . . .	75
5.13.2.13	VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L3 . . . . .	75
5.13.2.14	VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L4 . . . . .	76
5.13.2.15	VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L5 . . . . .	76
5.13.2.16	VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L6 . . . . .	76
5.13.2.17	VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L7 . . . . .	76
5.13.2.18	VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L8 . . . . .	76
5.13.2.19	VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L9 . . . . .	76
5.13.2.20	VDP_VIDEO_MIXER_FEATURE_INVERSE_TELECINE . . . . .	76
5.13.2.21	VDP_VIDEO_MIXER_FEATURE_LUMA_KEY . . . . .	77
5.13.2.22	VDP_VIDEO_MIXER_FEATURE_NOISE_REDUCTION . . . . .	77
5.13.2.23	VDP_VIDEO_MIXER_FEATURE_SHARPNESS . . . . .	77
5.13.2.24	VDP_VIDEO_MIXER_PARAMETER_CHROMA_TYPE . . . . .	77
5.13.2.25	VDP_VIDEO_MIXER_PARAMETER_LAYERS . . . . .	77
5.13.2.26	VDP_VIDEO_MIXER_PARAMETER_VIDEO_SURFACE_HEIGHT . . . . .	78
5.13.2.27	VDP_VIDEO_MIXER_PARAMETER_VIDEO_SURFACE_WIDTH . . . . .	78
5.13.3	Typedef Documentation . . . . .	78
5.13.3.1	VdpVideoMixer . . . . .	78
5.13.3.2	VdpVideoMixerAttribute . . . . .	78
5.13.3.3	VdpVideoMixerCreate . . . . .	78
5.13.3.4	VdpVideoMixerDestroy . . . . .	79
5.13.3.5	VdpVideoMixerFeature . . . . .	79
5.13.3.6	VdpVideoMixerGetAttributeValues . . . . .	79
5.13.3.7	VdpVideoMixerGetFeatureEnables . . . . .	80
5.13.3.8	VdpVideoMixerGetFeatureSupport . . . . .	80
5.13.3.9	VdpVideoMixerGetParameterValues . . . . .	80
5.13.3.10	VdpVideoMixerParameter . . . . .	81
5.13.3.11	VdpVideoMixerQueryAttributeSupport . . . . .	81
5.13.3.12	VdpVideoMixerQueryAttributeValueRange . . . . .	81
5.13.3.13	VdpVideoMixerQueryFeatureSupport . . . . .	82

5.13.3.14 VdpVideoMixerQueryParameterSupport . . . . .	82
5.13.3.15 VdpVideoMixerQueryParameterValueRange . . . . .	82
5.13.3.16 VdpVideoMixerRender . . . . .	83
5.13.3.17 VdpVideoMixerSetAttributeValues . . . . .	84
5.13.3.18 VdpVideoMixerSetFeatureEnables . . . . .	84
5.13.4 Enumeration Type Documentation . . . . .	84
5.13.4.1 VdpVideoMixerPictureStructure . . . . .	84
5.14 VdpPresentationQueue; Video presentation (display) object . . . . .	85
5.14.1 Detailed Description . . . . .	86
5.14.2 Typedef Documentation . . . . .	86
5.14.2.1 VdpPresentationQueue . . . . .	86
5.14.2.2 VdpPresentationQueueBlockUntilSurfaceIdle . . . . .	86
5.14.2.3 VdpPresentationQueueCreate . . . . .	87
5.14.2.4 VdpPresentationQueueDestroy . . . . .	87
5.14.2.5 VdpPresentationQueueDisplay . . . . .	87
5.14.2.6 VdpPresentationQueueGetBackgroundColor . . . . .	88
5.14.2.7 VdpPresentationQueueGetTime . . . . .	88
5.14.2.8 VdpPresentationQueueQuerySurfaceStatus . . . . .	88
5.14.2.9 VdpPresentationQueueSetBackgroundColor . . . . .	89
5.14.2.10 VdpPresentationQueueTarget . . . . .	89
5.14.2.11 VdpPresentationQueueTargetDestroy . . . . .	89
5.14.2.12 VdpTime . . . . .	90
5.14.3 Enumeration Type Documentation . . . . .	90
5.14.3.1 VdpPresentationQueueStatus . . . . .	90
5.15 Display Preemption . . . . .	91
5.15.1 Detailed Description . . . . .	91
5.15.2 Typedef Documentation . . . . .	91
5.15.2.1 VdpPreemptionCallback . . . . .	91
5.15.2.2 VdpPreemptionCallbackRegister . . . . .	92
5.16 Entry Point Retrieval . . . . .	93

5.16.1 Detailed Description . . . . .	94
5.16.2 Macro Definition Documentation . . . . .	94
5.16.2.1 VDP_FUNC_ID_BASE_WINSYS . . . . .	94
5.16.2.2 VDP_FUNC_ID_BITMAP_SURFACE_CREATE . . . . .	94
5.16.2.3 VDP_FUNC_ID_BITMAP_SURFACE_DESTROY . . . . .	94
5.16.2.4 VDP_FUNC_ID_BITMAP_SURFACE_GET_PARAMETERS . . . . .	94
5.16.2.5 VDP_FUNC_ID_BITMAP_SURFACE_PUT_BITS_NATIVE . . . . .	95
5.16.2.6 VDP_FUNC_ID_BITMAP_SURFACE_QUERY_CAPABILITIES . . . . .	95
5.16.2.7 VDP_FUNC_ID_DECODER_CREATE . . . . .	95
5.16.2.8 VDP_FUNC_ID_DECODER_DESTROY . . . . .	95
5.16.2.9 VDP_FUNC_ID_DECODER_GET_PARAMETERS . . . . .	95
5.16.2.10 VDP_FUNC_ID_DECODER_QUERY_CAPABILITIES . . . . .	95
5.16.2.11 VDP_FUNC_ID_DECODER_RENDER . . . . .	95
5.16.2.12 VDP_FUNC_ID_DEVICE_DESTROY . . . . .	95
5.16.2.13 VDP_FUNC_ID_GENERATE_CSC_MATRIX . . . . .	95
5.16.2.14 VDP_FUNC_ID_GET_API_VERSION . . . . .	95
5.16.2.15 VDP_FUNC_ID_GET_ERROR_STRING . . . . .	95
5.16.2.16 VDP_FUNC_ID_GET_INFORMATION_STRING . . . . .	95
5.16.2.17 VDP_FUNC_ID_GET_PROC_ADDRESS . . . . .	95
5.16.2.18 VDP_FUNC_ID_OUTPUT_SURFACE_CREATE . . . . .	95
5.16.2.19 VDP_FUNC_ID_OUTPUT_SURFACE_DESTROY . . . . .	95
5.16.2.20 VDP_FUNC_ID_OUTPUT_SURFACE_GET_BITS_NATIVE . . . . .	95
5.16.2.21 VDP_FUNC_ID_OUTPUT_SURFACE_GET_PARAMETERS . . . . .	95
5.16.2.22 VDP_FUNC_ID_OUTPUT_SURFACE_PUT_BITS_INDEXED . . . . .	95
5.16.2.23 VDP_FUNC_ID_OUTPUT_SURFACE_PUT_BITS_NATIVE . . . . .	95
5.16.2.24 VDP_FUNC_ID_OUTPUT_SURFACE_PUT_BITS_Y_CB_CR . . . . .	95
5.16.2.25 VDP_FUNC_ID_OUTPUT_SURFACE_QUERY_CAPABILITIES . . . . .	95
5.16.2.26 VDP_FUNC_ID_OUTPUT_SURFACE_QUERY_GET_PUT_BITS_NATIVE_C↔ APABILITIES . . . . .	95
5.16.2.27 VDP_FUNC_ID_OUTPUT_SURFACE_QUERY_PUT_BITS_INDEXED_CAPA↔ BILITIES . . . . .	95

5.16.2.28 VDP_FUNC_ID_OUTPUT_SURFACE_QUERY_PUT_BITS_Y_CB_CR_CAPABILITIES . . . . .	96
5.16.2.29 VDP_FUNC_ID_OUTPUT_SURFACE_RENDER_BITMAP_SURFACE . . . . .	96
5.16.2.30 VDP_FUNC_ID_OUTPUT_SURFACE_RENDER_OUTPUT_SURFACE . . . . .	96
5.16.2.31 VDP_FUNC_ID_OUTPUT_SURFACE_RENDER_VIDEO_SURFACE_LUMA . . . . .	96
5.16.2.32 VDP_FUNC_ID_PREEMPTION_CALLBACK_REGISTER . . . . .	96
5.16.2.33 VDP_FUNC_ID_PRESENTATION_QUEUE_BLOCK_UNTIL_SURFACE_IDLE . . . . .	96
5.16.2.34 VDP_FUNC_ID_PRESENTATION_QUEUE_CREATE . . . . .	96
5.16.2.35 VDP_FUNC_ID_PRESENTATION_QUEUE_DESTROY . . . . .	96
5.16.2.36 VDP_FUNC_ID_PRESENTATION_QUEUE_DISPLAY . . . . .	96
5.16.2.37 VDP_FUNC_ID_PRESENTATION_QUEUE_GET_BACKGROUND_COLOR . . . . .	96
5.16.2.38 VDP_FUNC_ID_PRESENTATION_QUEUE_GET_TIME . . . . .	96
5.16.2.39 VDP_FUNC_ID_PRESENTATION_QUEUE_QUERY_SURFACE_STATUS . . . . .	96
5.16.2.40 VDP_FUNC_ID_PRESENTATION_QUEUE_SET_BACKGROUND_COLOR . . . . .	96
5.16.2.41 VDP_FUNC_ID_PRESENTATION_QUEUE_TARGET_DESTROY . . . . .	96
5.16.2.42 VDP_FUNC_ID_VIDEO_MIXER_CREATE . . . . .	96
5.16.2.43 VDP_FUNC_ID_VIDEO_MIXER_DESTROY . . . . .	96
5.16.2.44 VDP_FUNC_ID_VIDEO_MIXER_GET_ATTRIBUTE_VALUES . . . . .	96
5.16.2.45 VDP_FUNC_ID_VIDEO_MIXER_GET_FEATURE_ENABLES . . . . .	96
5.16.2.46 VDP_FUNC_ID_VIDEO_MIXER_GET_FEATURE_SUPPORT . . . . .	96
5.16.2.47 VDP_FUNC_ID_VIDEO_MIXER_GET_PARAMETER_VALUES . . . . .	96
5.16.2.48 VDP_FUNC_ID_VIDEO_MIXER_QUERY_ATTRIBUTE_SUPPORT . . . . .	96
5.16.2.49 VDP_FUNC_ID_VIDEO_MIXER_QUERY_ATTRIBUTE_VALUE_RANGE . . . . .	96
5.16.2.50 VDP_FUNC_ID_VIDEO_MIXER_QUERY_FEATURE_SUPPORT . . . . .	96
5.16.2.51 VDP_FUNC_ID_VIDEO_MIXER_QUERY_PARAMETER_SUPPORT . . . . .	97
5.16.2.52 VDP_FUNC_ID_VIDEO_MIXER_QUERY_PARAMETER_VALUE_RANGE . . . . .	97
5.16.2.53 VDP_FUNC_ID_VIDEO_MIXER_RENDER . . . . .	97
5.16.2.54 VDP_FUNC_ID_VIDEO_MIXER_SET_ATTRIBUTE_VALUES . . . . .	97
5.16.2.55 VDP_FUNC_ID_VIDEO_MIXER_SET_FEATURE_ENABLES . . . . .	97
5.16.2.56 VDP_FUNC_ID_VIDEO_SURFACE_CREATE . . . . .	97
5.16.2.57 VDP_FUNC_ID_VIDEO_SURFACE_DESTROY . . . . .	97

5.16.2.58 VDP_FUNC_ID_VIDEO_SURFACE_GET_BITS_Y_CB_CR . . . . .	97
5.16.2.59 VDP_FUNC_ID_VIDEO_SURFACE_GET_PARAMETERS . . . . .	97
5.16.2.60 VDP_FUNC_ID_VIDEO_SURFACE_PUT_BITS_Y_CB_CR . . . . .	97
5.16.2.61 VDP_FUNC_ID_VIDEO_SURFACE_QUERY_CAPABILITIES . . . . .	97
5.16.2.62 VDP_FUNC_ID_VIDEO_SURFACE_QUERY_GET_PUT_BITS_Y_CB_CR_C↔ APABILITIES . . . . .	97
5.16.3 Typedef Documentation . . . . .	97
5.16.3.1 VdpFuncId . . . . .	97
5.16.3.2 VdpGetProcAddress . . . . .	97
5.17 Window System Integration Layer . . . . .	98
5.17.1 Detailed Description . . . . .	98
5.18 X11 Window System Integration Layer . . . . .	99
5.18.1 Detailed Description . . . . .	99
5.18.2 Library Layout . . . . .	99
5.18.3 Macro Definition Documentation . . . . .	100
5.18.3.1 VDP_FUNC_ID_PRESENTATION_QUEUE_TARGET_CREATE_X11 . . . . .	100
5.18.4 Typedef Documentation . . . . .	100
5.18.4.1 VdpDeviceCreateX11 . . . . .	100
5.18.4.2 VdpPresentationQueueTargetCreateX11 . . . . .	100
5.18.5 Variable Documentation . . . . .	101
5.18.5.1 vdp_device_create_x11 . . . . .	101



<b>6</b>	<b>Data Structure Documentation</b>	<b>103</b>
6.1	VdpBitstreamBuffer Struct Reference	103
6.1.1	Detailed Description	103
6.1.2	Field Documentation	103
6.1.2.1	bitstream	103
6.1.2.2	bitstream_bytes	103
6.1.2.3	struct_version	103
6.2	VdpColor Struct Reference	104
6.2.1	Detailed Description	104
6.2.2	Field Documentation	104
6.2.2.1	alpha	104
6.2.2.2	blue	104
6.2.2.3	green	104
6.2.2.4	red	104
6.3	VdpLayer Struct Reference	104
6.3.1	Detailed Description	105
6.3.2	Field Documentation	105
6.3.2.1	destination_rect	105
6.3.2.2	source_rect	105
6.3.2.3	source_surface	105
6.3.2.4	struct_version	105
6.4	VdpOutputSurfaceRenderBlendState Struct Reference	105
6.4.1	Detailed Description	106
6.4.2	Field Documentation	106
6.4.2.1	blend_constant	106
6.4.2.2	blend_equation_alpha	106
6.4.2.3	blend_equation_color	106
6.4.2.4	blend_factor_destination_alpha	106
6.4.2.5	blend_factor_destination_color	106
6.4.2.6	blend_factor_source_alpha	106

6.4.2.7	<code>blend_factor_source_color</code>	106
6.4.2.8	<code>struct_version</code>	106
6.5	VdpPictureInfoH264 Struct Reference	106
6.5.1	Detailed Description	107
6.5.2	Field Documentation	108
6.5.2.1	<code>bottom_field_flag</code>	108
6.5.2.2	<code>chroma_qp_index_offset</code>	108
6.5.2.3	<code>constrained_intra_pred_flag</code>	108
6.5.2.4	<code>deblocking_filter_control_present_flag</code>	108
6.5.2.5	<code>delta_pic_order_always_zero_flag</code>	108
6.5.2.6	<code>direct_8x8_inference_flag</code>	108
6.5.2.7	<code>entropy_coding_mode_flag</code>	108
6.5.2.8	<code>field_order_cnt</code>	108
6.5.2.9	<code>field_pic_flag</code>	108
6.5.2.10	<code>frame_mbs_only_flag</code>	108
6.5.2.11	<code>frame_num</code>	108
6.5.2.12	<code>is_reference</code>	108
6.5.2.13	<code>log2_max_frame_num_minus4</code>	108
6.5.2.14	<code>log2_max_pic_order_cnt_lsb_minus4</code>	108
6.5.2.15	<code>mb_adaptive_frame_field_flag</code>	108
6.5.2.16	<code>num_ref_frames</code>	108
6.5.2.17	<code>num_ref_idx_l0_active_minus1</code>	108
6.5.2.18	<code>num_ref_idx_l1_active_minus1</code>	108
6.5.2.19	<code>pic_init_qp_minus26</code>	108
6.5.2.20	<code>pic_order_cnt_type</code>	108
6.5.2.21	<code>pic_order_present_flag</code>	108
6.5.2.22	<code>redundant_pic_cnt_present_flag</code>	108
6.5.2.23	<code>referenceFrames</code>	108
6.5.2.24	<code>scaling_lists_4x4</code>	109
6.5.2.25	<code>scaling_lists_8x8</code>	109

6.5.2.26	second_chroma_qp_index_offset . . . . .	109
6.5.2.27	slice_count . . . . .	109
6.5.2.28	transform_8x8_mode_flag . . . . .	109
6.5.2.29	weighted_bipred_idc . . . . .	109
6.5.2.30	weighted_pred_flag . . . . .	109
6.6	VdpPictureInfoH264Predictive Struct Reference . . . . .	109
6.6.1	Detailed Description . . . . .	110
6.6.2	Field Documentation . . . . .	110
6.6.2.1	pictureInfo . . . . .	110
6.6.2.2	qpprime_y_zero_transform_bypass_flag . . . . .	110
6.6.2.3	separate_colour_plane_flag . . . . .	110
6.7	VdpPictureInfoHEVC Struct Reference . . . . .	110
6.7.1	Detailed Description . . . . .	112
6.7.2	Field Documentation . . . . .	113
6.7.2.1	amp_enabled_flag . . . . .	113
6.7.2.2	bit_depth_chroma_minus8 . . . . .	113
6.7.2.3	bit_depth_luma_minus8 . . . . .	113
6.7.2.4	cabac_init_present_flag . . . . .	113
6.7.2.5	chroma_format_idc . . . . .	113
6.7.2.6	column_width_minus1 . . . . .	113
6.7.2.7	constrained_intra_pred_flag . . . . .	113
6.7.2.8	cu_qp_delta_enabled_flag . . . . .	113
6.7.2.9	CurrPicOrderCntVal . . . . .	113
6.7.2.10	CurrRpsIdx . . . . .	113
6.7.2.11	deblocking_filter_control_present_flag . . . . .	113
6.7.2.12	deblocking_filter_override_enabled_flag . . . . .	113
6.7.2.13	dependent_slice_segments_enabled_flag . . . . .	113
6.7.2.14	diff_cu_qp_delta_depth . . . . .	113
6.7.2.15	entropy_coding_sync_enabled_flag . . . . .	114
6.7.2.16	IDRPicFlag . . . . .	114

6.7.2.17	init_qp_minus26	114
6.7.2.18	IsLongTerm	114
6.7.2.19	lists_modification_present_flag	114
6.7.2.20	log2_diff_max_min_luma_coding_block_size	114
6.7.2.21	log2_diff_max_min_pcm_luma_coding_block_size	114
6.7.2.22	log2_diff_max_min_transform_block_size	114
6.7.2.23	log2_max_pic_order_cnt_lsb_minus4	114
6.7.2.24	log2_min_luma_coding_block_size_minus3	114
6.7.2.25	log2_min_pcm_luma_coding_block_size_minus3	114
6.7.2.26	log2_min_transform_block_size_minus2	114
6.7.2.27	log2_parallel_merge_level_minus2	114
6.7.2.28	long_term_ref_pics_present_flag	114
6.7.2.29	loop_filter_across_tiles_enabled_flag	114
6.7.2.30	max_transform_hierarchy_depth_inter	115
6.7.2.31	max_transform_hierarchy_depth_intra	115
6.7.2.32	num_extra_slice_header_bits	115
6.7.2.33	num_long_term_ref_pics_sps	115
6.7.2.34	num_ref_idx_l0_default_active_minus1	115
6.7.2.35	num_ref_idx_l1_default_active_minus1	115
6.7.2.36	num_short_term_ref_pic_sets	115
6.7.2.37	num_tile_columns_minus1	115
6.7.2.38	num_tile_rows_minus1	115
6.7.2.39	NumDeltaPocsOfRefRpsIdx	115
6.7.2.40	NumLongTermPictureSliceHeaderBits	115
6.7.2.41	NumPocLtCurr	115
6.7.2.42	NumPocStCurrAfter	116
6.7.2.43	NumPocStCurrBefore	116
6.7.2.44	NumPocTotalCurr	116
6.7.2.45	NumShortTermPictureSliceHeaderBits	116
6.7.2.46	output_flag_present_flag	116

6.7.2.47	pcm_enabled_flag . . . . .	116
6.7.2.48	pcm_loop_filter_disabled_flag . . . . .	116
6.7.2.49	pcm_sample_bit_depth_chroma_minus1 . . . . .	116
6.7.2.50	pcm_sample_bit_depth_luma_minus1 . . . . .	116
6.7.2.51	pic_height_in_luma_samples . . . . .	116
6.7.2.52	pic_width_in_luma_samples . . . . .	116
6.7.2.53	PicOrderCntVal . . . . .	116
6.7.2.54	pps_beta_offset_div2 . . . . .	117
6.7.2.55	pps_cb_qp_offset . . . . .	117
6.7.2.56	pps_cr_qp_offset . . . . .	117
6.7.2.57	pps_deblocking_filter_disabled_flag . . . . .	117
6.7.2.58	pps_loop_filter_across_slices_enabled_flag . . . . .	117
6.7.2.59	pps_slice_chroma_qp_offsets_present_flag . . . . .	117
6.7.2.60	pps_tc_offset_div2 . . . . .	117
6.7.2.61	RAPPicFlag . . . . .	117
6.7.2.62	RefPics . . . . .	117
6.7.2.63	RefPicSetLtCurr . . . . .	117
6.7.2.64	RefPicSetStCurrAfter . . . . .	117
6.7.2.65	RefPicSetStCurrBefore . . . . .	117
6.7.2.66	row_height_minus1 . . . . .	118
6.7.2.67	sample_adaptive_offset_enabled_flag . . . . .	118
6.7.2.68	scaling_list_enabled_flag . . . . .	118
6.7.2.69	ScalingList16x16 . . . . .	118
6.7.2.70	ScalingList32x32 . . . . .	118
6.7.2.71	ScalingList4x4 . . . . .	118
6.7.2.72	ScalingList8x8 . . . . .	118
6.7.2.73	ScalingListDCCoeff16x16 . . . . .	118
6.7.2.74	ScalingListDCCoeff32x32 . . . . .	118
6.7.2.75	separate_colour_plane_flag . . . . .	118
6.7.2.76	sign_data_hiding_enabled_flag . . . . .	118

6.7.2.77	<code>slice_segment_header_extension_present_flag</code>	118
6.7.2.78	<code>sps_max_dec_pic_buffering_minus1</code>	118
6.7.2.79	<code>sps_temporal_mvp_enabled_flag</code>	119
6.7.2.80	<code>strong_intra_smoothing_enabled_flag</code>	119
6.7.2.81	<code>tiles_enabled_flag</code>	119
6.7.2.82	<code>transform_skip_enabled_flag</code>	119
6.7.2.83	<code>transquant_bypass_enabled_flag</code>	119
6.7.2.84	<code>uniform_spacing_flag</code>	119
6.7.2.85	<code>weighted_bipred_flag</code>	119
6.7.2.86	<code>weighted_pred_flag</code>	119
6.8	<code>VdpPictureInfoMPEG1Or2</code> Struct Reference	119
6.8.1	Detailed Description	120
6.8.2	Field Documentation	120
6.8.2.1	<code>alternate_scan</code>	120
6.8.2.2	<code>backward_reference</code>	120
6.8.2.3	<code>concealment_motion_vectors</code>	120
6.8.2.4	<code>f_code</code>	120
6.8.2.5	<code>forward_reference</code>	120
6.8.2.6	<code>frame_pred_frame_dct</code>	120
6.8.2.7	<code>full_pel_backward_vector</code>	120
6.8.2.8	<code>full_pel_forward_vector</code>	120
6.8.2.9	<code>intra_dc_precision</code>	120
6.8.2.10	<code>intra_quantizer_matrix</code>	120
6.8.2.11	<code>intra_vlc_format</code>	121
6.8.2.12	<code>non_intra_quantizer_matrix</code>	121
6.8.2.13	<code>picture_coding_type</code>	121
6.8.2.14	<code>picture_structure</code>	121
6.8.2.15	<code>q_scale_type</code>	121
6.8.2.16	<code>slice_count</code>	121
6.8.2.17	<code>top_field_first</code>	121

6.9	VdpPictureInfoMPEG4Part2 Struct Reference	121
6.9.1	Detailed Description	122
6.9.2	Field Documentation	122
6.9.2.1	alternate_vertical_scan_flag	122
6.9.2.2	backward_reference	122
6.9.2.3	forward_reference	122
6.9.2.4	interlaced	122
6.9.2.5	intra_quantizer_matrix	122
6.9.2.6	non_intra_quantizer_matrix	122
6.9.2.7	quant_type	122
6.9.2.8	quarter_sample	122
6.9.2.9	resync_marker_disable	122
6.9.2.10	rounding_control	122
6.9.2.11	short_video_header	122
6.9.2.12	top_field_first	122
6.9.2.13	trb	122
6.9.2.14	trd	122
6.9.2.15	vop_coding_type	122
6.9.2.16	vop_fcode_backward	122
6.9.2.17	vop_fcode_forward	122
6.9.2.18	vop_time_increment_resolution	122
6.10	VdpPictureInfoVC1 Struct Reference	123
6.10.1	Detailed Description	123
6.10.2	Field Documentation	123
6.10.2.1	backward_reference	123
6.10.2.2	deblockEnable	124
6.10.2.3	dquant	124
6.10.2.4	extended_dmv	124
6.10.2.5	extended_mv	124
6.10.2.6	fastuvmc	124

6.10.2.7	<a href="#">finterpflag</a>	124
6.10.2.8	<a href="#">forward_reference</a>	124
6.10.2.9	<a href="#">frame_coding_mode</a>	124
6.10.2.10	<a href="#">interlace</a>	124
6.10.2.11	<a href="#">loopfilter</a>	124
6.10.2.12	<a href="#">maxbframes</a>	125
6.10.2.13	<a href="#">multires</a>	125
6.10.2.14	<a href="#">overlap</a>	125
6.10.2.15	<a href="#">panscan_flag</a>	125
6.10.2.16	<a href="#">picture_type</a>	125
6.10.2.17	<a href="#">postprocflag</a>	125
6.10.2.18	<a href="#">pquant</a>	125
6.10.2.19	<a href="#">psf</a>	125
6.10.2.20	<a href="#">pulldown</a>	125
6.10.2.21	<a href="#">quantizer</a>	125
6.10.2.22	<a href="#">range_mapuv</a>	126
6.10.2.23	<a href="#">range_mapuv_flag</a>	126
6.10.2.24	<a href="#">range_mapy</a>	126
6.10.2.25	<a href="#">range_mapy_flag</a>	126
6.10.2.26	<a href="#">rangered</a>	126
6.10.2.27	<a href="#">reldist_flag</a>	126
6.10.2.28	<a href="#">slice_count</a>	126
6.10.2.29	<a href="#">syncmarker</a>	126
6.10.2.30	<a href="#">tfcntrflag</a>	126
6.10.2.31	<a href="#">vstransform</a>	126
6.11	<a href="#">VdpPoint Struct Reference</a>	127
6.11.1	<a href="#">Detailed Description</a>	127
6.11.2	<a href="#">Field Documentation</a>	127
6.11.2.1	<a href="#">x</a>	127
6.11.2.2	<a href="#">y</a>	127



6.12 VdpProcamp Struct Reference . . . . .	127
6.12.1 Detailed Description . . . . .	128
6.12.2 Field Documentation . . . . .	128
6.12.2.1 brightness . . . . .	128
6.12.2.2 contrast . . . . .	128
6.12.2.3 hue . . . . .	128
6.12.2.4 saturation . . . . .	128
6.12.2.5 struct_version . . . . .	128
6.13 VdpRect Struct Reference . . . . .	128
6.13.1 Detailed Description . . . . .	129
6.13.2 Field Documentation . . . . .	129
6.13.2.1 x0 . . . . .	129
6.13.2.2 x1 . . . . .	129
6.13.2.3 y0 . . . . .	129
6.13.2.4 y1 . . . . .	129
6.14 VdpReferenceFrameH264 Struct Reference . . . . .	129
6.14.1 Detailed Description . . . . .	130
6.14.2 Field Documentation . . . . .	130
6.14.2.1 bottom_is_reference . . . . .	130
6.14.2.2 field_order_cnt . . . . .	130
6.14.2.3 frame_idx . . . . .	130
6.14.2.4 is_long_term . . . . .	130
6.14.2.5 surface . . . . .	130
6.14.2.6 top_is_reference . . . . .	130
<b>7 File Documentation</b>	<b>131</b>
7.1 vdpau/vdpau.h File Reference . . . . .	131
7.1.1 Detailed Description . . . . .	143
7.2 vdpau/vdpau_x11.h File Reference . . . . .	143
7.2.1 Detailed Description . . . . .	144
<b>Index</b>	<b>145</b>



# Chapter 1

## Video Decode and Presentation API for Unix

### 1.1 Introduction

The Video Decode and Presentation API for Unix (VDPAU) provides a complete solution for decoding, post-processing, compositing, and displaying compressed or uncompressed video streams. These video streams may be combined (composited) with bitmap content, to implement OSDs and other application user interfaces.

### 1.2 API Partitioning

VDPAU is split into two distinct modules:

- [Core API](#)
- [Window System Integration Layer](#)

The intent is that most VDPAU functionality exists and operates identically across all possible Windowing Systems. This functionality is the [Core API](#).

However, a small amount of functionality must be included that is tightly coupled to the underlying Windowing System. This functionality is the [Window System Integration Layer](#). Possibly examples include:

- Creation of the initial VDPAU [VdpDevice](#) handle, since this act requires intimate knowledge of the underlying Window System, such as specific display handle or driver identification.
- Conversion of VDPAU surfaces to/from underlying Window System surface types, e.g. to allow manipulation of VDPAU-generated surfaces via native Window System APIs.

### 1.3 Object Types

VDPAU is roughly object oriented; most functionality is exposed by creating an object (handle) of a certain class (type), then executing various functions against that handle. The set of object classes supported, and their purpose, is discussed below.

### 1.3.1 Device Type

A [VdpDevice](#) is the root object in VDPAU's object system. The [Window System Integration Layer](#) allows creation of a [VdpDevice](#) object handle, from which all other API entry points can be retrieved and invoked.

### 1.3.2 Surface Types

A surface stores pixel information. Various types of surfaces existing for different purposes:

- [VdpVideoSurfaces](#) store decompressed YCbCr video frames in an implementation-defined internal format.
- [VdpOutputSurfaces](#) store RGB 4:4:4 data. They are legal render targets for video post-processing and compositing operations.
- [VdpBitmapSurfaces](#) store RGB 4:4:4 data. These surfaces are designed to contain read-only bitmap data, to be used for OSD or application UI compositing.

### 1.3.3 Transfer Types

A data transfer object reads data from a surface (or surfaces), processes it, and writes the result to another surface. Various types of processing are possible:

- [VdpDecoder](#) objects process compressed video data, and generate decompressed images.
- [VdpOutputSurfaces](#) have their own [rendering functionality](#).
- [VdpVideoMixer](#) objects perform video post-processing, de-interlacing, and compositing.
- [VdpPresentationQueue](#) is responsible for timestamp-based display of surfaces.

## 1.4 Data Flow

Compressed video data originates in the application's memory space. This memory is typically obtained using `malloc`, and filled via regular file or network read system calls. Alternatively, the application may `mmap` a file.

The compressed data is then processed using a [VdpDecoder](#), which will decompress the field or frame, and write the result into a [VdpVideoSurface](#). This action may require reading pixel data from some number of other [VdpVideoSurface](#) objects, depending on the type of compressed data and field/frame in question.

If the application wishes to display any form of OSD or user-interface, this must be created in a [VdpOutputSurface](#).

This process begins with the creation of [VdpBitmapSurface](#) objects to contain the OSD/UI's static data, such as individual glyphs.

[VdpOutputSurface rendering functionality](#) may be used to composite together various [VdpBitmapSurfaces](#) and [VdpOutputSurfaces](#), into another [VdpOutputSurface](#) "VdpOutputSurface".

Once video has been decoded, it must be post-processed. This involves various steps such as color space conversion, de-interlacing, and other video adjustments. This step is performed using an [VdpVideoMixer](#) object. This object can not only perform the aforementioned video post-processing, but also composite the video with a number of [VdpOutputSurfaces](#), thus allowing complex user interfaces to be built. The final result is written into another [VdpOutputSurface](#).

Note that at this point, the resultant [VdpOutputSurface](#) may be fed back through the above path, either using [VdpOutputSurface rendering functionality](#), or as input to the [VdpVideoMixer](#) object.

Finally, the resultant [VdpOutputSurface](#) must be displayed on screen. This is the job of the [VdpPresentationQueue](#) object.

## 1.5 Entry Point Retrieval

VDPAU is designed so that multiple implementations can be used without application changes. For example, VDPAU could be hosted on X11, or via direct GPU access.

The key technology behind this is the use of function pointers and a "get proc address" style API for all entry points. Put another way, functions are not called directly via global symbols set up by the linker, but rather through pointers.

In practical terms, the [Window System Integration Layer](#) provides factory functions which not only create and return [VdpDevice](#) objects, but also a function pointer to a [VdpGetProcAddress](#) function, through which all entry point function pointers will be retrieved.

### 1.5.1 Philosophy

It is entirely possible to envisage a simpler scheme whereby such function pointers are hidden. That is, the application would link against a wrapper library that exposed "real" functions. The application would then call such functions directly, by symbol, like any other function. The wrapper library would handle loading the appropriate back-end, and implementing a similar "get proc address" scheme internally.

However, the above scheme does not work well in the context of separated [Core API](#) and [Window System Integration Layer](#). In this scenario, one would require a separate wrapper library per Window System, since each Window System would have a different function name and prototype for the main factory function. If an application then wanted to be Window System agnostic (making final determination at run-time via some form of plugin), it may then need to link against two wrapper libraries, which would cause conflicts for all symbols other than the main factory function.

Another disadvantage of the wrapper library approach is the extra level of function call required; the wrapper library would internally implement the existing "get proc address" and "function pointer" style dispatch anyway. Exposing this directly to the application is slightly more efficient.

## 1.6 Multi-threading

All VDPAU functionality is fully thread-safe; any number of threads may call into any VDPAU functions at any time. VDPAU may not be called from signal-handlers.

Note, however, that this simply guarantees that internal VDPAU state will not be corrupted by thread usage, and that crashes and deadlocks will not occur. Completely arbitrary thread usage may not generate the results that an application desires. In particular, care must be taken when multiple threads are performing operations on the same VDPAU objects.

VDPAU implementations guarantee correct flow of surface content through the rendering pipeline, but only when function calls that read from or write to a surface return to the caller prior to any thread calling any other function(s) that read from or write to the surface. Invoking multiple reads from a surface in parallel is OK.

Note that this restriction is placed upon VDPAU function invocations, and specifically not upon any back-end hardware's physical rendering operations. VDPAU implementations are expected to internally synchronize such hardware operations.

In a single-threaded application, the above restriction comes naturally; each function call completes before it is possible to begin a new function call.

In a multi-threaded application, threads may need to be synchronized. For example, consider the situation where:

- Thread 1 is parsing compressed video data, passing them through a [VdpDecoder](#) object, and filling a ring-buffer of [VdpVideoSurfaces](#)
- Thread 2 is consuming those [VdpVideoSurfaces](#), and using a [VdpVideoMixer](#) to process them and composite them with UI.

In this case, the threads must synchronize to ensure that thread 1's call to [VdpDecoderRender](#) has returned prior to thread 2's call(s) to [VdpVideoMixerRender](#) that use that specific surface. This could be achieved using the following pseudo-code:

```
Queue<VdpVideoSurface> q_full_surfaces;
Queue<VdpVideoSurface> q_empty_surfaces;

thread_1() {
    for (;;) {
        VdpVideoSurface s = q_empty_surfaces.get();
        // Parse compressed stream here
        VdpDecoderRender(s, ...);
        q_full_surfaces.put(s);
    }
}

// This would need to be more complex if
// VdpVideoMixerRender were to be provided with more
// than one field/frame at a time.
thread_2() {
    for (;;) {
        // Possibly, other rendering operations to mixer
        // layer surfaces here.
        VdpOutputSurface t = ...;
        VdpPresentationQueueBlockUntilSurfaceIdle(t);
        VdpVideoSurface s = q_full_surfaces.get();
        VdpVideoMixerRender(s, t, ...);
        q_empty_surfaces.put(s);
        // Possibly, other rendering operations to "t" here
        VdpPresentationQueueDisplay(t, ...);
    }
}
```

Finally, note that VDPAU makes no guarantees regarding any level of parallelism in any given implementation. Put another way, use of multi-threading is not guaranteed to yield any performance gain, and in theory could even slightly reduce performance due to threading/synchronization overhead.

However, the intent of the threading requirements is to allow for e.g. video decoding and video mixer operations to proceed in parallel in hardware. Given a (presumably multi-threaded) application that kept each portion of the hardware busy, this would yield a performance increase.

## 1.7 Surface Endianness

When dealing with surface content, i.e. the input/output of Put/GetBits functions, applications must take care to access memory in the correct fashion, so as to avoid endianness issues.

By established convention in the 3D graphics world, RGBA data is defined to be an array of 32-bit pixels containing packed RGBA components, not as an array of bytes or interleaved RGBA components. VDPAU follows this convention. As such, applications are expected to access such surfaces as arrays of 32-bit components (i.e. using a 32-bit pointer), and not as interleaved arrays of 8-bit components (i.e. using an 8-bit pointer.) Deviation from this convention will lead to endianness issues, unless appropriate care is taken.

The same convention is followed for some packed YCbCr formats such as [VDP\\_YCBCR\\_FORMAT\\_Y8U8V8A8](#); i.e. they are considered arrays of 32-bit pixels, and hence should be accessed as such.

For YCbCr formats with chroma decimation and/or planar formats, however, this convention is awkward. Therefore, formats such as [VDP\\_YCBCR\\_FORMAT\\_NV12](#) are defined as arrays of (potentially interleaved) byte-sized components. Hence, applications should manipulate such data 8-bits at a time, using 8-bit pointers.

Note that one common usage for the input/output of Put/GetBits APIs is file I/O. Typical file I/O APIs treat all memory as a simple array of 8-bit values. This violates the rule requiring surface data to be accessed in its true native format. As such, applications may be required to solve endianness issues. Possible solutions include:

- Authoring static UI data files according to the endianness of the target execution platform.
- Conditionally byte-swapping Put/GetBits data buffers at run-time based on execution platform.

Note: Complete details regarding each surface format's precise pixel layout is included with the documentation of each surface type. For example, see [VDP\\_RGBA\\_FORMAT\\_B8G8R8A8](#).

## 1.8 Video Decoder Usage

VDPAU is a slice-level API. Put another way, VDPAU implementations accept "slice" data from the bitstream, and perform all required processing of those slices (e.g VLD decoding, IDCT, motion compensation, in-loop deblocking, etc.).

The client application is responsible for:

- Extracting the slices from the bitstream (e.g. parsing/demultiplexing container formats, scanning the data to determine slice start positions and slice sizes).
- Parsing various bitstream headers/structures (e.g. sequence header, sequence parameter set, picture parameter set, entry point structures, etc.) Various fields from the parsed header structures needs to be provided to VDPAU alongside the slice bitstream in a "picture info" structure.
- Surface management (e.g. H.264 DPB processing, display re-ordering)

It is recommended that applications pass solely the slice data to VDPAU; specifically that any header data structures be excluded from the portion of the bitstream passed to VDPAU. VDPAU implementations must operate correctly if non-slice data is included, at least for formats employing start codes to delimit slice data. However, any extra data may need to be uploaded to hardware for parsing thus lowering performance, and/or, in the worst case, may even overflow internal buffers that are sized solely for slice data.

The exact data that should be passed to VDPAU is detailed below for each supported format:

### 1.8.1 MPEG-1 and MPEG-2

Include all slices beginning with start codes 0x00000101 through 0x000001AF. The slice start code must be included for all slices.

### 1.8.2 H.264

Include all NALs with nal\_unit\_type of 1 or 5 (coded slice of non-IDR/IDR picture respectively). The complete slice start code (including 0x000001 prefix) must be included for all slices, even when the prefix is not included in the bitstream.

Note that if desired:

- The slice start code prefix may be included in a separate bitstream buffer array entry to the actual slice data extracted from the bitstream.
- Multiple bitstream buffer array entries (e.g. one per slice) may point at the same physical data storage for the slice start code prefix.

### 1.8.3 VC-1 Simple and Main Profile

VC-1 simple/main profile bitstreams always consist of a single slice per picture, and do not use start codes to delimit pictures. Instead, the container format must indicate where each picture begins/ends.

As such, no slice start codes should be included in the data passed to VDPAU; simply pass in the exact data from the bitstream.

Header information contained in the bitstream should be parsed by the application and passed to VDPAU using the "picture info" data structure; this header information explicitly must not be included in the bitstream data passed to VDPAU for this encoding format.

### 1.8.4 VC-1 Advanced Profile

Include all slices beginning with start codes 0x0000010D (frame), 0x0000010C (field) or 0x0000010B (slice). The slice start code should be included in all cases.

Some VC-1 advanced profile streams do not contain slice start codes; again, the container format must indicate where picture data begins and ends. In this case, pictures are assumed to be progressive and to contain a single slice. It is highly recommended that applications detect this condition, and add the missing start codes to the bitstream passed to VDPAU. However, VDPAU implementations must allow bitstreams with missing start codes, and act as if a 0x0000010D (frame) start code had been present.

Note that pictures containing multiple slices, or interlace streams, must contain a complete set of slice start codes in the original bitstream; without them, it is not possible to correctly parse and decode the stream.

The bitstream passed to VDPAU should contain all original emulation prevention bytes present in the original bitstream; do not remove these from the bitstream.

### 1.8.5 MPEG-4 Part 2 and DivX

Include all slices beginning with start codes 0x000001B6. The slice start code must be included for all slices.

### 1.8.6 H.265/HEVC - High Efficiency Video Codec

Include all video coding layer (VCL) NAL units, with `nal_unit_type` values of 0 (TRAIL\_N) through 31 (RSV\_VC↔L31) inclusive. In addition to parsing and providing NAL units, an H.265/HEVC decoder application using VDPAU for decoding must parse certain values of the first slice segment header in a VCL NAL unit and provide it through [VdpPictureInfoHEVC](#). Please see the documentation for [VdpPictureInfoHEVC](#) below for further details.

The complete slice start code (including the 0x000001 prefix) must be included for all slices, even when the prefix is not included in the bitstream.

Note that if desired:

- The slice start code prefix may be included in a separate bitstream buffer array entry to the actual slice data extracted from the bitstream.
- Multiple bitstream buffer array entries (e.g. one per slice) may point at the same physical data storage for the slice start code prefix.



## 1.9 Video Mixer Usage

### 1.9.1 VdpVideoSurface Content

Each [VdpVideoSurface](#) is expected to contain an entire frame's-worth of data, irrespective of whether an interlaced or progressive sequence is being decoded.

Depending on the exact encoding structure of the compressed video stream, the application may need to call [VdpDecoderRender](#) twice to fill a single [VdpVideoSurface](#). When the stream contains an encoded progressive frame, or a "frame coded" interlaced field-pair, a single [VdpDecoderRender](#) call will fill the entire surface. When the stream contains separately encoded interlaced fields, two [VdpDecoderRender](#) calls will be required; one for the top field, and one for the bottom field.

Implementation note: When [VdpDecoderRender](#) renders an interlaced field, this operation must not disturb the content of the other field in the surface.

### 1.9.2 VdpVideoMixer Surface List

An video stream is logically composed of a sequence of fields. An example is shown below, in display order, assuming top field first:

```
t0 b0 t1 b1 t2 b2 t3 b3 t4 b4 t5 b5 t6 b6 t7 b7 t8 b8 t9 b9
```

The canonical usage is to call [VdpVideoMixerRender](#) once for decoded field, in display order, to yield one post-processed frame for display.

For each call to [VdpVideoMixerRender](#), the field to be processed should be provided as the **video\_surface\_current** parameter.

To enable operation of advanced de-interlacing algorithms and/or post-processing algorithms, some past and/or future surfaces should be provided as context. These are provided in the **video\_surface\_past** and **video\_surface\_future** lists. In general, these lists may contain any number of surfaces. Specific implementations may have specific requirements determining the minimum required number of surfaces for optimal operation, and the maximum number of useful surfaces, beyond which surfaces are not used. It is recommended that in all cases other than plain bob/weave, at least 2 past and 1 future field be provided.

Note that it is entirely possible, in general, for any of the [VdpVideoMixer](#) post-processing steps other than de-interlacing to require access to multiple input fields/frames. For example, an motion-sensitive noise-reduction algorithm.

For example, when processing field t4, the [VdpVideoMixerRender](#) parameters may contain the following values, if the application chose to provide 3 fields of context for both the past and future:

```
current_picture_structure: VDP_VIDEO_MIXER_PICTURE_STRUCTURE_TOP_FIELD
past:    [b3, t3, b2]
current: t4
future:  [b4, t5, b5]
```

Note that for both the past/future lists, array index 0 represents the field temporally closest to current, in display order.

The [VdpVideoMixerRender](#) parameter **current\_picture\_structure** applies to **video\_surface\_current**. The picture structure for the other surfaces will be automatically derived from that for the current picture. The derivation algorithm is extremely simple; the concatenated list past/current/future is simply assumed to have an alternating top/bottom pattern throughout.

Continuing the example above, subsequent calls to [VdpVideoMixerRender](#) would provide the following sets of parameters:

```
current_picture_structure: VDP_VIDEO_MIXER_PICTURE_STRUCTURE_BOTTOM_FIELD
past:      [t4, b3, t3]
current:   b4
future:    [t5, b5, t6]
```

then:

```
current_picture_structure: VDP_VIDEO_MIXER_PICTURE_STRUCTURE_TOP_FIELD
past:      [b4, t4, b3]
current:   t5
future:    [b5, t6, b7]
```

In other words, the concatenated list of past/current/future frames simply forms a window that slides through the sequence of decoded fields.

It is syntactically legal for an application to choose not to provide a particular entry in the past or future lists. In this case, the "slot" in the surface list must be filled with the special value [VDP\\_INVALID\\_HANDLE](#), to explicitly indicate that the picture is missing; do not simply shuffle other surfaces together to fill in the gap. Note that entries should only be omitted under special circumstances, such as failed decode due to bitstream error during picture header parsing, since missing entries will typically cause advanced de-interlacing algorithms to experience significantly degraded operation.

Specific examples for different de-interlacing types are presented below.

### 1.9.3 Weave De-interlacing

Weave de-interlacing is the act of interleaving the lines of two temporally adjacent fields to form a frame for display.

To disable de-interlacing for progressive streams, simply specify **current\_picture\_structure** as [VDP\\_VIDEO\\_MIXER\\_PICTURE\\_STRUCTURE\\_FRAME](#); no de-interlacing will be applied.

Weave de-interlacing for interlaced streams is identical to disabling de-interlacing, as describe immediately above, because each [VdpVideoSurface](#); [Video Surface object](#) already contains an entire frame's worth (i.e. two fields) of picture data.

Inverse telecine is disabled when using weave de-interlacing.

Weave de-interlacing produces one output frame for each input frame. The application should make one [VdpVideoMixerRender](#) call per pair of decoded fields, or per decoded frame.

Weave de-interlacing requires no entries in the past/future lists.

All implementations must support weave de-interlacing.

### 1.9.4 Bob De-interlacing

Bob de-interlacing is the act of vertically scaling a single field to the size of a single frame.

To achieve bob de-interlacing, simply provide a single field as `video_surface_current`, and set `current_picture↔_structure` appropriately, to indicate whether a top or bottom field was provided.

Inverse telecine is disabled when using bob de-interlacing.

Bob de-interlacing produces one output frame for each input field. The application should make one `VdpVideo↔MixerRender` call per decoded field.

Bob de-interlacing requires no entries in the past/future lists.

Bob de-interlacing is the default when no advanced method is requested and enabled. Advanced de-interlacing algorithms may fall back to bob e.g. when required past/future fields are missing.

All implementations must support bob de-interlacing.

### 1.9.5 Advanced De-interlacing

Operation of both temporal and temporal-spatial de-interlacing is identical; the only difference is the internal processing the algorithm performs in generating the output frame.

These algorithms use various advanced processing on the pixels of both the current and various past/future fields in order to determine how best to de-interlacing individual portions of the image.

Inverse telecine may be enabled when using advanced de-interlacing.

Advanced de-interlacing produces one output frame for each input field. The application should make one `Vdp↔VideoMixerRender` call per decoded field.

Advanced de-interlacing requires entries in the past/future lists.

Availability of advanced de-interlacing algorithms is implementation dependent.

### 1.9.6 De-interlacing Rate

For all de-interlacing algorithms except weave, a choice may be made to call `VdpVideoMixerRender` for either each decoded field, or every second decoded field.

If `VdpVideoMixerRender` is called for every decoded field, the generated post-processed frame rate is equal to the decoded field rate. Put another way, the generated post-processed nominal field rate is equal to 2x the decoded field rate. This is standard practice.

If `VdpVideoMixerRender` is called for every second decoded field (say every top field), the generated post-processed frame rate is half to the decoded field rate. This mode of operation is thus referred to as "half-rate".

Implementations may choose whether to support half-rate de-interlacing or not. Regular full-rate de-interlacing should be supported by any supported advanced de-interlacing algorithm.

The descriptions of de-interlacing algorithms above assume that regular (not half-rate) operation is being performed, when detailing the number of `VdpVideoMixerRender` calls.

Recall that the concatenation of past/current/future surface lists simply forms a window into the stream of decoded fields. To achieve standard de-interlacing, the window is slid through the list of decoded fields one field at a time, and a call is made to `VdpVideoMixerRender` for each movement of the window. To achieve half-rate de-interlacing, the window is slid through the\* list of decoded fields two fields at a time, and a call is made to `VdpVideoMixerRender` for each movement of the window.

### 1.9.7 Inverse Telecine

Assuming the implementation supports it, inverse telecine may be enabled alongside any advanced de-interlacing algorithm. Inverse telecine is never active for bob or weave.

Operation of [VdpVideoMixerRender](#) with inverse telecine active is identical to the basic operation mechanisms describe above in every way; all inverse telecine processing is performed internally to the [VdpVideoMixer](#).

In particular, there is no provision way for [VdpVideoMixerRender](#) to indicate when identical input fields have been observed, and consequently identical output frames may have been produced.

De-interlacing (and inverse telecine) may be applied to streams that are marked as being progressive. This will allow detection of, and correct de-interlacing of, mixed interlace/progressive streams, bad edits, etc. To implement de-interlacing/inverse-telecine on progressive material, simply treat the stream of decoded frames as a stream of decoded fields, apply any telecine flags (see the next section), and then apply de-interlacing to those fields as described above.

Implementations are free to determine whether inverse telecine operates in conjunction with half-rate de-interlacing or not. It should always operate with regular de-interlacing, when advertized.

### 1.9.8 Telecine (Pull-Down) Flags

Some media delivery formats, e.g. DVD-Video, include flags that are intended to modify the decoded field sequence before display. This allows e.g. 24p content to be encoded at 48i, which saves space relative to a 60i encoded stream, but still displayed at 60i, to match target consumer display equipment.

If the inverse telecine option is not activated in the [VdpVideoMixer](#), these flags should be ignored, and the decoded fields passed directly to [VdpVideoMixerRender](#) as detailed above.

However, to make full use of the inverse telecine feature, these flags should be applied to the field stream, yielding another field stream with some repeated fields, before passing the field stream to [VdpVideoMixerRender](#). In this scenario, the sliding window mentioned in the descriptions above applies to the field stream after application of flags.

## 1.10 Extending the API

### 1.10.1 Enumerations and Other Constants

VDPAU defines a number of enumeration types.

When modifying VDPAU, existing enumeration constants must continue to exist (although they may be deprecated), and do so in the existing order.

The above discussion naturally applies to "manually" defined enumerations, using pre-processor macros, too.

### 1.10.2 Structures

In most case, VDPAU includes no provision for modifying existing structure definitions, although they may be deprecated.

New structures may be created, together with new API entry points or feature/attribute/parameter values, to expose new functionality.

A few structures are considered plausible candidates for future extension. Such structures include a version number as the first field, indicating the exact layout of the client-provided data. When changing such structures, the old structure must be preserved and a new structure created. This allows applications built against the old version of the structure to continue to interoperate. For example, to extend the [VdpProcamp](#) structure, define a new `VdpProcamp1` and update `VdpGenerateCSCMatrix` to take the new structure as an argument. Document in a comment that the caller must fill the `struct_version` field with the value 1. VDPAU implementations should use the `struct_version` field to determine which version of the structure the application was built against. Note that you cannot simply increment the value of `VDP_PROCAMP_VERSION` because applications recompiled against a newer version of [vdpau.h](#) but that have not been updated to use the new structure must still report that they're using version 0.

Note that the layouts of `VdpPictureInfo` structures are defined by their corresponding `VdpDecoderProfile` numbers, so no `struct_version` field is needed for them. This layout includes the size of the structure, so new profiles that extend existing functionality may incorporate the old `VdpPictureInfo` as a substructure, but may not modify existing `VdpPictureInfo` structures.

### 1.10.3 Functions

Existing functions may not be modified, although they may be deprecated.

New functions may be added at will. Note the enumeration requirements when modifying the enumeration that defines the list of entry points.

## 1.11 Display Preemption

Please note that the display may be preempted away from VDPAU at any time. See [Display Preemption](#) for more details.

### 1.11.1 Trademarks

VDPAU is a trademark of NVIDIA Corporation. You may freely use the VDPAU trademark, as long as trademark ownership is attributed to NVIDIA Corporation.



## Chapter 2

# Module Index

### 2.1 Modules

Here is a list of all modules:

Core API . . . . .	20
Basic Types . . . . .	22
Miscellaneous Types . . . . .	23
Error Handling . . . . .	29
Versioning . . . . .	32
VdpDevice; Primary API object . . . . .	34
VdpCSCMatrix; CSC Matrix Manipulation . . . . .	35
VdpVideoSurface; Video Surface object . . . . .	38
VdpOutputSurface; Output Surfaceobject . . . . .	43
VdpBitmapSurface; Bitmap Surfaceobject . . . . .	50
VdpOutputSurface Rendering Functionality . . . . .	54
VdpDecoder; Video Decoding object . . . . .	60
VdpVideoMixer; Video Post-processing and Compositing object . . . . .	70
VdpPresentationQueue; Video presentation (display) object . . . . .	85
Display Preemption . . . . .	91
Entry Point Retrieval . . . . .	93
Window System Integration Layer . . . . .	98
X11 Window System Integration Layer . . . . .	99





## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">VdpBitstreamBuffer</a>	
Application data buffer containing compressed video data . . . . .	103
<a href="#">VdpColor</a> . . . . .	104
<a href="#">VdpLayer</a>	
Definition of an additional <a href="#">VdpOutputSurface</a> layer in the compositing model . . . . .	104
<a href="#">VdpOutputSurfaceRenderBlendState</a>	
Complete blending operation definition . . . . .	105
<a href="#">VdpPictureInfoH264</a>	
Picture parameter information for an H.264 picture . . . . .	106
<a href="#">VdpPictureInfoH264Predictive</a>	
Picture parameter information for an H.264 Hi444PP picture . . . . .	109
<a href="#">VdpPictureInfoHEVC</a>	
Picture parameter information for an H.265/HEVC picture . . . . .	110
<a href="#">VdpPictureInfoMPEG1Or2</a>	
Picture parameter information for an MPEG 1 or MPEG 2 picture . . . . .	119
<a href="#">VdpPictureInfoMPEG4Part2</a>	
Picture parameter information for an MPEG-4 Part 2 picture . . . . .	121
<a href="#">VdpPictureInfoVC1</a>	
Picture parameter information for a VC1 picture . . . . .	123
<a href="#">VdpPoint</a>	
A location within a surface . . . . .	127
<a href="#">VdpProcamp</a>	
Procamp operation parameterization data . . . . .	127
<a href="#">VdpRect</a>	
A rectangular region of a surface . . . . .	128
<a href="#">VdpReferenceFrameH264</a>	
Information about an H.264 reference frame . . . . .	129



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

vdpuu/vdpau.h	
The Core API . . . . .	131
vdpuu/vdpau_x11.h	
X11 Window System Integration Layer . . . . .	143

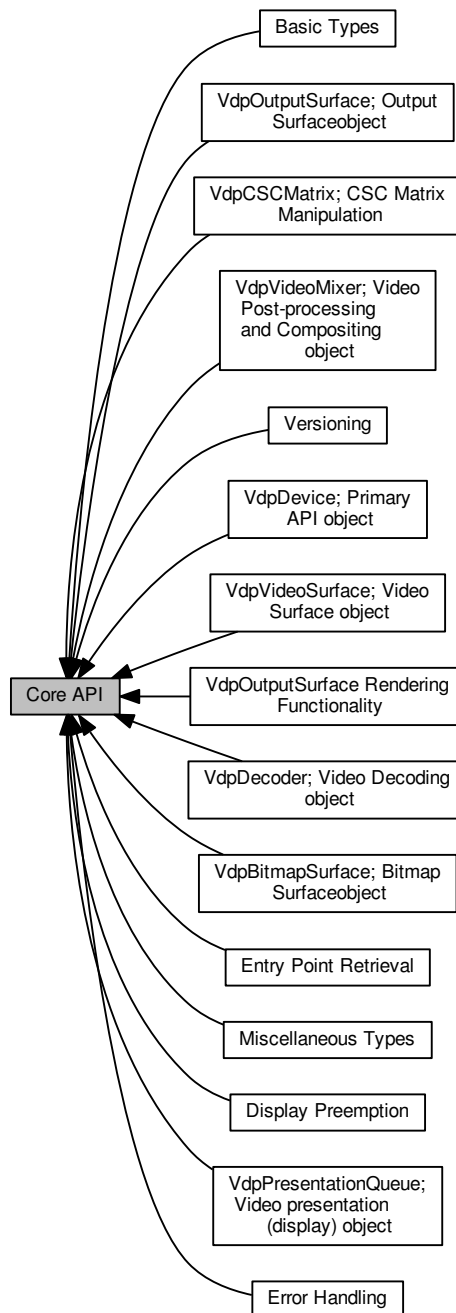


## **Chapter 5**

## **Module Documentation**

## 5.1 Core API

Collaboration diagram for Core API:



### Modules

- [Basic Types](#)
- [Miscellaneous Types](#)

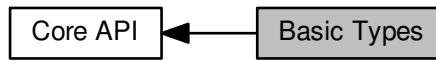
- [Error Handling](#)
- [Versioning](#)
- [VdpDevice](#); Primary API object
- [VdpCSCMatrix](#); CSC Matrix Manipulation
- [VdpVideoSurface](#); Video Surface object
- [VdpOutputSurface](#); Output Surfaceobject
- [VdpBitmapSurface](#); Bitmap Surfaceobject
- [VdpOutputSurface](#) Rendering Functionality
- [VdpDecoder](#); Video Decoding object
- [VdpVideoMixer](#); Video Post-processing and Compositing object
- [VdpPresentationQueue](#); Video presentation (display) object
- [Display Preemption](#)
- [Entry Point Retrieval](#)

### 5.1.1 Detailed Description

The core API encompasses all VDPAU functionality that operates in the same fashion across all Window Systems.

## 5.2 Basic Types

Collaboration diagram for Basic Types:



### Macros

- `#define VDP_TRUE 1`  
*A true `VdpBool` value.*
- `#define VDP_FALSE 0`  
*A false `VdpBool` value.*

### Typedefs

- `typedef int VdpBool`  
*A boolean value, holding `VDP_TRUE` or `VDP_FALSE`.*

#### 5.2.1 Detailed Description

VDP AU primarily uses ISO C99 types from `stdint.h`.

#### 5.2.2 Macro Definition Documentation

##### 5.2.2.1 `#define VDP_FALSE 0`

A false `VdpBool` value.

##### 5.2.2.2 `#define VDP_TRUE 1`

A true `VdpBool` value.

#### 5.2.3 Typedef Documentation

##### 5.2.3.1 `typedef int VdpBool`

A boolean value, holding `VDP_TRUE` or `VDP_FALSE`.



## 5.3 Miscellaneous Types

Collaboration diagram for Miscellaneous Types:



### Data Structures

- struct [VdpPoint](#)  
*A location within a surface.*
- struct [VdpRect](#)  
*A rectangular region of a surface.*
- struct [VdpColor](#)

### Macros

- #define [VDP\\_INVALID\\_HANDLE](#) 0xffffffffU  
*An invalid object handle value.*
- #define [VDP\\_CHROMA\\_TYPE\\_420](#)  
*4:2:0 chroma format.*
- #define [VDP\\_CHROMA\\_TYPE\\_422](#)  
*4:2:2 chroma format.*
- #define [VDP\\_CHROMA\\_TYPE\\_444](#)  
*4:4:4 chroma format.*
- #define [VDP\\_YCBCR\\_FORMAT\\_NV12](#)  
*The "NV12" YCbCr surface format.*
- #define [VDP\\_YCBCR\\_FORMAT\\_YV12](#)  
*The "YV12" YCbCr surface format.*
- #define [VDP\\_YCBCR\\_FORMAT\\_UYVY](#)  
*The "UYVY" YCbCr surface format.*
- #define [VDP\\_YCBCR\\_FORMAT\\_YUYV](#)  
*The "YUYV" YCbCr surface format.*
- #define [VDP\\_YCBCR\\_FORMAT\\_Y8U8V8A8](#)  
*A packed YCbCr format.*
- #define [VDP\\_YCBCR\\_FORMAT\\_V8U8Y8A8](#)  
*A packed YCbCr format.*
- #define [VDP\\_RGBA\\_FORMAT\\_B8G8R8A8](#)  
*A packed RGB format.*
- #define [VDP\\_RGBA\\_FORMAT\\_R8G8B8A8](#)  
*A packed RGB format.*
- #define [VDP\\_RGBA\\_FORMAT\\_R10G10B10A2](#)  
*A packed RGB format.*

- `#define VDP_RGBA_FORMAT_B10G10R10A2`  
*A packed RGB format.*
- `#define VDP_RGBA_FORMAT_A8`  
*An alpha-only surface format.*
- `#define VDP_INDEXED_FORMAT_A4I4`  
*A 4-bit indexed format, with alpha.*
- `#define VDP_INDEXED_FORMAT_I4A4`  
*A 4-bit indexed format, with alpha.*
- `#define VDP_INDEXED_FORMAT_A8I8`  
*A 8-bit indexed format, with alpha.*
- `#define VDP_INDEXED_FORMAT_I8A8`  
*A 8-bit indexed format, with alpha.*

## Typedefs

- `typedef uint32_t VdpChromaType`  
*The set of all chroma formats for [VdpVideoSurfaces](#).*
- `typedef uint32_t VdpYCbCrFormat`  
*The set of all known YCbCr surface formats.*
- `typedef uint32_t VdpRGBAFormat`  
*The set of all known RGB surface formats.*
- `typedef uint32_t VdpIndexedFormat`  
*The set of all known indexed surface formats.*

### 5.3.1 Detailed Description

### 5.3.2 Macro Definition Documentation

#### 5.3.2.1 `#define VDP_CHROMA_TYPE_420`

4:2:0 chroma format.

#### 5.3.2.2 `#define VDP_CHROMA_TYPE_422`

4:2:2 chroma format.

#### 5.3.2.3 `#define VDP_CHROMA_TYPE_444`

4:4:4 chroma format.

#### 5.3.2.4 `#define VDP_INDEXED_FORMAT_A4I4`

A 4-bit indexed format, with alpha.

This format has a single plane.

This plane is an array of byte-sized components. Within each byte, bits [7:4] contain I (index), and bits [3:0] contain A.

Applications should access this data via a `uint8_t` pointer.

#### 5.3.2.5 `#define VDP_INDEXED_FORMAT_A8I8`

A 8-bit indexed format, with alpha.

This format has a single plane.

This plane is an array of interleaved byte-sized A and I (index) components, in the order A, I, A, I.

Applications should access this data via a `uint8_t` pointer.

#### 5.3.2.6 `#define VDP_INDEXED_FORMAT_I4A4`

A 4-bit indexed format, with alpha.

This format has a single plane.

This plane is an array of byte-sized components. Within each byte, bits [7:4] contain A, and bits [3:0] contain I (index).

Applications should access this data via a `uint8_t` pointer.

#### 5.3.2.7 `#define VDP_INDEXED_FORMAT_I8A8`

A 8-bit indexed format, with alpha.

This format has a single plane.

This plane is an array of interleaved byte-sized A and I (index) components, in the order I, A, I, A.

Applications should access this data via a `uint8_t` pointer.

#### 5.3.2.8 `#define VDP_INVALID_HANDLE 0xffffffffU`

An invalid object handle value.

This value may be used to represent an invalid, or non-existent, object ([VdpDevice](#), [VdpVideoSurface](#), etc.)

Note that most APIs require valid object handles in all cases, and will fail when presented with this value.

#### 5.3.2.9 `#define VDP_RGBA_FORMAT_A8`

An alpha-only surface format.

This format has a single plane.

This plane is an array of byte-sized components.

Applications should access this data via a `uint8_t` pointer.

#### 5.3.2.10 `#define VDP_RGBA_FORMAT_B10G10R10A2`

A packed RGB format.

This format has a single plane.

This plane is an array packed 32-bit pixel data. Within each 32-bit pixel, bits [31:30] contain A, bits [29:20] contain R, bits [19:10] contain G, and bits [9:0] contain B.

Applications should access this data via a `uint32_t` pointer.

#### 5.3.2.11 `#define VDP_RGBA_FORMAT_B8G8R8A8`

A packed RGB format.

This format has a single plane.

This plane is an array packed 32-bit pixel data. Within each 32-bit pixel, bits [31:24] contain A, bits [23:16] contain R, bits [15:8] contain G, and bits [7:0] contain B.

Applications should access this data via a `uint32_t` pointer.

#### 5.3.2.12 `#define VDP_RGBA_FORMAT_R10G10B10A2`

A packed RGB format.

This format has a single plane.

This plane is an array packed 32-bit pixel data. Within each 32-bit pixel, bits [31:30] contain A, bits [29:20] contain B, bits [19:10] contain G, and bits [9:0] contain R.

Applications should access this data via a `uint32_t` pointer.

#### 5.3.2.13 `#define VDP_RGBA_FORMAT_R8G8B8A8`

A packed RGB format.

This format has a single plane.

This plane is an array packed 32-bit pixel data. Within each 32-bit pixel, bits [31:24] contain A, bits [23:16] contain B, bits [15:8] contain G, and bits [7:0] contain R.

Applications should access this data via a `uint32_t` pointer.

#### 5.3.2.14 `#define VDP_YCBCR_FORMAT_NV12`

The "NV12" YCbCr surface format.

This format has a two planes, a Y plane and a UV plane.

The Y plane is an array of byte-sized Y components. Applications should access this data via a `uint8_t` pointer.

The UV plane is an array of interleaved byte-sized U and V components, in the order U, V, U, V. Applications should access this data via a `uint8_t` pointer.

**5.3.2.15 #define VDP\_YCBCR\_FORMAT\_UYVY**

The "UYVY" YCbCr surface format.

This format may also be known as Y422, UYNV, HDYC.

This format has a single plane.

This plane is an array of interleaved byte-sized Y, U, and V components, in the order U, Y, V, Y, U, Y, V, Y.

Applications should access this data via a `uint8_t` pointer.

**5.3.2.16 #define VDP\_YCBCR\_FORMAT\_V8U8Y8A8**

A packed YCbCr format.

This format has a single plane.

This plane is an array packed 32-bit pixel data. Within each 32-bit pixel, bits [31:24] contain A, bits [23:16] contain Y, bits [15:8] contain U, and bits [7:0] contain V.

Applications should access this data via a `uint32_t` pointer.

**5.3.2.17 #define VDP\_YCBCR\_FORMAT\_Y8U8V8A8**

A packed YCbCr format.

This format has a single plane.

This plane is an array packed 32-bit pixel data. Within each 32-bit pixel, bits [31:24] contain A, bits [23:16] contain V, bits [15:8] contain U, and bits [7:0] contain Y.

Applications should access this data via a `uint32_t` pointer.

**5.3.2.18 #define VDP\_YCBCR\_FORMAT\_YUYV**

The "YUYV" YCbCr surface format.

This format may also be known as YUY2, YUNV, V422.

This format has a single plane.

This plane is an array of interleaved byte-sized Y, U, and V components, in the order Y, U, Y, V, Y, U, Y, V.

Applications should access this data via a `uint8_t` pointer.

**5.3.2.19 #define VDP\_YCBCR\_FORMAT\_YV12**

The "YV12" YCbCr surface format.

This format has a three planes, a Y plane, a V plane, and a U plane.

Each of the planes is an array of byte-sized components.

Applications should access this data via a `uint8_t` pointer.

### 5.3.3 Typedef Documentation

#### 5.3.3.1 typedef uint32\_t VdpChromaType

The set of all chroma formats for [VdpVideoSurfaces](#).

#### 5.3.3.2 typedef uint32\_t VdpIndexedFormat

The set of all known indexed surface formats.

#### 5.3.3.3 typedef uint32\_t VdpRGBAFormat

The set of all known RGB surface formats.

#### 5.3.3.4 typedef uint32\_t VdpYCbCrFormat

The set of all known YCbCr surface formats.

## 5.4 Error Handling

Collaboration diagram for Error Handling:



### Typedefs

- typedef char const \* [VdpGetErrorString\(VdpStatus status\)](#)

*Retrieve a string describing an error code.*

### Enumerations

- enum [VdpStatus](#) {  
[VDP\\_STATUS\\_OK](#) = 0, [VDP\\_STATUS\\_NO\\_IMPLEMENTATION](#), [VDP\\_STATUS\\_DISPLAY\\_PREEMPTED](#),  
[VDP\\_STATUS\\_INVALID\\_HANDLE](#),  
[VDP\\_STATUS\\_INVALID\\_POINTER](#), [VDP\\_STATUS\\_INVALID\\_CHROMA\\_TYPE](#), [VDP\\_STATUS\\_INVALID\\_](#)  
[D\\_Y\\_CB\\_CR\\_FORMAT](#), [VDP\\_STATUS\\_INVALID\\_RGBA\\_FORMAT](#),  
[VDP\\_STATUS\\_INVALID\\_INDEXED\\_FORMAT](#), [VDP\\_STATUS\\_INVALID\\_COLOR\\_STANDARD](#), [VDP\\_ST](#)  
[ATUS\\_INVALID\\_COLOR\\_TABLE\\_FORMAT](#), [VDP\\_STATUS\\_INVALID\\_BLEND\\_FACTOR](#),  
[VDP\\_STATUS\\_INVALID\\_BLEND\\_EQUATION](#), [VDP\\_STATUS\\_INVALID\\_FLAG](#), [VDP\\_STATUS\\_INVALID](#)  
[\\_DECODER\\_PROFILE](#), [VDP\\_STATUS\\_INVALID\\_VIDEO\\_MIXER\\_FEATURE](#),  
[VDP\\_STATUS\\_INVALID\\_VIDEO\\_MIXER\\_PARAMETER](#), [VDP\\_STATUS\\_INVALID\\_VIDEO\\_MIXER\\_ATT](#)  
[RIBUTE](#), [VDP\\_STATUS\\_INVALID\\_VIDEO\\_MIXER\\_PICTURE\\_STRUCTURE](#), [VDP\\_STATUS\\_INVALID](#)  
[FUNC\\_ID](#),  
[VDP\\_STATUS\\_INVALID\\_SIZE](#), [VDP\\_STATUS\\_INVALID\\_VALUE](#), [VDP\\_STATUS\\_INVALID\\_STRUCT\\_V](#)  
[ERSION](#), [VDP\\_STATUS\\_RESOURCES](#),  
[VDP\\_STATUS\\_HANDLE\\_DEVICE\\_MISMATCH](#), [VDP\\_STATUS\\_ERROR](#) }

*The set of all possible error codes.*

#### 5.4.1 Detailed Description

#### 5.4.2 Typedef Documentation

##### 5.4.2.1 typedef char const\* VdpGetErrorString(VdpStatus status)

Retrieve a string describing an error code.

#### Parameters

in	<i>status</i>	The error code.
----	---------------	-----------------

## Returns

A pointer to the string. Note that this is a statically allocated read-only string. As such, the application must not free the returned pointer. The pointer is valid as long as the VDPAU implementation is present within the application's address space.

### 5.4.3 Enumeration Type Documentation

#### 5.4.3.1 enum VdpStatus

The set of all possible error codes.

## Enumerator

- VDP\_STATUS\_OK** The operation completed successfully; no error.
- VDP\_STATUS\_NO\_IMPLEMENTATION** No backend implementation could be loaded.
- VDP\_STATUS\_DISPLAY\_PREEMPTED** The display was preempted, or a fatal error occurred.  
The application must re-initialize VDPAU.
- VDP\_STATUS\_INVALID\_HANDLE** An invalid handle value was provided.  
Either the handle does not exist at all, or refers to an object of an incorrect type.
- VDP\_STATUS\_INVALID\_POINTER** An invalid pointer was provided.  
Typically, this means that a NULL pointer was provided for an "output" parameter.
- VDP\_STATUS\_INVALID\_CHROMA\_TYPE** An invalid/unsupported [VdpChromaType](#) value was supplied.
- VDP\_STATUS\_INVALID\_Y\_CB\_CR\_FORMAT** An invalid/unsupported [VdpYCbCrFormat](#) value was supplied.
- VDP\_STATUS\_INVALID\_RGBA\_FORMAT** An invalid/unsupported [VdpRGBAFormat](#) value was supplied.
- VDP\_STATUS\_INVALID\_INDEXED\_FORMAT** An invalid/unsupported [VdpIndexedFormat](#) value was supplied.
- VDP\_STATUS\_INVALID\_COLOR\_STANDARD** An invalid/unsupported [VdpColorStandard](#) value was supplied.
- VDP\_STATUS\_INVALID\_COLOR\_TABLE\_FORMAT** An invalid/unsupported [VdpColorTableFormat](#) value was supplied.
- VDP\_STATUS\_INVALID\_BLEND\_FACTOR** An invalid/unsupported [VdpOutputSurfaceRenderBlendFactor](#) value was supplied.
- VDP\_STATUS\_INVALID\_BLEND\_EQUATION** An invalid/unsupported [VdpOutputSurfaceRenderBlendEquation](#) value was supplied.
- VDP\_STATUS\_INVALID\_FLAG** An invalid/unsupported flag value/combination was supplied.
- VDP\_STATUS\_INVALID\_DECODER\_PROFILE** An invalid/unsupported [VdpDecoderProfile](#) value was supplied.
- VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_FEATURE** An invalid/unsupported [VdpVideoMixerFeature](#) value was supplied.
- VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_PARAMETER** An invalid/unsupported [VdpVideoMixerParameter](#) value was supplied.
- VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_ATTRIBUTE** An invalid/unsupported [VdpVideoMixerAttribute](#) value was supplied.
- VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_PICTURE\_STRUCTURE** An invalid/unsupported [VdpVideoMixerPictureStructure](#) value was supplied.
- VDP\_STATUS\_INVALID\_FUNC\_ID** An invalid/unsupported [VdpFuncId](#) value was supplied.



**VDP\_STATUS\_INVALID\_SIZE** The size of a supplied object does not match the object it is being used with. For example, a [VdpVideoMixer](#) is configured to process [VdpVideoSurface](#) objects of a specific size. If presented with a [VdpVideoSurface](#) of a different size, this error will be raised.

**VDP\_STATUS\_INVALID\_VALUE** An invalid/unsupported value was supplied.

This is a catch-all error code for values of type other than those with a specific error code.

**VDP\_STATUS\_INVALID\_STRUCT\_VERSION** An invalid/unsupported structure version was specified in a versioned structure. This implies that the implementation is older than the header file the application was built against.

**VDP\_STATUS\_RESOURCES** The system does not have enough resources to complete the requested operation at this time.

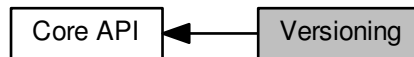
**VDP\_STATUS\_HANDLE\_DEVICE\_MISMATCH** The set of handles supplied are not all related to the same [VdpDevice](#).

When performing operations that operate on multiple surfaces, such as [VdpOutputSurfaceRender](#), [OutputSurface](#) or [VdpVideoMixerRender](#), all supplied surfaces must have been created within the context of the same [VdpDevice](#) object. This error is raised if they were not.

**VDP\_STATUS\_ERROR** A catch-all error, used when no other error code applies.

## 5.5 Versioning

Collaboration diagram for Versioning:



### Macros

- `#define VDDPAU_INTERFACE_VERSION 1`  
*The VDDPAU interface version described by this header file.*
- `#define VDDPAU_VERSION 1`  
*The VDDPAU version described by this header file.*

### Typedefs

- `typedef VdpStatus VdpGetApiVersion(uint32_t *api_version)`  
*Retrieve the VDDPAU version implemented by the backend.*
- `typedef VdpStatus VdpGetInformationString(char const **information_string)`  
*Retrieve an implementation-specific string description of the implementation. This typically includes detailed version information.*

#### 5.5.1 Detailed Description

#### 5.5.2 Macro Definition Documentation

##### 5.5.2.1 `#define VDDPAU_INTERFACE_VERSION 1`

The VDDPAU interface version described by this header file.

This version will only increase if a major incompatible change is made. For example, if the parameters passed to an existing function are modified, rather than simply adding new functions/enumerations), or if the mechanism used to load the backend driver is modified incompatibly. Such changes are unlikely.

This value also represents the DSO version of VDDPAU-related shared-libraries.

VDDPAU version numbers are simple integers that increase monotonically (typically by value 1).

### 5.5.2.2 `#define VDPAU_VERSION 1`

The VDPAU version described by this header file.

This version will increase whenever any non-documentation change is made to [vdpau.h](#), or related header files such as [vdpau\\_x11.h](#). Such changes typically involve the addition of new functions, constants, or features. Such changes are expected to be completely backwards-compatible.

VDPAU version numbers are simple integers that increase monotonically (typically by value 1).

## 5.5.3 Typedef Documentation

### 5.5.3.1 `typedef VdpStatus VdpGetApiVersion(uint32_t *api_version)`

Retrieve the VDPAU version implemented by the backend.

#### Parameters

out	<i>api_version</i>	The API version.
-----	--------------------	------------------

#### Returns

VdpStatus The completion status of the operation.

### 5.5.3.2 `typedef VdpStatus VdpGetInformationString(char const **information_string)`

Retrieve an implementation-specific string description of the implementation. This typically includes detailed version information.

#### Parameters

out	<i>information_string</i>	A pointer to the information string. Note that this is a statically allocated read-only string. As such, the application must not free the returned pointer. The pointer is valid as long as the implementation is present within the application's address space.
-----	---------------------------	--

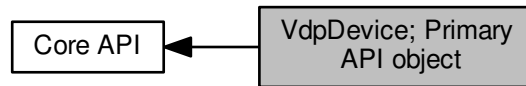
#### Returns

VdpStatus The completion status of the operation.

Note that the returned string is useful for information reporting. It is not intended that the application should parse this string in order to determine any information about the implementation.

## 5.6 VdpDevice; Primary API object

Collaboration diagram for VdpDevice; Primary API object:



### Typedefs

- typedef uint32\_t [VdpDevice](#)  
*An opaque handle representing a VdpDevice object.*
- typedef [VdpStatus](#) [VdpDeviceDestroy](#)([VdpDevice](#) device)  
*Destroy a VdpDevice.*

#### 5.6.1 Detailed Description

The VdpDevice is the root of the VDPAU object system. Using a VdpDevice object, all other object types may be created. See the sections describing those other object types for details on object creation.

Note that VdpDevice objects are created using the [Window System Integration Layer](#).

#### 5.6.2 Typedef Documentation

##### 5.6.2.1 typedef uint32\_t VdpDevice

An opaque handle representing a VdpDevice object.

##### 5.6.2.2 typedef VdpStatus VdpDeviceDestroy(VdpDevice device)

Destroy a VdpDevice.

###### Parameters

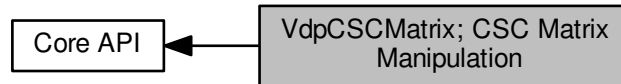
in	<i>device</i>	The device to destroy.
----	---------------	------------------------

###### Returns

VdpStatus The completion status of the operation.

## 5.7 VdpCSCMatrix; CSC Matrix Manipulation

Collaboration diagram for VdpCSCMatrix; CSC Matrix Manipulation:



### Data Structures

- struct [VdpProcamp](#)  
*Procamp operation parameterization data.*

### Macros

- `#define VDP_PROCAMB_VERSION 0`
- `#define VDP_COLOR_STANDARD_ITUR_BT_601`  
*ITU-R BT.601.*
- `#define VDP_COLOR_STANDARD_ITUR_BT_709`  
*ITU-R BT.709.*
- `#define VDP_COLOR_STANDARD_SMPTE_240M`  
*SMPTE-240M.*

### Typedefs

- `typedef float VdpCSCMatrix[3][4]`  
*Storage for a color space conversion matrix.*
- `typedef uint32_t VdpColorStandard`  
*YCbCr color space specification.*
- `typedef VdpStatus VdpGenerateCSCMatrix(VdpProcamp *procamp, VdpColorStandard standard, VdpCSCMatrix *csc_matrix)`  
*Generate a color space conversion matrix.*

#### 5.7.1 Detailed Description

When converting from YCbCr to RGB data formats, a color space conversion operation must be performed. This operation is parameterized using a "color space conversion matrix". The `VdpCSCMatrix` is a data structure representing this information.

## 5.7.2 Macro Definition Documentation

### 5.7.2.1 #define VDP\_COLOR\_STANDARD\_ITUR\_BT\_601

ITU-R BT.601.

### 5.7.2.2 #define VDP\_COLOR\_STANDARD\_ITUR\_BT\_709

ITU-R BT.709.

### 5.7.2.3 #define VDP\_COLOR\_STANDARD\_SMPTE\_240M

SMPTE-240M.

### 5.7.2.4 #define VDP\_PROCAMP\_VERSION 0

## 5.7.3 Typedef Documentation

### 5.7.3.1 typedef uint32\_t VdpColorStandard

YCbCr color space specification.

A number of YCbCr color spaces exist. This enumeration defines the specifications known to VDPAU.

### 5.7.3.2 typedef float VdpCSCMatrix[3][4]

Storage for a color space conversion matrix.

Note that the application may choose to construct the matrix content by either:

- Directly filling in the fields of the CSC matrix
- Using the [VdpGenerateCSCMatrix](#) helper function.

The color space conversion equation is as follows:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} m_{0,0} & m_{0,1} & m_{0,2} & m_{0,3} \\ m_{1,0} & m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,0} & m_{2,1} & m_{2,2} & m_{2,3} \end{pmatrix} * \begin{pmatrix} Y \\ Cb \\ Cr \\ 1.0 \end{pmatrix}$$

### 5.7.3.3 typedef VdpStatus VdpGenerateCSCMatrix(VdpProcamp \*procamp, VdpColorStandard standard, VdpCSCMatrix \*csc\_matrix)

Generate a color space conversion matrix.

**Parameters**

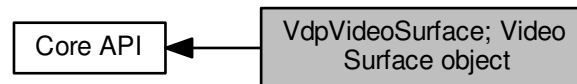
in	<i>procamp</i>	The procamp adjustments to make. If NULL, no adjustments will be made.
in	<i>standard</i>	The YCbCr color space to convert from.
out	<i>csc_matrix</i>	The CSC matrix to initialize.

**Returns**

VdpStatus The completion status of the operation.

## 5.8 VdpVideoSurface; Video Surface object

Collaboration diagram for VdpVideoSurface; Video Surface object:



### Typedefs

- typedef [VdpStatus](#) [VdpVideoSurfaceQueryCapabilities](#)([VdpDevice](#) device, [VdpChromaType](#) surface\_↔  
chroma\_type, [VdpBool](#) \*is\_supported, uint32\_t \*max\_width, uint32\_t \*max\_height)  
*Query the implementation's VdpVideoSurface capabilities.*
- typedef [VdpStatus](#) [VdpVideoSurfaceQueryGetPutBitsYCbCrCapabilities](#)([VdpDevice](#) device, [VdpChroma](#)↔  
[Type](#) surface\_chroma\_type, [VdpYCbCrFormat](#) bits\_ycbcr\_format, [VdpBool](#) \*is\_supported)  
*Query the implementation's VdpVideoSurface GetBits/PutBits capabilities.*
- typedef uint32\_t [VdpVideoSurface](#)  
*An opaque handle representing a VdpVideoSurface object.*
- typedef [VdpStatus](#) [VdpVideoSurfaceCreate](#)([VdpDevice](#) device, [VdpChromaType](#) chroma\_type, uint32\_↔  
t width, uint32\_t height, [VdpVideoSurface](#) \*surface)  
*Create a VdpVideoSurface.*
- typedef [VdpStatus](#) [VdpVideoSurfaceDestroy](#)([VdpVideoSurface](#) surface)  
*Destroy a VdpVideoSurface.*
- typedef [VdpStatus](#) [VdpVideoSurfaceGetParameters](#)([VdpVideoSurface](#) surface, [VdpChromaType](#) \*chroma↔  
\_type, uint32\_t \*width, uint32\_t \*height)  
*Retrieve the parameters used to create a VdpVideoSurface.*
- typedef [VdpStatus](#) [VdpVideoSurfaceGetBitsYCbCr](#)([VdpVideoSurface](#) surface, [VdpYCbCrFormat](#) destination↔  
\_ycbcr\_format, void \*const \*destination\_data, uint32\_t const \*destination\_pitches)  
*Copy image data from a VdpVideoSurface to application memory in a specified YCbCr format.*
- typedef [VdpStatus](#) [VdpVideoSurfacePutBitsYCbCr](#)([VdpVideoSurface](#) surface, [VdpYCbCrFormat](#) source\_↔  
ycbcr\_format, void const \*const \*source\_data, uint32\_t const \*source\_pitches)  
*Copy image data from application memory in a specific YCbCr format to a VdpVideoSurface.*

### 5.8.1 Detailed Description

A [VdpVideoSurface](#) stores YCbCr data in an internal format, with a variety of possible chroma sub-sampling options.

A [VdpVideoSurface](#) may be filled with:

- Data provided by the CPU via [VdpVideoSurfacePutBitsYCbCr](#) (i.e. software decode.)
- The result of applying a [VdpDecoder](#) to compressed video data.

[VdpVideoSurface](#) content may be accessed by:



- The application via [VdpVideoSurfaceGetBitsYCbCr](#)
- The Hardware that implements [VdpOutputSurface rendering functionality](#).
- The Hardware the implements [VdpVideoMixer](#) functionality.

VdpVideoSurfaces are not directly displayable. They must be converted into a displayable format using [VdpVideo↔Mixer](#) objects.

See [Video Mixer Usage](#) for additional information.

## 5.8.2 Typedef Documentation

### 5.8.2.1 typedef uint32\_t VdpVideoSurface

An opaque handle representing a VdpVideoSurface object.

### 5.8.2.2 typedef VdpStatus VdpVideoSurfaceCreate(VdpDevice device, VdpChromaType chroma\_type, uint32\_t width, uint32\_t height, VdpVideoSurface \*surface)

Create a VdpVideoSurface.

#### Parameters

in	<i>device</i>	The device that will contain the surface.
in	<i>chroma_type</i>	The chroma type of the new surface.
in	<i>width</i>	The width of the new surface.
in	<i>height</i>	The height of the new surface.
out	<i>surface</i>	The new surface's handle.

#### Returns

VdpStatus The completion status of the operation.

The memory backing the surface may not be initialized during creation. Applications are expected to initialize any region that they use, via [VdpDecoderRender](#) or [VdpVideoSurfacePutBitsYCbCr](#).

Note that certain widths/heights are impossible for specific values of chroma\_type. For example, the definition of VDP\_CHROMA\_TYPE\_420 implies that the width must be even, since each single chroma sample covers two luma samples horizontally. A similar argument applies to surface heights, although doubly so, since interlaced pictures must be supported; each field's height must itself be a multiple of 2. Hence the overall surface's height must be a multiple of 4.

Similar rules apply to other chroma\_type values.

Implementations may also impose additional restrictions on the surface sizes they support, potentially requiring additional rounding of actual surface sizes.

In most cases, this is not an issue, since:

- Video streams are encoded as an array of macro-blocks, which typically have larger size alignment requirements than video surfaces do.
- APIs such as [VdpVideoMixerRender](#) allow specification of a sub-region of the surface to read, which allows the padding data to be clipped away.

However, other APIs such as [VdpVideoSurfaceGetBitsYCbCr](#) and [VdpVideoSurfacePutBitsYCbCr](#) do not allow a sub-region to be specified, and always operate on surface size that was actually allocated, rather than the surface size that was requested. In this case, applications need to be aware of the actual surface size, in order to allocate appropriately sized buffers for the get-/put-bits operations.

For this reason, applications may need to call [VdpVideoSurfaceGetParameters](#) after creation, in order to retrieve the actual surface size.

#### 5.8.2.3 `typedef VdpStatus VdpVideoSurfaceDestroy(VdpVideoSurface surface)`

Destroy a `VdpVideoSurface`.

##### Parameters

in	<i>surface</i>	The surface's handle.
----	----------------	-----------------------

##### Returns

`VdpStatus` The completion status of the operation.

#### 5.8.2.4 `typedef VdpStatus VdpVideoSurfaceGetBitsYCbCr(VdpVideoSurface surface, VdpYCbCrFormat destination_ycbcr_format, void *const *destination_data, uint32_t const *destination_pitches)`

Copy image data from a `VdpVideoSurface` to application memory in a specified YCbCr format.

##### Parameters

in	<i>surface</i>	The surface's handle.
in	<i>destination_ycbcr_format</i>	The format of the application's data buffers.
in	<i>destination_data</i>	Pointers to the application data buffers into which the image data will be written. Note that this is an array of pointers, one per plane. The <i>destination_format</i> parameter will define how many planes are required.
in	<i>destination_pitches</i>	Pointers to the pitch values for the application data buffers. Note that this is an array of pointers, one per plane. The <i>destination_format</i> parameter will define how many planes are required.

##### Returns

`VdpStatus` The completion status of the operation.

#### 5.8.2.5 `typedef VdpStatus VdpVideoSurfaceGetParameters(VdpVideoSurface surface, VdpChromaType *chroma_type, uint32_t *width, uint32_t *height)`

Retrieve the parameters used to create a `VdpVideoSurface`.

## Parameters

in	<i>surface</i>	The surface's handle.
out	<i>chroma_type</i>	The chroma type of the surface.
out	<i>width</i>	The width of the surface.
out	<i>height</i>	The height of the surface.

## Returns

VdpStatus The completion status of the operation.

**5.8.2.6** `typedef VdpStatus VdpVideoSurfacePutBitsYCbCr(VdpVideoSurface surface, VdpYCbCrFormat source_ycbcr_format, void const *const *source_data, uint32_t const *source_pitches)`

Copy image data from application memory in a specific YCbCr format to a VdpVideoSurface.

## Parameters

in	<i>surface</i>	The surface's handle.
in	<i>source_ycbcr_format</i>	The format of the application's data buffers.
in	<i>source_data</i>	Pointers to the application data buffers from which the image data will be copied. Note that this is an array of pointers, one per plane. The source_format parameter will define how many planes are required.
in	<i>source_pitches</i>	Pointers to the pitch values for the application data buffers. Note that this is an array of pointers, one per plane. The source_format parameter will define how many planes are required.

## Returns

VdpStatus The completion status of the operation.

**5.8.2.7** `typedef VdpStatus VdpVideoSurfaceQueryCapabilities(VdpDevice device, VdpChromaType surface_chroma_type, VdpBool *is_supported, uint32_t *max_width, uint32_t *max_height)`

Query the implementation's VdpVideoSurface capabilities.

## Parameters

in	<i>device</i>	The device to query.
in	<i>surface_chroma_type</i>	The type of chroma type for which information is requested.
out	<i>is_supported</i>	Is this chroma type supported?
out	<i>max_width</i>	The maximum supported surface width for this chroma type.
out	<i>max_height</i>	The maximum supported surface height for this chroma type.

## Returns

VdpStatus The completion status of the operation.

5.8.2.8 `typedef VdpStatus VdpVideoSurfaceQueryGetPutBitsYCbCrCapabilities(VdpDevice device, VdpChromaType surface_chroma_type, VdpYCbCrFormat bits_ycbcr_format, VdpBool *is_supported)`

Query the implementation's VdpVideoSurface GetBits/PutBits capabilities.

#### Parameters

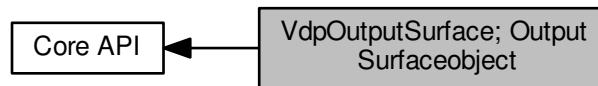
in	<i>device</i>	The device to query.
in	<i>surface_chroma_type</i>	The type of chroma type for which information is requested.
in	<i>bits_ycbcr_format</i>	The format of application "bits" buffer for which information is requested.
out	<i>is_supported</i>	Is this chroma type supported?

#### Returns

VdpStatus The completion status of the operation.

## 5.9 VdpOutputSurface; Output Surfaceobject

Collaboration diagram for VdpOutputSurface; Output Surfaceobject:



### Macros

- `#define VDP_COLOR_TABLE_FORMAT_B8G8R8X8`  
*8-bit per component packed into 32-bits*

### Typedefs

- `typedef uint32_t VdpColorTableFormat`  
*The set of all known color table formats, for use with [VdpOutputSurfacePutBitsIndexed](#).*
- `typedef VdpStatus VdpOutputSurfaceQueryCapabilities(VdpDevice device, VdpRGBAFormat surface_↔  
 rgba_format, VdpBool *is_supported, uint32_t *max_width, uint32_t *max_height)`  
*Query the implementation's VdpOutputSurface capabilities.*
- `typedef VdpStatus VdpOutputSurfaceQueryGetPutBitsNativeCapabilities(VdpDevice device, VdpRGBA↔  
 Format surface_rgba_format, VdpBool *is_supported)`  
*Query the implementation's capability to perform a PutBits operation using application data matching the surface's  
 format.*
- `typedef VdpStatus VdpOutputSurfaceQueryPutBitsIndexedCapabilities(VdpDevice device, VdpRGBAFormat  
 surface_rgba_format, VdpIndexedFormat bits_indexed_format, VdpColorTableFormat color_table_format,  
 VdpBool *is_supported)`  
*Query the implementation's capability to perform a PutBits operation using application data in a specific indexed  
 format.*
- `typedef VdpStatus VdpOutputSurfaceQueryPutBitsYCbCrCapabilities(VdpDevice device, VdpRGBAFormat  
 surface_rgba_format, VdpYCbCrFormat bits_ycbcr_format, VdpBool *is_supported)`  
*Query the implementation's capability to perform a PutBits operation using application data in a specific YCbCr/YUB  
 format.*
- `typedef uint32_t VdpOutputSurface`  
*An opaque handle representing a VdpOutputSurface object.*
- `typedef VdpStatus VdpOutputSurfaceCreate(VdpDevice device, VdpRGBAFormat rgba_format, uint32_t↔  
 width, uint32_t height, VdpOutputSurface *surface)`  
*Create a VdpOutputSurface.*
- `typedef VdpStatus VdpOutputSurfaceDestroy(VdpOutputSurface surface)`  
*Destroy a VdpOutputSurface.*
- `typedef VdpStatus VdpOutputSurfaceGetParameters(VdpOutputSurface surface, VdpRGBAFormat *rgba↔  
 _format, uint32_t *width, uint32_t *height)`  
*Retrieve the parameters used to create a VdpOutputSurface.*
- `typedef VdpStatus VdpOutputSurfaceGetBitsNative(VdpOutputSurface surface, VdpRect const *source_↔  
 rect, void *const *destination_data, uint32_t const *destination_pitches)`

*Copy image data from a VdpOutputSurface to application memory in the surface's native format.*

- typedef [VdpStatus](#) [VdpOutputSurfacePutBitsNative](#)([VdpOutputSurface](#) surface, void const \*const \*source↔\_data, uint32\_t const \*source\_pitches, [VdpRect](#) const \*destination\_rect)

*Copy image data from application memory in the surface's native format to a VdpOutputSurface.*

- typedef [VdpStatus](#) [VdpOutputSurfacePutBitsIndexed](#)([VdpOutputSurface](#) surface, [VdpIndexedFormat](#) source\_indexed\_format, void const \*const \*source\_data, uint32\_t const \*source\_pitch, [VdpRect](#) const \*destination\_rect, [VdpColorTableFormat](#) color\_table\_format, void const \*color\_table)

*Copy image data from application memory in a specific indexed format to a VdpOutputSurface.*

- typedef [VdpStatus](#) [VdpOutputSurfacePutBitsYCbCr](#)([VdpOutputSurface](#) surface, [VdpYCbCrFormat](#) source↔\_ycbcr\_format, void const \*const \*source\_data, uint32\_t const \*source\_pitches, [VdpRect](#) const \*destination\_rect, [VdpCSCMatrix](#) const \*csc\_matrix)

*Copy image data from application memory in a specific YCbCr format to a VdpOutputSurface.*

### 5.9.1 Detailed Description

A [VdpOutputSurface](#) stores RGBA data in a defined format.

A [VdpOutputSurface](#) may be filled with:

- Data provided by the CPU via the various [VdpOutputSurfacePutBits](#) functions.
- Using the [VdpOutputSurface rendering functionality](#).
- Using a [VdpVideoMixer](#) object.

[VdpOutputSurface](#) content may be accessed by:

- The application via the various [VdpOutputSurfaceGetBits](#) functions.
- The Hardware that implements [VdpOutputSurface rendering functionality](#).
- The Hardware the implements [VdpVideoMixer](#) functionality.
- The Hardware that implements [VdpPresentationQueue](#) functionality,

[VdpVideoSurfaces](#) are directly displayable using a [VdpPresentationQueue](#) object.

### 5.9.2 Macro Definition Documentation

#### 5.9.2.1 #define VDP\_COLOR\_TABLE\_FORMAT\_B8G8R8X8

8-bit per component packed into 32-bits

This format is an array of packed 32-bit RGB color values. Bits [31:24] are unused, bits [23:16] contain R, bits [15:8] contain G, and bits [7:0] contain B. Note: The format is physically an array of [uint32\\_t](#) values, and should be accessed as such by the application in order to avoid endianness issues.

### 5.9.3 Typedef Documentation

#### 5.9.3.1 typedef uint32\_t VdpColorTableFormat

The set of all known color table formats, for use with [VdpOutputSurfacePutBitsIndexed](#).

## 5.9.3.2 typedef uint32\_t VdpOutputSurface

An opaque handle representing a VdpOutputSurface object.

## 5.9.3.3 typedef VdpStatus VdpOutputSurfaceCreate(VdpDevice device, VdpRGBAFormat rgba\_format, uint32\_t width, uint32\_t height, VdpOutputSurface \*surface)

Create a VdpOutputSurface.

## Parameters

in	<i>device</i>	The device that will contain the surface.
in	<i>rgba_format</i>	The format of the new surface.
in	<i>width</i>	The width of the new surface.
in	<i>height</i>	The height of the new surface.
out	<i>surface</i>	The new surface's handle.

## Returns

VdpStatus The completion status of the operation.

The memory backing the surface will be initialized to 0 color and 0 alpha (i.e. black.)

## 5.9.3.4 typedef VdpStatus VdpOutputSurfaceDestroy(VdpOutputSurface surface)

Destroy a VdpOutputSurface.

## Parameters

in	<i>surface</i>	The surface's handle.
----	----------------	-----------------------

## Returns

VdpStatus The completion status of the operation.

## 5.9.3.5 typedef VdpStatus VdpOutputSurfaceGetBitsNative(VdpOutputSurface surface, VdpRect const \*source\_rect, void \*const \*destination\_data, uint32\_t const \*destination\_pitches)

Copy image data from a VdpOutputSurface to application memory in the surface's native format.

## Parameters

in	<i>surface</i>	The surface's handle.
in	<i>source_rect</i>	The sub-rectangle of the source surface to copy. If NULL, the entire surface will be retrieved.
in	<i>destination_data</i>	Pointers to the application data buffers into which the image data will be written. Note that this is an array of pointers, one per plane. The destination_format parameter will define how many planes are required.

**Parameters**

in	<i>destination_pitches</i>	Pointers to the pitch values for the application data buffers. Note that this is an array of pointers, one per plane. The <i>destination_format</i> parameter will define how many planes are required.
----	----------------------------	---

**Returns**

VdpStatus The completion status of the operation.

**5.9.3.6** `typedef VdpStatus VdpOutputSurfaceGetParameters(VdpOutputSurface surface, VdpRGBAFormat *rgba_format, uint32_t *width, uint32_t *height)`

Retrieve the parameters used to create a VdpOutputSurface.

**Parameters**

in	<i>surface</i>	The surface's handle.
out	<i>rgba_format</i>	The format of the surface.
out	<i>width</i>	The width of the surface.
out	<i>height</i>	The height of the surface.

**Returns**

VdpStatus The completion status of the operation.

**5.9.3.7** `typedef VdpStatus VdpOutputSurfacePutBitsIndexed(VdpOutputSurface surface, VdpIndexedFormat source_indexed_format, void const *const *source_data, uint32_t const *source_pitch, VdpRect const *destination_rect, VdpColorTableFormat color_table_format, void const *color_table)`

Copy image data from application memory in a specific indexed format to a VdpOutputSurface.

**Parameters**

in	<i>surface</i>	The surface's handle.
in	<i>source_indexed_format</i>	The format of the application's data buffers.
in	<i>source_data</i>	Pointers to the application data buffers from which the image data will be copied. Note that this is an array of pointers, one per plane. The <i>source_indexed_format</i> parameter will define how many planes are required.
in	<i>source_pitches</i>	Pointers to the pitch values for the application data buffers. Note that this is an array of pointers, one per plane. The <i>source_indexed_format</i> parameter will define how many planes are required.
in	<i>destination_rect</i>	The sub-rectangle of the surface to fill with application data. If NULL, the entire surface will be updated.
in	<i>color_table_format</i>	The format of the <i>color_table</i> .
in	<i>color_table</i>	A table that maps between source index and target color data. See <a href="#">VdpColorTableFormat</a> for details regarding the memory layout.



**Returns**

VdpStatus The completion status of the operation.

**5.9.3.8** `typedef VdpStatus VdpOutputSurfacePutBitsNative(VdpOutputSurface surface, void const *const *source_data, uint32_t const *source_pitches, VdpRect const *destination_rect)`

Copy image data from application memory in the surface's native format to a VdpOutputSurface.

**Parameters**

in	<i>surface</i>	The surface's handle.
in	<i>source_data</i>	Pointers to the application data buffers from which the image data will be copied. Note that this is an array of pointers, one per plane. The source_format parameter will define how many planes are required.
in	<i>source_pitches</i>	Pointers to the pitch values for the application data buffers. Note that this is an array of pointers, one per plane. The source_format parameter will define how many planes are required.
in	<i>destination_rect</i>	The sub-rectangle of the surface to fill with application data. If NULL, the entire surface will be updated.

**Returns**

VdpStatus The completion status of the operation.

**5.9.3.9** `typedef VdpStatus VdpOutputSurfacePutBitsYCbCr(VdpOutputSurface surface, VdpYCbCrFormat source_ycbcr_format, void const *const *source_data, uint32_t const *source_pitches, VdpRect const *destination_rect, VdpCSCMatrix const *csc_matrix)`

Copy image data from application memory in a specific YCbCr format to a VdpOutputSurface.

**Parameters**

in	<i>surface</i>	The surface's handle.
in	<i>source_ycbcr_format</i>	The format of the application's data buffers.
in	<i>source_data</i>	Pointers to the application data buffers from which the image data will be copied. Note that this is an array of pointers, one per plane. The source_ycbcr_format parameter will define how many planes are required.
in	<i>source_pitches</i>	Pointers to the pitch values for the application data buffers. Note that this is an array of pointers, one per plane. The source_ycbcr_format parameter will define how many planes are required.
in	<i>destination_rect</i>	The sub-rectangle of the surface to fill with application data. If NULL, the entire surface will be updated.
in	<i>csc_matrix</i>	The color space conversion matrix used by the copy operation. If NULL, a default matrix will be used internally. Th default matrix is equivalent to ITU-R BT.601 with no procamp changes.

**Returns**

VdpStatus The completion status of the operation.

**5.9.3.10** `typedef VdpStatus VdpOutputSurfaceQueryCapabilities(VdpDevice device, VdpRGBAFormat surface_rgba_format, VdpBool *is_supported, uint32_t *max_width, uint32_t *max_height)`

Query the implementation's VdpOutputSurface capabilities.

**Parameters**

in	<i>device</i>	The device to query.
in	<i>surface_rgba_format</i>	The surface format for which information is requested.
out	<i>is_supported</i>	Is this surface format supported?
out	<i>max_width</i>	The maximum supported surface width for this chroma type.
out	<i>max_height</i>	The maximum supported surface height for this chroma type.

**Returns**

VdpStatus The completion status of the operation.

**5.9.3.11** `typedef VdpStatus VdpOutputSurfaceQueryGetPutBitsNativeCapabilities(VdpDevice device, VdpRGBAFormat surface_rgba_format, VdpBool *is_supported)`

Query the implementation's capability to perform a PutBits operation using application data matching the surface's format.

**Parameters**

in	<i>device</i>	The device to query.
in	<i>surface_rgba_format</i>	The surface format for which information is requested.
out	<i>is_supported</i>	Is this surface format supported?

**Returns**

VdpStatus The completion status of the operation.

**5.9.3.12** `typedef VdpStatus VdpOutputSurfaceQueryPutBitsIndexedCapabilities(VdpDevice device, VdpRGBAFormat surface_rgba_format, VdpIndexedFormat bits_indexed_format, VdpColorTableFormat color_table_format, VdpBool *is_supported)`

Query the implementation's capability to perform a PutBits operation using application data in a specific indexed format.

**Parameters**

in	<i>device</i>	The device to query.
----	---------------	----------------------

**Parameters**

in	<i>surface_rgba_format</i>	The surface format for which information is requested.
in	<i>bits_indexed_format</i>	The format of the application data buffer.
in	<i>color_table_format</i>	The format of the color lookup table.
out	<i>is_supported</i>	Is this surface format supported?

**Returns**

VdpStatus The completion status of the operation.

**5.9.3.13** `typedef VdpStatus VdpOutputSurfaceQueryPutBitsYCbCrCapabilities(VdpDevice device, VdpRGBAFormat surface_rgba_format, VdpYCbCrFormat bits_ycbcr_format, VdpBool *is_supported)`

Query the implementation's capability to perform a PutBits operation using application data in a specific YCbCr/YUB format.

**Parameters**

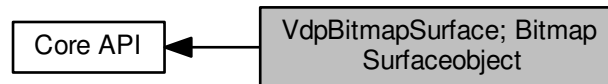
in	<i>device</i>	The device to query.
in	<i>surface_rgba_format</i>	The surface format for which information is requested.
in	<i>bits_ycbcr_format</i>	The format of the application data buffer.
out	<i>is_supported</i>	Is this surface format supported?

**Returns**

VdpStatus The completion status of the operation.

## 5.10 VdpBitmapSurface; Bitmap Surfaceobject

Collaboration diagram for VdpBitmapSurface; Bitmap Surfaceobject:



### Typedefs

- typedef [VdpStatus](#) [VdpBitmapSurfaceQueryCapabilities](#)([VdpDevice](#) device, [VdpRGBAFormat](#) surface\_↔  
rgba\_format, [VdpBool](#) \*is\_supported, uint32\_t \*max\_width, uint32\_t \*max\_height)  
*Query the implementation's VdpBitmapSurface capabilities.*
- typedef uint32\_t [VdpBitmapSurface](#)  
*An opaque handle representing a VdpBitmapSurface object.*
- typedef [VdpStatus](#) [VdpBitmapSurfaceCreate](#)([VdpDevice](#) device, [VdpRGBAFormat](#) rgba\_format, uint32\_↔  
t width, uint32\_t height, [VdpBool](#) frequently\_accessed, [VdpBitmapSurface](#) \*surface)  
*Create a VdpBitmapSurface.*
- typedef [VdpStatus](#) [VdpBitmapSurfaceDestroy](#)([VdpBitmapSurface](#) surface)  
*Destroy a VdpBitmapSurface.*
- typedef [VdpStatus](#) [VdpBitmapSurfaceGetParameters](#)([VdpBitmapSurface](#) surface, [VdpRGBAFormat](#) \*rgba\_↔  
\_format, uint32\_t \*width, uint32\_t \*height, [VdpBool](#) \*frequently\_accessed)  
*Retrieve the parameters used to create a VdpBitmapSurface.*
- typedef [VdpStatus](#) [VdpBitmapSurfacePutBitsNative](#)([VdpBitmapSurface](#) surface, void const \*const \*source\_↔  
\_data, uint32\_t const \*source\_pitches, [VdpRect](#) const \*destination\_rect)  
*Copy image data from application memory in the surface's native format to a VdpBitmapSurface.*

### 5.10.1 Detailed Description

A [VdpBitmapSurface](#) stores RGBA data in a defined format.

A [VdpBitmapSurface](#) may be filled with:

- Data provided by the CPU via the [VdpBitmapSurfacePutBitsNative](#) function.

[VdpBitmapSurface](#) content may be accessed by:

- The Hardware that implements [VdpOutputSurface rendering functionality](#)

[VdpBitmapSurface](#) objects are intended to store static read-only data, such as font glyphs, and the bitmaps used to compose an applications' user-interface.

The primary differences between [VdpBitmapSurfaces](#) and [VdpOutputSurfaces](#) are:

- You cannot render to a [VdpBitmapSurface](#), just upload native data via the PutBits API.
- The read-only nature of a [VdpBitmapSurface](#) gives the implementation more flexibility in its choice of data storage location for the bitmap data. For example, some implementations may choose to store some/all [VdpBitmapSurface](#) objects in system memory to relieve GPU memory pressure.
- [VdpBitmapSurface](#) and [VdpOutputSurface](#) may support different subsets of all known RGBA formats.

## 5.10.2 Typedef Documentation

### 5.10.2.1 typedef uint32\_t VdpBitmapSurface

An opaque handle representing a VdpBitmapSurface object.

### 5.10.2.2 typedef VdpStatus VdpBitmapSurfaceCreate(VdpDevice device, VdpRGBAFormat rgba\_format, uint32\_t width, uint32\_t height, VdpBool frequently\_accessed, VdpBitmapSurface \*surface)

Create a VdpBitmapSurface.

#### Parameters

in	<i>device</i>	The device that will contain the surface.
in	<i>rgba_format</i>	The format of the new surface.
in	<i>width</i>	The width of the new surface.
in	<i>height</i>	The height of the new surface.
in	<i>frequently_accessed</i>	Is this bitmap used frequently, or infrequently, by compositing options? Implementations may use this as a hint to determine how to allocate the underlying storage for the surface.
out	<i>surface</i>	The new surface's handle.

#### Returns

VdpStatus The completion status of the operation.

The memory backing the surface may not be initialized during creation. Applications are expected initialize any region that they use, via [VdpBitmapSurfacePutBitsNative](#).

### 5.10.2.3 typedef VdpStatus VdpBitmapSurfaceDestroy(VdpBitmapSurface surface)

Destroy a VdpBitmapSurface.

#### Parameters

in	<i>surface</i>	The surface's handle.
----	----------------	-----------------------

#### Returns

VdpStatus The completion status of the operation.

### 5.10.2.4 typedef VdpStatus VdpBitmapSurfaceGetParameters(VdpBitmapSurface surface, VdpRGBAFormat \*rgba\_format, uint32\_t \*width, uint32\_t \*height, VdpBool \*frequently\_accessed)

Retrieve the parameters used to create a VdpBitmapSurface.

**Parameters**

in	<i>surface</i>	The surface's handle.
out	<i>rgba_format</i>	The format of the surface.
out	<i>width</i>	The width of the surface.
out	<i>height</i>	The height of the surface.
out	<i>frequently_accessed</i>	The frequently_accessed state of the surface.

**Returns**

VdpStatus The completion status of the operation.

**5.10.2.5** `typedef VdpStatus VdpBitmapSurfacePutBitsNative(VdpBitmapSurface surface, void const *const *source_data, uint32_t const *source_pitches, VdpRect const *destination_rect)`

Copy image data from application memory in the surface's native format to a VdpBitmapSurface.

**Parameters**

in	<i>surface</i>	The surface's handle.
in	<i>source_data</i>	Pointers to the application data buffers from which the image data will be copied. Note that this is an array of pointers, one per plane. The source_format parameter will define how many planes are required.
in	<i>source_pitches</i>	Pointers to the pitch values for the application data buffers. Note that this is an array of pointers, one per plane. The source_format parameter will define how many planes are required.
in	<i>destination_rect</i>	The sub-rectangle of the surface to fill with application data. If NULL, the entire surface will be updated.

**Returns**

VdpStatus The completion status of the operation.

**5.10.2.6** `typedef VdpStatus VdpBitmapSurfaceQueryCapabilities(VdpDevice device, VdpRGBAFormat surface_rgba_format, VdpBool *is_supported, uint32_t *max_width, uint32_t *max_height)`

Query the implementation's VdpBitmapSurface capabilities.

**Parameters**

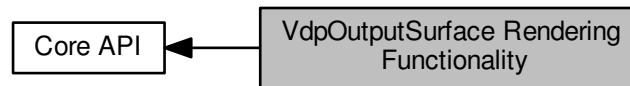
in	<i>device</i>	The device to query.
in	<i>surface_rgba_format</i>	The surface format for which information is requested.
out	<i>is_supported</i>	Is this surface format supported?
out	<i>max_width</i>	The maximum supported surface width for this chroma type.
out	<i>max_height</i>	The maximum supported surface height for this chroma type.

**Returns**

VdpStatus The completion status of the operation.

## 5.11 VdpOutputSurface Rendering Functionality

Collaboration diagram for VdpOutputSurface Rendering Functionality:



### Data Structures

- struct [VdpOutputSurfaceRenderBlendState](#)

*Complete blending operation definition.*

### Macros

- #define [VDP\\_OUTPUT\\_SURFACE\\_RENDER\\_BLEND\\_STATE\\_VERSION](#) 0
- #define [VDP\\_OUTPUT\\_SURFACE\\_RENDER\\_ROTATE\\_0](#)

*Do not rotate source\_surface prior to compositing.*

- #define [VDP\\_OUTPUT\\_SURFACE\\_RENDER\\_ROTATE\\_90](#)

*Rotate source\_surface 90 degrees clockwise prior to compositing.*

- #define [VDP\\_OUTPUT\\_SURFACE\\_RENDER\\_ROTATE\\_180](#)

*Rotate source\_surface 180 degrees prior to compositing.*

- #define [VDP\\_OUTPUT\\_SURFACE\\_RENDER\\_ROTATE\\_270](#)

*Rotate source\_surface 270 degrees clockwise prior to compositing.*

- #define [VDP\\_OUTPUT\\_SURFACE\\_RENDER\\_COLOR\\_PER\\_VERTEX](#)

*A separate color is used for each vertex of the smooth-shaded quad. Hence, colors array contains 4 elements rather than 1. See description of colors array.*

### Typedefs

- typedef [VdpStatus](#) [VdpOutputSurfaceRenderOutputSurface](#)([VdpOutputSurface](#) destination\_surface, [VdpRect](#) const \*destination\_rect, [VdpOutputSurface](#) source\_surface, [VdpRect](#) const \*source\_rect, [VdpColor](#) const \*colors, [VdpOutputSurfaceRenderBlendState](#) const \*blend\_state, uint32\_t flags)

*Composite a sub-rectangle of a [VdpOutputSurface](#) into a sub-rectangle of another [VdpOutputSurface](#); [Output Surfaceobject](#) [VdpOutputSurface](#).*

- typedef [VdpStatus](#) [VdpOutputSurfaceRenderBitmapSurface](#)([VdpOutputSurface](#) destination\_surface, [VdpRect](#) const \*destination\_rect, [VdpBitmapSurface](#) source\_surface, [VdpRect](#) const \*source\_rect, [VdpColor](#) const \*colors, [VdpOutputSurfaceRenderBlendState](#) const \*blend\_state, uint32\_t flags)

*Composite a sub-rectangle of a [VdpBitmapSurface](#) into a sub-rectangle of a [VdpOutputSurface](#); [Output Surfaceobject](#) [VdpOutputSurface](#).*



## Enumerations

- enum `VdpOutputSurfaceRenderBlendFactor` {  
`VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_ZERO` = 0, `VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_ONE` = 1, `VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_SRC_COLOR` = 2, `VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_ONE_MINUS_SRC_COLOR` = 3,  
`VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_SRC_ALPHA` = 4, `VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_ONE_MINUS_SRC_ALPHA` = 5, `VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_DST_ALPHA` = 6, `VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_ONE_MINUS_DST_ALPHA` = 7,  
`VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_DST_COLOR` = 8, `VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_ONE_MINUS_DST_COLOR` = 9, `VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_SRC_ALPHA_SATURATE` = 10, `VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_CONSTANT_COLOR` = 11,  
`VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_ONE_MINUS_CONSTANT_COLOR` = 12, `VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_CONSTANT_ALPHA` = 13, `VDP_OUTPUT_SURFACE_RENDER_BLEND_FACTOR_ONE_MINUS_CONSTANT_ALPHA` = 14 }

*The blending equation factors.*

- enum `VdpOutputSurfaceRenderBlendEquation` {  
`VDP_OUTPUT_SURFACE_RENDER_BLEND_EQUATION_SUBTRACT` = 0, `VDP_OUTPUT_SURFACE_RENDER_BLEND_EQUATION_REVERSE_SUBTRACT` = 1, `VDP_OUTPUT_SURFACE_RENDER_BLEND_EQUATION_ADD` = 2, `VDP_OUTPUT_SURFACE_RENDER_BLEND_EQUATION_MIN` = 3,  
`VDP_OUTPUT_SURFACE_RENDER_BLEND_EQUATION_MAX` = 4 }

*The blending equations.*

### 5.11.1 Detailed Description

`VdpOutputSurface` objects directly provide some rendering/compositing operations. These are described below.

### 5.11.2 Macro Definition Documentation

#### 5.11.2.1 `#define VDP_OUTPUT_SURFACE_RENDER_BLEND_STATE_VERSION 0`

#### 5.11.2.2 `#define VDP_OUTPUT_SURFACE_RENDER_COLOR_PER_VERTEX`

A separate color is used for each vertex of the smooth-shaded quad. Hence, colors array contains 4 elements rather than 1. See description of colors array.

#### 5.11.2.3 `#define VDP_OUTPUT_SURFACE_RENDER_ROTATE_0`

Do not rotate source\_surface prior to compositing.

#### 5.11.2.4 `#define VDP_OUTPUT_SURFACE_RENDER_ROTATE_180`

Rotate source\_surface 180 degrees prior to compositing.

#### 5.11.2.5 `#define VDP_OUTPUT_SURFACE_RENDER_ROTATE_270`

Rotate source\_surface 270 degrees clockwise prior to compositing.

#### 5.11.2.6 #define VDP\_OUTPUT\_SURFACE\_RENDER\_ROTATE\_90

Rotate `source_surface` 90 degrees clockwise prior to compositing.

### 5.11.3 Typedef Documentation

**5.11.3.1** `typedef VdpStatus VdpOutputSurfaceRenderBitmapSurface(VdpOutputSurface destination_surface, VdpRect const *destination_rect, VdpBitmapSurface source_surface, VdpRect const *source_rect, VdpColor const *colors, VdpOutputSurfaceRenderBlendState const *blend_state, uint32_t flags)`

Composite a sub-rectangle of a [VdpBitmapSurface](#) into a sub-rectangle of a [VdpOutputSurface](#); [Output Surface object](#) [VdpOutputSurface](#).

#### Parameters

in	<i>destination_surface</i>	The destination surface of the compositing operation.
in	<i>destination_rect</i>	The sub-rectangle of the destination surface to update. If NULL, the entire destination surface will be updated.
in	<i>source_surface</i>	The source surface for the compositing operation. The surface is treated as having four components: red, green, blue and alpha. Any missing components are treated as 1.0. For example, for an A8 <a href="#">VdpBitmapSurface</a> , alpha will come from the surface but red, green and blue will be treated as 1.0. If <code>source_surface</code> is <code>VDP_INVALID_HANDLE</code> , all components will be treated as 1.0. Note that <code>destination_surface</code> and <code>source_surface</code> must have been allocated via the same <a href="#">VdpDevice</a> .
in	<i>source_rect</i>	The sub-rectangle of the source surface to read from. If NULL, the entire <code>source_surface</code> will be read. Left/right or top/bottom co-ordinates may be swapped to flip the source. Any flip occurs prior to any requested rotation. Values from outside the source surface are valid and samples at those locations will be taken from the nearest edge.
in	<i>colors</i>	A pointer to an array of <a href="#">VdpColor</a> objects. If the flag <code>VDP_OUTPUT_SURFACE_RENDER_COLOR_PER_VERTEX</code> is set, VDPAU will use four entries from the array, and treat them as the colors corresponding to the upper-left, upper-right, lower-right and lower-left corners of the post-rotation source (i.e. indices 0, 1, 2 and 3 run clockwise from the upper left corner). If the flag <code>VDP_OUTPUT_SURFACE_RENDER_COLOR_PER_VERTEX</code> is not set, VDPAU will use the single <a href="#">VdpColor</a> for all four corners. If <code>colors</code> is NULL then red, green, blue and alpha values of 1.0 will be used.
in	<i>blend_state</i>	If a blend state is provided, the blend state will be used for the composite operation. If NULL, blending is effectively disabled, which is equivalent to a blend equation of ADD, source blend factors of ONE and destination blend factors of ZERO. See <a href="#">VdpOutputSurfaceRenderBlendState</a> for details regarding the mathematics of the blending operation.
in	<i>flags</i>	A set of flags influencing how the compositing operation works. <ul style="list-style-type: none"> <li>• <a href="#">VDP_OUTPUT_SURFACE_RENDER_ROTATE_0</a></li> <li>• <a href="#">VDP_OUTPUT_SURFACE_RENDER_ROTATE_90</a></li> <li>• <a href="#">VDP_OUTPUT_SURFACE_RENDER_ROTATE_180</a></li> <li>• <a href="#">VDP_OUTPUT_SURFACE_RENDER_ROTATE_270</a></li> <li>• <a href="#">VDP_OUTPUT_SURFACE_RENDER_COLOR_PER_VERTEX</a></li> </ul>

**Returns**

VdpStatus The completion status of the operation.

The general compositing pipeline is as follows.

1. Extract source\_rect from source\_surface.
2. The extracted source is rotated 0, 90, 180 or 270 degrees according to the flags.
3. The rotated source is component-wise multiplied by a smooth-shaded quad with a (potentially) different color at each vertex.
4. The resulting rotated, smooth-shaded quad is scaled to the size of destination\_rect and composited with destination\_surface using the provided blend state.

**5.11.3.2** `typedef VdpStatus VdpOutputSurfaceRenderOutputSurface(VdpOutputSurface destination_surface, VdpRect const *destination_rect, VdpOutputSurface source_surface, VdpRect const *source_rect, VdpColor const *colors, VdpOutputSurfaceRenderBlendState const *blend_state, uint32_t flags)`

Composite a sub-rectangle of a [VdpOutputSurface](#) into a sub-rectangle of another [VdpOutputSurface](#); [Output Surfaceobject](#) VdpOutputSurface.

**Parameters**

in	<i>destination_surface</i>	The destination surface of the compositing operation.
in	<i>destination_rect</i>	The sub-rectangle of the destination surface to update. If NULL, the entire destination surface will be updated.
in	<i>source_surface</i>	The source surface for the compositing operation. The surface is treated as having four components: red, green, blue and alpha. Any missing components are treated as 1.0. For example, for an A8 VdpOutputSurface, alpha will come from the surface but red, green and blue will be treated as 1.0. If source_surface is VDP_INVALID_HANDLE, all components will be treated as 1.0. Note that destination_surface and source_surface must have been allocated via the same <a href="#">VdpDevice</a> .
in	<i>source_rect</i>	The sub-rectangle of the source surface to read from. If NULL, the entire source_surface will be read. Left/right and/or top/bottom co-ordinates may be swapped to flip the source. Any flip occurs prior to any requested rotation. Values from outside the source surface are valid and samples at those locations will be taken from the nearest edge.
in	<i>colors</i>	A pointer to an array of <a href="#">VdpColor</a> objects. If the flag VDP_OUTPUT_SURFACE_RENDER_COLOR_PER_VERTEX is set, VDPAU will use four entries from the array, and treat them as the colors corresponding to the upper-left, upper-right, lower-right and lower-left corners of the post-rotation source (i.e. indices 0, 1, 2 and 3 run clockwise from the upper left corner). If the flag VDP_OUTPUT_SURFACE_RENDER_COLOR_PER_VERTEX is not set, VDPAU will use the single <a href="#">VdpColor</a> for all four corners. If colors is NULL then red, green, blue and alpha values of 1.0 will be used.
in	<i>blend_state</i>	If a blend state is provided, the blend state will be used for the composite operation. If NULL, blending is effectively disabled, which is equivalent to a blend equation of ADD, source blend factors of ONE and destination blend factors of ZERO. See <a href="#">VdpOutputSurfaceRenderBlendState</a> for details regarding the mathematics of the blending operation.

## Parameters

in	<i>flags</i>	A set of flags influencing how the compositing operation works. <ul style="list-style-type: none"> <li>• <a href="#">VDP_OUTPUT_SURFACE_RENDER_ROTATE_0</a></li> <li>• <a href="#">VDP_OUTPUT_SURFACE_RENDER_ROTATE_90</a></li> <li>• <a href="#">VDP_OUTPUT_SURFACE_RENDER_ROTATE_180</a></li> <li>• <a href="#">VDP_OUTPUT_SURFACE_RENDER_ROTATE_270</a></li> <li>• <a href="#">VDP_OUTPUT_SURFACE_RENDER_COLOR_PER_VERTEX</a></li> </ul>
----	--------------	---

## Returns

VdpStatus The completion status of the operation.

The general compositing pipeline is as follows.

1. Extract source\_rect from source\_surface.
2. The extracted source is rotated 0, 90, 180 or 270 degrees according to the flags.
3. The rotated source is component-wise multiplied by a smooth-shaded quad with a (potentially) different color at each vertex.
4. The resulting rotated, smooth-shaded quad is scaled to the size of destination\_rect and composited with destination\_surface using the provided blend state.

## 5.11.4 Enumeration Type Documentation

### 5.11.4.1 enum VdpOutputSurfaceRenderBlendEquation

The blending equations.

## Enumerator

***VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_EQUATION\_SUBTRACT***  
***VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_EQUATION\_REVERSE\_SUBTRACT***  
***VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_EQUATION\_ADD***  
***VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_EQUATION\_MIN***  
***VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_EQUATION\_MAX***

## 5.11.4.2 enum VdpOutputSurfaceRenderBlendFactor

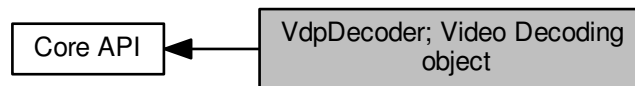
The blending equation factors.

Enumerator

*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ZERO*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_SRC\_COLOR*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE\_MINUS\_SRC\_COLOR*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_SRC\_ALPHA*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE\_MINUS\_SRC\_ALPHA*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_DST\_ALPHA*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE\_MINUS\_DST\_ALPHA*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_DST\_COLOR*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE\_MINUS\_DST\_COLOR*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_SRC\_ALPHA\_SATURATE*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_CONSTANT\_COLOR*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE\_MINUS\_CONSTANT\_COLOR*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_CONSTANT\_ALPHA*  
*VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE\_MINUS\_CONSTANT\_ALPHA*

## 5.12 VdpDecoder; Video Decoding object

Collaboration diagram for VdpDecoder; Video Decoding object:



### Data Structures

- struct [VdpBitstreamBuffer](#)  
*Application data buffer containing compressed video data.*
- struct [VdpPictureInfoMPEG1Or2](#)  
*Picture parameter information for an MPEG 1 or MPEG 2 picture.*
- struct [VdpReferenceFrameH264](#)  
*Information about an H.264 reference frame.*
- struct [VdpPictureInfoH264](#)  
*Picture parameter information for an H.264 picture.*
- struct [VdpPictureInfoH264Predictive](#)  
*Picture parameter information for an H.264 Hi444PP picture.*
- struct [VdpPictureInfoVC1](#)  
*Picture parameter information for a VC1 picture.*
- struct [VdpPictureInfoMPEG4Part2](#)  
*Picture parameter information for an MPEG-4 Part 2 picture.*
- struct [VdpPictureInfoHEVC](#)  
*Picture parameter information for an H.265/HEVC picture.*

### Macros

- `#define VDP_DECODER_PROFILE_MPEG1`
- `#define VDP_DECODER_PROFILE_MPEG2_SIMPLE`
- `#define VDP_DECODER_PROFILE_MPEG2_MAIN`
- `#define VDP_DECODER_PROFILE_H264_BASELINE`  
*MPEG 4 part 10 == H.264 == AVC.*
- `#define VDP_DECODER_PROFILE_H264_MAIN`
- `#define VDP_DECODER_PROFILE_H264_HIGH`
- `#define VDP_DECODER_PROFILE_VC1_SIMPLE`
- `#define VDP_DECODER_PROFILE_VC1_MAIN`
- `#define VDP_DECODER_PROFILE_VC1_ADVANCED`
- `#define VDP_DECODER_PROFILE_MPEG4_PART2_SP`
- `#define VDP_DECODER_PROFILE_MPEG4_PART2_ASP`
- `#define VDP_DECODER_PROFILE_DIVX4_QMOBILE`
- `#define VDP_DECODER_PROFILE_DIVX4_MOBILE`
- `#define VDP_DECODER_PROFILE_DIVX4_HOME_THEATER`

- #define VDP\_DECODER\_PROFILE\_DIVX4\_HD\_1080P
- #define VDP\_DECODER\_PROFILE\_DIVX5\_QMOBILE
- #define VDP\_DECODER\_PROFILE\_DIVX5\_MOBILE
- #define VDP\_DECODER\_PROFILE\_DIVX5\_HOME\_THEATER
- #define VDP\_DECODER\_PROFILE\_DIVX5\_HD\_1080P
- #define VDP\_DECODER\_PROFILE\_H264\_CONSTRAINED\_BASELINE
- #define VDP\_DECODER\_PROFILE\_H264\_EXTENDED
- #define VDP\_DECODER\_PROFILE\_H264\_PROGRESSIVE\_HIGH
- #define VDP\_DECODER\_PROFILE\_H264\_CONSTRAINED\_HIGH
- #define VDP\_DECODER\_PROFILE\_H264\_HIGH\_444\_PREDICTIVE

*Support for 8 bit depth only.*

- #define VDP\_DECODER\_PROFILE\_HEVC\_MAIN
  - MPEG-H Part 2 == H.265 == HEVC.*
- #define VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_10
- #define VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_STILL
- #define VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_12
- #define VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_444
- #define VDP\_DECODER\_LEVEL\_MPEG1\_NA
- #define VDP\_DECODER\_LEVEL\_MPEG2\_LL
- #define VDP\_DECODER\_LEVEL\_MPEG2\_ML
- #define VDP\_DECODER\_LEVEL\_MPEG2\_HL14
- #define VDP\_DECODER\_LEVEL\_MPEG2\_HL
- #define VDP\_DECODER\_LEVEL\_H264\_1
- #define VDP\_DECODER\_LEVEL\_H264\_1b
- #define VDP\_DECODER\_LEVEL\_H264\_1\_1
- #define VDP\_DECODER\_LEVEL\_H264\_1\_2
- #define VDP\_DECODER\_LEVEL\_H264\_1\_3
- #define VDP\_DECODER\_LEVEL\_H264\_2
- #define VDP\_DECODER\_LEVEL\_H264\_2\_1
- #define VDP\_DECODER\_LEVEL\_H264\_2\_2
- #define VDP\_DECODER\_LEVEL\_H264\_3
- #define VDP\_DECODER\_LEVEL\_H264\_3\_1
- #define VDP\_DECODER\_LEVEL\_H264\_3\_2
- #define VDP\_DECODER\_LEVEL\_H264\_4
- #define VDP\_DECODER\_LEVEL\_H264\_4\_1
- #define VDP\_DECODER\_LEVEL\_H264\_4\_2
- #define VDP\_DECODER\_LEVEL\_H264\_5
- #define VDP\_DECODER\_LEVEL\_H264\_5\_1
- #define VDP\_DECODER\_LEVEL\_VC1\_SIMPLE\_LOW
- #define VDP\_DECODER\_LEVEL\_VC1\_SIMPLE\_MEDIUM
- #define VDP\_DECODER\_LEVEL\_VC1\_MAIN\_LOW
- #define VDP\_DECODER\_LEVEL\_VC1\_MAIN\_MEDIUM
- #define VDP\_DECODER\_LEVEL\_VC1\_MAIN\_HIGH
- #define VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L0
- #define VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L1
- #define VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L2
- #define VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L3
- #define VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L4
- #define VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_SP\_L0
- #define VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_SP\_L1
- #define VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_SP\_L2
- #define VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_SP\_L3
- #define VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_ASP\_L0
- #define VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_ASP\_L1

- `#define VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L2`
- `#define VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L3`
- `#define VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L4`
- `#define VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L5`
- `#define VDP_DECODER_LEVEL_DIVX_NA`
- `#define VDP_DECODER_LEVEL_HEVC_1 30`
- `#define VDP_DECODER_LEVEL_HEVC_2`
- `#define VDP_DECODER_LEVEL_HEVC_2_1`
- `#define VDP_DECODER_LEVEL_HEVC_3`
- `#define VDP_DECODER_LEVEL_HEVC_3_1`
- `#define VDP_DECODER_LEVEL_HEVC_4`
- `#define VDP_DECODER_LEVEL_HEVC_4_1`
- `#define VDP_DECODER_LEVEL_HEVC_5`
- `#define VDP_DECODER_LEVEL_HEVC_5_1`
- `#define VDP_DECODER_LEVEL_HEVC_5_2`
- `#define VDP_DECODER_LEVEL_HEVC_6`
- `#define VDP_DECODER_LEVEL_HEVC_6_1`
- `#define VDP_DECODER_LEVEL_HEVC_6_2`
- `#define VDP_BITSTREAM_BUFFER_VERSION 0`

## Typedefs

- `typedef uint32_t VdpDecoderProfile`  
*The set of all known compressed video formats, and associated profiles, that may be decoded.*
- `typedef VdpStatus VdpDecoderQueryCapabilities(VdpDevice device, VdpDecoderProfile profile, VdpBool *is_supported, uint32_t *max_level, uint32_t *max_macroblocks, uint32_t *max_width, uint32_t *max_height)`  
*Query the implementation's VdpDecoder capabilities.*
- `typedef uint32_t VdpDecoder`  
*An opaque handle representing a VdpDecoder object.*
- `typedef VdpStatus VdpDecoderCreate(VdpDevice device, VdpDecoderProfile profile, uint32_t width, uint32_t height, uint32_t max_references, VdpDecoder *decoder)`  
*Create a VdpDecoder.*
- `typedef VdpStatus VdpDecoderDestroy(VdpDecoder decoder)`  
*Destroy a VdpDecoder.*
- `typedef VdpStatus VdpDecoderGetParameters(VdpDecoder decoder, VdpDecoderProfile *profile, uint32_t *width, uint32_t *height)`  
*Retrieve the parameters used to create a VdpDecoder.*
- `typedef void VdpPictureInfo`  
*A generic "picture information" type.*
- `typedef VdpPictureInfoMPEG4Part2 VdpPictureInfoDivX4`  
*Picture parameter information for a DivX 4 picture.*
- `typedef VdpPictureInfoMPEG4Part2 VdpPictureInfoDivX5`  
*Picture parameter information for a DivX 5 picture.*
- `typedef VdpStatus VdpDecoderRender(VdpDecoder decoder, VdpVideoSurface target, VdpPictureInfo const *picture_info, uint32_t bitstream_buffer_count, VdpBitstreamBuffer const *bitstream_buffers)`  
*Decode a compressed field/frame and render the result into a VdpVideoSurface.*



### 5.12.1 Detailed Description

The VdpDecoder object decodes compressed video data, writing the results to a [VdpVideoSurface](#).

A specific VDP AU implementation may support decoding multiple types of compressed video data. However, VdpDecoder objects are able to decode a specific type of compressed video data. This type must be specified during creation.

### 5.12.2 Macro Definition Documentation

5.12.2.1 `#define VDP_BITSTREAM_BUFFER_VERSION 0`

5.12.2.2 `#define VDP_DECODER_LEVEL_DIVX_NA`

5.12.2.3 `#define VDP_DECODER_LEVEL_H264_1`

5.12.2.4 `#define VDP_DECODER_LEVEL_H264_1_1`

5.12.2.5 `#define VDP_DECODER_LEVEL_H264_1_2`

5.12.2.6 `#define VDP_DECODER_LEVEL_H264_1_3`

5.12.2.7 `#define VDP_DECODER_LEVEL_H264_1b`

5.12.2.8 `#define VDP_DECODER_LEVEL_H264_2`

5.12.2.9 `#define VDP_DECODER_LEVEL_H264_2_1`

5.12.2.10 `#define VDP_DECODER_LEVEL_H264_2_2`

5.12.2.11 `#define VDP_DECODER_LEVEL_H264_3`

5.12.2.12 `#define VDP_DECODER_LEVEL_H264_3_1`

5.12.2.13 `#define VDP_DECODER_LEVEL_H264_3_2`

5.12.2.14 `#define VDP_DECODER_LEVEL_H264_4`

5.12.2.15 `#define VDP_DECODER_LEVEL_H264_4_1`

5.12.2.16 `#define VDP_DECODER_LEVEL_H264_4_2`

5.12.2.17 `#define VDP_DECODER_LEVEL_H264_5`

5.12.2.18 `#define VDP_DECODER_LEVEL_H264_5_1`

5.12.2.19 `#define VDP_DECODER_LEVEL_HEVC_1 30`

The VDP AU H.265/HEVC decoder levels correspond to the values of `general_level_idc` as described in the H.265 Specification, Annex A, Table A.1. The enumeration values are equal to thirty times the level number.

5.12.2.20 `#define VDP_DECODER_LEVEL_HEVC_2`

5.12.2.21 `#define VDP_DECODER_LEVEL_HEVC_2_1`

5.12.2.22 `#define VDP_DECODER_LEVEL_HEVC_3`

5.12.2.23 `#define VDP_DECODER_LEVEL_HEVC_3_1`

5.12.2.24 `#define VDP_DECODER_LEVEL_HEVC_4`

5.12.2.25 `#define VDP_DECODER_LEVEL_HEVC_4_1`

5.12.2.26 `#define VDP_DECODER_LEVEL_HEVC_5`

5.12.2.27 `#define VDP_DECODER_LEVEL_HEVC_5_1`

5.12.2.28 `#define VDP_DECODER_LEVEL_HEVC_5_2`

5.12.2.29 `#define VDP_DECODER_LEVEL_HEVC_6`

5.12.2.30 `#define VDP_DECODER_LEVEL_HEVC_6_1`

5.12.2.31 `#define VDP_DECODER_LEVEL_HEVC_6_2`

5.12.2.32 `#define VDP_DECODER_LEVEL_MPEG1_NA`

5.12.2.33 `#define VDP_DECODER_LEVEL_MPEG2_HL`

5.12.2.34 `#define VDP_DECODER_LEVEL_MPEG2_HL14`

5.12.2.35 `#define VDP_DECODER_LEVEL_MPEG2_LL`

5.12.2.36 `#define VDP_DECODER_LEVEL_MPEG2_ML`

5.12.2.37 `#define VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L0`

5.12.2.38 `#define VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L1`

5.12.2.39 `#define VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L2`

5.12.2.40 `#define VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L3`

5.12.2.41 `#define VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L4`

5.12.2.42 `#define VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L5`

5.12.2.43 `#define VDP_DECODER_LEVEL_MPEG4_PART2_SP_L0`

5.12.2.44 `#define VDP_DECODER_LEVEL_MPEG4_PART2_SP_L1`

5.12.2.45 `#define VDP_DECODER_LEVEL_MPEG4_PART2_SP_L2`

5.12.2.46 `#define VDP_DECODER_LEVEL_MPEG4_PART2_SP_L3`

5.12.2.47 `#define VDP_DECODER_LEVEL_VC1_ADVANCED_L0`

5.12.2.48 `#define VDP_DECODER_LEVEL_VC1_ADVANCED_L1`

5.12.2.49 `#define VDP_DECODER_LEVEL_VC1_ADVANCED_L2`

5.12.2.50 `#define VDP_DECODER_LEVEL_VC1_ADVANCED_L3`

5.12.2.51 `#define VDP_DECODER_LEVEL_VC1_ADVANCED_L4`

5.12.2.52 `#define VDP_DECODER_LEVEL_VC1_MAIN_HIGH`

5.12.2.53 `#define VDP_DECODER_LEVEL_VC1_MAIN_LOW`

5.12.2.54 `#define VDP_DECODER_LEVEL_VC1_MAIN_MEDIUM`

5.12.2.55 `#define VDP_DECODER_LEVEL_VC1_SIMPLE_LOW`

5.12.2.56 `#define VDP_DECODER_LEVEL_VC1_SIMPLE_MEDIUM`

5.12.2.57 `#define VDP_DECODER_PROFILE_DIVX4_HD_1080P`

5.12.2.58 `#define VDP_DECODER_PROFILE_DIVX4_HOME_THEATER`

5.12.2.59 `#define VDP_DECODER_PROFILE_DIVX4_MOBILE`

5.12.2.60 `#define VDP_DECODER_PROFILE_DIVX4_QMOBILE`

5.12.2.61 `#define VDP_DECODER_PROFILE_DIVX5_HD_1080P`

5.12.2.62 `#define VDP_DECODER_PROFILE_DIVX5_HOME_THEATER`

5.12.2.63 `#define VDP_DECODER_PROFILE_DIVX5_MOBILE`

5.12.2.64 `#define VDP_DECODER_PROFILE_DIVX5_QMOBILE`

5.12.2.65 `#define VDP_DECODER_PROFILE_H264_BASELINE`

MPEG 4 part 10 == H.264 == AVC.

5.12.2.66 `#define VDP_DECODER_PROFILE_H264_CONSTRAINED_BASELINE`

5.12.2.67 `#define VDP_DECODER_PROFILE_H264_CONSTRAINED_HIGH`

5.12.2.68 `#define VDP_DECODER_PROFILE_H264_EXTENDED`

5.12.2.69 `#define VDP_DECODER_PROFILE_H264_HIGH`

5.12.2.70 `#define VDP_DECODER_PROFILE_H264_HIGH_444_PREDICTIVE`

Support for 8 bit depth only.

5.12.2.71 `#define VDP_DECODER_PROFILE_H264_MAIN`

5.12.2.72 `#define VDP_DECODER_PROFILE_H264_PROGRESSIVE_HIGH`

5.12.2.73 `#define VDP_DECODER_PROFILE_HEVC_MAIN`

MPEG-H Part 2 == H.265 == HEVC.

5.12.2.74 `#define VDP_DECODER_PROFILE_HEVC_MAIN_10`

5.12.2.75 `#define VDP_DECODER_PROFILE_HEVC_MAIN_12`

5.12.2.76 `#define VDP_DECODER_PROFILE_HEVC_MAIN_444`

5.12.2.77 `#define VDP_DECODER_PROFILE_HEVC_MAIN_STILL`

5.12.2.78 `#define VDP_DECODER_PROFILE_MPEG1`

5.12.2.79 `#define VDP_DECODER_PROFILE_MPEG2_MAIN`

5.12.2.80 `#define VDP_DECODER_PROFILE_MPEG2_SIMPLE`

5.12.2.81 `#define VDP_DECODER_PROFILE_MPEG4_PART2_ASP`

5.12.2.82 `#define VDP_DECODER_PROFILE_MPEG4_PART2_SP`

5.12.2.83 `#define VDP_DECODER_PROFILE_VC1_ADVANCED`

5.12.2.84 `#define VDP_DECODER_PROFILE_VC1_MAIN`

5.12.2.85 `#define VDP_DECODER_PROFILE_VC1_SIMPLE`

### 5.12.3 Typedef Documentation

5.12.3.1 `typedef uint32_t VdpDecoder`

An opaque handle representing a VdpDecoder object.

5.12.3.2 `typedef VdpStatus VdpDecoderCreate(VdpDevice device, VdpDecoderProfile profile, uint32_t width, uint32_t height, uint32_t max_references, VdpDecoder *decoder)`

Create a VdpDecoder.

## Parameters

in	<i>device</i>	The device that will contain the surface.
in	<i>profile</i>	The video format the decoder will decode.
in	<i>width</i>	The width of the new surface.
in	<i>height</i>	The height of the new surface.
in	<i>max_references</i>	The maximum number of references that may be used by a single frame in the stream to be decoded. This parameter exists mainly for formats such as H.264, where different streams may use a different number of references. Requesting too many references may waste memory, but decoding should still operate correctly. Requesting too few references will cause decoding to fail.
out	<i>decoder</i>	The new decoder's handle.

## Returns

VdpStatus The completion status of the operation.

5.12.3.3 `typedef VdpStatus VdpDecoderDestroy(VdpDecoder decoder)`

Destroy a VdpDecoder.

## Parameters

in	<i>surface</i>	The decoder's handle.
----	----------------	-----------------------

## Returns

VdpStatus The completion status of the operation.

5.12.3.4 `typedef VdpStatus VdpDecoderGetParameters(VdpDecoder decoder, VdpDecoderProfile *profile, uint32_t *width, uint32_t *height)`

Retrieve the parameters used to create a VdpDecoder.

## Parameters

in	<i>surface</i>	The surface's handle.
out	<i>profile</i>	The video format used to create the decoder.
out	<i>width</i>	The width of surfaces decode by the decoder.
out	<i>height</i>	The height of surfaces decode by the decoder

## Returns

VdpStatus The completion status of the operation.

5.12.3.5 `typedef uint32_t VdpDecoderProfile`

The set of all known compressed video formats, and associated profiles, that may be decoded.

**5.12.3.6** `typedef VdpStatus VdpDecoderQueryCapabilities(VdpDevice device, VdpDecoderProfile profile, VdpBool *is_supported, uint32_t *max_level, uint32_t *max_macroblocks, uint32_t *max_width, uint32_t *max_height)`

Query the implementation's VdpDecoder capabilities.

#### Parameters

in	<i>device</i>	The device to query.
in	<i>profile</i>	The decoder profile for which information is requested.
out	<i>is_supported</i>	Is this profile supported?
out	<i>max_level</i>	The maximum specification level supported for this profile.
out	<i>max_macroblocks</i>	The maximum supported surface size in macroblocks. Note that this could be greater than that dictated by the maximum level.
out	<i>max_width</i>	The maximum supported surface width for this profile. Note that this could be greater than that dictated by the maximum level.
out	<i>max_height</i>	The maximum supported surface height for this profile. Note that this could be greater than that dictated by the maximum level.

#### Returns

VdpStatus The completion status of the operation.

**5.12.3.7** `typedef VdpStatus VdpDecoderRender(VdpDecoder decoder, VdpVideoSurface target, VdpPictureInfo const *picture_info, uint32_t bitstream_buffer_count, VdpBitstreamBuffer const *bitstream_buffers)`

Decode a compressed field/frame and render the result into a [VdpVideoSurface](#).

#### Parameters

in	<i>decoder</i>	The decoder object that will perform the decode operation.
in	<i>target</i>	The video surface to render to.
in	<i>picture_info</i>	A (pointer to a) structure containing information about the picture to be decoded. Note that the appropriate type of VdpPictureInfo* structure must be provided to match to profile that the decoder was created for.
in	<i>bitstream_buffer_count</i>	The number of bitstream buffers containing compressed data for this picture.
in	<i>bitstream_buffers</i>	An array of bitstream buffers.

#### Returns

VdpStatus The completion status of the operation.

See [Video Mixer Usage](#) for additional information.

**5.12.3.8** `typedef void VdpPictureInfo`

A generic "picture information" type.

This type serves solely to document the expected usage of a generic (void \*) function parameter. In actual usage, the application is expected to physically provide a pointer to an instance of one of the "real" VdpPictureInfo\* structures, picking the type appropriate for the decoder object in question.

#### 5.12.3.9 `typedef VdpPictureInfoMPEG4Part2 VdpPictureInfoDivX4`

Picture parameter information for a DivX 4 picture.

Due to similarites between MPEG-4 Part 2 and DivX 4, the picture parameter structure is re-used.

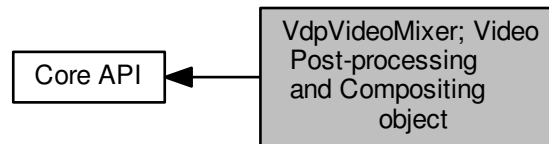
#### 5.12.3.10 `typedef VdpPictureInfoMPEG4Part2 VdpPictureInfoDivX5`

Picture parameter information for a DivX 5 picture.

Due to similarites between MPEG-4 Part 2 and DivX 5, the picture parameter structure is re-used.

## 5.13 VdpVideoMixer; Video Post-processing and Compositing object

Collaboration diagram for VdpVideoMixer; Video Post-processing and Compositing object:



### Data Structures

- struct [VdpLayer](#)  
Definition of an additional [VdpOutputSurface](#) layer in the compositing model.

### Macros

- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_DEINTERLACE\\_TEMPORAL](#)  
A *VdpVideoMixerFeature*.
- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_DEINTERLACE\\_TEMPORAL\\_SPATIAL](#)  
A *VdpVideoMixerFeature*.
- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_INVERSE\\_TELECINE](#)  
A *VdpVideoMixerFeature*.
- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_NOISE\\_REDUCTION](#)  
A *VdpVideoMixerFeature*.
- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_SHARPNESS](#)  
A *VdpVideoMixerFeature*.
- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_LUMA\\_KEY](#)  
A *VdpVideoMixerFeature*.
- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L1](#)  
A *VdpVideoMixerFeature*.
- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L2](#)  
A *VdpVideoMixerFeature*.
- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L3](#)  
A *VdpVideoMixerFeature*.
- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L4](#)  
A *VdpVideoMixerFeature*.
- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L5](#)  
A *VdpVideoMixerFeature*.
- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L6](#)  
A *VdpVideoMixerFeature*.
- #define [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L7](#)  
A *VdpVideoMixerFeature*.



- `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L8`  
*A VdpVideoMixerFeature.*
- `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L9`  
*A VdpVideoMixerFeature.*
- `#define VDP_VIDEO_MIXER_PARAMETER_VIDEO_SURFACE_WIDTH`  
*The exact width of input video surfaces.*
- `#define VDP_VIDEO_MIXER_PARAMETER_VIDEO_SURFACE_HEIGHT`  
*The exact height of input video surfaces.*
- `#define VDP_VIDEO_MIXER_PARAMETER_CHROMA_TYPE`  
*The chroma type of the input video surfaces the will process.*
- `#define VDP_VIDEO_MIXER_PARAMETER_LAYERS`  
*The number of auxiliary layers in the mixer's compositing model.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_BACKGROUND_COLOR`  
*The background color in the VdpVideoMixer's compositing model.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_CSC_MATRIX`  
*The color-space conversion matrix used by the VdpVideoMixer.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_NOISE_REDUCTION_LEVEL`  
*The amount of noise reduction algorithm to apply.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_SHARPNESS_LEVEL`  
*The amount of sharpening, or blurring, to apply.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_LUMA_KEY_MIN_LUMA`  
*The minimum luma value for the luma key algorithm.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_LUMA_KEY_MAX_LUMA`  
*The maximum luma value for the luma key algorithm.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_SKIP_CHROMA_DEINTERLACE`  
*Whether de-interlacers should operate solely on luma, and bob chroma.*
- `#define VDP_LAYER_VERSION 0`

## Typedefs

- `typedef uint32_t VdpVideoMixerFeature`  
*A VdpVideoMixer feature that must be requested at creation time to be used.*
- `typedef uint32_t VdpVideoMixerParameter`  
*A VdpVideoMixer creation parameter.*
- `typedef uint32_t VdpVideoMixerAttribute`  
*An adjustable attribute of VdpVideoMixer operation.*
- `typedef VdpStatus VdpVideoMixerQueryFeatureSupport(VdpDevice device, VdpVideoMixerFeature feature, VdpBool *is_supported)`  
*Query the implementation's support for a specific feature.*
- `typedef VdpStatus VdpVideoMixerQueryParameterSupport(VdpDevice device, VdpVideoMixerParameter parameter, VdpBool *is_supported)`  
*Query the implementation's support for a specific parameter.*
- `typedef VdpStatus VdpVideoMixerQueryAttributeSupport(VdpDevice device, VdpVideoMixerAttribute attribute, VdpBool *is_supported)`  
*Query the implementation's support for a specific attribute.*
- `typedef VdpStatus VdpVideoMixerQueryParameterValueRange(VdpDevice device, VdpVideoMixerParameter parameter, void *min_value, void *max_value)`  
*Query the implementation's supported for a specific parameter.*
- `typedef VdpStatus VdpVideoMixerQueryAttributeValueRange(VdpDevice device, VdpVideoMixerAttribute attribute, void *min_value, void *max_value)`  
*Query the implementation's supported for a specific attribute.*

- typedef uint32\_t [VdpVideoMixer](#)  
*An opaque handle representing a [VdpVideoMixer](#) object.*
- typedef [VdpStatus](#) [VdpVideoMixerCreate](#)([VdpDevice](#) device, uint32\_t feature\_count, [VdpVideoMixerFeature](#) const \*features, uint32\_t parameter\_count, [VdpVideoMixerParameter](#) const \*parameters, void const \*const \*parameter\_values, [VdpVideoMixer](#) \*mixer)  
*Create a [VdpVideoMixer](#).*
- typedef [VdpStatus](#) [VdpVideoMixerSetFeatureEnables](#)([VdpVideoMixer](#) mixer, uint32\_t feature\_count, [VdpVideoMixerFeature](#) const \*features, [VdpBool](#) const \*feature\_enables)  
*Enable or disable features.*
- typedef [VdpStatus](#) [VdpVideoMixerSetAttributeValues](#)([VdpVideoMixer](#) mixer, uint32\_t attribute\_count, [VdpVideoMixerAttribute](#) const \*attributes, void const \*const \*attribute\_values)  
*Set attribute values.*
- typedef [VdpStatus](#) [VdpVideoMixerGetFeatureSupport](#)([VdpVideoMixer](#) mixer, uint32\_t feature\_count, [VdpVideoMixerFeature](#) const \*features, [VdpBool](#) \*feature\_supports)  
*Retrieve whether features were requested at creation time.*
- typedef [VdpStatus](#) [VdpVideoMixerGetFeatureEnables](#)([VdpVideoMixer](#) mixer, uint32\_t feature\_count, [VdpVideoMixerFeature](#) const \*features, [VdpBool](#) \*feature\_enables)  
*Retrieve whether features are enabled.*
- typedef [VdpStatus](#) [VdpVideoMixerGetParameterValues](#)([VdpVideoMixer](#) mixer, uint32\_t parameter\_count, [VdpVideoMixerParameter](#) const \*parameters, void \*const \*parameter\_values)  
*Retrieve parameter values given at creation time.*
- typedef [VdpStatus](#) [VdpVideoMixerGetAttributeValues](#)([VdpVideoMixer](#) mixer, uint32\_t attribute\_count, [VdpVideoMixerAttribute](#) const \*attributes, void \*const \*attribute\_values)  
*Retrieve current attribute values.*
- typedef [VdpStatus](#) [VdpVideoMixerDestroy](#)([VdpVideoMixer](#) mixer)  
*Destroy a [VdpVideoMixer](#).*
- typedef [VdpStatus](#) [VdpVideoMixerRender](#)([VdpVideoMixer](#) mixer, [VdpOutputSurface](#) background\_surface, [VdpRect](#) const \*background\_source\_rect, [VdpVideoMixerPictureStructure](#) current\_picture\_structure, uint32\_t video\_surface\_past\_count, [VdpVideoSurface](#) const \*video\_surface\_past, [VdpVideoSurface](#) video\_surface\_current, uint32\_t video\_surface\_future\_count, [VdpVideoSurface](#) const \*video\_surface\_future, [VdpRect](#) const \*video\_source\_rect, [VdpOutputSurface](#) destination\_surface, [VdpRect](#) const \*destination\_rect, [VdpRect](#) const \*destination\_video\_rect, uint32\_t layer\_count, [VdpLayer](#) const \*layers)  
*Perform a video post-processing and compositing operation.*

## Enumerations

- enum [VdpVideoMixerPictureStructure](#) { [VDP\\_VIDEO\\_MIXER\\_PICTURE\\_STRUCTURE\\_TOP\\_FIELD](#), [VDP\\_VIDEO\\_MIXER\\_PICTURE\\_STRUCTURE\\_BOTTOM\\_FIELD](#), [VDP\\_VIDEO\\_MIXER\\_PICTURE\\_STRUCTURE\\_FRAME](#) }  
*The structure of the picture present in a [VdpVideoSurface](#).*

### 5.13.1 Detailed Description

[VdpVideoMixer](#) can perform some subset of the following post-processing steps on video:

- De-interlacing
  - Various types, with or without inverse telecine
- Noise-reduction
- Sharpness adjustment

- Color space conversion to RGB
- Chroma format upscaling to 4:4:4

A VdpVideoMixer takes a source [VdpVideoSurface](#) VdpVideoSurface and performs various video processing steps on it (potentially using information from past or future video surfaces). It scales the video and converts it to RGB, then optionally composites it with multiple auxiliary [VdpOutputSurfaces](#) before writing the result to the destination [VdpOutputSurface](#).

The video mixer compositing model is as follows:

- A rectangle will be rendered on an output surface. No pixels will be rendered outside of this output rectangle. The contents of this rectangle will be a composite of many layers.
- The first layer is the background color. The background color will fill the entire rectangle.
- The second layer is the processed video which has been converted to RGB. These pixels will overwrite the background color of the first layer except where the second layer's rectangle does not completely cover the output rectangle. In those regions the background color will continue to show. If any portion of the second layer's output rectangle is outside of the output rectangle, those portions will be clipped.
- The third layer contains some number of auxiliary layers (in the form of [VdpOutputSurfaces](#)) which will be composited using the alpha value from the those surfaces. The compositing operations are equivalent to rendering with [VdpOutputSurfaceRenderOutputSurface](#) using a source blend factor of SOURCE\_ALPHA, a destination blend factor of ONE\_MINUS\_SOURCE\_ALPHA and an equation of ADD.

### 5.13.2 Macro Definition Documentation

#### 5.13.2.1 #define VDP\_LAYER\_VERSION 0

#### 5.13.2.2 #define VDP\_VIDEO\_MIXER\_ATTRIBUTE\_BACKGROUND\_COLOR

The background color in the VdpVideoMixer's compositing model.

This attribute's type is [VdpColor](#).

This parameter defaults to black (all color components 0.0 and alpha 1.0).

The application may not query this parameter's supported range, since the type is not scalar.

#### 5.13.2.3 #define VDP\_VIDEO\_MIXER\_ATTRIBUTE\_CSC\_MATRIX

The color-space conversion matrix used by the VdpVideoMixer.

This attribute's type is [VdpCSCMatrix](#); [CSC Matrix Manipulation](#).

Note: When using [VdpVideoMixerGetAttributeValues](#) to retrieve the current CSC matrix, the attribute\_values array must contain a pointer to a pointer a VdpCSCMatrix (VdpCSCMatrix\*\* as a void \*). The get function will either initialize the referenced CSC matrix to the current value, or clear the supplied pointer to NULL, if the previous set call supplied a value of NULL in parameter\_values, to request the default matrix.

```
1 VdpCSCMatrix  matrix;
2 VdpCSCMatrix * matrix_ptr;
3 void * attribute_values[] = {&matrix_ptr};
4 VdpStatus st = vdp_video_mixer_get_attribute_values(..., attribute_values, ...);
```

This parameter defaults to a matrix suitable for ITU-R BT.601 input surfaces, with no procamp adjustments.

The application may not query this parameter's supported range, since the type is not scalar.

#### 5.13.2.4 `#define VDP_VIDEO_MIXER_ATTRIBUTE_LUMA_KEY_MAX_LUMA`

The maximum luma value for the luma key algorithm.

This attribute's type is float.

This parameter defaults to 1.0.

The application may query this parameter's supported range. However, the range is fixed as 0.0...1.0.

#### 5.13.2.5 `#define VDP_VIDEO_MIXER_ATTRIBUTE_LUMA_KEY_MIN_LUMA`

The minimum luma value for the luma key algorithm.

This attribute's type is float.

This parameter defaults to 0.0.

The application may query this parameter's supported range. However, the range is fixed as 0.0...1.0.

#### 5.13.2.6 `#define VDP_VIDEO_MIXER_ATTRIBUTE_NOISE_REDUCTION_LEVEL`

The amount of noise reduction algorithm to apply.

This attribute's type is float.

This parameter defaults to 0.0, which equates to no noise reduction.

The application may query this parameter's supported range. However, the range is fixed as 0.0...1.0.

#### 5.13.2.7 `#define VDP_VIDEO_MIXER_ATTRIBUTE_SHARPNESS_LEVEL`

The amount of sharpening, or blurring, to apply.

This attribute's type is float.

This parameter defaults to 0.0, which equates to no sharpening.

Positive values request sharpening. Negative values request blurring.

The application may query this parameter's supported range. However, the range is fixed as -1.0...1.0.

#### 5.13.2.8 `#define VDP_VIDEO_MIXER_ATTRIBUTE_SKIP_CHROMA_DEINTERLACE`

Whether de-interlacers should operate solely on luma, and bob chroma.

Note: This attribute only affects advanced de-interlacing algorithms, not bob or weave.

This attribute's type is `uint8_t`.

This parameter defaults to 0.

The application may query this parameter's supported range. However, the range is fixed as 0 (no/off) ... 1 (yes/on).

#### 5.13.2.9 `#define VDP_VIDEO_MIXER_FEATURE_DEINTERLACE_TEMPORAL`

A VdpVideoMixerFeature.

When requested and enabled, motion adaptive temporal deinterlacing will be used on interlaced content.

When multiple de-interlacing options are requested and enabled, the back-end implementation chooses the best algorithm to apply.

#### 5.13.2.10 `#define VDP_VIDEO_MIXER_FEATURE_DEINTERLACE_TEMPORAL_SPATIAL`

A VdpVideoMixerFeature.

When requested and enabled, this enables a more advanced version of temporal de-interlacing, that additionally uses edge-guided spatial interpolation.

When multiple de-interlacing options are requested and enabled, the back-end implementation chooses the best algorithm to apply.

#### 5.13.2.11 `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L1`

A VdpVideoMixerFeature.

A VDPAAU implementation may support multiple scaling algorithms of differing quality, and may potentially support a different subset of algorithms on different hardware.

In some cases, higher quality algorithms may require more resources (memory size, memory bandwidth, etc.) to operate. Hence, these high quality algorithms must be explicitly requested and enabled by the client application. This allows applications operating in a resource-constrained environment to have some level of control over resource usage.

Basic scaling is always built into any video mixer, and is known as level 0. Scaling quality increases beginning with optional level 1, through optional level 9.

If an application requests and enables multiple high quality scaling algorithms, the highest level enabled scaling algorithm will be used.

#### 5.13.2.12 `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L2`

A VdpVideoMixerFeature.

See [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L1](#) for details.

#### 5.13.2.13 `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L3`

A VdpVideoMixerFeature.

See [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L1](#) for details.

#### 5.13.2.14 `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L4`

A `VdpVideoMixerFeature`.

See [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L1](#) for details.

#### 5.13.2.15 `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L5`

A `VdpVideoMixerFeature`.

See [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L1](#) for details.

#### 5.13.2.16 `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L6`

A `VdpVideoMixerFeature`.

See [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L1](#) for details.

#### 5.13.2.17 `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L7`

A `VdpVideoMixerFeature`.

See [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L1](#) for details.

#### 5.13.2.18 `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L8`

A `VdpVideoMixerFeature`.

See [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L1](#) for details.

#### 5.13.2.19 `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L9`

A `VdpVideoMixerFeature`.

See [VDP\\_VIDEO\\_MIXER\\_FEATURE\\_HIGH\\_QUALITY\\_SCALING\\_L1](#) for details.

#### 5.13.2.20 `#define VDP_VIDEO_MIXER_FEATURE_INVERSE_TELECINE`

A `VdpVideoMixerFeature`.

When requested and enabled, cadence detection will be enabled on interlaced content and the video mixer will try to extract progressive frames from pull-down material.

#### 5.13.2.21 `#define VDP_VIDEO_MIXER_FEATURE_LUMA_KEY`

A VdpVideoMixerFeature.

When requested and enabled, the alpha of the rendered surface, which is normally set to the alpha of the background color, will be forced to 0.0 on pixels corresponding to source video surface luminance values in the range specified by attributes `VDP_VIDEO_MIXER_ATTRIBUTE_LUMA_KEY_MIN_LUMA` to `VDP_VIDEO_MIXER_ATTRIBUTE_LUMA_KEY_MAX_LUMA`. This keying is performed after scaling and de-interlacing.

#### 5.13.2.22 `#define VDP_VIDEO_MIXER_FEATURE_NOISE_REDUCTION`

A VdpVideoMixerFeature.

When requested and enabled, a noise reduction algorithm will be applied to the video.

#### 5.13.2.23 `#define VDP_VIDEO_MIXER_FEATURE_SHARPNESS`

A VdpVideoMixerFeature.

When requested and enabled, a sharpening algorithm will be applied to the video.

#### 5.13.2.24 `#define VDP_VIDEO_MIXER_PARAMETER_CHROMA_TYPE`

The chroma type of the input video surfaces the will process.

This parameter's type is VdpChromaType.

If not specified, this parameter defaults to `VDP_CHROMA_TYPE_420`.

The application may not query this application's supported range, since it is a potentially disjoint enumeration.

#### 5.13.2.25 `#define VDP_VIDEO_MIXER_PARAMETER_LAYERS`

The number of auxiliary layers in the mixer's compositing model.

Note that this indicates the maximum number of layers that may be processed by a given `VdpVideoMixer` object. Each individual `VdpVideoMixerRender` invocation may choose to use a different number of actual layers, from 0 up to this limit.

This attribute's type is `uint32_t`.

If not specified, this parameter defaults to 0.

The application may query this parameter's supported range.

#### 5.13.2.26 `#define VDP_VIDEO_MIXER_PARAMETER_VIDEO_SURFACE_HEIGHT`

The exact height of input video surfaces.

This parameter's type is `uint32_t`.

This parameter defaults to 0 if not specified, which entails that it must be specified.

The application may query this parameter's supported range.

#### 5.13.2.27 `#define VDP_VIDEO_MIXER_PARAMETER_VIDEO_SURFACE_WIDTH`

The exact width of input video surfaces.

This parameter's type is `uint32_t`.

This parameter defaults to 0 if not specified, which entails that it must be specified.

The application may query this parameter's supported range.

### 5.13.3 Typedef Documentation

#### 5.13.3.1 `typedef uint32_t VdpVideoMixer`

An opaque handle representing a `VdpVideoMixer` object.

#### 5.13.3.2 `typedef uint32_t VdpVideoMixerAttribute`

An adjustable attribute of `VdpVideoMixer` operation.

Various attributes of `VdpVideoMixer` operation may be adjusted at any time. Each attribute is named via a specific `VdpVideoMixerAttribute` value.

Each attribute has a specific type, and specific default value if not specified at `VdpVideoMixer` creation time. The application may query the legal supported range for some attributes.

#### 5.13.3.3 `typedef VdpStatus VdpVideoMixerCreate(VdpDevice device, uint32_t feature_count, VdpVideoMixerFeature const *features, uint32_t parameter_count, VdpVideoMixerParameter const *parameters, void const *const *parameter_values, VdpVideoMixer *mixer)`

Create a `VdpVideoMixer`.

##### Parameters

in	<i>device</i>	The device that will contain the mixer.
in	<i>feature_count</i>	The number of features to request.
in	<i>features</i>	The list of features to request.
in	<i>parameter_count</i>	The number of parameters to set.
in	<i>parameters</i>	The list of parameters to set.
in	<i>parameter_values</i>	The values for the parameters. Note that each entry in the value array is a pointer to the actual value. In other words, the values themselves are not cast to "void *" and passed "inside" the array.
out	<i>mixer</i>	The new mixer's handle.



**Returns**

VdpStatus The completion status of the operation.

Initially, all requested features will be disabled. They can be enabled using [VdpVideoMixerSetFeatureEnables](#).

Initially, all attributes will have default values. Values can be changed using [VdpVideoMixerSetAttributeValues](#).

**5.13.3.4 typedef VdpStatus VdpVideoMixerDestroy(VdpVideoMixer mixer)**

Destroy a VdpVideoMixer.

**Parameters**

in	<i>device</i>	The device to destroy.
----	---------------	------------------------

**Returns**

VdpStatus The completion status of the operation.

**5.13.3.5 typedef uint32\_t VdpVideoMixerFeature**

A VdpVideoMixer feature that must be requested at creation time to be used.

Certain advanced VdpVideoMixer features are optional, and the ability to use those features at all must be requested when the VdpVideoMixer object is created. Each feature is named via a specific VdpVideoMixerFeature value.

Once requested, these features are permanently available within that specific VdpVideoMixer object. All features that are not explicitly requested at creation time default to being permanently unavailable.

Even when requested, all features default to being initially disabled. However, applications can subsequently enable and disable features at any time. See [VdpVideoMixerSetFeatureEnables](#).

Some features allow configuration of their operation. Each configurable item is an [VdpVideoMixerAttribute](#). These attributes may be manipulated at any time using [VdpVideoMixerSetAttributeValues](#).

**5.13.3.6 typedef VdpStatus VdpVideoMixerGetAttributeValues(VdpVideoMixer mixer, uint32\_t attribute\_count, VdpVideoMixerAttribute const \*attributes, void \*const \*attribute\_values)**

Retrieve current attribute values.

**Parameters**

in	<i>mixer</i>	The mixer to manipulate.
in	<i>attribute_count</i>	The number of attributes to query.
in	<i>attributes</i>	The list of attributes to query.
out	<i>attribute_values</i>	The list of current values for the attributes. Note that each entry in the value array is a pointer to storage that will receive the actual value. If the attribute's type is a pointer itself, please closely read the documentation for that attribute type for any other data passing requirements.

**Returns**

VdpStatus The completion status of the operation.

**5.13.3.7** `typedef VdpStatus VdpVideoMixerGetFeatureEnables(VdpVideoMixer mixer, uint32_t feature_count, VdpVideoMixerFeature const *features, VdpBool *feature_enables)`

Retrieve whether features are enabled.

**Parameters**

in	<i>mixer</i>	The mixer to manipulate.
in	<i>feature_count</i>	The number of features to query.
in	<i>features</i>	The list of features to query.
out	<i>feature_enabled</i>	A list of values indicating whether the feature is enabled.

**Returns**

VdpStatus The completion status of the operation.

**5.13.3.8** `typedef VdpStatus VdpVideoMixerGetFeatureSupport(VdpVideoMixer mixer, uint32_t feature_count, VdpVideoMixerFeature const *features, VdpBool *feature_supports)`

Retrieve whether features were requested at creation time.

**Parameters**

in	<i>mixer</i>	The mixer to query.
in	<i>feature_count</i>	The number of features to query.
in	<i>features</i>	The list of features to query.
out	<i>feature_supported</i>	A list of values indicating whether the feature was requested, and hence is available.

**Returns**

VdpStatus The completion status of the operation.

**5.13.3.9** `typedef VdpStatus VdpVideoMixerGetParameterValues(VdpVideoMixer mixer, uint32_t parameter_count, VdpVideoMixerParameter const *parameters, void *const *parameter_values)`

Retrieve parameter values given at creation time.

**Parameters**

in	<i>mixer</i>	The mixer to manipulate.
in	<i>parameter_count</i>	The number of parameters to query.
in	<i>parameters</i>	The list of parameters to query.
out	<i>parameter_values</i>	The list of current values for the parameters. Note that each entry in the value array is a pointer to storage that will receive the actual value. If the attribute's type is a pointer itself, please closely read the documentation for that attribute type for any other data passing requirements.
		Generated by Doxygen

**Returns**

VdpStatus The completion status of the operation.

**5.13.3.10 typedef uint32\_t VdpVideoMixerParameter**

A VdpVideoMixer creation parameter.

When a VdpVideoMixer is created, certain parameters may be supplied. Each parameter is named via a specific VdpVideoMixerParameter value.

Each parameter has a specific type, and specific default value if not specified at VdpVideoMixer creation time. The application may query the legal supported range for some parameters.

**5.13.3.11 typedef VdpStatus VdpVideoMixerQueryAttributeSupport(VdpDevice device, VdpVideoMixerAttribute attribute, VdpBool \*is\_supported)**

Query the implementation's support for a specific attribute.

**Parameters**

in	<i>device</i>	The device to query.
in	<i>feature</i>	The feature for which support is to be queried.
out	<i>is_supported</i>	Is the specified feature supported?

**Returns**

VdpStatus The completion status of the operation.

**5.13.3.12 typedef VdpStatus VdpVideoMixerQueryAttributeValueRange(VdpDevice device, VdpVideoMixerAttribute attribute, void \*min\_value, void \*max\_value)**

Query the implementation's supported for a specific attribute.

**Parameters**

in	<i>device</i>	The device to query.
in	<i>attribute</i>	The attribute for which support is to be queried.
out	<i>min_value</i>	The minimum supported value.
out	<i>max_value</i>	The maximum supported value.

**Returns**

VdpStatus The completion status of the operation.

**5.13.3.13** `typedef VdpStatus VdpVideoMixerQueryFeatureSupport(VdpDevice device, VdpVideoMixerFeature feature, VdpBool *is_supported)`

Query the implementation's support for a specific feature.

#### Parameters

in	<i>device</i>	The device to query.
in	<i>feature</i>	The feature for which support is to be queried.
out	<i>is_supported</i>	Is the specified feature supported?

#### Returns

VdpStatus The completion status of the operation.

**5.13.3.14** `typedef VdpStatus VdpVideoMixerQueryParameterSupport(VdpDevice device, VdpVideoMixerParameter parameter, VdpBool *is_supported)`

Query the implementation's support for a specific parameter.

#### Parameters

in	<i>device</i>	The device to query.
in	<i>parameter</i>	The parameter for which support is to be queried.
out	<i>is_supported</i>	Is the specified parameter supported?

#### Returns

VdpStatus The completion status of the operation.

**5.13.3.15** `typedef VdpStatus VdpVideoMixerQueryParameterValueRange(VdpDevice device, VdpVideoMixerParameter parameter, void *min_value, void *max_value)`

Query the implementation's supported for a specific parameter.

#### Parameters

in	<i>device</i>	The device to query.
in	<i>parameter</i>	The parameter for which support is to be queried.
out	<i>min_value</i>	The minimum supported value.
out	<i>max_value</i>	The maximum supported value.

#### Returns

VdpStatus The completion status of the operation.

```
5.13.3.16 typedef VdpStatus VdpVideoMixerRender(VdpVideoMixer mixer, VdpOutputSurface background_surface,
VdpRect const *background_source_rect, VdpVideoMixerPictureStructure current_picture_structure,
uint32_t video_surface_past_count, VdpVideoSurface const *video_surface_past, VdpVideoSurface
video_surface_current, uint32_t video_surface_future_count, VdpVideoSurface const *video_surface_future,
VdpRect const *video_source_rect, VdpOutputSurface destination_surface, VdpRect const *destination_rect,
VdpRect const *destination_video_rect, uint32_t layer_count, VdpLayer const *layers)
```

Perform a video post-processing and compositing operation.

#### Parameters

in	<i>mixer</i>	The mixer object that will perform the mixing/rendering operation.
in	<i>background_surface</i>	A background image. If set to any value other than <code>VDP_INVALID_HANDLE</code> , the specific surface will be used instead of the background color as the first layer in the mixer's compositing process.
in	<i>background_source_rect</i>	When <i>background_surface</i> is specified, this parameter indicates the portion of <i>background_surface</i> that will be used as the background layer. The specified region will be extracted and scaled to match the size of <i>destination_rect</i> . If <code>NULL</code> , the entire <i>background_surface</i> will be used.
in	<i>current_picture_structure</i>	The picture structure of the field/frame to be processed. This field/frame is presented in the <b>video_surface_current</b> parameter. If frame, then all <b>video_surface_*</b> parameters are assumed to be frames. If field, then all <i>video_surface_*</i> parameters are assumed to be fields, with alternating top/bottom-ness derived from <i>video_surface_current</i> .
in	<i>video_surfaces_past_count</i>	The number of provided fields/frames prior to the current picture.
in	<i>video_surfaces_past</i>	The fields/frames prior to the current field/frame. Note that array index 0 is the field/frame temporally nearest to the current field/frame, with increasing array indices used for older frames. Unavailable entries may be set to <code>VDP_INVALID_HANDLE</code> .
in	<i>video_surface_current</i>	The field/frame to be processed.
in	<i>video_surfaces_future_count</i>	The number of provided fields/frames following the current picture.
in	<i>video_surfaces_future</i>	The fields/frames that follow the current field/frame. Note that array index 0 is the field/frame temporally nearest to the current field/frame, with increasing array indices used for newer frames. Unavailable entries may be set to <code>VDP_INVALID_HANDLE</code> .
in	<i>video_source_rect</i>	The sub-rectangle of the source video surface to extract and process. If <code>NULL</code> , the entire surface will be used. Left/right and/or top/bottom co-ordinates may be swapped to flip the source. Values from outside the video surface are valid and samples at those locations will be taken from the nearest edge.
in	<i>destination_surface</i>	
in	<i>destination_rect</i>	The sub-rectangle of the destination surface to modify. Note that rectangle clips all other actions.
in	<i>destination_video_rect</i>	The sub-rectangle of the destination surface that will contain the processed video. This rectangle is relative to the entire destination surface. This rectangle is clipped by <b>destination_rect</b> . If <code>NULL</code> , the destination rectangle will be sized to match the source rectangle, and will be located at the origin.
in	<i>layer_count</i>	The number of additional layers to composite above the video.
in	<i>layers</i>	The array of additional layers to composite above the video.

#### Returns

VdpStatus The completion status of the operation.

For a complete discussion of how to use this API, please see [Video Mixer Usage](#).

**5.13.3.17** `typedef VdpStatus VdpVideoMixerSetAttributeValues(VdpVideoMixer mixer, uint32_t attribute_count, VdpVideoMixerAttribute const *attributes, void const *const *attribute_values)`

Set attribute values.

#### Parameters

in	<i>mixer</i>	The mixer to manipulate.
in	<i>attribute_count</i>	The number of attributes to set.
in	<i>attributes</i>	The list of attributes to set.
in	<i>attribute_values</i>	The values for the attributes. Note that each entry in the value array is a pointer to the actual value. In other words, the values themselves are not cast to "void *" and passed "inside" the array. A NULL pointer requests that the default value be set for that attribute.

#### Returns

VdpStatus The completion status of the operation.

**5.13.3.18** `typedef VdpStatus VdpVideoMixerSetFeatureEnables(VdpVideoMixer mixer, uint32_t feature_count, VdpVideoMixerFeature const *features, VdpBool const *feature_enables)`

Enable or disable features.

#### Parameters

in	<i>mixer</i>	The mixer to manipulate.
in	<i>feature_count</i>	The number of features to enable/disable.
in	<i>features</i>	The list of features to enable/disable.
in	<i>feature_enables</i>	The list of new feature enable values.

#### Returns

VdpStatus The completion status of the operation.

## 5.13.4 Enumeration Type Documentation

**5.13.4.1** `enum VdpVideoMixerPictureStructure`

The structure of the picture present in a [VdpVideoSurface](#).

#### Enumerator

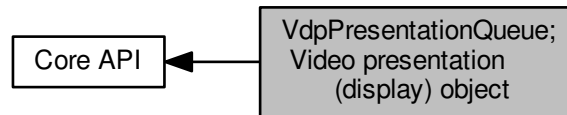
**VDP\_VIDEO\_MIXER\_PICTURE\_STRUCTURE\_TOP\_FIELD** The picture is a field, and is the top field of the surface.

**VDP\_VIDEO\_MIXER\_PICTURE\_STRUCTURE\_BOTTOM\_FIELD** The picture is a field, and is the bottom field of the surface.

**VDP\_VIDEO\_MIXER\_PICTURE\_STRUCTURE\_FRAME** The picture is a frame, and hence is the entire surface.

## 5.14 VdpPresentationQueue; Video presentation (display) object

Collaboration diagram for VdpPresentationQueue; Video presentation (display) object:



### Typedefs

- typedef uint64\_t [VdpTime](#)  
The representation of a point in time.
- typedef uint32\_t [VdpPresentationQueueTarget](#)  
An opaque handle representing the location where video will be presented.
- typedef [VdpStatus](#) [VdpPresentationQueueTargetDestroy](#)([VdpPresentationQueueTarget](#) presentation\_queue\_target)  
Destroy a [VdpPresentationQueueTarget](#).
- typedef uint32\_t [VdpPresentationQueue](#)  
An opaque handle representing a presentation queue object.
- typedef [VdpStatus](#) [VdpPresentationQueueCreate](#)([VdpDevice](#) device, [VdpPresentationQueueTarget](#) presentation\_queue\_target, [VdpPresentationQueue](#) \*presentation\_queue)  
Create a [VdpPresentationQueue](#).
- typedef [VdpStatus](#) [VdpPresentationQueueDestroy](#)([VdpPresentationQueue](#) presentation\_queue)  
Destroy a [VdpPresentationQueue](#).
- typedef [VdpStatus](#) [VdpPresentationQueueSetBackgroundColor](#)([VdpPresentationQueue](#) presentation\_queue, [VdpColor](#) \*const background\_color)  
Configure the background color setting.
- typedef [VdpStatus](#) [VdpPresentationQueueGetBackgroundColor](#)([VdpPresentationQueue](#) presentation\_queue, [VdpColor](#) \*background\_color)  
Retrieve the current background color setting.
- typedef [VdpStatus](#) [VdpPresentationQueueGetTime](#)([VdpPresentationQueue](#) presentation\_queue, [VdpTime](#) \*current\_time)  
Retrieve the presentation queue's "current" time.
- typedef [VdpStatus](#) [VdpPresentationQueueDisplay](#)([VdpPresentationQueue](#) presentation\_queue, [VdpOutputSurface](#) surface, uint32\_t clip\_width, uint32\_t clip\_height, [VdpTime](#) earliest\_presentation\_time)  
Enter a surface into the presentation queue.
- typedef [VdpStatus](#) [VdpPresentationQueueBlockUntilSurfaceIdle](#)([VdpPresentationQueue](#) presentation\_queue, [VdpOutputSurface](#) surface, [VdpTime](#) \*first\_presentation\_time)  
Wait for a surface to finish being displayed.
- typedef [VdpStatus](#) [VdpPresentationQueueQuerySurfaceStatus](#)([VdpPresentationQueue](#) presentation\_queue, [VdpOutputSurface](#) surface, [VdpPresentationQueueStatus](#) \*status, [VdpTime](#) \*first\_presentation\_time)  
Poll the current queue status of a surface.

## Enumerations

- enum [VdpPresentationQueueStatus](#) { [VDP\\_PRESENTATION\\_QUEUE\\_STATUS\\_IDLE](#), [VDP\\_PRESENTATION\\_QUEUE\\_STATUS\\_QUEUED](#), [VDP\\_PRESENTATION\\_QUEUE\\_STATUS\\_VISIBLE](#) }

*The status of a surface within a presentation queue.*

### 5.14.1 Detailed Description

The `VdpPresentationQueue` manages a queue of surfaces and associated timestamps. For each surface in the queue, once the associated timestamp is reached, the surface is displayed to the user. This timestamp-based approach yields high quality video delivery.

The exact location of the displayed content is Window System specific. For this reason, the [Window System Integration Layer](#) provides an API to create a [VdpPresentationQueueTarget](#) object (e.g. via [VdpPresentationQueueTargetCreateX11](#)) which encapsulates this information.

Note that the presentation queue performs no scaling of surfaces to match the display target's size, aspect ratio, etc.

Surfaces that are too large to fit into the display target will be clipped. Surfaces that are too small to fill the display target will be aligned to the top-left corner of the display target, with the balance of the display target being filled with a constant configurable "background" color.

Note that the presentation queue operates in a manner that is semantically equivalent to an overlay surface, with any required color key painting hidden internally. However, implementations are free to use whatever semantically equivalent technique they wish. Note that implementations that actually use color-keyed overlays will typically use the "background" color as the overlay color key value, so this color should be chosen with care.

### 5.14.2 Typedef Documentation

#### 5.14.2.1 typedef uint32\_t VdpPresentationQueue

An opaque handle representing a presentation queue object.

#### 5.14.2.2 typedef VdpStatus VdpPresentationQueueBlockUntilSurfaceIdle(VdpPresentationQueue presentation\_queue, VdpOutputSurface surface, VdpTime \*first\_presentation\_time)

Wait for a surface to finish being displayed.

##### Parameters

in	<i>presentation_queue</i>	The queue to query.
in	<i>surface</i>	The surface to wait for.
out	<i>first_presentation_time</i>	The timestamp of the VSYNC at which this surface was first displayed. Note that 0 means the surface was never displayed.

##### Returns

`VdpStatus` The completion status of the operation.



Note that this API would block forever if queried about the surface most recently added to a presentation queue. That is because there would be no other surface that could possibly replace that surface as the currently displayed surface, and hence that surface would never become idle. For that reason, this function will return an error in that case.

**5.14.2.3** `typedef VdpStatus VdpPresentationQueueCreate(VdpDevice device, VdpPresentationQueueTarget presentation_queue_target, VdpPresentationQueue *presentation_queue)`

Create a VdpPresentationQueue.

#### Parameters

in	<i>device</i>	The device that will contain the queue.
in	<i>presentation_queue_target</i>	The location to display the content.
out	<i>presentation_queue</i>	The new queue's handle.

#### Returns

VdpStatus The completion status of the operation.

Note: The initial value for the background color will be set to an implementation-defined value.

**5.14.2.4** `typedef VdpStatus VdpPresentationQueueDestroy(VdpPresentationQueue presentation_queue)`

Destroy a VdpPresentationQueue.

#### Parameters

in	<i>presentation_queue</i>	The queue to destroy.
----	---------------------------	-----------------------

#### Returns

VdpStatus The completion status of the operation.

**5.14.2.5** `typedef VdpStatus VdpPresentationQueueDisplay(VdpPresentationQueue presentation_queue, VdpOutputSurface surface, uint32_t clip_width, uint32_t clip_height, VdpTime earliest_presentation_time)`

Enter a surface into the presentation queue.

#### Parameters

in	<i>presentation_queue</i>	The queue to query.
in	<i>surface</i>	The surface to enter into the queue.
in	<i>clip_width</i>	If set to a non-zero value, the presentation queue will display only clip_width pixels of the surface (anchored to the top-left corner of the surface).
in	<i>clip_height</i>	If set to a non-zero value, the presentation queue will display only clip_height lines of the surface (anchored to the top-left corner of the surface).
in	<i>earliest_presentation_time</i>	The timestamp associated with the surface. The presentation queue will not display the surface until the presentation queue's current time is at least this value.

**Returns**

VdpStatus The completion status of the operation.

Applications may choose to allow resizing of the presentation queue target (which may be e.g. a regular Window when using an X11-based implementation).

**clip\_width** and **clip\_height** may be used to limit the size of the displayed region of a surface, in order to match the specific region that was rendered to.

In turn, this allows the application to allocate over-sized (e.g. screen-sized) surfaces, but render to a region that matches the current size of the video window.

Using this technique, an application's response to window resizing may simply be to render to, and display, a different region of the surface, rather than de-/re-allocation of surfaces to match the updated window size.

Implementations may impose an upper bound on the number of entries contained by the presentation queue at a given time. This limit is likely different to the number of [VdpOutputSurfaces](#) that may be allocated at a given time. This limit applies to entries in the QUEUED or VISIBLE state only. In other words, entries that have transitioned from a QUEUED or VISIBLE state to an IDLE state do not count toward this limit.

**5.14.2.6** `typedef VdpStatus VdpPresentationQueueGetBackgroundColor(VdpPresentationQueue presentation_queue, VdpColor *background_color)`

Retrieve the current background color setting.

**Parameters**

in	<i>presentation_queue</i>	The queue to query.
out	<i>background_color</i>	The current background color.

**5.14.2.7** `typedef VdpStatus VdpPresentationQueueGetTime(VdpPresentationQueue presentation_queue, VdpTime *current_time)`

Retrieve the presentation queue's "current" time.

**Parameters**

in	<i>presentation_queue</i>	The queue to query.
out	<i>current_time</i>	The current time, which may represent a point between display VSYNC events.

**Returns**

VdpStatus The completion status of the operation.

**5.14.2.8** `typedef VdpStatus VdpPresentationQueueQuerySurfaceStatus(VdpPresentationQueue presentation_queue, VdpOutputSurface surface, VdpPresentationQueueStatus *status, VdpTime *first_presentation_time)`

Poll the current queue status of a surface.

## Parameters

in	<i>presentation_queue</i>	The queue to query.
in	<i>surface</i>	The surface to query.
out	<i>status</i>	The current status of the surface within the queue.
out	<i>first_presentation_time</i>	The timestamp of the VSYNC at which this surface was first displayed. Note that 0 means the surface was never displayed.

## Returns

VdpStatus The completion status of the operation.

5.14.2.9 `typedef VdpStatus VdpPresentationQueueSetBackgroundColor(VdpPresentationQueue presentation_queue, VdpColor *const background_color)`

Configure the background color setting.

## Parameters

in	<i>presentation_queue</i>	The queue to manipulate.
in	<i>background_color</i>	The new background color.

Note: Implementations may choose whether to apply the new background color value immediately, or defer it until the next surface is presented.

5.14.2.10 `typedef uint32_t VdpPresentationQueueTarget`

An opaque handle representing the location where video will be presented.

VdpPresentationQueueTarget are created using a [Window System Integration Layer](#) specific API, such as [VdpPresentationQueueTargetCreateX11](#).

5.14.2.11 `typedef VdpStatus VdpPresentationQueueTargetDestroy(VdpPresentationQueueTarget presentation_queue_target)`

Destroy a VdpPresentationQueueTarget.

## Parameters

in	<i>presentation_queue_target</i>	The target to destroy.
----	----------------------------------	------------------------

## Returns

VdpStatus The completion status of the operation.

#### 5.14.2.12 `typedef uint64_t VdpTime`

The representation of a point in time.

VdpTime timestamps are intended to be a high-precision timing system, potentially independent from any other time domain in the system.

Time is represented in units of nanoseconds. The origin (i.e. the time represented by a value of 0) is implementation dependent.

### 5.14.3 Enumeration Type Documentation

#### 5.14.3.1 `enum VdpPresentationQueueStatus`

The status of a surface within a presentation queue.

Enumerator

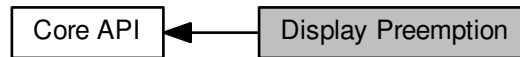
**`VDP_PRESENTATION_QUEUE_STATUS_IDLE`** The surface is not queued or currently visible.

**`VDP_PRESENTATION_QUEUE_STATUS_QUEUED`** The surface is in the queue, and not currently visible.

**`VDP_PRESENTATION_QUEUE_STATUS_VISIBLE`** The surface is the currently visible surface.

## 5.15 Display Preemption

Collaboration diagram for Display Preemption:



### Typedefs

- typedef void [VdpPreemptionCallback](#)([VdpDevice](#) device, void \*context)  
*A callback to notify the client application that a device's display has been preempted.*
- typedef [VdpStatus](#) [VdpPreemptionCallbackRegister](#)([VdpDevice](#) device, [VdpPreemptionCallback](#) callback, void \*context)  
*Configure the display preemption callback.*

### 5.15.1 Detailed Description

The Window System may operate within a frame-work (such as Linux's VT switching) where the display is shared between the Window System (e.g. X) and some other output mechanism (e.g. the VT.) Given this scenario, the Window System's control of the display could be preempted, and restored, at any time.

VDPAU does not mandate that implementations hide such preemptions from VDPAU client applications; doing so may impose extreme burdens upon VDPAU implementations. Equally, however, implementations are free to hide such preemptions from client applications.

VDPAU allows implementations to inform the client application when such a preemption has occurred, and then refuse to continue further operation.

Similarly, some form of fatal hardware error could prevent further operation of the VDPAU implementation, without a complete re-initialization.

The following discusses the behavior of implementations that choose not to hide preemption from client applications.

When preemption occurs, VDPAU internally destroys all objects; the client application need not do this. However, if the client application wishes to continue operation, it must recreate all objects that it uses. It is probable that this recreation will not succeed until the display ownership is restored to the Window System.

Once preemption has occurred, all VDPAU entry points will return the specific error code [VDP\\_STATUS\\_DISPLAY\\_PREEMPTED](#).

VDPAU client applications may also be notified of such preemptions and fatal errors via a callback. See [VdpPreemptionCallbackRegister](#) for more details.

### 5.15.2 Typedef Documentation

#### 5.15.2.1 typedef void VdpPreemptionCallback(VdpDevice device, void \*context)

A callback to notify the client application that a device's display has been preempted.

**Parameters**

in	<i>device</i>	The device that had its display preempted.
in	<i>context</i>	The client-supplied callback context information.

**Returns**

void No return value

**5.15.2.2** `typedef VdpStatus VdpPreemptionCallbackRegister(VdpDevice device, VdpPreemptionCallback callback, void *context)`

Configure the display preemption callback.

**Parameters**

in	<i>device</i>	The device to be monitored for preemption.
in	<i>callback</i>	The client application's callback function. If NULL, the callback is unregistered.
in	<i>context</i>	The client-supplied callback context information. This information will be passed to the callback function if/when invoked.

**Returns**

VdpStatus The completion status of the operation.

## 5.16 Entry Point Retrieval

Collaboration diagram for Entry Point Retrieval:



### Macros

- #define VDP\_FUNC\_ID\_GET\_ERROR\_STRING
- #define VDP\_FUNC\_ID\_GET\_PROC\_ADDRESS
- #define VDP\_FUNC\_ID\_GET\_API\_VERSION
- #define VDP\_FUNC\_ID\_GET\_INFORMATION\_STRING
- #define VDP\_FUNC\_ID\_DEVICE\_DESTROY
- #define VDP\_FUNC\_ID\_GENERATE\_CSC\_MATRIX
- #define VDP\_FUNC\_ID\_VIDEO\_SURFACE\_QUERY\_CAPABILITIES
- #define VDP\_FUNC\_ID\_VIDEO\_SURFACE\_QUERY\_GET\_PUT\_BITS\_Y\_CB\_CR\_CAPABILITIES
- #define VDP\_FUNC\_ID\_VIDEO\_SURFACE\_CREATE
- #define VDP\_FUNC\_ID\_VIDEO\_SURFACE\_DESTROY
- #define VDP\_FUNC\_ID\_VIDEO\_SURFACE\_GET\_PARAMETERS
- #define VDP\_FUNC\_ID\_VIDEO\_SURFACE\_GET\_BITS\_Y\_CB\_CR
- #define VDP\_FUNC\_ID\_VIDEO\_SURFACE\_PUT\_BITS\_Y\_CB\_CR
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_QUERY\_CAPABILITIES
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_QUERY\_GET\_PUT\_BITS\_NATIVE\_CAPABILITIES
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_QUERY\_PUT\_BITS\_INDEXED\_CAPABILITIES
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_QUERY\_PUT\_BITS\_Y\_CB\_CR\_CAPABILITIES
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_CREATE
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_DESTROY
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_GET\_PARAMETERS
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_GET\_BITS\_NATIVE
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_PUT\_BITS\_NATIVE
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_PUT\_BITS\_INDEXED
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_PUT\_BITS\_Y\_CB\_CR
- #define VDP\_FUNC\_ID\_BITMAP\_SURFACE\_QUERY\_CAPABILITIES
- #define VDP\_FUNC\_ID\_BITMAP\_SURFACE\_CREATE
- #define VDP\_FUNC\_ID\_BITMAP\_SURFACE\_DESTROY
- #define VDP\_FUNC\_ID\_BITMAP\_SURFACE\_GET\_PARAMETERS
- #define VDP\_FUNC\_ID\_BITMAP\_SURFACE\_PUT\_BITS\_NATIVE
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_RENDER\_OUTPUT\_SURFACE
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_RENDER\_BITMAP\_SURFACE
- #define VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_RENDER\_VIDEO\_SURFACE\_LUMA
- #define VDP\_FUNC\_ID\_DECODER\_QUERY\_CAPABILITIES
- #define VDP\_FUNC\_ID\_DECODER\_CREATE
- #define VDP\_FUNC\_ID\_DECODER\_DESTROY
- #define VDP\_FUNC\_ID\_DECODER\_GET\_PARAMETERS
- #define VDP\_FUNC\_ID\_DECODER\_RENDER

- `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_FEATURE_SUPPORT`
- `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_PARAMETER_SUPPORT`
- `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_ATTRIBUTE_SUPPORT`
- `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_PARAMETER_VALUE_RANGE`
- `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_ATTRIBUTE_VALUE_RANGE`
- `#define VDP_FUNC_ID_VIDEO_MIXER_CREATE`
- `#define VDP_FUNC_ID_VIDEO_MIXER_SET_FEATURE_ENABLES`
- `#define VDP_FUNC_ID_VIDEO_MIXER_SET_ATTRIBUTE_VALUES`
- `#define VDP_FUNC_ID_VIDEO_MIXER_GET_FEATURE_SUPPORT`
- `#define VDP_FUNC_ID_VIDEO_MIXER_GET_FEATURE_ENABLES`
- `#define VDP_FUNC_ID_VIDEO_MIXER_GET_PARAMETER_VALUES`
- `#define VDP_FUNC_ID_VIDEO_MIXER_GET_ATTRIBUTE_VALUES`
- `#define VDP_FUNC_ID_VIDEO_MIXER_DESTROY`
- `#define VDP_FUNC_ID_VIDEO_MIXER_RENDER`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_TARGET_DESTROY`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_CREATE`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_DESTROY`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_SET_BACKGROUND_COLOR`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_GET_BACKGROUND_COLOR`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_GET_TIME`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_DISPLAY`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_BLOCK_UNTIL_SURFACE_IDLE`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_QUERY_SURFACE_STATUS`
- `#define VDP_FUNC_ID_PREEMPTION_CALLBACK_REGISTER`
- `#define VDP_FUNC_ID_BASE_WINSYS 0x1000`

## Typedefs

- `typedef uint32_t VdpFuncId`  
*A type suitable for `VdpGetProcAddress`'s **function\_id** parameter.*
- `typedef VdpStatus VdpGetProcAddress(VdpDevice device, VdpFuncId function_id, void **function_pointer)`  
*Retrieve a VDPAU function pointer.*

### 5.16.1 Detailed Description

In order to facilitate multiple implementations of VDPAU co-existing within a single process, all functionality is available via function pointers. The mechanism to retrieve those function pointers is described below.

### 5.16.2 Macro Definition Documentation

5.16.2.1 `#define VDP_FUNC_ID_BASE_WINSYS 0x1000`

5.16.2.2 `#define VDP_FUNC_ID_BITMAP_SURFACE_CREATE`

5.16.2.3 `#define VDP_FUNC_ID_BITMAP_SURFACE_DESTROY`

5.16.2.4 `#define VDP_FUNC_ID_BITMAP_SURFACE_GET_PARAMETERS`



5.16.2.5 `#define VDP_FUNC_ID_BITMAP_SURFACE_PUT_BITS_NATIVE`

5.16.2.6 `#define VDP_FUNC_ID_BITMAP_SURFACE_QUERY_CAPABILITIES`

5.16.2.7 `#define VDP_FUNC_ID_DECODER_CREATE`

5.16.2.8 `#define VDP_FUNC_ID_DECODER_DESTROY`

5.16.2.9 `#define VDP_FUNC_ID_DECODER_GET_PARAMETERS`

5.16.2.10 `#define VDP_FUNC_ID_DECODER_QUERY_CAPABILITIES`

5.16.2.11 `#define VDP_FUNC_ID_DECODER_RENDER`

5.16.2.12 `#define VDP_FUNC_ID_DEVICE_DESTROY`

5.16.2.13 `#define VDP_FUNC_ID_GENERATE_CSC_MATRIX`

5.16.2.14 `#define VDP_FUNC_ID_GET_API_VERSION`

5.16.2.15 `#define VDP_FUNC_ID_GET_ERROR_STRING`

5.16.2.16 `#define VDP_FUNC_ID_GET_INFORMATION_STRING`

5.16.2.17 `#define VDP_FUNC_ID_GET_PROC_ADDRESS`

5.16.2.18 `#define VDP_FUNC_ID_OUTPUT_SURFACE_CREATE`

5.16.2.19 `#define VDP_FUNC_ID_OUTPUT_SURFACE_DESTROY`

5.16.2.20 `#define VDP_FUNC_ID_OUTPUT_SURFACE_GET_BITS_NATIVE`

5.16.2.21 `#define VDP_FUNC_ID_OUTPUT_SURFACE_GET_PARAMETERS`

5.16.2.22 `#define VDP_FUNC_ID_OUTPUT_SURFACE_PUT_BITS_INDEXED`

5.16.2.23 `#define VDP_FUNC_ID_OUTPUT_SURFACE_PUT_BITS_NATIVE`

5.16.2.24 `#define VDP_FUNC_ID_OUTPUT_SURFACE_PUT_BITS_Y_CB_CR`

5.16.2.25 `#define VDP_FUNC_ID_OUTPUT_SURFACE_QUERY_CAPABILITIES`

5.16.2.26 `#define VDP_FUNC_ID_OUTPUT_SURFACE_QUERY_GET_PUT_BITS_NATIVE_CAPABILITIES`

5.16.2.27 `#define VDP_FUNC_ID_OUTPUT_SURFACE_QUERY_PUT_BITS_INDEXED_CAPABILITIES`

5.16.2.28 `#define VDP_FUNC_ID_OUTPUT_SURFACE_QUERY_PUT_BITS_Y_CB_CR_CAPABILITIES`

5.16.2.29 `#define VDP_FUNC_ID_OUTPUT_SURFACE_RENDER_BITMAP_SURFACE`

5.16.2.30 `#define VDP_FUNC_ID_OUTPUT_SURFACE_RENDER_OUTPUT_SURFACE`

5.16.2.31 `#define VDP_FUNC_ID_OUTPUT_SURFACE_RENDER_VIDEO_SURFACE_LUMA`

5.16.2.32 `#define VDP_FUNC_ID_PREEMPTION_CALLBACK_REGISTER`

5.16.2.33 `#define VDP_FUNC_ID_PRESENTATION_QUEUE_BLOCK_UNTIL_SURFACE_IDLE`

5.16.2.34 `#define VDP_FUNC_ID_PRESENTATION_QUEUE_CREATE`

5.16.2.35 `#define VDP_FUNC_ID_PRESENTATION_QUEUE_DESTROY`

5.16.2.36 `#define VDP_FUNC_ID_PRESENTATION_QUEUE_DISPLAY`

5.16.2.37 `#define VDP_FUNC_ID_PRESENTATION_QUEUE_GET_BACKGROUND_COLOR`

5.16.2.38 `#define VDP_FUNC_ID_PRESENTATION_QUEUE_GET_TIME`

5.16.2.39 `#define VDP_FUNC_ID_PRESENTATION_QUEUE_QUERY_SURFACE_STATUS`

5.16.2.40 `#define VDP_FUNC_ID_PRESENTATION_QUEUE_SET_BACKGROUND_COLOR`

5.16.2.41 `#define VDP_FUNC_ID_PRESENTATION_QUEUE_TARGET_DESTROY`

5.16.2.42 `#define VDP_FUNC_ID_VIDEO_MIXER_CREATE`

5.16.2.43 `#define VDP_FUNC_ID_VIDEO_MIXER_DESTROY`

5.16.2.44 `#define VDP_FUNC_ID_VIDEO_MIXER_GET_ATTRIBUTE_VALUES`

5.16.2.45 `#define VDP_FUNC_ID_VIDEO_MIXER_GET_FEATURE_ENABLES`

5.16.2.46 `#define VDP_FUNC_ID_VIDEO_MIXER_GET_FEATURE_SUPPORT`

5.16.2.47 `#define VDP_FUNC_ID_VIDEO_MIXER_GET_PARAMETER_VALUES`

5.16.2.48 `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_ATTRIBUTE_SUPPORT`

5.16.2.49 `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_ATTRIBUTE_VALUE_RANGE`

5.16.2.50 `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_FEATURE_SUPPORT`

5.16.2.51 `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_PARAMETER_SUPPORT`

5.16.2.52 `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_PARAMETER_VALUE_RANGE`

5.16.2.53 `#define VDP_FUNC_ID_VIDEO_MIXER_RENDER`

5.16.2.54 `#define VDP_FUNC_ID_VIDEO_MIXER_SET_ATTRIBUTE_VALUES`

5.16.2.55 `#define VDP_FUNC_ID_VIDEO_MIXER_SET_FEATURE_ENABLES`

5.16.2.56 `#define VDP_FUNC_ID_VIDEO_SURFACE_CREATE`

5.16.2.57 `#define VDP_FUNC_ID_VIDEO_SURFACE_DESTROY`

5.16.2.58 `#define VDP_FUNC_ID_VIDEO_SURFACE_GET_BITS_Y_CB_CR`

5.16.2.59 `#define VDP_FUNC_ID_VIDEO_SURFACE_GET_PARAMETERS`

5.16.2.60 `#define VDP_FUNC_ID_VIDEO_SURFACE_PUT_BITS_Y_CB_CR`

5.16.2.61 `#define VDP_FUNC_ID_VIDEO_SURFACE_QUERY_CAPABILITIES`

5.16.2.62 `#define VDP_FUNC_ID_VIDEO_SURFACE_QUERY_GET_PUT_BITS_Y_CB_CR_CAPABILITIES`

### 5.16.3 Typedef Documentation

5.16.3.1 `typedef uint32_t VdpFuncId`

A type suitable for [VdpGetProcAddress](#)'s `function_id` parameter.

5.16.3.2 `typedef VdpStatus VdpGetProcAddress(VdpDevice device, VdpFuncId function_id, void **function_pointer)`

Retrieve a VDDPAU function pointer.

#### Parameters

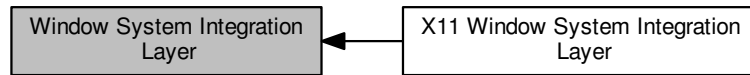
in	<i>device</i>	The device that the function will operate against.
in	<i>function_id</i>	The specific function to retrieve.
out	<i>function_pointer</i>	The actual pointer for the application to call.

#### Returns

VdpStatus The completion status of the operation.

## 5.17 Window System Integration Layer

Collaboration diagram for Window System Integration Layer:



### Modules

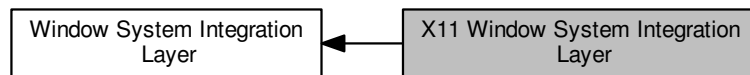
- [X11 Window System Integration Layer](#)

### 5.17.1 Detailed Description

The set of VDPAU functionality specific to an individual Windowing System.

## 5.18 X11 Window System Integration Layer

Collaboration diagram for X11 Window System Integration Layer:



- typedef `VdpStatus VdpDeviceCreateX11`(Display \*display, int screen, `VdpDevice` \*device, `VdpGetProcAddress` \*\*get\_proc\_address)  
Create a `VdpDevice` object for use with X11.
- typedef `VdpStatus VdpPresentationQueueTargetCreateX11`(`VdpDevice` device, Drawable drawable, `VdpPresentationQueueTarget` \*target)  
Create a `VdpPresentationQueueTarget` for use with X11.
- `VdpDeviceCreateX11 vdp_device_create_x11`  
Create a `VdpDevice` object for use with X11. This is an actual symbol of type `VdpDeviceCreateX11`.
- #define `VDP_FUNC_ID_PRESENTATION_QUEUE_TARGET_CREATE_X11`

### 5.18.1 Detailed Description

The set of VDPAU functionality specific to usage with the X Window System.

### 5.18.2 Library Layout

An X11-oriented VDPAU installation consists of the following components:

- Header files. These files are located in the standard system header file path.
  - `vdpa/vdpau.h`
  - `vdpa/vdpau_x11.h`
- The VDPAU wrapper library. These files are located in the standard system (possibly X11-specific) library path.
  - `libvdpa.so.1` (runtime)
  - `libvdpa.so` (development)
- Back-end driver files. These files are located in a system-defined library path, which is configurable at compile time but is typically `/usr/lib/vdpa`. Use `pkg-config --variable=moduledir vdpau` to locate the driver install path.
  - `$moduledir/libvdpa_%s.so.1` For example:
  - `/usr/lib/vdpa/libvdpa_nvidia.so.1`
  - `/usr/lib/vdpa/libvdpa_intel.so.1`
  - `/usr/lib/vdpa/libvdpa_ati.so.1` The library path can be overridden by the `VDPAU_DRIVER_PATH` environment variable.

The VDPAU wrapper library implements just one function; `vdp_device_create_x11`. The wrapper implements this function by dynamically loading the appropriate back-end driver file mentioned above. When available, the wrapper uses the DRI2 extension's DRI2Connect request with the driver type 'DRI2DriverVDPAU' to determine which back-end driver to load. If that fails, the wrapper library hard-codes the driver name as "nvidia", although this can be overridden using the environment variable VDPAU\_DRIVER.

The back-end driver is expected to implement a function named `vdp_imp_device_create_x11`. The wrapper will call this function to actually implement the `vdp_device_create_x11` application call.

Note that it is theoretically possible for an application to create multiple `VdpDevice` objects. In this case, the wrapper library may load multiple back-end drivers into the same application, and/or invoke a specific back-end driver's `VdpImpDeviceCreateX11` multiple times. The wrapper library imposes no policy regarding whether the application may instantiate multiple `VdpDevice` objects for the same display and/or screen. However, back-end drivers are free to limit the number of `VdpDevice` objects as required by their implementation.

### 5.18.3 Macro Definition Documentation

#### 5.18.3.1 `#define VDP_FUNC_ID_PRESENTATION_QUEUE_TARGET_CREATE_X11`

### 5.18.4 Typedef Documentation

#### 5.18.4.1 `typedef VdpStatus VdpDeviceCreateX11(Display *display, int screen, VdpDevice *device, VdpGetProcAddress **get_proc_address)`

Create a `VdpDevice` object for use with X11.

##### Parameters

in	<i>display</i>	The X Display that the <code>VdpDevice</code> will operate against.
in	<i>screen</i>	The X screen that the <code>VdpDevice</code> will operate against.
out	<i>device</i>	The new device's handle.
out	<i>get_proc_address</i>	The <code>get_proc_address</code> entry point to use with this device.

##### Returns

`VdpStatus` The completion status of the operation.

#### 5.18.4.2 `typedef VdpStatus VdpPresentationQueueTargetCreateX11(VdpDevice device, Drawable drawable, VdpPresentationQueueTarget *target)`

Create a `VdpPresentationQueueTarget` for use with X11.

##### Parameters

in	<i>device</i>	The device that will contain the queue target.
in	<i>drawable</i>	The X11 Drawable that the presentation queue will present into.
out	<i>target</i>	The new queue target's handle.

**Returns**

VdpStatus The completion status of the operation.

Note: VDPAU expects to own the entire drawable for the duration of time that the presentation queue target exists. In particular, implementations may choose to manipulate client-visible X11 window state as required. As such, it is recommended that applications create a dedicated window for the presentation queue target, as a child (grand-child, ...) of their top-level application window.

Applications may also create child-windows of the presentation queue target, which will cover any presented video in the normal fashion. VDPAU implementations will not manipulate such child windows in any fashion.

**5.18.5 Variable Documentation****5.18.5.1 VdpDeviceCreateX11 vdp\_device\_create\_x11**

Create a VdpDevice object for use with X11. This is an actual symbol of type [VdpDeviceCreateX11](#).





## Chapter 6

# Data Structure Documentation

### 6.1 VdpBitstreamBuffer Struct Reference

Application data buffer containing compressed video data.

```
#include <vdpau.h>
```

#### Data Fields

- uint32\_t [struct\\_version](#)
- void const \* [bitstream](#)
- uint32\_t [bitstream\\_bytes](#)

#### 6.1.1 Detailed Description

Application data buffer containing compressed video data.

#### 6.1.2 Field Documentation

##### 6.1.2.1 void const\* VdpBitstreamBuffer::bitstream

A pointer to the bitstream data bytes

##### 6.1.2.2 uint32\_t VdpBitstreamBuffer::bitstream\_bytes

The number of data bytes

##### 6.1.2.3 uint32\_t VdpBitstreamBuffer::struct\_version

This field must be filled with VDP\_BITSTREAM\_BUFFER\_VERSION

The documentation for this struct was generated from the following file:

- [vdpau/vdpau.h](#)

## 6.2 VdpColor Struct Reference

```
#include <vdpu.h>
```

### Data Fields

- float [red](#)
- float [green](#)
- float [blue](#)
- float [alpha](#)

### 6.2.1 Detailed Description

A constant RGBA color.

Note that the components are stored as float values in the range 0.0...1.0 rather than format-specific integer values. This allows [VdpColor](#) values to be independent from the exact surface format(s) in use.

### 6.2.2 Field Documentation

6.2.2.1 float [VdpColor::alpha](#)

6.2.2.2 float [VdpColor::blue](#)

6.2.2.3 float [VdpColor::green](#)

6.2.2.4 float [VdpColor::red](#)

The documentation for this struct was generated from the following file:

- [vdpu/vdpu.h](#)

## 6.3 VdpLayer Struct Reference

Definition of an additional [VdpOutputSurface](#) layer in the compositing model.

```
#include <vdpu.h>
```

### Data Fields

- uint32\_t [struct\\_version](#)
- [VdpOutputSurface](#) [source\\_surface](#)
- [VdpRect](#) const \* [source\\_rect](#)
- [VdpRect](#) const \* [destination\\_rect](#)

### 6.3.1 Detailed Description

Definition of an additional [VdpOutputSurface](#) layer in the compositing model.

### 6.3.2 Field Documentation

#### 6.3.2.1 `VdpRect const* VdpLayer::destination_rect`

The sub-rectangle of the destination surface to map this layer into. This rectangle is relative to the entire destination surface. This rectangle will be clipped by [VdpVideoMixerRender](#)'s **destination\_rect**. If NULL, the destination rectangle will be sized to match the source rectangle, and will be located at the origin.

#### 6.3.2.2 `VdpRect const* VdpLayer::source_rect`

The sub-rectangle of the source surface to use. If NULL, the entire source surface will be used.

#### 6.3.2.3 `VdpOutputSurface VdpLayer::source_surface`

The surface to composite from.

#### 6.3.2.4 `uint32_t VdpLayer::struct_version`

This field must be filled with VDP\_LAYER\_VERSION

The documentation for this struct was generated from the following file:

- [vdpau/vdpau.h](#)

## 6.4 VdpOutputSurfaceRenderBlendState Struct Reference

Complete blending operation definition.

```
#include <vdpau.h>
```

### Data Fields

- `uint32_t struct_version`
- `VdpOutputSurfaceRenderBlendFactor blend_factor_source_color`
- `VdpOutputSurfaceRenderBlendFactor blend_factor_destination_color`
- `VdpOutputSurfaceRenderBlendFactor blend_factor_source_alpha`
- `VdpOutputSurfaceRenderBlendFactor blend_factor_destination_alpha`
- `VdpOutputSurfaceRenderBlendEquation blend_equation_color`
- `VdpOutputSurfaceRenderBlendEquation blend_equation_alpha`
- `VdpColor blend_constant`

### 6.4.1 Detailed Description

Complete blending operation definition.

A "blend state" operation controls the math behind certain rendering operations.

The blend math is the familiar OpenGL blend math:

$$dst.a = equation(blendFactorDstAlpha * dst.a, blendFactorSrcAlpha * src.a);$$

$$dst.rgb = equation(blendFactorDstColor * dst.rgb, blendFactorSrcColor * src.rgb);$$

Note that when equation is MIN or MAX, the blend factors and constants are ignored, and are treated as if they were 1.0.

### 6.4.2 Field Documentation

6.4.2.1 **VdpColor** VdpOutputSurfaceRenderBlendState::blend\_constant

6.4.2.2 **VdpOutputSurfaceRenderBlendEquation** VdpOutputSurfaceRenderBlendState::blend\_equation\_alpha

6.4.2.3 **VdpOutputSurfaceRenderBlendEquation** VdpOutputSurfaceRenderBlendState::blend\_equation\_color

6.4.2.4 **VdpOutputSurfaceRenderBlendFactor** VdpOutputSurfaceRenderBlendState::blend\_factor\_destination\_alpha

6.4.2.5 **VdpOutputSurfaceRenderBlendFactor** VdpOutputSurfaceRenderBlendState::blend\_factor\_destination\_color

6.4.2.6 **VdpOutputSurfaceRenderBlendFactor** VdpOutputSurfaceRenderBlendState::blend\_factor\_source\_alpha

6.4.2.7 **VdpOutputSurfaceRenderBlendFactor** VdpOutputSurfaceRenderBlendState::blend\_factor\_source\_color

6.4.2.8 **uint32\_t** VdpOutputSurfaceRenderBlendState::struct\_version

This field must be filled with VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_STATE\_VERSION

The documentation for this struct was generated from the following file:

- [vdpau/vdpau.h](#)

## 6.5 VdpPictureInfoH264 Struct Reference

Picture parameter information for an H.264 picture.

```
#include <vdpau.h>
```

## Data Fields

- uint32\_t [slice\\_count](#)
- int32\_t [field\\_order\\_cnt](#) [2]
- [VdpBool](#) [is\\_reference](#)
- [VdpReferenceFrameH264](#) [referenceFrames](#) [16]

### H.264 bitstream

*Copies of the H.264 bitstream fields.*

- uint16\_t [frame\\_num](#)
- uint8\_t [field\\_pic\\_flag](#)
- uint8\_t [bottom\\_field\\_flag](#)
- uint8\_t [num\\_ref\\_frames](#)
- uint8\_t [mb\\_adaptive\\_frame\\_field\\_flag](#)
- uint8\_t [constrained\\_intra\\_pred\\_flag](#)
- uint8\_t [weighted\\_pred\\_flag](#)
- uint8\_t [weighted\\_bipred\\_idc](#)
- uint8\_t [frame\\_mbs\\_only\\_flag](#)
- uint8\_t [transform\\_8x8\\_mode\\_flag](#)
- int8\_t [chroma\\_qp\\_index\\_offset](#)
- int8\_t [second\\_chroma\\_qp\\_index\\_offset](#)
- int8\_t [pic\\_init\\_qp\\_minus26](#)
- uint8\_t [num\\_ref\\_idx\\_l0\\_active\\_minus1](#)
- uint8\_t [num\\_ref\\_idx\\_l1\\_active\\_minus1](#)
- uint8\_t [log2\\_max\\_frame\\_num\\_minus4](#)
- uint8\_t [pic\\_order\\_cnt\\_type](#)
- uint8\_t [log2\\_max\\_pic\\_order\\_cnt\\_lsb\\_minus4](#)
- uint8\_t [delta\\_pic\\_order\\_always\\_zero\\_flag](#)
- uint8\_t [direct\\_8x8\\_inference\\_flag](#)
- uint8\_t [entropy\\_coding\\_mode\\_flag](#)
- uint8\_t [pic\\_order\\_present\\_flag](#)
- uint8\_t [deblocking\\_filter\\_control\\_present\\_flag](#)
- uint8\_t [redundant\\_pic\\_cnt\\_present\\_flag](#)
- uint8\_t [scaling\\_lists\\_4x4](#) [6][16]
- uint8\_t [scaling\\_lists\\_8x8](#) [2][64]

### 6.5.1 Detailed Description

Picture parameter information for an H.264 picture.

Note: The [referenceFrames](#) array must contain the "DPB" as defined by the H.264 specification. In particular, once a reference frame has been decoded to a surface, that surface must continue to appear in the DPB until no longer required to predict any future frame. Once a surface is removed from the DPB, it can no longer be used as a reference, unless decoded again.

Also note that only surfaces previously generated using [VdpDecoderRender](#) may be used as reference frames. In particular, surfaces filled using any "put bits" API will not work.

Note: References to bitstream fields below may refer to data literally parsed from the bitstream, or derived from the bitstream using a mechanism described in the specification.

Note: VDPAU clients must use [VdpPictureInfoH264Predictive](#) to describe the attributes of a frame being decoded with VDP\_DECODER\_PROFILE\_H264\_HIGH\_444\_PREDICTIVE.

## 6.5.2 Field Documentation

6.5.2.1 `uint8_t VdpPictureInfoH264::bottom_field_flag`

6.5.2.2 `int8_t VdpPictureInfoH264::chroma_qp_index_offset`

6.5.2.3 `uint8_t VdpPictureInfoH264::constrained_intra_pred_flag`

6.5.2.4 `uint8_t VdpPictureInfoH264::deblocking_filter_control_present_flag`

6.5.2.5 `uint8_t VdpPictureInfoH264::delta_pic_order_always_zero_flag`

6.5.2.6 `uint8_t VdpPictureInfoH264::direct_8x8_inference_flag`

6.5.2.7 `uint8_t VdpPictureInfoH264::entropy_coding_mode_flag`

6.5.2.8 `int32_t VdpPictureInfoH264::field_order_cnt[2]`

[0]: top, [1]: bottom

6.5.2.9 `uint8_t VdpPictureInfoH264::field_pic_flag`

6.5.2.10 `uint8_t VdpPictureInfoH264::frame_mbs_only_flag`

6.5.2.11 `uint16_t VdpPictureInfoH264::frame_num`

6.5.2.12 `VdpBool VdpPictureInfoH264::is_reference`

Will the decoded frame be used as a reference later.

6.5.2.13 `uint8_t VdpPictureInfoH264::log2_max_frame_num_minus4`

6.5.2.14 `uint8_t VdpPictureInfoH264::log2_max_pic_order_cnt_lsb_minus4`

6.5.2.15 `uint8_t VdpPictureInfoH264::mb_adaptive_frame_field_flag`

6.5.2.16 `uint8_t VdpPictureInfoH264::num_ref_frames`

6.5.2.17 `uint8_t VdpPictureInfoH264::num_ref_idx_l0_active_minus1`

6.5.2.18 `uint8_t VdpPictureInfoH264::num_ref_idx_l1_active_minus1`

6.5.2.19 `int8_t VdpPictureInfoH264::pic_init_qp_minus26`

6.5.2.20 `uint8_t VdpPictureInfoH264::pic_order_cnt_type`

6.5.2.21 `uint8_t VdpPictureInfoH264::pic_order_present_flag`

6.5.2.22 `uint8_t VdpPictureInfoH264::redundant_pic_cnt_present_flag`

6.5.2.23 `VdpReferenceFrameH264 VdpPictureInfoH264::referenceFrames[16]`

See [VdpPictureInfoH264](#) for instructions regarding this field.

6.5.2.24 `uint8_t VdpPictureInfoH264::scaling_lists_4x4[6][16]`

Convert to raster order.

6.5.2.25 `uint8_t VdpPictureInfoH264::scaling_lists_8x8[2][64]`

Convert to raster order.

6.5.2.26 `int8_t VdpPictureInfoH264::second_chroma_qp_index_offset`

6.5.2.27 `uint32_t VdpPictureInfoH264::slice_count`

Number of slices in the bitstream provided.

6.5.2.28 `uint8_t VdpPictureInfoH264::transform_8x8_mode_flag`

6.5.2.29 `uint8_t VdpPictureInfoH264::weighted_bipred_idc`

6.5.2.30 `uint8_t VdpPictureInfoH264::weighted_pred_flag`

The documentation for this struct was generated from the following file:

- [vdpau/vdpau.h](#)

## 6.6 VdpPictureInfoH264Predictive Struct Reference

Picture parameter information for an H.264 Hi444PP picture.

```
#include <vdpau.h>
```

### Data Fields

- [VdpPictureInfoH264 pictureInfo](#)

### H.264 bitstream

*Copies of the H.264 bitstream fields.*

- `uint8_t qprime_y_zero_transform_bypass_flag`
- `uint8_t separate_colour_plane_flag`

### 6.6.1 Detailed Description

Picture parameter information for an H.264 Hi444PP picture.

Note: VDPAU clients must use [VdpPictureInfoH264Predictive](#) to describe the attributes of a frame being decoded with VDP\_DECODER\_PROFILE\_H264\_HIGH\_444\_PREDICTIVE.

Note: software drivers may choose to honor values of `qpprime_y_zero_transform_bypass_flag` greater than 1 for internal use.

### 6.6.2 Field Documentation

#### 6.6.2.1 `VdpPictureInfoH264 VdpPictureInfoH264Predictive::pictureInfo`

[VdpPictureInfoH264](#) struct.

#### 6.6.2.2 `uint8_t VdpPictureInfoH264Predictive::qpprime_y_zero_transform_bypass_flag`

0 - lossless disabled 1 - lossless enabled

#### 6.6.2.3 `uint8_t VdpPictureInfoH264Predictive::separate_colour_plane_flag`

0 - disabled 1 - enabled

The documentation for this struct was generated from the following file:

- [vdpau/vdpau.h](#)

## 6.7 VdpPictureInfoHEVC Struct Reference

Picture parameter information for an H.265/HEVC picture.

```
#include <vdpau.h>
```



## Data Fields

- `int32_t CurrPicOrderCntVal`
- `VdpVideoSurface RefPics` [16]
- `int32_t PicOrderCntVal` [16]
- `uint8_t IsLongTerm` [16]
- `uint8_t NumPocStCurrBefore`
- `uint8_t NumPocStCurrAfter`
- `uint8_t NumPocLtCurr`
- `uint8_t RefPicSetStCurrBefore` [8]
- `uint8_t RefPicSetStCurrAfter` [8]
- `uint8_t RefPicSetLtCurr` [8]

## HEVC Sequence Parameter Set

*Copies of the HEVC Sequence Parameter Set bitstream fields.*

- `uint8_t chroma_format_idc`
- `uint8_t separate_colour_plane_flag`
- `uint32_t pic_width_in_luma_samples`
- `uint32_t pic_height_in_luma_samples`
- `uint8_t bit_depth_luma_minus8`
- `uint8_t bit_depth_chroma_minus8`
- `uint8_t log2_max_pic_order_cnt_lsb_minus4`
- `uint8_t sps_max_dec_pic_buffering_minus1`
- `uint8_t log2_min_luma_coding_block_size_minus3`
- `uint8_t log2_diff_max_min_luma_coding_block_size`
- `uint8_t log2_min_transform_block_size_minus2`
- `uint8_t log2_diff_max_min_transform_block_size`
- `uint8_t max_transform_hierarchy_depth_inter`
- `uint8_t max_transform_hierarchy_depth_intra`
- `uint8_t scaling_list_enabled_flag`
- `uint8_t ScalingList4x4` [6][16]
- `uint8_t ScalingList8x8` [6][64]
- `uint8_t ScalingList16x16` [6][64]
- `uint8_t ScalingList32x32` [2][64]
- `uint8_t ScalingListDCCoeff16x16` [6]
- `uint8_t ScalingListDCCoeff32x32` [2]
- `uint8_t amp_enabled_flag`
- `uint8_t sample_adaptive_offset_enabled_flag`
- `uint8_t pcm_enabled_flag`
- `uint8_t pcm_sample_bit_depth_luma_minus1`
- `uint8_t pcm_sample_bit_depth_chroma_minus1`
- `uint8_t log2_min_pcm_luma_coding_block_size_minus3`
- `uint8_t log2_diff_max_min_pcm_luma_coding_block_size`
- `uint8_t pcm_loop_filter_disabled_flag`
- `uint8_t num_short_term_ref_pic_sets`
- `uint8_t long_term_ref_pics_present_flag`
- `uint8_t num_long_term_ref_pics_sps`
- `uint8_t sps_temporal_mvp_enabled_flag`
- `uint8_t strong_intra_smoothing_enabled_flag`

## HEVC Picture Parameter Set

*Copies of the HEVC Picture Parameter Set bitstream fields.*

- `uint8_t dependent_slice_segments_enabled_flag`
- `uint8_t output_flag_present_flag`
- `uint8_t num_extra_slice_header_bits`
- `uint8_t sign_data_hiding_enabled_flag`
- `uint8_t cabac_init_present_flag`
- `uint8_t num_ref_idx_l0_default_active_minus1`

- [uint8\\_t num\\_ref\\_idx\\_l1\\_default\\_active\\_minus1](#)
- [int8\\_t init\\_qp\\_minus26](#)
- [uint8\\_t constrained\\_intra\\_pred\\_flag](#)
- [uint8\\_t transform\\_skip\\_enabled\\_flag](#)
- [uint8\\_t cu\\_qp\\_delta\\_enabled\\_flag](#)
- [uint8\\_t diff\\_cu\\_qp\\_delta\\_depth](#)
- [int8\\_t pps\\_cb\\_qp\\_offset](#)
- [int8\\_t pps\\_cr\\_qp\\_offset](#)
- [uint8\\_t pps\\_slice\\_chroma\\_qp\\_offsets\\_present\\_flag](#)
- [uint8\\_t weighted\\_pred\\_flag](#)
- [uint8\\_t weighted\\_bipred\\_flag](#)
- [uint8\\_t transquant\\_bypass\\_enabled\\_flag](#)
- [uint8\\_t tiles\\_enabled\\_flag](#)
- [uint8\\_t entropy\\_coding\\_sync\\_enabled\\_flag](#)
- [uint8\\_t num\\_tile\\_columns\\_minus1](#)
- [uint8\\_t num\\_tile\\_rows\\_minus1](#)
- [uint8\\_t uniform\\_spacing\\_flag](#)
- [uint16\\_t column\\_width\\_minus1 \[20\]](#)
- [uint16\\_t row\\_height\\_minus1 \[22\]](#)
- [uint8\\_t loop\\_filter\\_across\\_tiles\\_enabled\\_flag](#)
- [uint8\\_t pps\\_loop\\_filter\\_across\\_slices\\_enabled\\_flag](#)
- [uint8\\_t deblocking\\_filter\\_control\\_present\\_flag](#)
- [uint8\\_t deblocking\\_filter\\_override\\_enabled\\_flag](#)
- [uint8\\_t pps\\_deblocking\\_filter\\_disabled\\_flag](#)
- [int8\\_t pps\\_beta\\_offset\\_div2](#)
- [int8\\_t pps\\_tc\\_offset\\_div2](#)
- [uint8\\_t lists\\_modification\\_present\\_flag](#)
- [uint8\\_t log2\\_parallel\\_merge\\_level\\_minus2](#)
- [uint8\\_t slice\\_segment\\_header\\_extension\\_present\\_flag](#)

### HEVC Slice Segment Header

*Copies of the HEVC Slice Segment Header bitstream fields and calculated values detailed in the specification.*

- [uint8\\_t IDRPicFlag](#)
- [uint8\\_t RAPPicFlag](#)
- [uint8\\_t CurrRpsIdx](#)
- [uint32\\_t NumPocTotalCurr](#)
- [uint32\\_t NumDeltaPocsOfRefRpsIdx](#)
- [uint32\\_t NumShortTermPictureSliceHeaderBits](#)
- [uint32\\_t NumLongTermPictureSliceHeaderBits](#)

## 6.7.1 Detailed Description

Picture parameter information for an H.265/HEVC picture.

References to bitstream fields below may refer to data literally parsed from the bitstream, or derived from the bitstream using a mechanism described in Rec. ITU-T H.265 (04/2013), hereafter referred to as "the H.265/HEVC Specification".

VDP AU H.265/HEVC implementations implement the portion of the decoding process described by clauses 8.4, 8.5, 8.6 and 8.7 of the the H.265/HEVC Specification. [VdpPictureInfoHEVC](#) provides enough data to complete this portion of the decoding process, plus additional information not defined in the H.265/HEVC Specification that may be useful to particular implementations.

Client applications must supply every field in this struct.

## 6.7.2 Field Documentation

6.7.2.1 `uint8_t VdpPictureInfoHEVC::amp_enabled_flag`

6.7.2.2 `uint8_t VdpPictureInfoHEVC::bit_depth_chroma_minus8`

6.7.2.3 `uint8_t VdpPictureInfoHEVC::bit_depth_luma_minus8`

6.7.2.4 `uint8_t VdpPictureInfoHEVC::cabac_init_present_flag`

6.7.2.5 `uint8_t VdpPictureInfoHEVC::chroma_format_idc`

6.7.2.6 `uint16_t VdpPictureInfoHEVC::column_width_minus1[20]`

Only need to set 0..num\_tile\_columns\_minus1. The struct definition reserves up to the maximum of 20. Invalid values are ignored.

6.7.2.7 `uint8_t VdpPictureInfoHEVC::constrained_intra_pred_flag`

6.7.2.8 `uint8_t VdpPictureInfoHEVC::cu_qp_delta_enabled_flag`

6.7.2.9 `int32_t VdpPictureInfoHEVC::CurrPicOrderCntVal`

Slice Decoding Process - Picture Order Count The value of PicOrderCntVal of the picture in the access unit containing the SEI message. The picture being decoded.

6.7.2.10 `uint8_t VdpPictureInfoHEVC::CurrRpsIdx`

See section 7.4.7.1 of the specification.

6.7.2.11 `uint8_t VdpPictureInfoHEVC::deblocking_filter_control_present_flag`

6.7.2.12 `uint8_t VdpPictureInfoHEVC::deblocking_filter_override_enabled_flag`

Only valid if deblocking\_filter\_control\_present\_flag is set. Ignored otherwise.

6.7.2.13 `uint8_t VdpPictureInfoHEVC::dependent_slice_segments_enabled_flag`

6.7.2.14 `uint8_t VdpPictureInfoHEVC::diff_cu_qp_delta_depth`

Only needed if cu\_qp\_delta\_enabled\_flag is set. Ignored otherwise.

6.7.2.15 `uint8_t VdpPictureInfoHEVC::entropy_coding_sync_enabled_flag`

6.7.2.16 `uint8_t VdpPictureInfoHEVC::IDRPicFlag`

Set to 1 if `nal_unit_type` is equal to `IDR_W_RADL` or `IDR_N_LP`. Set to zero otherwise.

6.7.2.17 `int8_t VdpPictureInfoHEVC::init_qp_minus26`

6.7.2.18 `uint8_t VdpPictureInfoHEVC::IsLongTerm[16]`

Array used to specify whether a particular `RefPic` is a long term reference. A value of "1" indicates a long-term reference.

6.7.2.19 `uint8_t VdpPictureInfoHEVC::lists_modification_present_flag`

6.7.2.20 `uint8_t VdpPictureInfoHEVC::log2_diff_max_min_luma_coding_block_size`

6.7.2.21 `uint8_t VdpPictureInfoHEVC::log2_diff_max_min_pcm_luma_coding_block_size`

Only needs to be set if `pcm_enabled_flag` is set. Ignored otherwise.

6.7.2.22 `uint8_t VdpPictureInfoHEVC::log2_diff_max_min_transform_block_size`

6.7.2.23 `uint8_t VdpPictureInfoHEVC::log2_max_pic_order_cnt_lsb_minus4`

6.7.2.24 `uint8_t VdpPictureInfoHEVC::log2_min_luma_coding_block_size_minus3`

6.7.2.25 `uint8_t VdpPictureInfoHEVC::log2_min_pcm_luma_coding_block_size_minus3`

Only needs to be set if `pcm_enabled_flag` is set. Ignored otherwise.

6.7.2.26 `uint8_t VdpPictureInfoHEVC::log2_min_transform_block_size_minus2`

6.7.2.27 `uint8_t VdpPictureInfoHEVC::log2_parallel_merge_level_minus2`

6.7.2.28 `uint8_t VdpPictureInfoHEVC::long_term_ref_pics_present_flag`

6.7.2.29 `uint8_t VdpPictureInfoHEVC::loop_filter_across_tiles_enabled_flag`

Only needed if `tiles_enabled_flag` is set. Invalid values are ignored.

6.7.2.30 `uint8_t VdpPictureInfoHEVC::max_transform_hierarchy_depth_inter`

6.7.2.31 `uint8_t VdpPictureInfoHEVC::max_transform_hierarchy_depth_intra`

6.7.2.32 `uint8_t VdpPictureInfoHEVC::num_extra_slice_header_bits`

6.7.2.33 `uint8_t VdpPictureInfoHEVC::num_long_term_ref_pics_sps`

Only needed if `long_term_ref_pics_present_flag` is set. Ignored otherwise.

6.7.2.34 `uint8_t VdpPictureInfoHEVC::num_ref_idx_l0_default_active_minus1`

6.7.2.35 `uint8_t VdpPictureInfoHEVC::num_ref_idx_l1_default_active_minus1`

6.7.2.36 `uint8_t VdpPictureInfoHEVC::num_short_term_ref_pic_sets`

Per spec, when zero, assume `short_term_ref_pic_set_sps_flag` is also zero.

6.7.2.37 `uint8_t VdpPictureInfoHEVC::num_tile_columns_minus1`

Only valid if `tiles_enabled_flag` is set. Ignored otherwise.

6.7.2.38 `uint8_t VdpPictureInfoHEVC::num_tile_rows_minus1`

Only valid if `tiles_enabled_flag` is set. Ignored otherwise.

6.7.2.39 `uint32_t VdpPictureInfoHEVC::NumDeltaPocsOfRefRpsIdx`

Corresponds to specification field, `NumDeltaPocs[RefRpsIdx]`. Only applicable when `short_term_ref_pic_set_sps_flag == 0`. Implementations will ignore this value in other cases. See 7.4.8.

6.7.2.40 `uint32_t VdpPictureInfoHEVC::NumLongTermPictureSliceHeaderBits`

Second, VDP AU requires the number of bits used for long term reference pictures in the `slice_segment_header`. This is equal to the number of bits used for the contents of the block beginning with "if(long\_term\_ref\_pics\_present\_flag)".

6.7.2.41 `uint8_t VdpPictureInfoHEVC::NumPocLtCurr`

Copy of specification field, see Section 8.3.2 of the H.265/HEVC Specification.

**6.7.2.42 uint8\_t VdpPictureInfoHEVC::NumPocStCurrAfter**

Copy of specification field, see Section 8.3.2 of the H.265/HEVC Specification.

**6.7.2.43 uint8\_t VdpPictureInfoHEVC::NumPocStCurrBefore**

Copy of specification field, see Section 8.3.2 of the H.265/HEVC Specification.

**6.7.2.44 uint32\_t VdpPictureInfoHEVC::NumPocTotalCurr**

See section 7.4.7.2 of the specification.

**6.7.2.45 uint32\_t VdpPictureInfoHEVC::NumShortTermPictureSliceHeaderBits**

Section 7.6.3.1 of the H.265/HEVC Specification defines the syntax of the slice\_segment\_header. This header contains information that some VDP AU implementations may choose to skip. The VDP AU API requires client applications to track the number of bits used in the slice header for structures associated with short term and long term reference pictures. First, VDP AU requires the number of bits used by the short\_term\_ref\_pic\_set array in the slice\_segment\_header.

**6.7.2.46 uint8\_t VdpPictureInfoHEVC::output\_flag\_present\_flag****6.7.2.47 uint8\_t VdpPictureInfoHEVC::pcm\_enabled\_flag****6.7.2.48 uint8\_t VdpPictureInfoHEVC::pcm\_loop\_filter\_disabled\_flag**

Only needs to be set if pcm\_enabled\_flag is set. Ignored otherwise.

**6.7.2.49 uint8\_t VdpPictureInfoHEVC::pcm\_sample\_bit\_depth\_chroma\_minus1**

Only needs to be set if pcm\_enabled\_flag is set. Ignored otherwise.

**6.7.2.50 uint8\_t VdpPictureInfoHEVC::pcm\_sample\_bit\_depth\_luma\_minus1**

Only needs to be set if pcm\_enabled\_flag is set. Ignored otherwise.

**6.7.2.51 uint32\_t VdpPictureInfoHEVC::pic\_height\_in\_luma\_samples****6.7.2.52 uint32\_t VdpPictureInfoHEVC::pic\_width\_in\_luma\_samples****6.7.2.53 int32\_t VdpPictureInfoHEVC::PicOrderCntVal[16]**

Array of picture order counts. These correspond to positions in the RefPics array.

6.7.2.54 `int8_t VdpPictureInfoHEVC::pps_beta_offset_div2`

Only valid if `deblocking_filter_control_present_flag` is set and `pps_deblocking_filter_disabled_flag` is not set. Ignored otherwise.

6.7.2.55 `int8_t VdpPictureInfoHEVC::pps_cb_qp_offset`6.7.2.56 `int8_t VdpPictureInfoHEVC::pps_cr_qp_offset`6.7.2.57 `uint8_t VdpPictureInfoHEVC::pps_deblocking_filter_disabled_flag`

Only valid if `deblocking_filter_control_present_flag` is set. Ignored otherwise.

6.7.2.58 `uint8_t VdpPictureInfoHEVC::pps_loop_filter_across_slices_enabled_flag`6.7.2.59 `uint8_t VdpPictureInfoHEVC::pps_slice_chroma_qp_offsets_present_flag`6.7.2.60 `int8_t VdpPictureInfoHEVC::pps_tc_offset_div2`

Only valid if `deblocking_filter_control_present_flag` is set and `pps_deblocking_filter_disabled_flag` is not set. Ignored otherwise.

6.7.2.61 `uint8_t VdpPictureInfoHEVC::RAPPicFlag`

Set to 1 if `nal_unit_type` in the range of `BLA_W_LP` to `RSV_IRAP_VCL23`, inclusive. Set to zero otherwise.

6.7.2.62 `VdpVideoSurface VdpPictureInfoHEVC::RefPics[16]`

Slice Decoding Process - Reference Picture Sets Array of video reference surfaces. Set any unused positions to `VDP_INVALID_HANDLE`.

6.7.2.63 `uint8_t VdpPictureInfoHEVC::RefPicSetLtCurr[8]`

Reference Picture Set list, one of the long-term RPS. These correspond to positions in the `RefPics` array.

6.7.2.64 `uint8_t VdpPictureInfoHEVC::RefPicSetStCurrAfter[8]`

Reference Picture Set list, one of the short-term RPS. These correspond to positions in the `RefPics` array.

6.7.2.65 `uint8_t VdpPictureInfoHEVC::RefPicSetStCurrBefore[8]`

Reference Picture Set list, one of the short-term RPS. These correspond to positions in the `RefPics` array.

#### 6.7.2.66 `uint16_t VdpPictureInfoHEVC::row_height_minus1[22]`

Only need to set 0..num\_tile\_rows\_minus1. The struct definition reserves up to the maximum of 22. Invalid values are ignored.

#### 6.7.2.67 `uint8_t VdpPictureInfoHEVC::sample_adaptive_offset_enabled_flag`

#### 6.7.2.68 `uint8_t VdpPictureInfoHEVC::scaling_list_enabled_flag`

#### 6.7.2.69 `uint8_t VdpPictureInfoHEVC::ScalingList16x16[6][64]`

Scaling List for 16x16 quantization matrix, indexed as `ScalingList16x16[matrixId][i]`.

#### 6.7.2.70 `uint8_t VdpPictureInfoHEVC::ScalingList32x32[2][64]`

Scaling List for 32x32 quantization matrix, indexed as `ScalingList32x32[matrixId][i]`.

#### 6.7.2.71 `uint8_t VdpPictureInfoHEVC::ScalingList4x4[6][16]`

Scaling lists, in diagonal order, to be used for this frame. Scaling List for 4x4 quantization matrix, indexed as `ScalingList4x4[matrixId][i]`.

#### 6.7.2.72 `uint8_t VdpPictureInfoHEVC::ScalingList8x8[6][64]`

Scaling List for 8x8 quantization matrix, indexed as `ScalingList8x8[matrixId][i]`.

#### 6.7.2.73 `uint8_t VdpPictureInfoHEVC::ScalingListDCCoeff16x16[6]`

Scaling List DC Coefficients for 16x16, indexed as `ScalingListDCCoeff16x16[matrixId]`.

#### 6.7.2.74 `uint8_t VdpPictureInfoHEVC::ScalingListDCCoeff32x32[2]`

Scaling List DC Coefficients for 32x32, indexed as `ScalingListDCCoeff32x32[matrixId]`.

#### 6.7.2.75 `uint8_t VdpPictureInfoHEVC::separate_colour_plane_flag`

Only valid if `chroma_format_idc == 3`. Ignored otherwise.

#### 6.7.2.76 `uint8_t VdpPictureInfoHEVC::sign_data_hiding_enabled_flag`

#### 6.7.2.77 `uint8_t VdpPictureInfoHEVC::slice_segment_header_extension_present_flag`

#### 6.7.2.78 `uint8_t VdpPictureInfoHEVC::sps_max_dec_pic_buffering_minus1`

Provides the value corresponding to the `nuh_temporal_id` of the frame to be decoded.



- 6.7.2.79 `uint8_t VdpPictureInfoHEVC::sps_temporal_mvp_enabled_flag`
- 6.7.2.80 `uint8_t VdpPictureInfoHEVC::strong_intra_smoothing_enabled_flag`
- 6.7.2.81 `uint8_t VdpPictureInfoHEVC::tiles_enabled_flag`
- 6.7.2.82 `uint8_t VdpPictureInfoHEVC::transform_skip_enabled_flag`
- 6.7.2.83 `uint8_t VdpPictureInfoHEVC::transquant_bypass_enabled_flag`
- 6.7.2.84 `uint8_t VdpPictureInfoHEVC::uniform_spacing_flag`

Only valid if `tiles_enabled_flag` is set. Ignored otherwise.

- 6.7.2.85 `uint8_t VdpPictureInfoHEVC::weighted_bipred_flag`
- 6.7.2.86 `uint8_t VdpPictureInfoHEVC::weighted_pred_flag`

The documentation for this struct was generated from the following file:

- [vdpau/vdpau.h](#)

## 6.8 VdpPictureInfoMPEG1Or2 Struct Reference

Picture parameter information for an MPEG 1 or MPEG 2 picture.

```
#include <vdpau.h>
```

### Data Fields

- [VdpVideoSurface forward\\_reference](#)
- [VdpVideoSurface backward\\_reference](#)
- `uint32_t slice_count`

### MPEG bitstream

*Copies of the MPEG bitstream fields.*

- `uint8_t picture_structure`
- `uint8_t picture_coding_type`
- `uint8_t intra_dc_precision`
- `uint8_t frame_pred_frame_dct`
- `uint8_t concealment_motion_vectors`
- `uint8_t intra_vlc_format`
- `uint8_t alternate_scan`
- `uint8_t q_scale_type`
- `uint8_t top_field_first`
- `uint8_t full_pel_forward_vector`
- `uint8_t full_pel_backward_vector`
- `uint8_t f_code [2][2]`
- `uint8_t intra_quantizer_matrix [64]`
- `uint8_t non_intra_quantizer_matrix [64]`

### 6.8.1 Detailed Description

Picture parameter information for an MPEG 1 or MPEG 2 picture.

Note: References to bitstream fields below may refer to data literally parsed from the bitstream, or derived from the bitstream using a mechanism described in the specification.

### 6.8.2 Field Documentation

6.8.2.1 `uint8_t VdpPictureInfoMPEG1Or2::alternate_scan`

6.8.2.2 `VdpVideoSurface VdpPictureInfoMPEG1Or2::backward_reference`

Reference used by B frames. Set to `VDP_INVALID_HANDLE` when not used.

6.8.2.3 `uint8_t VdpPictureInfoMPEG1Or2::concealment_motion_vectors`

6.8.2.4 `uint8_t VdpPictureInfoMPEG1Or2::f_code[2][2]`

For MPEG-1, fill both horizontal and vertical entries.

6.8.2.5 `VdpVideoSurface VdpPictureInfoMPEG1Or2::forward_reference`

Reference used by B and P frames. Set to `VDP_INVALID_HANDLE` when not used.

6.8.2.6 `uint8_t VdpPictureInfoMPEG1Or2::frame_pred_frame_dct`

6.8.2.7 `uint8_t VdpPictureInfoMPEG1Or2::full_pel_backward_vector`

MPEG-1 only. For MPEG-2, set to 0.

6.8.2.8 `uint8_t VdpPictureInfoMPEG1Or2::full_pel_forward_vector`

MPEG-1 only. For MPEG-2, set to 0.

6.8.2.9 `uint8_t VdpPictureInfoMPEG1Or2::intra_dc_precision`

6.8.2.10 `uint8_t VdpPictureInfoMPEG1Or2::intra_quantizer_matrix[64]`

Convert to raster order.

6.8.2.11 `uint8_t VdpPictureInfoMPEG1Or2::intra_vlc_format`

6.8.2.12 `uint8_t VdpPictureInfoMPEG1Or2::non_intra_quantizer_matrix[64]`

Convert to raster order.

6.8.2.13 `uint8_t VdpPictureInfoMPEG1Or2::picture_coding_type`

6.8.2.14 `uint8_t VdpPictureInfoMPEG1Or2::picture_structure`

6.8.2.15 `uint8_t VdpPictureInfoMPEG1Or2::q_scale_type`

6.8.2.16 `uint32_t VdpPictureInfoMPEG1Or2::slice_count`

Number of slices in the bitstream provided.

6.8.2.17 `uint8_t VdpPictureInfoMPEG1Or2::top_field_first`

The documentation for this struct was generated from the following file:

- [vdpau/vdpau.h](#)

## 6.9 VdpPictureInfoMPEG4Part2 Struct Reference

Picture parameter information for an MPEG-4 Part 2 picture.

```
#include <vdpau.h>
```

### Data Fields

- [VdpVideoSurface forward\\_reference](#)
- [VdpVideoSurface backward\\_reference](#)

### MPEG 4 part 2 bitstream

*Copies of the MPEG 4 part 2 bitstream fields.*

- `int32_t trd [2]`
- `int32_t trb [2]`
- `uint16_t vop_time_increment_resolution`
- `uint8_t vop_coding_type`
- `uint8_t vop_fcode_forward`
- `uint8_t vop_fcode_backward`
- `uint8_t resync_marker_disable`
- `uint8_t interlaced`
- `uint8_t quant_type`
- `uint8_t quarter_sample`
- `uint8_t short_video_header`
- `uint8_t rounding_control`
- `uint8_t alternate_vertical_scan_flag`
- `uint8_t top_field_first`
- `uint8_t intra_quantizer_matrix [64]`
- `uint8_t non_intra_quantizer_matrix [64]`

### 6.9.1 Detailed Description

Picture parameter information for an MPEG-4 Part 2 picture.

Note: References to bitstream fields below may refer to data literally parsed from the bitstream, or derived from the bitstream using a mechanism described in the specification.

### 6.9.2 Field Documentation

6.9.2.1 `uint8_t VdpPictureInfoMPEG4Part2::alternate_vertical_scan_flag`

6.9.2.2 `VdpVideoSurface VdpPictureInfoMPEG4Part2::backward_reference`

Reference used by B frames. Set to `VDP_INVALID_HANDLE` when not used.

6.9.2.3 `VdpVideoSurface VdpPictureInfoMPEG4Part2::forward_reference`

Reference used by B and P frames. Set to `VDP_INVALID_HANDLE` when not used.

6.9.2.4 `uint8_t VdpPictureInfoMPEG4Part2::interlaced`

6.9.2.5 `uint8_t VdpPictureInfoMPEG4Part2::intra_quantizer_matrix[64]`

6.9.2.6 `uint8_t VdpPictureInfoMPEG4Part2::non_intra_quantizer_matrix[64]`

6.9.2.7 `uint8_t VdpPictureInfoMPEG4Part2::quant_type`

6.9.2.8 `uint8_t VdpPictureInfoMPEG4Part2::quarter_sample`

6.9.2.9 `uint8_t VdpPictureInfoMPEG4Part2::resync_marker_disable`

6.9.2.10 `uint8_t VdpPictureInfoMPEG4Part2::rounding_control`

Derived from `vop_rounding_type` bitstream field.

6.9.2.11 `uint8_t VdpPictureInfoMPEG4Part2::short_video_header`

6.9.2.12 `uint8_t VdpPictureInfoMPEG4Part2::top_field_first`

6.9.2.13 `int32_t VdpPictureInfoMPEG4Part2::trb[2]`

6.9.2.14 `int32_t VdpPictureInfoMPEG4Part2::trd[2]`

6.9.2.15 `uint8_t VdpPictureInfoMPEG4Part2::vop_coding_type`

6.9.2.16 `uint8_t VdpPictureInfoMPEG4Part2::vop_fcode_backward`

6.9.2.17 `uint8_t VdpPictureInfoMPEG4Part2::vop_fcode_forward`

6.9.2.18 `uint16_t VdpPictureInfoMPEG4Part2::vop_time_increment_resolution`

The documentation for this struct was generated from the following file:

- `vdpu/vdpau.h`

## 6.10 VdpPictureInfoVC1 Struct Reference

Picture parameter information for a VC1 picture.

```
#include <vdpu.h>
```

### Data Fields

- [VdpVideoSurface](#) forward\_reference
- [VdpVideoSurface](#) backward\_reference
- uint32\_t slice\_count
- uint8\_t picture\_type
- uint8\_t frame\_coding\_mode
- uint8\_t deblockEnable
- uint8\_t pquant

### VC-1 bitstream

*Copies of the VC-1 bitstream fields.*

- uint8\_t postprocflag
- uint8\_t pulldown
- uint8\_t interlace
- uint8\_t tfcntrflag
- uint8\_t finterpfalg
- uint8\_t psf
- uint8\_t dquant
- uint8\_t panscan\_flag
- uint8\_t reldist\_flag
- uint8\_t quantizer
- uint8\_t extended\_mv
- uint8\_t extended\_dmv
- uint8\_t overlap
- uint8\_t vstransform
- uint8\_t loopfilter
- uint8\_t fastuvmc
- uint8\_t range\_mapy\_flag
- uint8\_t range\_mapy
- uint8\_t range\_mapuv\_flag
- uint8\_t range\_mapuv
- uint8\_t multires
- uint8\_t syncmarker
- uint8\_t rangered
- uint8\_t maxbframes

### 6.10.1 Detailed Description

Picture parameter information for a VC1 picture.

Note: References to bitstream fields below may refer to data literally parsed from the bitstream, or derived from the bitstream using a mechanism described in the specification.

### 6.10.2 Field Documentation

#### 6.10.2.1 VdpVideoSurface VdpPictureInfoVC1::backward\_reference

Reference used by B frames. Set to VDP\_INVALID\_HANDLE when not used.

#### 6.10.2.2 `uint8_t VdpPictureInfoVC1::deblockEnable`

Out-of-loop deblocking enable. Bit 0 of POSTPROC from VC-1 7.1.1.27 Note that bit 1 of POSTPROC (dering enable) should not be included.

#### 6.10.2.3 `uint8_t VdpPictureInfoVC1::dquant`

See VC-1 6.2.8.

#### 6.10.2.4 `uint8_t VdpPictureInfoVC1::extended_dmv`

See VC-1 6.2.14.

#### 6.10.2.5 `uint8_t VdpPictureInfoVC1::extended_mv`

See VC-1 6.2.7.

#### 6.10.2.6 `uint8_t VdpPictureInfoVC1::fastuvmc`

See VC-1 6.2.6.

#### 6.10.2.7 `uint8_t VdpPictureInfoVC1::finterpflag`

See VC-1 6.1.11.

#### 6.10.2.8 `VdpVideoSurface VdpPictureInfoVC1::forward_reference`

Reference used by B and P frames. Set to `VDP_INVALID_HANDLE` when not used.

#### 6.10.2.9 `uint8_t VdpPictureInfoVC1::frame_coding_mode`

Progressive=0, Frame-interlace=2, Field-interlace=3; see VC-1 7.1.1.15.

#### 6.10.2.10 `uint8_t VdpPictureInfoVC1::interlace`

See VC-1 6.1.9.

#### 6.10.2.11 `uint8_t VdpPictureInfoVC1::loopfilter`

See VC-1 6.2.5.

**6.10.2.12    uint8\_t VdpPictureInfoVC1::maxbframes**

See VC-1 J.1.17. Only used by simple and main profiles.

**6.10.2.13    uint8\_t VdpPictureInfoVC1::multires**

See VC-1 J.1.10. Only used by simple and main profiles.

**6.10.2.14    uint8\_t VdpPictureInfoVC1::overlap**

See VC-1 6.2.10.

**6.10.2.15    uint8\_t VdpPictureInfoVC1::panscan\_flag**

See VC-1 6.2.3.

**6.10.2.16    uint8\_t VdpPictureInfoVC1::picture\_type**

I=0, P=1, B=3, BI=4 from 7.1.1.4.

**6.10.2.17    uint8\_t VdpPictureInfoVC1::postprocflag**

See VC-1 6.1.5.

**6.10.2.18    uint8\_t VdpPictureInfoVC1::pquant**

Parameter used by VC-1 Annex H deblocking algorithm. Note that VDP AU implementations may choose which deblocking algorithm to use. See VC-1 7.1.1.6

**6.10.2.19    uint8\_t VdpPictureInfoVC1::psf**

See VC-1 6.1.3.

**6.10.2.20    uint8\_t VdpPictureInfoVC1::pulldown**

See VC-1 6.1.8.

**6.10.2.21    uint8\_t VdpPictureInfoVC1::quantizer**

See VC-1 6.2.11.

6.10.2.22 `uint8_t VdpPictureInfoVC1::range_mapuv`

6.10.2.23 `uint8_t VdpPictureInfoVC1::range_mapuv_flag`

See VC-1 6.2.16.

6.10.2.24 `uint8_t VdpPictureInfoVC1::range_mapy`

6.10.2.25 `uint8_t VdpPictureInfoVC1::range_mapy_flag`

See VC-1 6.12.15.

6.10.2.26 `uint8_t VdpPictureInfoVC1::rangered`

VC-1 SP/MP range reduction control. Only used by simple and main profiles. Bit 0: Copy of ranged VC-1 bitstream field; See VC-1 J.1.17. Bit 1: Copy of ranged VC-1 bitstream field; See VC-1 7.1.13.

6.10.2.27 `uint8_t VdpPictureInfoVC1::refdist_flag`

See VC-1 6.2.4.

6.10.2.28 `uint32_t VdpPictureInfoVC1::slice_count`

Number of slices in the bitstream provided.

6.10.2.29 `uint8_t VdpPictureInfoVC1::syncmarker`

See VC-1 J.1.16. Only used by simple and main profiles.

6.10.2.30 `uint8_t VdpPictureInfoVC1::tfcntrflag`

See VC-1 6.1.10.

6.10.2.31 `uint8_t VdpPictureInfoVC1::vstransform`

See VC-1 6.2.9.

The documentation for this struct was generated from the following file:

- [vdpau/vdpau.h](#)



## 6.11 VdpPoint Struct Reference

A location within a surface.

```
#include <vdpau.h>
```

### Data Fields

- `uint32_t x`
- `uint32_t y`

### 6.11.1 Detailed Description

A location within a surface.

The VDPAU co-ordinate system has its origin at the top-left of a surface, with x and y components increasing right and down.

### 6.11.2 Field Documentation

#### 6.11.2.1 `uint32_t VdpPoint::x`

X co-ordinate.

#### 6.11.2.2 `uint32_t VdpPoint::y`

Y co-ordinate.

The documentation for this struct was generated from the following file:

- `vdpau/vdpau.h`

## 6.12 VdpProcamp Struct Reference

Procamp operation parameterization data.

```
#include <vdpau.h>
```

### Data Fields

- `uint32_t struct_version`
- `float brightness`
- `float contrast`
- `float saturation`
- `float hue`

### 6.12.1 Detailed Description

Procamp operation parameterization data.

When performing a color space conversion operation, various adjustments can be performed at the same time, such as brightness and contrast. This structure defines the level of adjustments to make.

### 6.12.2 Field Documentation

#### 6.12.2.1 float VdpProcamp::brightness

Brightness adjustment amount. A value clamped between -1.0 and 1.0. 0.0 represents no modification.

#### 6.12.2.2 float VdpProcamp::contrast

Contrast adjustment amount. A value clamped between 0.0 and 10.0. 1.0 represents no modification.

#### 6.12.2.3 float VdpProcamp::hue

Hue adjustment amount. A value clamped between -PI and PI. 0.0 represents no modification.

#### 6.12.2.4 float VdpProcamp::saturation

Saturation adjustment amount. A value clamped between 0.0 and 10.0. 1.0 represents no modification.

#### 6.12.2.5 uint32\_t VdpProcamp::struct\_version

This field must be filled with VDP\_PROPCAMP\_VERSION

The documentation for this struct was generated from the following file:

- [vdpau/vdpau.h](#)

## 6.13 VdpRect Struct Reference

A rectangular region of a surface.

```
#include <vdpau.h>
```

### Data Fields

- uint32\_t [x0](#)
- uint32\_t [y0](#)
- uint32\_t [x1](#)
- uint32\_t [y1](#)

### 6.13.1 Detailed Description

A rectangular region of a surface.

The co-ordinates are top-left inclusive, bottom-right exclusive.

The VDP AU co-ordinate system has its origin at the top-left of a surface, with x and y components increasing right and down.

### 6.13.2 Field Documentation

#### 6.13.2.1 `uint32_t VdpRect::x0`

Left X co-ordinate. Inclusive.

#### 6.13.2.2 `uint32_t VdpRect::x1`

Right X co-ordinate. Exclusive.

#### 6.13.2.3 `uint32_t VdpRect::y0`

Top Y co-ordinate. Inclusive.

#### 6.13.2.4 `uint32_t VdpRect::y1`

Bottom Y co-ordinate. Exclusive.

The documentation for this struct was generated from the following file:

- [vdpau/vdpau.h](#)

## 6.14 VdpReferenceFrameH264 Struct Reference

Information about an H.264 reference frame.

```
#include <vdpau.h>
```

### Data Fields

- [VdpVideoSurface surface](#)
- [VdpBool is\\_long\\_term](#)
- [VdpBool top\\_is\\_reference](#)
- [VdpBool bottom\\_is\\_reference](#)
- [int32\\_t field\\_order\\_cnt](#) [2]
- [uint16\\_t frame\\_idx](#)

### 6.14.1 Detailed Description

Information about an H.264 reference frame.

Note: References to bitstream fields below may refer to data literally parsed from the bitstream, or derived from the bitstream using a mechanism described in the specification.

### 6.14.2 Field Documentation

#### 6.14.2.1 `VdpBool VdpReferenceFrameH264::bottom_is_reference`

Is the bottom field used as a reference. Set to `VDP_FALSE` for unused entries.

#### 6.14.2.2 `int32_t VdpReferenceFrameH264::field_order_cnt[2]`

[0]: top, [1]: bottom

#### 6.14.2.3 `uint16_t VdpReferenceFrameH264::frame_idx`

Copy of the H.264 bitstream field: `frame_num` from `slice_header` for short-term references, `LongTermPicNum` from decoding algorithm for long-term references.

#### 6.14.2.4 `VdpBool VdpReferenceFrameH264::is_long_term`

Is this a long term reference (else short term).

#### 6.14.2.5 `VdpVideoSurface VdpReferenceFrameH264::surface`

The surface that contains the reference image. Set to `VDP_INVALID_HANDLE` for unused entries.

#### 6.14.2.6 `VdpBool VdpReferenceFrameH264::top_is_reference`

Is the top field used as a reference. Set to `VDP_FALSE` for unused entries.

The documentation for this struct was generated from the following file:

- [vdpau/vdpau.h](#)

## Chapter 7

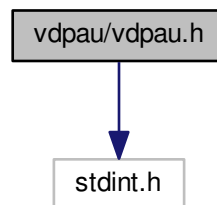
# File Documentation

### 7.1 vdpau/vdpau.h File Reference

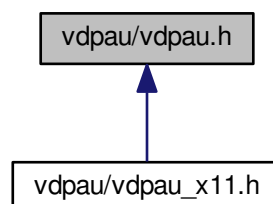
The Core API.

```
#include <stdint.h>
```

Include dependency graph for vdpau.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [VdpPoint](#)  
*A location within a surface.*
- struct [VdpRect](#)  
*A rectangular region of a surface.*
- struct [VdpColor](#)
- struct [VdpProcamp](#)  
*Procamp operation parameterization data.*
- struct [VdpOutputSurfaceRenderBlendState](#)  
*Complete blending operation definition.*
- struct [VdpBitstreamBuffer](#)  
*Application data buffer containing compressed video data.*
- struct [VdpPictureInfoMPEG1Or2](#)  
*Picture parameter information for an MPEG 1 or MPEG 2 picture.*
- struct [VdpReferenceFrameH264](#)  
*Information about an H.264 reference frame.*
- struct [VdpPictureInfoH264](#)  
*Picture parameter information for an H.264 picture.*
- struct [VdpPictureInfoH264Predictive](#)  
*Picture parameter information for an H.264 Hi444PP picture.*
- struct [VdpPictureInfoVC1](#)  
*Picture parameter information for a VC1 picture.*
- struct [VdpPictureInfoMPEG4Part2](#)  
*Picture parameter information for an MPEG-4 Part 2 picture.*
- struct [VdpPictureInfoHEVC](#)  
*Picture parameter information for an H.265/HEVC picture.*
- struct [VdpLayer](#)  
*Definition of an additional [VdpOutputSurface](#) layer in the compositing model.*

## Macros

- #define [VDP\\_TRUE](#) 1  
*A true [VdpBool](#) value.*
- #define [VDP\\_FALSE](#) 0  
*A false [VdpBool](#) value.*
- #define [VDP\\_INVALID\\_HANDLE](#) 0xffffffffU  
*An invalid object handle value.*
- #define [VDP\\_CHROMA\\_TYPE\\_420](#)  
*4:2:0 chroma format.*
- #define [VDP\\_CHROMA\\_TYPE\\_422](#)  
*4:2:2 chroma format.*
- #define [VDP\\_CHROMA\\_TYPE\\_444](#)  
*4:4:4 chroma format.*
- #define [VDP\\_YCBCR\\_FORMAT\\_NV12](#)  
*The "NV12" YCbCr surface format.*
- #define [VDP\\_YCBCR\\_FORMAT\\_YV12](#)  
*The "YV12" YCbCr surface format.*
- #define [VDP\\_YCBCR\\_FORMAT\\_UYVY](#)  
*The "UYVY" YCbCr surface format.*

- #define `VDP_YCBCR_FORMAT_YUYV`  
*The "YUYV" YCbCr surface format.*
- #define `VDP_YCBCR_FORMAT_Y8U8V8A8`  
*A packed YCbCr format.*
- #define `VDP_YCBCR_FORMAT_V8U8Y8A8`  
*A packed YCbCr format.*
- #define `VDP_RGBA_FORMAT_B8G8R8A8`  
*A packed RGB format.*
- #define `VDP_RGBA_FORMAT_R8G8B8A8`  
*A packed RGB format.*
- #define `VDP_RGBA_FORMAT_R10G10B10A2`  
*A packed RGB format.*
- #define `VDP_RGBA_FORMAT_B10G10R10A2`  
*A packed RGB format.*
- #define `VDP_RGBA_FORMAT_A8`  
*An alpha-only surface format.*
- #define `VDP_INDEXED_FORMAT_A4I4`  
*A 4-bit indexed format, with alpha.*
- #define `VDP_INDEXED_FORMAT_I4A4`  
*A 4-bit indexed format, with alpha.*
- #define `VDP_INDEXED_FORMAT_A8I8`  
*A 8-bit indexed format, with alpha.*
- #define `VDP_INDEXED_FORMAT_I8A8`  
*A 8-bit indexed format, with alpha.*
- #define `VDPAU_INTERFACE_VERSION` 1  
*The VDPAU interface version described by this header file.*
- #define `VDPAU_VERSION` 1  
*The VDPAU version described by this header file.*
- #define `VDP_PROCOMP_VERSION` 0
- #define `VDP_COLOR_STANDARD_ITUR_BT_601`  
*ITU-R BT.601.*
- #define `VDP_COLOR_STANDARD_ITUR_BT_709`  
*ITU-R BT.709.*
- #define `VDP_COLOR_STANDARD_SMPTE_240M`  
*SMPTE-240M.*
- #define `VDP_COLOR_TABLE_FORMAT_B8G8R8X8`  
*8-bit per component packed into 32-bits*
- #define `VDP_OUTPUT_SURFACE_RENDER_BLEND_STATE_VERSION` 0
- #define `VDP_OUTPUT_SURFACE_RENDER_ROTATE_0`  
*Do not rotate source\_surface prior to compositing.*
- #define `VDP_OUTPUT_SURFACE_RENDER_ROTATE_90`  
*Rotate source\_surface 90 degrees clockwise prior to compositing.*
- #define `VDP_OUTPUT_SURFACE_RENDER_ROTATE_180`  
*Rotate source\_surface 180 degrees prior to compositing.*
- #define `VDP_OUTPUT_SURFACE_RENDER_ROTATE_270`  
*Rotate source\_surface 270 degrees clockwise prior to compositing.*
- #define `VDP_OUTPUT_SURFACE_RENDER_COLOR_PER_VERTEX`  
*A separate color is used for each vertex of the smooth-shaded quad. Hence, colors array contains 4 elements rather than 1. See description of colors array.*
- #define `VDP_DECODER_PROFILE_MPEG1`
- #define `VDP_DECODER_PROFILE_MPEG2_SIMPLE`

- #define VDP\_DECODER\_PROFILE\_MPEG2\_MAIN
- #define VDP\_DECODER\_PROFILE\_H264\_BASELINE
  - MPEG 4 part 10 == H.264 == AVC.*
- #define VDP\_DECODER\_PROFILE\_H264\_MAIN
- #define VDP\_DECODER\_PROFILE\_H264\_HIGH
- #define VDP\_DECODER\_PROFILE\_VC1\_SIMPLE
- #define VDP\_DECODER\_PROFILE\_VC1\_MAIN
- #define VDP\_DECODER\_PROFILE\_VC1\_ADVANCED
- #define VDP\_DECODER\_PROFILE\_MPEG4\_PART2\_SP
- #define VDP\_DECODER\_PROFILE\_MPEG4\_PART2\_ASP
- #define VDP\_DECODER\_PROFILE\_DIVX4\_QMOBILE
- #define VDP\_DECODER\_PROFILE\_DIVX4\_MOBILE
- #define VDP\_DECODER\_PROFILE\_DIVX4\_HOME\_THEATER
- #define VDP\_DECODER\_PROFILE\_DIVX4\_HD\_1080P
- #define VDP\_DECODER\_PROFILE\_DIVX5\_QMOBILE
- #define VDP\_DECODER\_PROFILE\_DIVX5\_MOBILE
- #define VDP\_DECODER\_PROFILE\_DIVX5\_HOME\_THEATER
- #define VDP\_DECODER\_PROFILE\_DIVX5\_HD\_1080P
- #define VDP\_DECODER\_PROFILE\_H264\_CONstrained\_BASELINE
- #define VDP\_DECODER\_PROFILE\_H264\_EXTENDED
- #define VDP\_DECODER\_PROFILE\_H264\_PROGRESSIVE\_HIGH
- #define VDP\_DECODER\_PROFILE\_H264\_CONstrained\_HIGH
- #define VDP\_DECODER\_PROFILE\_H264\_HIGH\_444\_PREDICTIVE
  - Support for 8 bit depth only.*
- #define VDP\_DECODER\_PROFILE\_HEVC\_MAIN
  - MPEG-H Part 2 == H.265 == HEVC.*
- #define VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_10
- #define VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_STILL
- #define VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_12
- #define VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_444
- #define VDP\_DECODER\_LEVEL\_MPEG1\_NA
- #define VDP\_DECODER\_LEVEL\_MPEG2\_LL
- #define VDP\_DECODER\_LEVEL\_MPEG2\_ML
- #define VDP\_DECODER\_LEVEL\_MPEG2\_HL14
- #define VDP\_DECODER\_LEVEL\_MPEG2\_HL
- #define VDP\_DECODER\_LEVEL\_H264\_1
- #define VDP\_DECODER\_LEVEL\_H264\_1b
- #define VDP\_DECODER\_LEVEL\_H264\_1\_1
- #define VDP\_DECODER\_LEVEL\_H264\_1\_2
- #define VDP\_DECODER\_LEVEL\_H264\_1\_3
- #define VDP\_DECODER\_LEVEL\_H264\_2
- #define VDP\_DECODER\_LEVEL\_H264\_2\_1
- #define VDP\_DECODER\_LEVEL\_H264\_2\_2
- #define VDP\_DECODER\_LEVEL\_H264\_3
- #define VDP\_DECODER\_LEVEL\_H264\_3\_1
- #define VDP\_DECODER\_LEVEL\_H264\_3\_2
- #define VDP\_DECODER\_LEVEL\_H264\_4
- #define VDP\_DECODER\_LEVEL\_H264\_4\_1
- #define VDP\_DECODER\_LEVEL\_H264\_4\_2
- #define VDP\_DECODER\_LEVEL\_H264\_5
- #define VDP\_DECODER\_LEVEL\_H264\_5\_1
- #define VDP\_DECODER\_LEVEL\_VC1\_SIMPLE\_LOW
- #define VDP\_DECODER\_LEVEL\_VC1\_SIMPLE\_MEDIUM
- #define VDP\_DECODER\_LEVEL\_VC1\_MAIN\_LOW



- #define `VDP_DECODER_LEVEL_VC1_MAIN_MEDIUM`
- #define `VDP_DECODER_LEVEL_VC1_MAIN_HIGH`
- #define `VDP_DECODER_LEVEL_VC1_ADVANCED_L0`
- #define `VDP_DECODER_LEVEL_VC1_ADVANCED_L1`
- #define `VDP_DECODER_LEVEL_VC1_ADVANCED_L2`
- #define `VDP_DECODER_LEVEL_VC1_ADVANCED_L3`
- #define `VDP_DECODER_LEVEL_VC1_ADVANCED_L4`
- #define `VDP_DECODER_LEVEL_MPEG4_PART2_SP_L0`
- #define `VDP_DECODER_LEVEL_MPEG4_PART2_SP_L1`
- #define `VDP_DECODER_LEVEL_MPEG4_PART2_SP_L2`
- #define `VDP_DECODER_LEVEL_MPEG4_PART2_SP_L3`
- #define `VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L0`
- #define `VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L1`
- #define `VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L2`
- #define `VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L3`
- #define `VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L4`
- #define `VDP_DECODER_LEVEL_MPEG4_PART2_ASP_L5`
- #define `VDP_DECODER_LEVEL_DIVX_NA`
- #define `VDP_DECODER_LEVEL_HEVC_1_30`
- #define `VDP_DECODER_LEVEL_HEVC_2`
- #define `VDP_DECODER_LEVEL_HEVC_2_1`
- #define `VDP_DECODER_LEVEL_HEVC_3`
- #define `VDP_DECODER_LEVEL_HEVC_3_1`
- #define `VDP_DECODER_LEVEL_HEVC_4`
- #define `VDP_DECODER_LEVEL_HEVC_4_1`
- #define `VDP_DECODER_LEVEL_HEVC_5`
- #define `VDP_DECODER_LEVEL_HEVC_5_1`
- #define `VDP_DECODER_LEVEL_HEVC_5_2`
- #define `VDP_DECODER_LEVEL_HEVC_6`
- #define `VDP_DECODER_LEVEL_HEVC_6_1`
- #define `VDP_DECODER_LEVEL_HEVC_6_2`
- #define `VDP_BITSTREAM_BUFFER_VERSION` 0
- #define `VDP_VIDEO_MIXER_FEATURE_DEINTERLACE_TEMPORAL`  
*A VdpVideoMixerFeature.*
- #define `VDP_VIDEO_MIXER_FEATURE_DEINTERLACE_TEMPORAL_SPATIAL`  
*A VdpVideoMixerFeature.*
- #define `VDP_VIDEO_MIXER_FEATURE_INVERSE_TELECINE`  
*A VdpVideoMixerFeature.*
- #define `VDP_VIDEO_MIXER_FEATURE_NOISE_REDUCTION`  
*A VdpVideoMixerFeature.*
- #define `VDP_VIDEO_MIXER_FEATURE_SHARPNESS`  
*A VdpVideoMixerFeature.*
- #define `VDP_VIDEO_MIXER_FEATURE_LUMA_KEY`  
*A VdpVideoMixerFeature.*
- #define `VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L1`  
*A VdpVideoMixerFeature.*
- #define `VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L2`  
*A VdpVideoMixerFeature.*
- #define `VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L3`  
*A VdpVideoMixerFeature.*
- #define `VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L4`  
*A VdpVideoMixerFeature.*
- #define `VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L5`

- *A VdpVideoMixerFeature.*
- `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L6`  
*A VdpVideoMixerFeature.*
- `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L7`  
*A VdpVideoMixerFeature.*
- `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L8`  
*A VdpVideoMixerFeature.*
- `#define VDP_VIDEO_MIXER_FEATURE_HIGH_QUALITY_SCALING_L9`  
*A VdpVideoMixerFeature.*
- `#define VDP_VIDEO_MIXER_PARAMETER_VIDEO_SURFACE_WIDTH`  
*The exact width of input video surfaces.*
- `#define VDP_VIDEO_MIXER_PARAMETER_VIDEO_SURFACE_HEIGHT`  
*The exact height of input video surfaces.*
- `#define VDP_VIDEO_MIXER_PARAMETER_CHROMA_TYPE`  
*The chroma type of the input video surfaces the will process.*
- `#define VDP_VIDEO_MIXER_PARAMETER_LAYERS`  
*The number of auxiliary layers in the mixer's compositing model.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_BACKGROUND_COLOR`  
*The background color in the VdpVideoMixer's compositing model.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_CSC_MATRIX`  
*The color-space conversion matrix used by the VdpVideoMixer.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_NOISE_REDUCTION_LEVEL`  
*The amount of noise reduction algorithm to apply.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_SHARPNESS_LEVEL`  
*The amount of sharpening, or blurring, to apply.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_LUMA_KEY_MIN_LUMA`  
*The minimum luma value for the luma key algorithm.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_LUMA_KEY_MAX_LUMA`  
*The maximum luma value for the luma key algorithm.*
- `#define VDP_VIDEO_MIXER_ATTRIBUTE_SKIP_CHROMA_DEINTERLACE`  
*Whether de-interlacers should operate solely on luma, and bob chroma.*
- `#define VDP_LAYER_VERSION 0`
- `#define VDP_FUNC_ID_GET_ERROR_STRING`
- `#define VDP_FUNC_ID_GET_PROC_ADDRESS`
- `#define VDP_FUNC_ID_GET_API_VERSION`
- `#define VDP_FUNC_ID_GET_INFORMATION_STRING`
- `#define VDP_FUNC_ID_DEVICE_DESTROY`
- `#define VDP_FUNC_ID_GENERATE_CSC_MATRIX`
- `#define VDP_FUNC_ID_VIDEO_SURFACE_QUERY_CAPABILITIES`
- `#define VDP_FUNC_ID_VIDEO_SURFACE_QUERY_GET_PUT_BITS_Y_CB_CR_CAPABILITIES`
- `#define VDP_FUNC_ID_VIDEO_SURFACE_CREATE`
- `#define VDP_FUNC_ID_VIDEO_SURFACE_DESTROY`
- `#define VDP_FUNC_ID_VIDEO_SURFACE_GET_PARAMETERS`
- `#define VDP_FUNC_ID_VIDEO_SURFACE_GET_BITS_Y_CB_CR`
- `#define VDP_FUNC_ID_VIDEO_SURFACE_PUT_BITS_Y_CB_CR`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_QUERY_CAPABILITIES`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_QUERY_GET_PUT_BITS_NATIVE_CAPABILITIES`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_QUERY_PUT_BITS_INDEXED_CAPABILITIES`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_QUERY_PUT_BITS_Y_CB_CR_CAPABILITIES`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_CREATE`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_DESTROY`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_GET_PARAMETERS`

- `#define VDP_FUNC_ID_OUTPUT_SURFACE_GET_BITS_NATIVE`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_PUT_BITS_NATIVE`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_PUT_BITS_INDEXED`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_PUT_BITS_Y_CB_CR`
- `#define VDP_FUNC_ID_BITMAP_SURFACE_QUERY_CAPABILITIES`
- `#define VDP_FUNC_ID_BITMAP_SURFACE_CREATE`
- `#define VDP_FUNC_ID_BITMAP_SURFACE_DESTROY`
- `#define VDP_FUNC_ID_BITMAP_SURFACE_GET_PARAMETERS`
- `#define VDP_FUNC_ID_BITMAP_SURFACE_PUT_BITS_NATIVE`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_RENDER_OUTPUT_SURFACE`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_RENDER_BITMAP_SURFACE`
- `#define VDP_FUNC_ID_OUTPUT_SURFACE_RENDER_VIDEO_SURFACE_LUMA`
- `#define VDP_FUNC_ID_DECODER_QUERY_CAPABILITIES`
- `#define VDP_FUNC_ID_DECODER_CREATE`
- `#define VDP_FUNC_ID_DECODER_DESTROY`
- `#define VDP_FUNC_ID_DECODER_GET_PARAMETERS`
- `#define VDP_FUNC_ID_DECODER_RENDER`
- `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_FEATURE_SUPPORT`
- `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_PARAMETER_SUPPORT`
- `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_ATTRIBUTE_SUPPORT`
- `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_PARAMETER_VALUE_RANGE`
- `#define VDP_FUNC_ID_VIDEO_MIXER_QUERY_ATTRIBUTE_VALUE_RANGE`
- `#define VDP_FUNC_ID_VIDEO_MIXER_CREATE`
- `#define VDP_FUNC_ID_VIDEO_MIXER_SET_FEATURE_ENABLES`
- `#define VDP_FUNC_ID_VIDEO_MIXER_SET_ATTRIBUTE_VALUES`
- `#define VDP_FUNC_ID_VIDEO_MIXER_GET_FEATURE_SUPPORT`
- `#define VDP_FUNC_ID_VIDEO_MIXER_GET_FEATURE_ENABLES`
- `#define VDP_FUNC_ID_VIDEO_MIXER_GET_PARAMETER_VALUES`
- `#define VDP_FUNC_ID_VIDEO_MIXER_GET_ATTRIBUTE_VALUES`
- `#define VDP_FUNC_ID_VIDEO_MIXER_DESTROY`
- `#define VDP_FUNC_ID_VIDEO_MIXER_RENDER`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_TARGET_DESTROY`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_CREATE`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_DESTROY`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_SET_BACKGROUND_COLOR`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_GET_BACKGROUND_COLOR`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_GET_TIME`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_DISPLAY`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_BLOCK_UNTIL_SURFACE_IDLE`
- `#define VDP_FUNC_ID_PRESENTATION_QUEUE_QUERY_SURFACE_STATUS`
- `#define VDP_FUNC_ID_PREEMPTION_CALLBACK_REGISTER`
- `#define VDP_FUNC_ID_BASE_WINSYS 0x1000`

## Typedefs

- typedef int `VdpBool`  
*A boolean value, holding `VDP_TRUE` or `VDP_FALSE`.*
- typedef uint32\_t `VdpChromaType`  
*The set of all chroma formats for `VdpVideoSurfaces`.*
- typedef uint32\_t `VdpYCbCrFormat`  
*The set of all known YCbCr surface formats.*
- typedef uint32\_t `VdpRGBAFormat`  
*The set of all known RGB surface formats.*

- typedef uint32\_t [VdpIndexedFormat](#)  
*The set of all known indexed surface formats.*
- typedef char const \* [VdpGetErrorString](#)([VdpStatus](#) status)  
*Retrieve a string describing an error code.*
- typedef [VdpStatus](#) [VdpGetApiVersion](#)(uint32\_t \*api\_version)  
*Retrieve the VDPAU version implemented by the backend.*
- typedef [VdpStatus](#) [VdpGetInformationString](#)(char const \*\*information\_string)  
*Retrieve an implementation-specific string description of the implementation. This typically includes detailed version information.*
- typedef uint32\_t [VdpDevice](#)  
*An opaque handle representing a VdpDevice object.*
- typedef [VdpStatus](#) [VdpDeviceDestroy](#)([VdpDevice](#) device)  
*Destroy a VdpDevice.*
- typedef float [VdpCSCMatrix](#)[3][4]  
*Storage for a color space conversion matrix.*
- typedef uint32\_t [VdpColorStandard](#)  
*YCbCr color space specification.*
- typedef [VdpStatus](#) [VdpGenerateCSCMatrix](#)([VdpProcamp](#) \*procamp, [VdpColorStandard](#) standard, [VdpCSCMatrix](#) \*csc\_matrix)  
*Generate a color space conversion matrix.*
- typedef [VdpStatus](#) [VdpVideoSurfaceQueryCapabilities](#)([VdpDevice](#) device, [VdpChromaType](#) surface\_chroma\_type, [VdpBool](#) \*is\_supported, uint32\_t \*max\_width, uint32\_t \*max\_height)  
*Query the implementation's VdpVideoSurface capabilities.*
- typedef [VdpStatus](#) [VdpVideoSurfaceQueryGetPutBitsYCbCrCapabilities](#)([VdpDevice](#) device, [VdpChromaType](#) surface\_chroma\_type, [VdpYCbCrFormat](#) bits\_ycbcr\_format, [VdpBool](#) \*is\_supported)  
*Query the implementation's VdpVideoSurface GetBits/PutBits capabilities.*
- typedef uint32\_t [VdpVideoSurface](#)  
*An opaque handle representing a VdpVideoSurface object.*
- typedef [VdpStatus](#) [VdpVideoSurfaceCreate](#)([VdpDevice](#) device, [VdpChromaType](#) chroma\_type, uint32\_t width, uint32\_t height, [VdpVideoSurface](#) \*surface)  
*Create a VdpVideoSurface.*
- typedef [VdpStatus](#) [VdpVideoSurfaceDestroy](#)([VdpVideoSurface](#) surface)  
*Destroy a VdpVideoSurface.*
- typedef [VdpStatus](#) [VdpVideoSurfaceGetParameters](#)([VdpVideoSurface](#) surface, [VdpChromaType](#) \*chroma\_type, uint32\_t \*width, uint32\_t \*height)  
*Retrieve the parameters used to create a VdpVideoSurface.*
- typedef [VdpStatus](#) [VdpVideoSurfaceGetBitsYCbCr](#)([VdpVideoSurface](#) surface, [VdpYCbCrFormat](#) destination\_ycbcr\_format, void \*const \*destination\_data, uint32\_t const \*destination\_pitches)  
*Copy image data from a VdpVideoSurface to application memory in a specified YCbCr format.*
- typedef [VdpStatus](#) [VdpVideoSurfacePutBitsYCbCr](#)([VdpVideoSurface](#) surface, [VdpYCbCrFormat](#) source\_ycbcr\_format, void const \*const \*source\_data, uint32\_t const \*source\_pitches)  
*Copy image data from application memory in a specific YCbCr format to a VdpVideoSurface.*
- typedef uint32\_t [VdpColorTableFormat](#)  
*The set of all known color table formats, for use with [VdpOutputSurfacePutBitsIndexed](#).*
- typedef [VdpStatus](#) [VdpOutputSurfaceQueryCapabilities](#)([VdpDevice](#) device, [VdpRGBAFormat](#) surface\_rgba\_format, [VdpBool](#) \*is\_supported, uint32\_t \*max\_width, uint32\_t \*max\_height)  
*Query the implementation's VdpOutputSurface capabilities.*
- typedef [VdpStatus](#) [VdpOutputSurfaceQueryGetPutBitsNativeCapabilities](#)([VdpDevice](#) device, [VdpRGBAFormat](#) surface\_rgba\_format, [VdpBool](#) \*is\_supported)  
*Query the implementation's capability to perform a PutBits operation using application data matching the surface's format.*

- typedef [VdpStatus](#) [VdpOutputSurfaceQueryPutBitsIndexedCapabilities](#)([VdpDevice](#) device, [VdpRGBAFormat](#) surface\_rgba\_format, [VdpIndexedFormat](#) bits\_indexed\_format, [VdpColorTableFormat](#) color\_table\_format, [VdpBool](#) \*is\_supported)  
*Query the implementation's capability to perform a PutBits operation using application data in a specific indexed format.*
- typedef [VdpStatus](#) [VdpOutputSurfaceQueryPutBitsYCbCrCapabilities](#)([VdpDevice](#) device, [VdpRGBAFormat](#) surface\_rgba\_format, [VdpYCbCrFormat](#) bits\_ycbcr\_format, [VdpBool](#) \*is\_supported)  
*Query the implementation's capability to perform a PutBits operation using application data in a specific YCbCr/YUB format.*
- typedef uint32\_t [VdpOutputSurface](#)  
*An opaque handle representing a VdpOutputSurface object.*
- typedef [VdpStatus](#) [VdpOutputSurfaceCreate](#)([VdpDevice](#) device, [VdpRGBAFormat](#) rgba\_format, uint32\_t width, uint32\_t height, [VdpOutputSurface](#) \*surface)  
*Create a VdpOutputSurface.*
- typedef [VdpStatus](#) [VdpOutputSurfaceDestroy](#)([VdpOutputSurface](#) surface)  
*Destroy a VdpOutputSurface.*
- typedef [VdpStatus](#) [VdpOutputSurfaceGetParameters](#)([VdpOutputSurface](#) surface, [VdpRGBAFormat](#) \*rgba\_format, uint32\_t \*width, uint32\_t \*height)  
*Retrieve the parameters used to create a VdpOutputSurface.*
- typedef [VdpStatus](#) [VdpOutputSurfaceGetBitsNative](#)([VdpOutputSurface](#) surface, [VdpRect](#) const \*source\_rect, void \*const \*destination\_data, uint32\_t const \*destination\_pitches)  
*Copy image data from a VdpOutputSurface to application memory in the surface's native format.*
- typedef [VdpStatus](#) [VdpOutputSurfacePutBitsNative](#)([VdpOutputSurface](#) surface, void const \*const \*source\_data, uint32\_t const \*source\_pitches, [VdpRect](#) const \*destination\_rect)  
*Copy image data from application memory in the surface's native format to a VdpOutputSurface.*
- typedef [VdpStatus](#) [VdpOutputSurfacePutBitsIndexed](#)([VdpOutputSurface](#) surface, [VdpIndexedFormat](#) source\_indexed\_format, void const \*const \*source\_data, uint32\_t const \*source\_pitch, [VdpRect](#) const \*destination\_rect, [VdpColorTableFormat](#) color\_table\_format, void const \*color\_table)  
*Copy image data from application memory in a specific indexed format to a VdpOutputSurface.*
- typedef [VdpStatus](#) [VdpOutputSurfacePutBitsYCbCr](#)([VdpOutputSurface](#) surface, [VdpYCbCrFormat](#) source\_ycbcr\_format, void const \*const \*source\_data, uint32\_t const \*source\_pitches, [VdpRect](#) const \*destination\_rect, [VdpCSCMatrix](#) const \*csc\_matrix)  
*Copy image data from application memory in a specific YCbCr format to a VdpOutputSurface.*
- typedef [VdpStatus](#) [VdpBitmapSurfaceQueryCapabilities](#)([VdpDevice](#) device, [VdpRGBAFormat](#) surface\_rgba\_format, [VdpBool](#) \*is\_supported, uint32\_t \*max\_width, uint32\_t \*max\_height)  
*Query the implementation's VdpBitmapSurface capabilities.*
- typedef uint32\_t [VdpBitmapSurface](#)  
*An opaque handle representing a VdpBitmapSurface object.*
- typedef [VdpStatus](#) [VdpBitmapSurfaceCreate](#)([VdpDevice](#) device, [VdpRGBAFormat](#) rgba\_format, uint32\_t width, uint32\_t height, [VdpBool](#) frequently\_accessed, [VdpBitmapSurface](#) \*surface)  
*Create a VdpBitmapSurface.*
- typedef [VdpStatus](#) [VdpBitmapSurfaceDestroy](#)([VdpBitmapSurface](#) surface)  
*Destroy a VdpBitmapSurface.*
- typedef [VdpStatus](#) [VdpBitmapSurfaceGetParameters](#)([VdpBitmapSurface](#) surface, [VdpRGBAFormat](#) \*rgba\_format, uint32\_t \*width, uint32\_t \*height, [VdpBool](#) \*frequently\_accessed)  
*Retrieve the parameters used to create a VdpBitmapSurface.*
- typedef [VdpStatus](#) [VdpBitmapSurfacePutBitsNative](#)([VdpBitmapSurface](#) surface, void const \*const \*source\_data, uint32\_t const \*source\_pitches, [VdpRect](#) const \*destination\_rect)  
*Copy image data from application memory in the surface's native format to a VdpBitmapSurface.*
- typedef [VdpStatus](#) [VdpOutputSurfaceRenderOutputSurface](#)([VdpOutputSurface](#) destination\_surface, [VdpRect](#) const \*destination\_rect, [VdpOutputSurface](#) source\_surface, [VdpRect](#) const \*source\_rect, [VdpColor](#) const \*colors, [VdpOutputSurfaceRenderBlendState](#) const \*blend\_state, uint32\_t flags)  
*Composite a sub-rectangle of a VdpOutputSurface into a sub-rectangle of another VdpOutputSurface; Output Surface object VdpOutputSurface.*

- typedef [VdpStatus](#) [VdpOutputSurfaceRenderBitmapSurface](#)([VdpOutputSurface](#) destination\_surface, [VdpRect](#) const \*destination\_rect, [VdpBitmapSurface](#) source\_surface, [VdpRect](#) const \*source\_rect, [VdpColor](#) const \*colors, [VdpOutputSurfaceRenderBlendState](#) const \*blend\_state, uint32\_t flags)  
*Composite a sub-rectangle of a [VdpBitmapSurface](#) into a sub-rectangle of a [VdpOutputSurface](#); Output Surface object [VdpOutputSurface](#).*
- typedef uint32\_t [VdpDecoderProfile](#)  
*The set of all known compressed video formats, and associated profiles, that may be decoded.*
- typedef [VdpStatus](#) [VdpDecoderQueryCapabilities](#)([VdpDevice](#) device, [VdpDecoderProfile](#) profile, [VdpBool](#) \*is\_supported, uint32\_t \*max\_level, uint32\_t \*max\_macroblocks, uint32\_t \*max\_width, uint32\_t \*max\_height)  
*Query the implementation's [VdpDecoder](#) capabilities.*
- typedef uint32\_t [VdpDecoder](#)  
*An opaque handle representing a [VdpDecoder](#) object.*
- typedef [VdpStatus](#) [VdpDecoderCreate](#)([VdpDevice](#) device, [VdpDecoderProfile](#) profile, uint32\_t width, uint32\_t height, uint32\_t max\_references, [VdpDecoder](#) \*decoder)  
*Create a [VdpDecoder](#).*
- typedef [VdpStatus](#) [VdpDecoderDestroy](#)([VdpDecoder](#) decoder)  
*Destroy a [VdpDecoder](#).*
- typedef [VdpStatus](#) [VdpDecoderGetParameters](#)([VdpDecoder](#) decoder, [VdpDecoderProfile](#) \*profile, uint32\_t \*width, uint32\_t \*height)  
*Retrieve the parameters used to create a [VdpDecoder](#).*
- typedef void [VdpPictureInfo](#)  
*A generic "picture information" type.*
- typedef [VdpPictureInfoMPEG4Part2](#) [VdpPictureInfoDivX4](#)  
*Picture parameter information for a DivX 4 picture.*
- typedef [VdpPictureInfoMPEG4Part2](#) [VdpPictureInfoDivX5](#)  
*Picture parameter information for a DivX 5 picture.*
- typedef [VdpStatus](#) [VdpDecoderRender](#)([VdpDecoder](#) decoder, [VdpVideoSurface](#) target, [VdpPictureInfo](#) const \*picture\_info, uint32\_t bitstream\_buffer\_count, [VdpBitstreamBuffer](#) const \*bitstream\_buffers)  
*Decode a compressed field/frame and render the result into a [VdpVideoSurface](#).*
- typedef uint32\_t [VdpVideoMixerFeature](#)  
*A [VdpVideoMixer](#) feature that must be requested at creation time to be used.*
- typedef uint32\_t [VdpVideoMixerParameter](#)  
*A [VdpVideoMixer](#) creation parameter.*
- typedef uint32\_t [VdpVideoMixerAttribute](#)  
*An adjustable attribute of [VdpVideoMixer](#) operation.*
- typedef [VdpStatus](#) [VdpVideoMixerQueryFeatureSupport](#)([VdpDevice](#) device, [VdpVideoMixerFeature](#) feature, [VdpBool](#) \*is\_supported)  
*Query the implementation's support for a specific feature.*
- typedef [VdpStatus](#) [VdpVideoMixerQueryParameterSupport](#)([VdpDevice](#) device, [VdpVideoMixerParameter](#) parameter, [VdpBool](#) \*is\_supported)  
*Query the implementation's support for a specific parameter.*
- typedef [VdpStatus](#) [VdpVideoMixerQueryAttributeSupport](#)([VdpDevice](#) device, [VdpVideoMixerAttribute](#) attribute, [VdpBool](#) \*is\_supported)  
*Query the implementation's support for a specific attribute.*
- typedef [VdpStatus](#) [VdpVideoMixerQueryParameterValueRange](#)([VdpDevice](#) device, [VdpVideoMixerParameter](#) parameter, void \*min\_value, void \*max\_value)  
*Query the implementation's supported for a specific parameter.*
- typedef [VdpStatus](#) [VdpVideoMixerQueryAttributeValueRange](#)([VdpDevice](#) device, [VdpVideoMixerAttribute](#) attribute, void \*min\_value, void \*max\_value)  
*Query the implementation's supported for a specific attribute.*
- typedef uint32\_t [VdpVideoMixer](#)



- An opaque handle representing a VdpVideoMixer object.*
- typedef [VdpStatus](#) [VdpVideoMixerCreate](#)([VdpDevice](#) device, uint32\_t feature\_count, [VdpVideoMixerFeature](#) const \*features, uint32\_t parameter\_count, [VdpVideoMixerParameter](#) const \*parameters, void const \*const \*parameter\_values, [VdpVideoMixer](#) \*mixer)
- Create a VdpVideoMixer.*
- typedef [VdpStatus](#) [VdpVideoMixerSetFeatureEnables](#)([VdpVideoMixer](#) mixer, uint32\_t feature\_count, [VdpVideoMixerFeature](#) const \*features, [VdpBool](#) const \*feature\_enables)
- Enable or disable features.*
- typedef [VdpStatus](#) [VdpVideoMixerSetAttributeValues](#)([VdpVideoMixer](#) mixer, uint32\_t attribute\_count, [VdpVideoMixerAttribute](#) const \*attributes, void const \*const \*attribute\_values)
- Set attribute values.*
- typedef [VdpStatus](#) [VdpVideoMixerGetFeatureSupport](#)([VdpVideoMixer](#) mixer, uint32\_t feature\_count, [VdpVideoMixerFeature](#) const \*features, [VdpBool](#) \*feature\_supports)
- Retrieve whether features were requested at creation time.*
- typedef [VdpStatus](#) [VdpVideoMixerGetFeatureEnables](#)([VdpVideoMixer](#) mixer, uint32\_t feature\_count, [VdpVideoMixerFeature](#) const \*features, [VdpBool](#) \*feature\_enables)
- Retrieve whether features are enabled.*
- typedef [VdpStatus](#) [VdpVideoMixerGetParameterValues](#)([VdpVideoMixer](#) mixer, uint32\_t parameter\_count, [VdpVideoMixerParameter](#) const \*parameters, void \*const \*parameter\_values)
- Retrieve parameter values given at creation time.*
- typedef [VdpStatus](#) [VdpVideoMixerGetAttributeValues](#)([VdpVideoMixer](#) mixer, uint32\_t attribute\_count, [VdpVideoMixerAttribute](#) const \*attributes, void \*const \*attribute\_values)
- Retrieve current attribute values.*
- typedef [VdpStatus](#) [VdpVideoMixerDestroy](#)([VdpVideoMixer](#) mixer)
- Destroy a VdpVideoMixer.*
- typedef [VdpStatus](#) [VdpVideoMixerRender](#)([VdpVideoMixer](#) mixer, [VdpOutputSurface](#) background\_surface, [VdpRect](#) const \*background\_source\_rect, [VdpVideoMixerPictureStructure](#) current\_picture\_structure, uint32\_t video\_surface\_past\_count, [VdpVideoSurface](#) const \*video\_surface\_past, [VdpVideoSurface](#) video\_surface\_current, uint32\_t video\_surface\_future\_count, [VdpVideoSurface](#) const \*video\_surface\_future, [VdpRect](#) const \*video\_source\_rect, [VdpOutputSurface](#) destination\_surface, [VdpRect](#) const \*destination\_video\_rect, [VdpRect](#) const \*destination\_video\_rect, uint32\_t layer\_count, [VdpLayer](#) const \*layers)
- Perform a video post-processing and compositing operation.*
- typedef uint64\_t [VdpTime](#)
- The representation of a point in time.*
- typedef uint32\_t [VdpPresentationQueueTarget](#)
- An opaque handle representing the location where video will be presented.*
- typedef [VdpStatus](#) [VdpPresentationQueueTargetDestroy](#)([VdpPresentationQueueTarget](#) presentation\_queue\_target)
- Destroy a VdpPresentationQueueTarget.*
- typedef uint32\_t [VdpPresentationQueue](#)
- An opaque handle representing a presentation queue object.*
- typedef [VdpStatus](#) [VdpPresentationQueueCreate](#)([VdpDevice](#) device, [VdpPresentationQueueTarget](#) presentation\_queue\_target, [VdpPresentationQueue](#) \*presentation\_queue)
- Create a VdpPresentationQueue.*
- typedef [VdpStatus](#) [VdpPresentationQueueDestroy](#)([VdpPresentationQueue](#) presentation\_queue)
- Destroy a VdpPresentationQueue.*
- typedef [VdpStatus](#) [VdpPresentationQueueSetBackgroundColor](#)([VdpPresentationQueue](#) presentation\_queue, [VdpColor](#) \*const background\_color)
- Configure the background color setting.*
- typedef [VdpStatus](#) [VdpPresentationQueueGetBackgroundColor](#)([VdpPresentationQueue](#) presentation\_queue, [VdpColor](#) \*background\_color)
- Retrieve the current background color setting.*

- typedef `VdpStatus VdpPresentationQueueGetTime(VdpPresentationQueue presentation_queue, VdpTime *current_time)`  
Retrieve the presentation queue's "current" time.
- typedef `VdpStatus VdpPresentationQueueDisplay(VdpPresentationQueue presentation_queue, VdpOutputSurface surface, uint32_t clip_width, uint32_t clip_height, VdpTime earliest_presentation_time)`  
Enter a surface into the presentation queue.
- typedef `VdpStatus VdpPresentationQueueBlockUntilSurfaceIdle(VdpPresentationQueue presentation_queue, VdpOutputSurface surface, VdpTime *first_presentation_time)`  
Wait for a surface to finish being displayed.
- typedef `VdpStatus VdpPresentationQueueQuerySurfaceStatus(VdpPresentationQueue presentation_queue, VdpOutputSurface surface, VdpPresentationQueueStatus *status, VdpTime *first_presentation_time)`  
Poll the current queue status of a surface.
- typedef void `VdpPreemptionCallback(VdpDevice device, void *context)`  
A callback to notify the client application that a device's display has been preempted.
- typedef `VdpStatus VdpPreemptionCallbackRegister(VdpDevice device, VdpPreemptionCallback callback, void *context)`  
Configure the display preemption callback.
- typedef uint32\_t `VdpFuncId`  
A type suitable for `VdpGetProcAddress`'s **function\_id** parameter.
- typedef `VdpStatus VdpGetProcAddress(VdpDevice device, VdpFuncId function_id, void **function_pointer)`  
Retrieve a VDP AU function pointer.

## Enumerations

- enum `VdpStatus` {  
VDP\_STATUS\_OK = 0, VDP\_STATUS\_NO\_IMPLEMENTATION, VDP\_STATUS\_DISPLAY\_PREEMPTED,  
VDP\_STATUS\_INVALID\_HANDLE,  
VDP\_STATUS\_INVALID\_POINTER, VDP\_STATUS\_INVALID\_CHROMA\_TYPE, VDP\_STATUS\_INVALID\_Y\_CB\_CR\_FORMAT, VDP\_STATUS\_INVALID\_RGBA\_FORMAT,  
VDP\_STATUS\_INVALID\_INDEXED\_FORMAT, VDP\_STATUS\_INVALID\_COLOR\_STANDARD, VDP\_STATUS\_INVALID\_COLOR\_TABLE\_FORMAT, VDP\_STATUS\_INVALID\_BLEND\_FACTOR,  
VDP\_STATUS\_INVALID\_BLEND\_EQUATION, VDP\_STATUS\_INVALID\_FLAG, VDP\_STATUS\_INVALID\_DECODER\_PROFILE, VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_FEATURE,  
VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_PARAMETER, VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_ATTRIBUTE, VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_PICTURE\_STRUCTURE, VDP\_STATUS\_INVALID\_FUNC\_ID,  
VDP\_STATUS\_INVALID\_SIZE, VDP\_STATUS\_INVALID\_VALUE, VDP\_STATUS\_INVALID\_STRUCTURE\_VERSION, VDP\_STATUS\_RESOURCES,  
VDP\_STATUS\_HANDLE\_DEVICE\_MISMATCH, VDP\_STATUS\_ERROR }  
The set of all possible error codes.
- enum `VdpOutputSurfaceRenderBlendFactor` {  
VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ZERO = 0, VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE = 1, VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_SRC\_COLOR = 2,  
VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE\_MINUS\_SRC\_COLOR = 3,  
VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_SRC\_ALPHA = 4, VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE\_MINUS\_SRC\_ALPHA = 5, VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_DST\_ALPHA = 6,  
VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE\_MINUS\_DST\_ALPHA = 7,  
VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_DST\_COLOR = 8, VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE\_MINUS\_DST\_COLOR = 9, VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_SRC\_ALPHA\_SATURATE = 10,  
VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_CONSTANT\_COLOR = 11,  
VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE\_MINUS\_CONSTANT\_COLOR = 12, VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_CONSTANT\_ALPHA = 13, VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR\_ONE\_MINUS\_CONSTANT\_ALPHA = 14 }  
The set of all possible blend factors.



*The blending equation factors.*

- enum [VdpOutputSurfaceRenderBlendEquation](#) {  
[VDP\\_OUTPUT\\_SURFACE\\_RENDER\\_BLEND\\_EQUATION\\_SUBTRACT](#) = 0, [VDP\\_OUTPUT\\_SURFACE\\_RENDER\\_BLEND\\_EQUATION\\_REVERSE\\_SUBTRACT](#) = 1, [VDP\\_OUTPUT\\_SURFACE\\_RENDER\\_BLEND\\_EQUATION\\_ADD](#) = 2, [VDP\\_OUTPUT\\_SURFACE\\_RENDER\\_BLEND\\_EQUATION\\_MIN](#) = 3, [VDP\\_OUTPUT\\_SURFACE\\_RENDER\\_BLEND\\_EQUATION\\_MAX](#) = 4 }

*The blending equations.*

- enum [VdpVideoMixerPictureStructure](#) { [VDP\\_VIDEO\\_MIXER\\_PICTURE\\_STRUCTURE\\_TOP\\_FIELD](#), [VDP\\_VIDEO\\_MIXER\\_PICTURE\\_STRUCTURE\\_BOTTOM\\_FIELD](#), [VDP\\_VIDEO\\_MIXER\\_PICTURE\\_STRUCTURE\\_FRAME](#) }

*The structure of the picture present in a [VdpVideoSurface](#).*

- enum [VdpPresentationQueueStatus](#) { [VDP\\_PRESENTATION\\_QUEUE\\_STATUS\\_IDLE](#), [VDP\\_PRESENTATION\\_QUEUE\\_STATUS\\_QUEUED](#), [VDP\\_PRESENTATION\\_QUEUE\\_STATUS\\_VISIBLE](#) }

*The status of a surface within a presentation queue.*

### 7.1.1 Detailed Description

The Core API.

This file contains the [Core API](#).

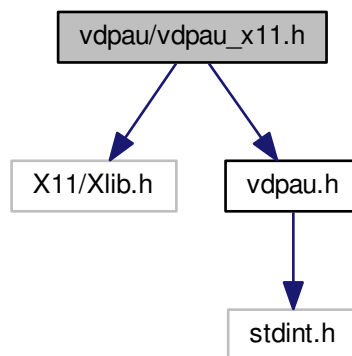
## 7.2 vdpau/vdpau\_x11.h File Reference

X11 Window System Integration Layer.

```
#include <X11/Xlib.h>
```

```
#include "vdpau.h"
```

Include dependency graph for vdpau\_x11.h:



- #define [VDP\\_FUNC\\_ID\\_PRESENTATION\\_QUEUE\\_TARGET\\_CREATE\\_X11](#)
- typedef [VdpStatus](#) [VdpDeviceCreateX11](#)(Display \*display, int screen, [VdpDevice](#) \*device, [VdpGetProcAddress](#) \*\*get\_proc\_address)

*Create a VdpDevice object for use with X11.*

- typedef [VdpStatus](#) [VdpPresentationQueueTargetCreateX11](#)([VdpDevice](#) device, Drawable drawable, [VdpPresentationQueueTarget](#) \*target)

*Create a VdpPresentationQueueTarget for use with X11.*

- [VdpDeviceCreateX11](#) vdp\_device\_create\_x11

*Create a VdpDevice object for use with X11. This is an actual symbol of type [VdpDeviceCreateX11](#).*

### 7.2.1 Detailed Description

X11 Window System Integration Layer.

This file contains the [X11 Window System Integration Layer](#) X11 Window System Integration Layer.

# Index

- alpha
  - VdpColor, [104](#)
- alternate\_scan
  - VdpPictureInfoMPEG1Or2, [120](#)
- alternate\_vertical\_scan\_flag
  - VdpPictureInfoMPEG4Part2, [122](#)
- amp\_enabled\_flag
  - VdpPictureInfoHEVC, [113](#)
- backward\_reference
  - VdpPictureInfoMPEG1Or2, [120](#)
  - VdpPictureInfoMPEG4Part2, [122](#)
  - VdpPictureInfoVC1, [123](#)
- Basic Types, [22](#)
  - VDP\_FALSE, [22](#)
  - VDP\_TRUE, [22](#)
  - VdpBool, [22](#)
- bit\_depth\_chroma\_minus8
  - VdpPictureInfoHEVC, [113](#)
- bit\_depth\_luma\_minus8
  - VdpPictureInfoHEVC, [113](#)
- bitstream
  - VdpBitstreamBuffer, [103](#)
- bitstream\_bytes
  - VdpBitstreamBuffer, [103](#)
- blend\_constant
  - VdpOutputSurfaceRenderBlendState, [106](#)
- blend\_equation\_alpha
  - VdpOutputSurfaceRenderBlendState, [106](#)
- blend\_equation\_color
  - VdpOutputSurfaceRenderBlendState, [106](#)
- blend\_factor\_destination\_alpha
  - VdpOutputSurfaceRenderBlendState, [106](#)
- blend\_factor\_destination\_color
  - VdpOutputSurfaceRenderBlendState, [106](#)
- blend\_factor\_source\_alpha
  - VdpOutputSurfaceRenderBlendState, [106](#)
- blend\_factor\_source\_color
  - VdpOutputSurfaceRenderBlendState, [106](#)
- blue
  - VdpColor, [104](#)
- bottom\_field\_flag
  - VdpPictureInfoH264, [108](#)
- bottom\_is\_reference
  - VdpReferenceFrameH264, [130](#)
- brightness
  - VdpProcamp, [128](#)
- cabac\_init\_present\_flag
  - VdpPictureInfoHEVC, [113](#)
- chroma\_format\_idc
  - VdpPictureInfoHEVC, [113](#)
- chroma\_qp\_index\_offset
  - VdpPictureInfoH264, [108](#)
- column\_width\_minus1
  - VdpPictureInfoHEVC, [113](#)
- concealment\_motion\_vectors
  - VdpPictureInfoMPEG1Or2, [120](#)
- constrained\_intra\_pred\_flag
  - VdpPictureInfoH264, [108](#)
  - VdpPictureInfoHEVC, [113](#)
- contrast
  - VdpProcamp, [128](#)
- Core API, [20](#)
- cu\_qp\_delta\_enabled\_flag
  - VdpPictureInfoHEVC, [113](#)
- CurrPicOrderCntVal
  - VdpPictureInfoHEVC, [113](#)
- CurrRpsIdx
  - VdpPictureInfoHEVC, [113](#)
- deblockEnable
  - VdpPictureInfoVC1, [123](#)
- deblocking\_filter\_control\_present\_flag
  - VdpPictureInfoH264, [108](#)
  - VdpPictureInfoHEVC, [113](#)
- deblocking\_filter\_override\_enabled\_flag
  - VdpPictureInfoHEVC, [113](#)
- delta\_pic\_order\_always\_zero\_flag
  - VdpPictureInfoH264, [108](#)
- dependent\_slice\_segments\_enabled\_flag
  - VdpPictureInfoHEVC, [113](#)
- destination\_rect
  - VdpLayer, [105](#)
- diff\_cu\_qp\_delta\_depth
  - VdpPictureInfoHEVC, [113](#)
- direct\_8x8\_inference\_flag
  - VdpPictureInfoH264, [108](#)
- Display Preemption, [91](#)
  - VdpPreemptionCallback, [91](#)
  - VdpPreemptionCallbackRegister, [92](#)
- dquant
  - VdpPictureInfoVC1, [124](#)
- entropy\_coding\_mode\_flag
  - VdpPictureInfoH264, [108](#)
- entropy\_coding\_sync\_enabled\_flag
  - VdpPictureInfoHEVC, [113](#)
- Entry Point Retrieval, [93](#)
  - VDP\_FUNC\_ID\_BASE\_WINSYS, [94](#)

- VDP\_FUNC\_ID\_BITMAP\_SURFACE\_CREATE, 94
- VDP\_FUNC\_ID\_BITMAP\_SURFACE\_DESTROY, 94
- VDP\_FUNC\_ID\_BITMAP\_SURFACE\_GET\_PARAMETERS, 94
- VDP\_FUNC\_ID\_BITMAP\_SURFACE\_PUT\_BITS\_NATIVE, 94
- VDP\_FUNC\_ID\_BITMAP\_SURFACE\_QUERY\_CAPABILITIES, 95
- VDP\_FUNC\_ID\_DECODER\_CREATE, 95
- VDP\_FUNC\_ID\_DECODER\_DESTROY, 95
- VDP\_FUNC\_ID\_DECODER\_GET\_PARAMETERS, 95
- VDP\_FUNC\_ID\_DECODER\_QUERY\_CAPABILITIES, 95
- VDP\_FUNC\_ID\_DECODER\_RENDER, 95
- VDP\_FUNC\_ID\_DEVICE\_DESTROY, 95
- VDP\_FUNC\_ID\_GENERATE\_CSC\_MATRIX, 95
- VDP\_FUNC\_ID\_GET\_API\_VERSION, 95
- VDP\_FUNC\_ID\_GET\_ERROR\_STRING, 95
- VDP\_FUNC\_ID\_GET\_INFORMATION\_STRING, 95
- VDP\_FUNC\_ID\_GET\_PROC\_ADDRESS, 95
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_CREATE, 95
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_DESTROY, 95
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_GET\_BITS\_NATIVE, 95
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_GET\_PARAMETERS, 95
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_PUT\_BITS\_INDEXED, 95
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_PUT\_BITS\_NATIVE, 95
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_PUT\_BITS\_Y\_CB\_CR, 95
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_QUERY\_CAPABILITIES, 95
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_QUERY\_GET\_PUT\_BITS\_NATIVE\_CAPABILITIES, 95
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_QUERY\_PUT\_BITS\_INDEXED\_CAPABILITIES, 95
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_QUERY\_PUT\_BITS\_Y\_CB\_CR\_CAPABILITIES, 95
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_RENDER\_BITMAP\_SURFACE, 96
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_RENDER\_OUTPUT\_SURFACE, 96
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_RENDER\_VIDEO\_SURFACE\_LUMA, 96
- VDP\_FUNC\_ID\_PREEMPTION\_CALLBACK\_REGISTER, 96
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_BLOCK\_UNTIL\_SURFACE\_IDLE, 96
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_CREATE, 96
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_DESTROY, 96
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_DISPLAY, 96
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_GET\_BACKGROUND\_COLOR, 96
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_GET\_TIME, 96
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_QUERY\_SURFACE\_STATUS, 96
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_SET\_BACKGROUND\_COLOR, 96
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_TARGET\_DESTROY, 96
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_CREATE, 96
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_DESTROY, 96
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_GET\_ATTRIBUTES\_VALUES, 96
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_GET\_FEATURE\_ENABLES, 96
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_GET\_FEATURE\_SUPPORT, 96
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_GET\_PARAMETER\_VALUES, 96
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_QUERY\_ATTRIBUTES\_SUPPORT, 96
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_QUERY\_ATTRIBUTES\_VALUE\_RANGE, 96
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_QUERY\_FEATURE\_SUPPORT, 96
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_QUERY\_PARAMETER\_SUPPORT, 96
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_QUERY\_PARAMETER\_VALUE\_RANGE, 97
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_RENDER, 97
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_SET\_ATTRIBUTES\_VALUES, 97
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_SET\_FEATURE\_ENABLES, 97
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_CREATE, 97
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_DESTROY, 97
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_GET\_BITS\_Y\_CB\_CR, 97
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_GET\_PARAMETERS, 97
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_PUT\_BITS\_Y\_CB\_CR, 97
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_QUERY\_CAPABILITIES, 97
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_QUERY\_GET\_PUT\_BITS\_Y\_CB\_CR\_CAPABILITIES, 97
- VdpFuncId, 97
- VdpGetProcAddress, 97
- Error Handling, 29
- VDP\_STATUS\_DISPLAY\_PREEMPTED, 30

- VDP\_STATUS\_ERROR, 31
- VDP\_STATUS\_HANDLE\_DEVICE\_MISMATCH, 31
- VDP\_STATUS\_INVALID\_BLEND\_EQUATION, 30
- VDP\_STATUS\_INVALID\_BLEND\_FACTOR, 30
- VDP\_STATUS\_INVALID\_CHROMA\_TYPE, 30
- VDP\_STATUS\_INVALID\_COLOR\_STANDARD, 30
- VDP\_STATUS\_INVALID\_COLOR\_TABLE\_FORMAT, 30
- VDP\_STATUS\_INVALID\_DECODER\_PROFILE, 30
- VDP\_STATUS\_INVALID\_FLAG, 30
- VDP\_STATUS\_INVALID\_FUNC\_ID, 30
- VDP\_STATUS\_INVALID\_HANDLE, 30
- VDP\_STATUS\_INVALID\_INDEXED\_FORMAT, 30
- VDP\_STATUS\_INVALID\_POINTER, 30
- VDP\_STATUS\_INVALID\_RGBA\_FORMAT, 30
- VDP\_STATUS\_INVALID\_SIZE, 30
- VDP\_STATUS\_INVALID\_STRUCT\_VERSION, 31
- VDP\_STATUS\_INVALID\_VALUE, 31
- VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_ATTRIBUTES, 30
- VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_FEATURE, 30
- VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_PARAMETER, 30
- VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_PICTURE\_STRUCTURE, 30
- VDP\_STATUS\_INVALID\_Y\_CB\_CR\_FORMAT, 30
- VDP\_STATUS\_NO\_IMPLEMENTATION, 30
- VDP\_STATUS\_OK, 30
- VDP\_STATUS\_RESOURCES, 31
- VdpGetErrorString, 29
- VdpStatus, 30
- extended\_dmv
  - VdpPictureInfoVC1, 124
- extended\_mv
  - VdpPictureInfoVC1, 124
- f\_code
  - VdpPictureInfoMPEG1Or2, 120
- fastuvmc
  - VdpPictureInfoVC1, 124
- field\_order\_cnt
  - VdpPictureInfoH264, 108
  - VdpReferenceFrameH264, 130
- field\_pic\_flag
  - VdpPictureInfoH264, 108
- finterpflag
  - VdpPictureInfoVC1, 124
- forward\_reference
  - VdpPictureInfoMPEG1Or2, 120
  - VdpPictureInfoMPEG4Part2, 122
  - VdpPictureInfoVC1, 124
- frame\_coding\_mode
  - VdpPictureInfoVC1, 124
- frame\_idx
  - VdpReferenceFrameH264, 130
- frame\_mbs\_only\_flag
  - VdpPictureInfoH264, 108
- frame\_num
  - VdpPictureInfoH264, 108
- frame\_pred\_frame\_dct
  - VdpPictureInfoMPEG1Or2, 120
- full\_pel\_backward\_vector
  - VdpPictureInfoMPEG1Or2, 120
- full\_pel\_forward\_vector
  - VdpPictureInfoMPEG1Or2, 120
- green
  - VdpColor, 104
- hue
  - VdpProcamp, 128
- IDRPicFlag
  - VdpPictureInfoHEVC, 114
- init\_qp\_minus26
  - VdpPictureInfoHEVC, 114
- interlace
  - VdpPictureInfoVC1, 124
- interlaced
  - VdpPictureInfoMPEG4Part2, 122
- intra\_dc\_precision
  - VdpPictureInfoMPEG1Or2, 120
- intra\_quantizer\_matrix
  - VdpPictureInfoMPEG1Or2, 120
  - VdpPictureInfoMPEG4Part2, 122
- intra\_vlc\_format
  - VdpPictureInfoMPEG1Or2, 120
- is\_long\_term
  - VdpReferenceFrameH264, 130
- is\_reference
  - VdpPictureInfoH264, 108
- IsLongTerm
  - VdpPictureInfoHEVC, 114
- lists\_modification\_present\_flag
  - VdpPictureInfoHEVC, 114
- log2\_diff\_max\_min\_luma\_coding\_block\_size
  - VdpPictureInfoHEVC, 114
- log2\_diff\_max\_min\_pcm\_luma\_coding\_block\_size
  - VdpPictureInfoHEVC, 114
- log2\_diff\_max\_min\_transform\_block\_size
  - VdpPictureInfoHEVC, 114
- log2\_max\_frame\_num\_minus4
  - VdpPictureInfoH264, 108
- log2\_max\_pic\_order\_cnt\_lsb\_minus4
  - VdpPictureInfoH264, 108
  - VdpPictureInfoHEVC, 114
- log2\_min\_luma\_coding\_block\_size\_minus3
  - VdpPictureInfoHEVC, 114
- log2\_min\_pcm\_luma\_coding\_block\_size\_minus3
  - VdpPictureInfoHEVC, 114
- log2\_min\_transform\_block\_size\_minus2
  - VdpPictureInfoHEVC, 114
- log2\_parallel\_merge\_level\_minus2

- VdpPictureInfoHEVC, 114
- long\_term\_ref\_pics\_present\_flag
  - VdpPictureInfoHEVC, 114
- loop\_filter\_across\_tiles\_enabled\_flag
  - VdpPictureInfoHEVC, 114
- loopfilter
  - VdpPictureInfoVC1, 124
- max\_transform\_hierarchy\_depth\_inter
  - VdpPictureInfoHEVC, 114
- max\_transform\_hierarchy\_depth\_intra
  - VdpPictureInfoHEVC, 115
- maxbframes
  - VdpPictureInfoVC1, 124
- mb\_adaptive\_frame\_field\_flag
  - VdpPictureInfoH264, 108
- Miscellaneous Types, 23
  - VDP\_CHROMA\_TYPE\_420, 24
  - VDP\_CHROMA\_TYPE\_422, 24
  - VDP\_CHROMA\_TYPE\_444, 24
  - VDP\_INDEXED\_FORMAT\_A4I4, 24
  - VDP\_INDEXED\_FORMAT\_A8I8, 24
  - VDP\_INDEXED\_FORMAT\_I4A4, 25
  - VDP\_INDEXED\_FORMAT\_I8A8, 25
  - VDP\_INVALID\_HANDLE, 25
  - VDP\_RGBA\_FORMAT\_A8, 25
  - VDP\_RGBA\_FORMAT\_B10G10R10A2, 25
  - VDP\_RGBA\_FORMAT\_B8G8R8A8, 26
  - VDP\_RGBA\_FORMAT\_R10G10B10A2, 26
  - VDP\_RGBA\_FORMAT\_R8G8B8A8, 26
  - VDP\_YCBCR\_FORMAT\_NV12, 26
  - VDP\_YCBCR\_FORMAT\_UYVY, 26
  - VDP\_YCBCR\_FORMAT\_V8U8Y8A8, 27
  - VDP\_YCBCR\_FORMAT\_Y8U8V8A8, 27
  - VDP\_YCBCR\_FORMAT\_YUYV, 27
  - VDP\_YCBCR\_FORMAT\_YV12, 27
  - VdpChromaType, 28
  - VdpIndexedFormat, 28
  - VdpRGBAFormat, 28
  - VdpYCbCrFormat, 28
- multires
  - VdpPictureInfoVC1, 125
- non\_intra\_quantizer\_matrix
  - VdpPictureInfoMPEG1Or2, 121
  - VdpPictureInfoMPEG4Part2, 122
- num\_extra\_slice\_header\_bits
  - VdpPictureInfoHEVC, 115
- num\_long\_term\_ref\_pics\_sps
  - VdpPictureInfoHEVC, 115
- num\_ref\_frames
  - VdpPictureInfoH264, 108
- num\_ref\_idx\_l0\_active\_minus1
  - VdpPictureInfoH264, 108
- num\_ref\_idx\_l0\_default\_active\_minus1
  - VdpPictureInfoHEVC, 115
- num\_ref\_idx\_l1\_active\_minus1
  - VdpPictureInfoH264, 108
- num\_ref\_idx\_l1\_default\_active\_minus1
  - VdpPictureInfoH264, 108
- num\_short\_term\_ref\_pic\_sets
  - VdpPictureInfoHEVC, 115
- num\_tile\_columns\_minus1
  - VdpPictureInfoHEVC, 115
- num\_tile\_rows\_minus1
  - VdpPictureInfoHEVC, 115
- NumDeltaPocsOfRefRpsIdx
  - VdpPictureInfoHEVC, 115
- NumLongTermPictureSliceHeaderBits
  - VdpPictureInfoHEVC, 115
- NumPocLtCurr
  - VdpPictureInfoHEVC, 115
- NumPocStCurrAfter
  - VdpPictureInfoHEVC, 115
- NumPocStCurrBefore
  - VdpPictureInfoHEVC, 116
- NumPocTotalCurr
  - VdpPictureInfoHEVC, 116
- NumShortTermPictureSliceHeaderBits
  - VdpPictureInfoHEVC, 116
- output\_flag\_present\_flag
  - VdpPictureInfoHEVC, 116
- overlap
  - VdpPictureInfoVC1, 125
- panscan\_flag
  - VdpPictureInfoVC1, 125
- pcm\_enabled\_flag
  - VdpPictureInfoHEVC, 116
- pcm\_loop\_filter\_disabled\_flag
  - VdpPictureInfoHEVC, 116
- pcm\_sample\_bit\_depth\_chroma\_minus1
  - VdpPictureInfoHEVC, 116
- pcm\_sample\_bit\_depth\_luma\_minus1
  - VdpPictureInfoHEVC, 116
- pic\_height\_in\_luma\_samples
  - VdpPictureInfoHEVC, 116
- pic\_init\_qp\_minus26
  - VdpPictureInfoH264, 108
- pic\_order\_cnt\_type
  - VdpPictureInfoH264, 108
- pic\_order\_present\_flag
  - VdpPictureInfoH264, 108
- pic\_width\_in\_luma\_samples
  - VdpPictureInfoHEVC, 116
- PicOrderCntVal
  - VdpPictureInfoHEVC, 116
- picture\_coding\_type
  - VdpPictureInfoMPEG1Or2, 121
- picture\_structure
  - VdpPictureInfoMPEG1Or2, 121
- picture\_type
  - VdpPictureInfoVC1, 125
- pictureInfo
  - VdpPictureInfoH264Predictive, 110
- postprocflag
  - VdpPictureInfoVC1, 125

- pps\_beta\_offset\_div2
  - VdpPictureInfoHEVC, [116](#)
- pps\_cb\_qp\_offset
  - VdpPictureInfoHEVC, [117](#)
- pps\_cr\_qp\_offset
  - VdpPictureInfoHEVC, [117](#)
- pps\_deblocking\_filter\_disabled\_flag
  - VdpPictureInfoHEVC, [117](#)
- pps\_loop\_filter\_across\_slices\_enabled\_flag
  - VdpPictureInfoHEVC, [117](#)
- pps\_slice\_chroma\_qp\_offsets\_present\_flag
  - VdpPictureInfoHEVC, [117](#)
- pps\_tc\_offset\_div2
  - VdpPictureInfoHEVC, [117](#)
- pquant
  - VdpPictureInfoVC1, [125](#)
- psf
  - VdpPictureInfoVC1, [125](#)
- pulldown
  - VdpPictureInfoVC1, [125](#)
- q\_scale\_type
  - VdpPictureInfoMPEG1Or2, [121](#)
- qpprime\_y\_zero\_transform\_bypass\_flag
  - VdpPictureInfoH264Predictive, [110](#)
- quant\_type
  - VdpPictureInfoMPEG4Part2, [122](#)
- quantizer
  - VdpPictureInfoVC1, [125](#)
- quarter\_sample
  - VdpPictureInfoMPEG4Part2, [122](#)
- RAPPicFlag
  - VdpPictureInfoHEVC, [117](#)
- range\_mapuv
  - VdpPictureInfoVC1, [125](#)
- range\_mapuv\_flag
  - VdpPictureInfoVC1, [126](#)
- range\_mapy
  - VdpPictureInfoVC1, [126](#)
- range\_mapy\_flag
  - VdpPictureInfoVC1, [126](#)
- rangered
  - VdpPictureInfoVC1, [126](#)
- red
  - VdpColor, [104](#)
- redundant\_pic\_cnt\_present\_flag
  - VdpPictureInfoH264, [108](#)
- RefPicSetLtCurr
  - VdpPictureInfoHEVC, [117](#)
- RefPicSetStCurrAfter
  - VdpPictureInfoHEVC, [117](#)
- RefPicSetStCurrBefore
  - VdpPictureInfoHEVC, [117](#)
- RefPics
  - VdpPictureInfoHEVC, [117](#)
- refdist\_flag
  - VdpPictureInfoVC1, [126](#)
- referenceFrames
  - VdpPictureInfoH264, [108](#)
- resync\_marker\_disable
  - VdpPictureInfoMPEG4Part2, [122](#)
- rounding\_control
  - VdpPictureInfoMPEG4Part2, [122](#)
- row\_height\_minus1
  - VdpPictureInfoHEVC, [117](#)
- sample\_adaptive\_offset\_enabled\_flag
  - VdpPictureInfoHEVC, [118](#)
- saturation
  - VdpProcamp, [128](#)
- scaling\_list\_enabled\_flag
  - VdpPictureInfoHEVC, [118](#)
- scaling\_lists\_4x4
  - VdpPictureInfoH264, [108](#)
- scaling\_lists\_8x8
  - VdpPictureInfoH264, [109](#)
- ScalingList16x16
  - VdpPictureInfoHEVC, [118](#)
- ScalingList32x32
  - VdpPictureInfoHEVC, [118](#)
- ScalingList4x4
  - VdpPictureInfoHEVC, [118](#)
- ScalingList8x8
  - VdpPictureInfoHEVC, [118](#)
- ScalingListDCCoeff16x16
  - VdpPictureInfoHEVC, [118](#)
- ScalingListDCCoeff32x32
  - VdpPictureInfoHEVC, [118](#)
- second\_chroma\_qp\_index\_offset
  - VdpPictureInfoH264, [109](#)
- separate\_colour\_plane\_flag
  - VdpPictureInfoH264Predictive, [110](#)
  - VdpPictureInfoHEVC, [118](#)
- short\_video\_header
  - VdpPictureInfoMPEG4Part2, [122](#)
- sign\_data\_hiding\_enabled\_flag
  - VdpPictureInfoHEVC, [118](#)
- slice\_count
  - VdpPictureInfoH264, [109](#)
  - VdpPictureInfoMPEG1Or2, [121](#)
  - VdpPictureInfoVC1, [126](#)
- slice\_segment\_header\_extension\_present\_flag
  - VdpPictureInfoHEVC, [118](#)
- source\_rect
  - VdpLayer, [105](#)
- source\_surface
  - VdpLayer, [105](#)
- sps\_max\_dec\_pic\_buffering\_minus1
  - VdpPictureInfoHEVC, [118](#)
- sps\_temporal\_mvp\_enabled\_flag
  - VdpPictureInfoHEVC, [118](#)
- strong\_intra\_smoothing\_enabled\_flag
  - VdpPictureInfoHEVC, [119](#)
- struct\_version
  - VdpBitstreamBuffer, [103](#)
  - VdpLayer, [105](#)
  - VdpOutputSurfaceRenderBlendState, [106](#)



- VdpProcamp, [128](#)
- surface
  - VdpReferenceFrameH264, [130](#)
- syncmarker
  - VdpPictureInfoVC1, [126](#)
- tfcntrflag
  - VdpPictureInfoVC1, [126](#)
- tiles\_enabled\_flag
  - VdpPictureInfoHEVC, [119](#)
- top\_field\_first
  - VdpPictureInfoMPEG1Or2, [121](#)
  - VdpPictureInfoMPEG4Part2, [122](#)
- top\_is\_reference
  - VdpReferenceFrameH264, [130](#)
- transform\_8x8\_mode\_flag
  - VdpPictureInfoH264, [109](#)
- transform\_skip\_enabled\_flag
  - VdpPictureInfoHEVC, [119](#)
- transquant\_bypass\_enabled\_flag
  - VdpPictureInfoHEVC, [119](#)
- trb
  - VdpPictureInfoMPEG4Part2, [122](#)
- trd
  - VdpPictureInfoMPEG4Part2, [122](#)
- uniform\_spacing\_flag
  - VdpPictureInfoHEVC, [119](#)
- VDP\_BITSTREAM\_BUFFER\_VERSION
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_CHROMA\_TYPE\_420
  - Miscellaneous Types, [24](#)
- VDP\_CHROMA\_TYPE\_422
  - Miscellaneous Types, [24](#)
- VDP\_CHROMA\_TYPE\_444
  - Miscellaneous Types, [24](#)
- VDP\_COLOR\_STANDARD\_ITUR\_BT\_601
  - VdpCSCMatrix; CSC Matrix Manipulation, [36](#)
- VDP\_COLOR\_STANDARD\_ITUR\_BT\_709
  - VdpCSCMatrix; CSC Matrix Manipulation, [36](#)
- VDP\_COLOR\_STANDARD\_SMPTE\_240M
  - VdpCSCMatrix; CSC Matrix Manipulation, [36](#)
- VDP\_COLOR\_TABLE\_FORMAT\_B8G8R8X8
  - VdpOutputSurface; Output Surfaceobject, [44](#)
- VDP\_DECODER\_LEVEL\_DIVX\_NA
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_1
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_1\_1
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_1\_2
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_1\_3
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_1b
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_2
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_2\_1
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_2\_2
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_3
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_3\_1
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_3\_2
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_4
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_4\_1
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_4\_2
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_5
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_H264\_5\_1
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_HEVC\_1
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_HEVC\_2
  - VdpDecoder; Video Decoding object, [63](#)
- VDP\_DECODER\_LEVEL\_HEVC\_2\_1
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_HEVC\_3
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_HEVC\_3\_1
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_HEVC\_4
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_HEVC\_4\_1
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_HEVC\_5
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_HEVC\_5\_1
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_HEVC\_5\_2
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_HEVC\_6
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_HEVC\_6\_1
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_HEVC\_6\_2
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_MPEG1\_NA
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_MPEG2\_HL14
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_MPEG2\_HL
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_MPEG2\_LL
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_MPEG2\_ML
  - VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_ASP\_L0
  - VdpDecoder; Video Decoding object, [64](#)



- VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_ASP\_L1  
VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_ASP\_L2  
VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_ASP\_L3  
VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_ASP\_L4  
VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_ASP\_L5  
VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_SP\_L0  
VdpDecoder; Video Decoding object, [64](#)
- VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_SP\_L1  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_SP\_L2  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_SP\_L3  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L0  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L1  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L2  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L3  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L4  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_LEVEL\_VC1\_MAIN\_HIGH  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_LEVEL\_VC1\_MAIN\_LOW  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_LEVEL\_VC1\_MAIN\_MEDIUM  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_LEVEL\_VC1\_SIMPLE\_LOW  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_LEVEL\_VC1\_SIMPLE\_MEDIUM  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_PROFILE\_DIVX4\_HD\_1080P  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_PROFILE\_DIVX4\_HOME\_THEATER  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_PROFILE\_DIVX4\_MOBILE  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_PROFILE\_DIVX4\_QMOBILE  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_PROFILE\_DIVX5\_HD\_1080P  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_PROFILE\_DIVX5\_HOME\_THEATER  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_PROFILE\_DIVX5\_MOBILE  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_PROFILE\_DIVX5\_QMOBILE  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_PROFILE\_H264\_BASELINE  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_PROFILE\_H264\_CONSTRAINED↔  
\_BASELINE  
VdpDecoder; Video Decoding object, [65](#)
- VDP\_DECODER\_PROFILE\_H264\_CONSTRAINED↔  
\_HIGH  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_H264\_EXTENDED  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_H264\_HIGH\_444\_PRE↔  
DICTIVE  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_H264\_HIGH  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_H264\_MAIN  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_H264\_PROGRESSIVE↔  
\_HIGH  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_10  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_12  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_444  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_STILL  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_HEVC\_MAIN  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_MPEG1  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_MPEG2\_MAIN  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_MPEG2\_SIMPLE  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_MPEG4\_PART2\_ASP  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_MPEG4\_PART2\_SP  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_VC1\_ADVANCED  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_VC1\_MAIN  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_DECODER\_PROFILE\_VC1\_SIMPLE  
VdpDecoder; Video Decoding object, [66](#)
- VDP\_FALSE  
Basic Types, [22](#)
- VDP\_FUNC\_ID\_BASE\_WINSYS  
Entry Point Retrieval, [94](#)
- VDP\_FUNC\_ID\_BITMAP\_SURFACE\_CREATE  
Entry Point Retrieval, [94](#)
- VDP\_FUNC\_ID\_BITMAP\_SURFACE\_DESTROY  
Entry Point Retrieval, [94](#)
- VDP\_FUNC\_ID\_BITMAP\_SURFACE\_GET\_PARAM↔  
ETERS  
Entry Point Retrieval, [94](#)
- VDP\_FUNC\_ID\_BITMAP\_SURFACE\_PUT\_BITS\_N↔  
ATIVE  
Entry Point Retrieval, [94](#)
- VDP\_FUNC\_ID\_BITMAP\_SURFACE\_QUERY\_CAP↔  
ABILITIES

- Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_DECODER\_CREATE
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_DECODER\_DESTROY
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_DECODER\_GET\_PARAMETERS
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_DECODER\_QUERY\_CAPABILITIES
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_DECODER\_RENDER
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_DEVICE\_DESTROY
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_GENERATE\_CSC\_MATRIX
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_GET\_API\_VERSION
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_GET\_ERROR\_STRING
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_GET\_INFORMATION\_STRING
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_GET\_PROC\_ADDRESS
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_CREATE
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_DESTROY
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_GET\_BITS\_NATIVE
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_GET\_PARAMETERS
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_PUT\_BITS\_INDEXED
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_PUT\_BITS\_NATIVE
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_PUT\_BITS\_NATIVE\_CB\_CR
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_QUERY\_CAPABILITIES
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_QUERY\_GET\_PUT\_BITS\_NATIVE\_CAPABILITIES
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_QUERY\_PUT\_BITS\_INDEXED\_CAPABILITIES
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_QUERY\_PUT\_BITS\_Y\_CB\_CR\_CAPABILITIES
  - Entry Point Retrieval, [95](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_RENDER\_BITMAP\_SURFACE
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_RENDER\_OUTPUT\_SURFACE
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_OUTPUT\_SURFACE\_RENDER\_VIDEO\_SURFACE\_LUMA
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_PREEMPTION\_CALLBACK\_REGISTER
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_BLOCK\_UNTIL\_SURFACE\_IDLE
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_CREATE
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_DESTROY
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_DISPLAY
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_GET\_BACKGROUND\_COLOR
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_GET\_TILE\_ME
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_QUERY\_SURFACE\_STATUS
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_SET\_BACKGROUND\_COLOR
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_TARGET\_CREATE\_X11
  - X11 Window System Integration Layer, [100](#)
- VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_TARGET\_DESTROY
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_CREATE
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_DESTROY
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_GET\_ATTRIBUTE\_VALUES
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_GET\_FEATURE\_ENABLED
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_GET\_FEATURE\_SUPPORT
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_GET\_PARAMETER\_VALUES
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_QUERY\_ATTRIBUTE\_SUPPORT
  - Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_QUERY\_ATTRIBUTE\_VALUE\_RANGE
  - Entry Point Retrieval, [96](#)

- VDP\_FUNC\_ID\_VIDEO\_MIXER\_QUERY\_FEATURE↔  
\_SUPPORT  
Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_QUERY\_PARAMETERS↔  
\_SUPPORT  
Entry Point Retrieval, [96](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_QUERY\_PARAMETERS↔  
\_VALUE\_RANGE  
Entry Point Retrieval, [97](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_RENDER  
Entry Point Retrieval, [97](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_SET\_ATTRIBUTES↔  
\_VALUES  
Entry Point Retrieval, [97](#)
- VDP\_FUNC\_ID\_VIDEO\_MIXER\_SET\_FEATURES↔  
\_ENABLES  
Entry Point Retrieval, [97](#)
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_CREATE  
Entry Point Retrieval, [97](#)
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_DESTROY  
Entry Point Retrieval, [97](#)
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_GET\_BITS\_Y↔  
\_CB\_CR  
Entry Point Retrieval, [97](#)
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_GET\_PARAMETERS↔  
\_TERS  
Entry Point Retrieval, [97](#)
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_PUT\_BITS\_Y↔  
\_CB\_CR  
Entry Point Retrieval, [97](#)
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_QUERY\_CAPABILITIES↔  
\_ILITIES  
Entry Point Retrieval, [97](#)
- VDP\_FUNC\_ID\_VIDEO\_SURFACE\_QUERY\_GET↔  
\_PUT\_BITS\_Y\_CB\_CR\_CAPABILITIES  
Entry Point Retrieval, [97](#)
- VDP\_INDEXED\_FORMAT\_A4I4  
Miscellaneous Types, [24](#)
- VDP\_INDEXED\_FORMAT\_A8I8  
Miscellaneous Types, [24](#)
- VDP\_INDEXED\_FORMAT\_I4A4  
Miscellaneous Types, [25](#)
- VDP\_INDEXED\_FORMAT\_I8A8  
Miscellaneous Types, [25](#)
- VDP\_INVALID\_HANDLE  
Miscellaneous Types, [25](#)
- VDP\_LAYER\_VERSION  
VdpVideoMixer; Video Post-processing and Compositing object, [73](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_EQU↔  
\_ATION\_ADD  
VdpOutputSurface Rendering Functionality, [58](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_EQU↔  
\_ATION\_MAX  
VdpOutputSurface Rendering Functionality, [58](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_EQU↔  
\_ATION\_MIN  
VdpOutputSurface Rendering Functionality, [58](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_EQU↔  
\_ATION\_REVERSE\_SUBTRACT  
VdpOutputSurface Rendering Functionality, [58](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_EQU↔  
\_ATION\_SUBTRACT  
VdpOutputSurface Rendering Functionality, [58](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_CONSTANT\_ALPHA  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_CONSTANT\_COLOR  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_DST\_ALPHA  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_DST\_COLOR  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_ONE\_MINUS\_CONSTANT\_ALPHA  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_ONE\_MINUS\_CONSTANT\_COLOR  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_ONE\_MINUS\_DST\_ALPHA  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_ONE\_MINUS\_DST\_COLOR  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_ONE\_MINUS\_SRC\_ALPHA  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_ONE\_MINUS\_SRC\_COLOR  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_ONE  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_SRC\_ALPHA\_SATURATE  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_SRC\_ALPHA  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_SRC\_COLOR  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_FACTOR↔  
\_TOR\_ZERO  
VdpOutputSurface Rendering Functionality, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_STATE↔  
\_TE\_VERSION  
VdpOutputSurface Rendering Functionality, [55](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_COLOR\_PER↔  
\_VERTEX  
VdpOutputSurface Rendering Functionality, [55](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_ROTATE\_0

- VdpOutputSurface Rendering Functionality, [55](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_ROTATE\_180
  - VdpOutputSurface Rendering Functionality, [55](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_ROTATE\_270
  - VdpOutputSurface Rendering Functionality, [55](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_ROTATE\_90
  - VdpOutputSurface Rendering Functionality, [55](#)
- VDP\_PRESENTATION\_QUEUE\_STATUS\_IDLE
  - VdpPresentationQueue; Video presentation (display) object, [90](#)
- VDP\_PRESENTATION\_QUEUE\_STATUS\_QUEUED
  - VdpPresentationQueue; Video presentation (display) object, [90](#)
- VDP\_PRESENTATION\_QUEUE\_STATUS\_VISIBLE
  - VdpPresentationQueue; Video presentation (display) object, [90](#)
- VDP\_PROCAMP\_VERSION
  - VdpCSCMatrix; CSC Matrix Manipulation, [36](#)
- VDP\_RGBA\_FORMAT\_A8
  - Miscellaneous Types, [25](#)
- VDP\_RGBA\_FORMAT\_B10G10R10A2
  - Miscellaneous Types, [25](#)
- VDP\_RGBA\_FORMAT\_B8G8R8A8
  - Miscellaneous Types, [26](#)
- VDP\_RGBA\_FORMAT\_R10G10B10A2
  - Miscellaneous Types, [26](#)
- VDP\_RGBA\_FORMAT\_R8G8B8A8
  - Miscellaneous Types, [26](#)
- VDP\_STATUS\_DISPLAY\_PREEMPTED
  - Error Handling, [30](#)
- VDP\_STATUS\_ERROR
  - Error Handling, [31](#)
- VDP\_STATUS\_HANDLE\_DEVICE\_MISMATCH
  - Error Handling, [31](#)
- VDP\_STATUS\_INVALID\_BLEND\_EQUATION
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_BLEND\_FACTOR
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_CHROMA\_TYPE
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_COLOR\_STANDARD
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_COLOR\_TABLE\_FORMAT
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_DECODER\_PROFILE
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_FLAG
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_FUNC\_ID
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_HANDLE
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_INDEXED\_FORMAT
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_POINTER
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_RGBA\_FORMAT
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_SIZE
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_STRUCT\_VERSION
  - Error Handling, [31](#)
- VDP\_STATUS\_INVALID\_VALUE
  - Error Handling, [31](#)
- VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_ATTRIBUTE
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_FEATURE
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_PARAMETER
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_VIDEO\_MIXER\_PICTURE\_STRUCTURE
  - Error Handling, [30](#)
- VDP\_STATUS\_INVALID\_Y\_CB\_CR\_FORMAT
  - Error Handling, [30](#)
- VDP\_STATUS\_NO\_IMPLEMENTATION
  - Error Handling, [30](#)
- VDP\_STATUS\_OK
  - Error Handling, [30](#)
- VDP\_STATUS\_RESOURCES
  - Error Handling, [31](#)
- VDP\_TRUE
  - Basic Types, [22](#)
- VDP\_VIDEO\_MIXER\_ATTRIBUTE\_BACKGROUND\_COLOR
  - VdpVideoMixer; Video Post-processing and Compositing object, [73](#)
- VDP\_VIDEO\_MIXER\_ATTRIBUTE\_CSC\_MATRIX
  - VdpVideoMixer; Video Post-processing and Compositing object, [73](#)
- VDP\_VIDEO\_MIXER\_ATTRIBUTE\_LUMA\_KEY\_MATRIX\_LUMA
  - VdpVideoMixer; Video Post-processing and Compositing object, [73](#)
- VDP\_VIDEO\_MIXER\_ATTRIBUTE\_LUMA\_KEY\_MATRIX\_LUMA
  - VdpVideoMixer; Video Post-processing and Compositing object, [74](#)
- VDP\_VIDEO\_MIXER\_ATTRIBUTE\_NOISE\_REDUCTION\_LEVEL
  - VdpVideoMixer; Video Post-processing and Compositing object, [74](#)
- VDP\_VIDEO\_MIXER\_ATTRIBUTE\_SHARPNESS\_LEVEL
  - VdpVideoMixer; Video Post-processing and Compositing object, [74](#)
- VDP\_VIDEO\_MIXER\_ATTRIBUTE\_SKIP\_CHROMA\_DEINTERLACE
  - VdpVideoMixer; Video Post-processing and Compositing object, [74](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_DEINTERLACE\_TEMPORAL\_SPATIAL
  - VdpVideoMixer; Video Post-processing and Compositing object, [75](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_DEINTERLACE\_TEMPORAL\_SPATIAL
  - VdpVideoMixer; Video Post-processing and Compositing object, [75](#)

- EMPORAL
- VdpVideoMixer; Video Post-processing and Compositing object, [74](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L1
  - VdpVideoMixer; Video Post-processing and Compositing object, [75](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L2
  - VdpVideoMixer; Video Post-processing and Compositing object, [75](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L3
  - VdpVideoMixer; Video Post-processing and Compositing object, [75](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L4
  - VdpVideoMixer; Video Post-processing and Compositing object, [75](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L5
  - VdpVideoMixer; Video Post-processing and Compositing object, [76](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L6
  - VdpVideoMixer; Video Post-processing and Compositing object, [76](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L7
  - VdpVideoMixer; Video Post-processing and Compositing object, [76](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L8
  - VdpVideoMixer; Video Post-processing and Compositing object, [76](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L9
  - VdpVideoMixer; Video Post-processing and Compositing object, [76](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_INVERSE\_TELECONVERSION
  - VdpVideoMixer; Video Post-processing and Compositing object, [76](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_LUMA\_KEY
  - VdpVideoMixer; Video Post-processing and Compositing object, [76](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_NOISE\_REDUCTION
  - VdpVideoMixer; Video Post-processing and Compositing object, [77](#)
- VDP\_VIDEO\_MIXER\_FEATURE\_SHARPNESS
  - VdpVideoMixer; Video Post-processing and Compositing object, [77](#)
- VDP\_VIDEO\_MIXER\_PARAMETER\_CHROMA\_TYPE
  - VdpVideoMixer; Video Post-processing and Compositing object, [77](#)
- VDP\_VIDEO\_MIXER\_PARAMETER\_LAYERS
  - VdpVideoMixer; Video Post-processing and Compositing object, [77](#)
- VDP\_VIDEO\_MIXER\_PARAMETER\_VIDEO\_SURFACE\_HEIGHT
  - VdpVideoMixer; Video Post-processing and Compositing object, [77](#)
- VDP\_VIDEO\_MIXER\_PARAMETER\_VIDEO\_SURFACE\_WIDTH
  - VdpVideoMixer; Video Post-processing and Compositing object, [78](#)
- VDP\_VIDEO\_MIXER\_PICTURE\_STRUCTURE\_BOTTOM\_FIELD
  - VdpVideoMixer; Video Post-processing and Compositing object, [84](#)
- VDP\_VIDEO\_MIXER\_PICTURE\_STRUCTURE\_FRAME\_ME
  - VdpVideoMixer; Video Post-processing and Compositing object, [84](#)
- VDP\_VIDEO\_MIXER\_PICTURE\_STRUCTURE\_TOP\_FIELD
  - VdpVideoMixer; Video Post-processing and Compositing object, [84](#)
- VDP\_YCBCR\_FORMAT\_NV12
  - Miscellaneous Types, [26](#)
- VDP\_YCBCR\_FORMAT\_UYVY
  - Miscellaneous Types, [26](#)
- VDP\_YCBCR\_FORMAT\_V8U8Y8A8
  - Miscellaneous Types, [27](#)
- VDP\_YCBCR\_FORMAT\_Y8U8V8A8
  - Miscellaneous Types, [27](#)
- VDP\_YCBCR\_FORMAT\_YUYV
  - Miscellaneous Types, [27](#)
- VDP\_YCBCR\_FORMAT\_YV12
  - Miscellaneous Types, [27](#)
- VDPAU\_INTERFACE\_VERSION
  - Versioning, [32](#)
- VDPAU\_VERSION
  - Versioning, [32](#)
- vdp\_device\_create\_x11
  - X11 Window System Integration Layer, [101](#)
- VdpBitmapSurface
  - VdpBitmapSurface; Bitmap Surfaceobject, [51](#)
- VdpBitmapSurface; Bitmap Surfaceobject, [50](#)
  - VdpBitmapSurface, [51](#)
  - VdpBitmapSurfaceCreate, [51](#)
  - VdpBitmapSurfaceDestroy, [51](#)
  - VdpBitmapSurfaceGetParameters, [51](#)
  - VdpBitmapSurfacePutBitsNative, [52](#)
  - VdpBitmapSurfaceQueryCapabilities, [52](#)
- VdpBitmapSurfaceCreate
  - VdpBitmapSurface; Bitmap Surfaceobject, [51](#)
- VdpBitmapSurfaceDestroy
  - VdpBitmapSurface; Bitmap Surfaceobject, [51](#)
- VdpBitmapSurfaceGetParameters
  - VdpBitmapSurface; Bitmap Surfaceobject, [51](#)
- VdpBitmapSurfacePutBitsNative
  - VdpBitmapSurface; Bitmap Surfaceobject, [52](#)
- VdpBitmapSurfaceQueryCapabilities
  - VdpBitmapSurface; Bitmap Surfaceobject, [52](#)



- VdpBitstreamBuffer, 103
  - bitstream, 103
  - bitstream\_bytes, 103
  - struct\_version, 103
- VdpBool
  - Basic Types, 22
- VdpCSCMatrix
  - VdpCSCMatrix; CSC Matrix Manipulation, 36
- VdpCSCMatrix; CSC Matrix Manipulation, 35
  - VDP\_COLOR\_STANDARD\_ITUR\_BT\_601, 36
  - VDP\_COLOR\_STANDARD\_ITUR\_BT\_709, 36
  - VDP\_COLOR\_STANDARD\_SMPTE\_240M, 36
  - VDP\_PROCAMP\_VERSION, 36
  - VdpCSCMatrix, 36
  - VdpColorStandard, 36
  - VdpGenerateCSCMatrix, 36
- VdpChromaType
  - Miscellaneous Types, 28
- VdpColor, 104
  - alpha, 104
  - blue, 104
  - green, 104
  - red, 104
- VdpColorStandard
  - VdpCSCMatrix; CSC Matrix Manipulation, 36
- VdpColorTableFormat
  - VdpOutputSurface; Output Surfaceobject, 44
- VdpDecoder
  - VdpDecoder; Video Decoding object, 66
- VdpDecoder; Video Decoding object, 60
  - VDP\_BITSTREAM\_BUFFER\_VERSION, 63
  - VDP\_DECODER\_LEVEL\_DIVX\_NA, 63
  - VDP\_DECODER\_LEVEL\_H264\_1, 63
  - VDP\_DECODER\_LEVEL\_H264\_1\_1, 63
  - VDP\_DECODER\_LEVEL\_H264\_1\_2, 63
  - VDP\_DECODER\_LEVEL\_H264\_1\_3, 63
  - VDP\_DECODER\_LEVEL\_H264\_1b, 63
  - VDP\_DECODER\_LEVEL\_H264\_2, 63
  - VDP\_DECODER\_LEVEL\_H264\_2\_1, 63
  - VDP\_DECODER\_LEVEL\_H264\_2\_2, 63
  - VDP\_DECODER\_LEVEL\_H264\_3, 63
  - VDP\_DECODER\_LEVEL\_H264\_3\_1, 63
  - VDP\_DECODER\_LEVEL\_H264\_3\_2, 63
  - VDP\_DECODER\_LEVEL\_H264\_4, 63
  - VDP\_DECODER\_LEVEL\_H264\_4\_1, 63
  - VDP\_DECODER\_LEVEL\_H264\_4\_2, 63
  - VDP\_DECODER\_LEVEL\_H264\_5, 63
  - VDP\_DECODER\_LEVEL\_H264\_5\_1, 63
  - VDP\_DECODER\_LEVEL\_HEVC\_1, 63
  - VDP\_DECODER\_LEVEL\_HEVC\_2, 63
  - VDP\_DECODER\_LEVEL\_HEVC\_2\_1, 64
  - VDP\_DECODER\_LEVEL\_HEVC\_3, 64
  - VDP\_DECODER\_LEVEL\_HEVC\_3\_1, 64
  - VDP\_DECODER\_LEVEL\_HEVC\_4, 64
  - VDP\_DECODER\_LEVEL\_HEVC\_4\_1, 64
  - VDP\_DECODER\_LEVEL\_HEVC\_5, 64
  - VDP\_DECODER\_LEVEL\_HEVC\_5\_1, 64
  - VDP\_DECODER\_LEVEL\_HEVC\_5\_2, 64
  - VDP\_DECODER\_LEVEL\_HEVC\_6, 64
  - VDP\_DECODER\_LEVEL\_HEVC\_6\_1, 64
  - VDP\_DECODER\_LEVEL\_HEVC\_6\_2, 64
  - VDP\_DECODER\_LEVEL\_MPEG1\_NA, 64
  - VDP\_DECODER\_LEVEL\_MPEG2\_HL14, 64
  - VDP\_DECODER\_LEVEL\_MPEG2\_HL, 64
  - VDP\_DECODER\_LEVEL\_MPEG2\_LL, 64
  - VDP\_DECODER\_LEVEL\_MPEG2\_ML, 64
  - VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_AS↵  
P\_L0, 64
  - VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_AS↵  
P\_L1, 64
  - VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_AS↵  
P\_L2, 64
  - VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_AS↵  
P\_L3, 64
  - VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_AS↵  
P\_L4, 64
  - VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_AS↵  
P\_L5, 64
  - VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_SP↵  
\_L0, 64
  - VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_SP↵  
\_L1, 65
  - VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_SP↵  
\_L2, 65
  - VDP\_DECODER\_LEVEL\_MPEG4\_PART2\_SP↵  
\_L3, 65
  - VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L0, 65
  - VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L1, 65
  - VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L2, 65
  - VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L3, 65
  - VDP\_DECODER\_LEVEL\_VC1\_ADVANCED\_L4, 65
  - VDP\_DECODER\_LEVEL\_VC1\_MAIN\_HIGH, 65
  - VDP\_DECODER\_LEVEL\_VC1\_MAIN\_LOW, 65
  - VDP\_DECODER\_LEVEL\_VC1\_MAIN\_MEDIUM, 65
  - VDP\_DECODER\_LEVEL\_VC1\_SIMPLE\_LOW, 65
  - VDP\_DECODER\_LEVEL\_VC1\_SIMPLE\_MEDI↵  
UM, 65
  - VDP\_DECODER\_PROFILE\_DIVX4\_HD\_1080P, 65
  - VDP\_DECODER\_PROFILE\_DIVX4\_HOME\_TH↵  
EATER, 65
  - VDP\_DECODER\_PROFILE\_DIVX4\_MOBILE, 65
  - VDP\_DECODER\_PROFILE\_DIVX4\_QMOBILE, 65
  - VDP\_DECODER\_PROFILE\_DIVX5\_HD\_1080P, 65
  - VDP\_DECODER\_PROFILE\_DIVX5\_HOME\_TH↵  
EATER, 65
  - VDP\_DECODER\_PROFILE\_DIVX5\_MOBILE, 65
  - VDP\_DECODER\_PROFILE\_DIVX5\_QMOBILE, 65

- 65
- VDP\_DECODER\_PROFILE\_H264\_BASELINE, 65
- VDP\_DECODER\_PROFILE\_H264\_CONSTRAI↔  
NED\_BASELINE, 65
- VDP\_DECODER\_PROFILE\_H264\_CONSTRAI↔  
NED\_HIGH, 66
- VDP\_DECODER\_PROFILE\_H264\_EXTENDED,  
66
- VDP\_DECODER\_PROFILE\_H264\_HIGH\_444\_↔  
PREDICTIVE, 66
- VDP\_DECODER\_PROFILE\_H264\_HIGH, 66
- VDP\_DECODER\_PROFILE\_H264\_MAIN, 66
- VDP\_DECODER\_PROFILE\_H264\_PROGRES↔  
SIVE\_HIGH, 66
- VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_10, 66
- VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_12, 66
- VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_444,  
66
- VDP\_DECODER\_PROFILE\_HEVC\_MAIN\_STILL,  
66
- VDP\_DECODER\_PROFILE\_HEVC\_MAIN, 66
- VDP\_DECODER\_PROFILE\_MPEG1, 66
- VDP\_DECODER\_PROFILE\_MPEG2\_MAIN, 66
- VDP\_DECODER\_PROFILE\_MPEG2\_SIMPLE, 66
- VDP\_DECODER\_PROFILE\_MPEG4\_PART2\_A↔  
SP, 66
- VDP\_DECODER\_PROFILE\_MPEG4\_PART2\_SP,  
66
- VDP\_DECODER\_PROFILE\_VC1\_ADVANCED,  
66
- VDP\_DECODER\_PROFILE\_VC1\_MAIN, 66
- VDP\_DECODER\_PROFILE\_VC1\_SIMPLE, 66
- VdpDecoder, 66
- VdpDecoderCreate, 66
- VdpDecoderDestroy, 67
- VdpDecoderGetParameters, 67
- VdpDecoderProfile, 67
- VdpDecoderQueryCapabilities, 67
- VdpDecoderRender, 68
- VdpPictureInfo, 68
- VdpPictureInfoDivX4, 68
- VdpPictureInfoDivX5, 69
- VdpDecoderCreate
  - VdpDecoder; Video Decoding object, 66
- VdpDecoderDestroy
  - VdpDecoder; Video Decoding object, 67
- VdpDecoderGetParameters
  - VdpDecoder; Video Decoding object, 67
- VdpDecoderProfile
  - VdpDecoder; Video Decoding object, 67
- VdpDecoderQueryCapabilities
  - VdpDecoder; Video Decoding object, 67
- VdpDecoderRender
  - VdpDecoder; Video Decoding object, 68
- VdpDevice
  - VdpDevice; Primary API object, 34
- VdpDevice; Primary API object, 34
- VdpDevice; Primary API object, 34
- VdpDeviceDestroy, 34
- VdpDeviceCreateX11
  - X11 Window System Integration Layer, 100
- VdpDeviceDestroy
  - VdpDevice; Primary API object, 34
- VdpFuncId
  - Entry Point Retrieval, 97
- VdpGenerateCSCMatrix
  - VdpCSCMatrix; CSC Matrix Manipulation, 36
- VdpGetApiVersion
  - Versioning, 33
- VdpGetErrorString
  - Error Handling, 29
- VdpGetInformationString
  - Versioning, 33
- VdpGetProcAddress
  - Entry Point Retrieval, 97
- VdpIndexedFormat
  - Miscellaneous Types, 28
- VdpLayer, 104
  - destination\_rect, 105
  - source\_rect, 105
  - source\_surface, 105
  - struct\_version, 105
- VdpOutputSurface
  - VdpOutputSurface; Output Surfaceobject, 44
- VdpOutputSurface Rendering Functionality, 54
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
EQUATION\_ADD, 58
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
EQUATION\_MAX, 58
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
EQUATION\_MIN, 58
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
EQUATION\_REVERSE\_SUBTRACT, 58
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
EQUATION\_SUBTRACT, 58
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_CONSTANT\_ALPHA, 59
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_CONSTANT\_COLOR, 59
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_DST\_ALPHA, 59
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_DST\_COLOR, 59
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_ONE\_MINUS\_CONSTANT\_ALP↔  
HA, 59
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_ONE\_MINUS\_CONSTANT\_COL↔  
OR, 59
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_ONE\_MINUS\_DST\_ALPHA, 59
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_ONE\_MINUS\_DST\_COLOR, 59
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_ONE\_MINUS\_SRC\_ALPHA, 59
  - VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔

- FACTOR\_ONE\_MINUS\_SRC\_COLOR, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_ONE, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_SRC\_ALPHA\_SATURATE, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_SRC\_ALPHA, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_SRC\_COLOR, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
FACTOR\_ZERO, [59](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_BLEND\_↔  
STATE\_VERSION, [55](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_COLOR\_↔  
PER\_VERTEX, [55](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_ROTATE\_↔  
\_0, [55](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_ROTATE\_↔  
\_180, [55](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_ROTATE\_↔  
\_270, [55](#)
- VDP\_OUTPUT\_SURFACE\_RENDER\_ROTATE\_↔  
\_90, [55](#)
- VdpOutputSurfaceRenderBitmapSurface, [56](#)
- VdpOutputSurfaceRenderBlendEquation, [58](#)
- VdpOutputSurfaceRenderBlendFactor, [58](#)
- VdpOutputSurfaceRenderOutputSurface, [57](#)
- VdpOutputSurface; Output Surfaceobject, [43](#)
- VDP\_COLOR\_TABLE\_FORMAT\_B8G8R8X8, [44](#)
- VdpColorTableFormat, [44](#)
- VdpOutputSurface, [44](#)
- VdpOutputSurfaceCreate, [45](#)
- VdpOutputSurfaceDestroy, [45](#)
- VdpOutputSurfaceGetBitsNative, [45](#)
- VdpOutputSurfaceGetParameters, [46](#)
- VdpOutputSurfacePutBitsIndexed, [46](#)
- VdpOutputSurfacePutBitsNative, [47](#)
- VdpOutputSurfacePutBitsYCbCr, [47](#)
- VdpOutputSurfaceQueryCapabilities, [48](#)
- VdpOutputSurfaceQueryGetPutBitsNative↔  
Capabilities, [48](#)
- VdpOutputSurfaceQueryPutBitsIndexedCapabilities,  
[48](#)
- VdpOutputSurfaceQueryPutBitsYCbCrCapabilities,  
[49](#)
- VdpOutputSurfaceCreate  
VdpOutputSurface; Output Surfaceobject, [45](#)
- VdpOutputSurfaceDestroy  
VdpOutputSurface; Output Surfaceobject, [45](#)
- VdpOutputSurfaceGetBitsNative  
VdpOutputSurface; Output Surfaceobject, [45](#)
- VdpOutputSurfaceGetParameters  
VdpOutputSurface; Output Surfaceobject, [46](#)
- VdpOutputSurfacePutBitsIndexed  
VdpOutputSurface; Output Surfaceobject, [46](#)
- VdpOutputSurfacePutBitsNative  
VdpOutputSurface; Output Surfaceobject, [47](#)
- VdpOutputSurfacePutBitsYCbCr  
VdpOutputSurface; Output Surfaceobject, [47](#)
- VdpOutputSurfaceQueryCapabilities  
VdpOutputSurface; Output Surfaceobject, [48](#)
- VdpOutputSurfaceQueryGetPutBitsNativeCapabilities  
VdpOutputSurface; Output Surfaceobject, [48](#)
- VdpOutputSurfaceQueryPutBitsIndexedCapabilities  
VdpOutputSurface; Output Surfaceobject, [48](#)
- VdpOutputSurfaceQueryPutBitsYCbCrCapabilities  
VdpOutputSurface; Output Surfaceobject, [49](#)
- VdpOutputSurfaceRenderBitmapSurface  
VdpOutputSurface Rendering Functionality, [56](#)
- VdpOutputSurfaceRenderBlendEquation  
VdpOutputSurface Rendering Functionality, [58](#)
- VdpOutputSurfaceRenderBlendFactor  
VdpOutputSurface Rendering Functionality, [58](#)
- VdpOutputSurfaceRenderBlendState, [105](#)  
blend\_constant, [106](#)  
blend\_equation\_alpha, [106](#)  
blend\_equation\_color, [106](#)  
blend\_factor\_destination\_alpha, [106](#)  
blend\_factor\_destination\_color, [106](#)  
blend\_factor\_source\_alpha, [106](#)  
blend\_factor\_source\_color, [106](#)  
struct\_version, [106](#)
- VdpOutputSurfaceRenderOutputSurface  
VdpOutputSurface Rendering Functionality, [57](#)
- VdpPictureInfo  
VdpDecoder; Video Decoding object, [68](#)
- VdpPictureInfoDivX4  
VdpDecoder; Video Decoding object, [68](#)
- VdpPictureInfoDivX5  
VdpDecoder; Video Decoding object, [69](#)
- VdpPictureInfoH264, [106](#)  
bottom\_field\_flag, [108](#)  
chroma\_qp\_index\_offset, [108](#)  
constrained\_intra\_pred\_flag, [108](#)  
deblocking\_filter\_control\_present\_flag, [108](#)  
delta\_pic\_order\_always\_zero\_flag, [108](#)  
direct\_8x8\_inference\_flag, [108](#)  
entropy\_coding\_mode\_flag, [108](#)  
field\_order\_cnt, [108](#)  
field\_pic\_flag, [108](#)  
frame\_mbs\_only\_flag, [108](#)  
frame\_num, [108](#)  
is\_reference, [108](#)  
log2\_max\_frame\_num\_minus4, [108](#)  
log2\_max\_pic\_order\_cnt\_lsb\_minus4, [108](#)  
mb\_adaptive\_frame\_field\_flag, [108](#)  
num\_ref\_frames, [108](#)  
num\_ref\_idx\_l0\_active\_minus1, [108](#)  
num\_ref\_idx\_l1\_active\_minus1, [108](#)  
pic\_init\_qp\_minus26, [108](#)  
pic\_order\_cnt\_type, [108](#)  
pic\_order\_present\_flag, [108](#)  
redundant\_pic\_cnt\_present\_flag, [108](#)  
referenceFrames, [108](#)  
scaling\_lists\_4x4, [108](#)  
scaling\_lists\_8x8, [109](#)



- second\_chroma\_qp\_index\_offset, 109
- slice\_count, 109
- transform\_8x8\_mode\_flag, 109
- weighted\_bipred\_idc, 109
- weighted\_pred\_flag, 109
- VdpPictureInfoH264Predictive, 109
  - pictureInfo, 110
  - qpprime\_y\_zero\_transform\_bypass\_flag, 110
  - separate\_colour\_plane\_flag, 110
- VdpPictureInfoHEVC, 110
  - amp\_enabled\_flag, 113
  - bit\_depth\_chroma\_minus8, 113
  - bit\_depth\_luma\_minus8, 113
  - cabac\_init\_present\_flag, 113
  - chroma\_format\_idc, 113
  - column\_width\_minus1, 113
  - constrained\_intra\_pred\_flag, 113
  - cu\_qp\_delta\_enabled\_flag, 113
  - CurrPicOrderCntVal, 113
  - CurrRpsIdx, 113
  - deblocking\_filter\_control\_present\_flag, 113
  - deblocking\_filter\_override\_enabled\_flag, 113
  - dependent\_slice\_segments\_enabled\_flag, 113
  - diff\_cu\_qp\_delta\_depth, 113
  - entropy\_coding\_sync\_enabled\_flag, 113
  - IDRPicFlag, 114
  - init\_qp\_minus26, 114
  - IsLongTerm, 114
  - lists\_modification\_present\_flag, 114
  - log2\_diff\_max\_min\_luma\_coding\_block\_size, 114
  - log2\_diff\_max\_min\_pcm\_luma\_coding\_block\_size, 114
  - log2\_diff\_max\_min\_transform\_block\_size, 114
  - log2\_max\_pic\_order\_cnt\_lsb\_minus4, 114
  - log2\_min\_luma\_coding\_block\_size\_minus3, 114
  - log2\_min\_pcm\_luma\_coding\_block\_size\_minus3, 114
  - log2\_min\_transform\_block\_size\_minus2, 114
  - log2\_parallel\_merge\_level\_minus2, 114
  - long\_term\_ref\_pics\_present\_flag, 114
  - loop\_filter\_across\_tiles\_enabled\_flag, 114
  - max\_transform\_hierarchy\_depth\_inter, 114
  - max\_transform\_hierarchy\_depth\_intra, 115
  - num\_extra\_slice\_header\_bits, 115
  - num\_long\_term\_ref\_pics\_sps, 115
  - num\_ref\_idx\_l0\_default\_active\_minus1, 115
  - num\_ref\_idx\_l1\_default\_active\_minus1, 115
  - num\_short\_term\_ref\_pic\_sets, 115
  - num\_tile\_columns\_minus1, 115
  - num\_tile\_rows\_minus1, 115
  - NumDeltaPocsOfRefRpsIdx, 115
  - NumLongTermPictureSliceHeaderBits, 115
  - NumPocLtCurr, 115
  - NumPocStCurrAfter, 115
  - NumPocStCurrBefore, 116
  - NumPocTotalCurr, 116
  - NumShortTermPictureSliceHeaderBits, 116
  - output\_flag\_present\_flag, 116
  - pcm\_enabled\_flag, 116
  - pcm\_loop\_filter\_disabled\_flag, 116
  - pcm\_sample\_bit\_depth\_chroma\_minus1, 116
  - pcm\_sample\_bit\_depth\_luma\_minus1, 116
  - pic\_height\_in\_luma\_samples, 116
  - pic\_width\_in\_luma\_samples, 116
  - PicOrderCntVal, 116
  - pps\_beta\_offset\_div2, 116
  - pps\_cb\_qp\_offset, 117
  - pps\_cr\_qp\_offset, 117
  - pps\_deblocking\_filter\_disabled\_flag, 117
  - pps\_loop\_filter\_across\_slices\_enabled\_flag, 117
  - pps\_slice\_chroma\_qp\_offsets\_present\_flag, 117
  - pps\_tc\_offset\_div2, 117
  - RAPPicFlag, 117
  - RefPicSetLtCurr, 117
  - RefPicSetStCurrAfter, 117
  - RefPicSetStCurrBefore, 117
  - RefPics, 117
  - row\_height\_minus1, 117
  - sample\_adaptive\_offset\_enabled\_flag, 118
  - scaling\_list\_enabled\_flag, 118
  - ScalingList16x16, 118
  - ScalingList32x32, 118
  - ScalingList4x4, 118
  - ScalingList8x8, 118
  - ScalingListDCCoeff16x16, 118
  - ScalingListDCCoeff32x32, 118
  - separate\_colour\_plane\_flag, 118
  - sign\_data\_hiding\_enabled\_flag, 118
  - slice\_segment\_header\_extension\_present\_flag, 118
  - sps\_max\_dec\_pic\_buffering\_minus1, 118
  - sps\_temporal\_mvp\_enabled\_flag, 118
  - strong\_intra\_smoothing\_enabled\_flag, 119
  - tiles\_enabled\_flag, 119
  - transform\_skip\_enabled\_flag, 119
  - transquant\_bypass\_enabled\_flag, 119
  - uniform\_spacing\_flag, 119
  - weighted\_bipred\_flag, 119
  - weighted\_pred\_flag, 119
  - VdpPictureInfoMPEG1Or2, 119
    - alternate\_scan, 120
    - backward\_reference, 120
    - concealment\_motion\_vectors, 120
    - f\_code, 120
    - forward\_reference, 120
    - frame\_pred\_frame\_dct, 120
    - full\_pel\_backward\_vector, 120
    - full\_pel\_forward\_vector, 120
    - intra\_dc\_precision, 120
    - intra\_quantizer\_matrix, 120
    - intra\_vlc\_format, 120
    - non\_intra\_quantizer\_matrix, 121
    - picture\_coding\_type, 121
    - picture\_structure, 121
    - q\_scale\_type, 121
    - slice\_count, 121

- top\_field\_first, 121
- VdpPictureInfoMPEG4Part2, 121
  - alternate\_vertical\_scan\_flag, 122
  - backward\_reference, 122
  - forward\_reference, 122
  - interlaced, 122
  - intra\_quantizer\_matrix, 122
  - non\_intra\_quantizer\_matrix, 122
  - quant\_type, 122
  - quarter\_sample, 122
  - resync\_marker\_disable, 122
  - rounding\_control, 122
  - short\_video\_header, 122
  - top\_field\_first, 122
  - trb, 122
  - trd, 122
  - vop\_coding\_type, 122
  - vop\_fcode\_backward, 122
  - vop\_fcode\_forward, 122
  - vop\_time\_increment\_resolution, 122
- VdpPictureInfoVC1, 123
  - backward\_reference, 123
  - deblockEnable, 123
  - dquant, 124
  - extended\_dmv, 124
  - extended\_mv, 124
  - fastuvmc, 124
  - finterpflag, 124
  - forward\_reference, 124
  - frame\_coding\_mode, 124
  - interlace, 124
  - loopfilter, 124
  - maxbframes, 124
  - multires, 125
  - overlap, 125
  - panscan\_flag, 125
  - picture\_type, 125
  - postprocflag, 125
  - pquant, 125
  - psf, 125
  - pulldown, 125
  - quantizer, 125
  - range\_mapuv, 125
  - range\_mapuv\_flag, 126
  - range\_mapy, 126
  - range\_mapy\_flag, 126
  - rangered, 126
  - refdist\_flag, 126
  - slice\_count, 126
  - syncmarker, 126
  - tfcntrflag, 126
  - vstransform, 126
- VdpPoint, 127
  - x, 127
  - y, 127
- VdpPreemptionCallback
  - Display Preemption, 91
- VdpPreemptionCallbackRegister
  - Display Preemption, 92
- VdpPresentationQueue
  - VdpPresentationQueue; Video presentation (display) object, 86
- VdpPresentationQueue; Video presentation (display) object, 85
  - VDP\_PRESENTATION\_QUEUE\_STATUS\_IDLE, 90
  - VDP\_PRESENTATION\_QUEUE\_STATUS\_QU↔EUED, 90
  - VDP\_PRESENTATION\_QUEUE\_STATUS\_VISI↔BLE, 90
- VdpPresentationQueue, 86
  - VdpPresentationQueueBlockUntilSurfaceIdle, 86
  - VdpPresentationQueueCreate, 87
  - VdpPresentationQueueDestroy, 87
  - VdpPresentationQueueDisplay, 87
  - VdpPresentationQueueGetBackgroundColor, 88
  - VdpPresentationQueueGetTime, 88
  - VdpPresentationQueueQuerySurfaceStatus, 88
  - VdpPresentationQueueSetBackgroundColor, 89
  - VdpPresentationQueueStatus, 90
  - VdpPresentationQueueTarget, 89
  - VdpPresentationQueueTargetDestroy, 89
  - VdpTime, 89
- VdpPresentationQueueBlockUntilSurfaceIdle
  - VdpPresentationQueue; Video presentation (display) object, 86
- VdpPresentationQueueCreate
  - VdpPresentationQueue; Video presentation (display) object, 87
- VdpPresentationQueueDestroy
  - VdpPresentationQueue; Video presentation (display) object, 87
- VdpPresentationQueueDisplay
  - VdpPresentationQueue; Video presentation (display) object, 87
- VdpPresentationQueueGetBackgroundColor
  - VdpPresentationQueue; Video presentation (display) object, 88
- VdpPresentationQueueGetTime
  - VdpPresentationQueue; Video presentation (display) object, 88
- VdpPresentationQueueQuerySurfaceStatus
  - VdpPresentationQueue; Video presentation (display) object, 88
- VdpPresentationQueueSetBackgroundColor
  - VdpPresentationQueue; Video presentation (display) object, 89
- VdpPresentationQueueStatus
  - VdpPresentationQueue; Video presentation (display) object, 90
- VdpPresentationQueueTarget
  - VdpPresentationQueue; Video presentation (display) object, 89
- VdpPresentationQueueTargetCreateX11
  - X11 Window System Integration Layer, 100
- VdpPresentationQueueTargetDestroy

- VdpPresentationQueue; Video presentation (display) object, [89](#)
- VdpProcamp, [127](#)
  - brightness, [128](#)
  - contrast, [128](#)
  - hue, [128](#)
  - saturation, [128](#)
  - struct\_version, [128](#)
- VdpRGBAFormat
  - Miscellaneous Types, [28](#)
- VdpRect, [128](#)
  - x0, [129](#)
  - x1, [129](#)
  - y0, [129](#)
  - y1, [129](#)
- VdpReferenceFrameH264, [129](#)
  - bottom\_is\_reference, [130](#)
  - field\_order\_cnt, [130](#)
  - frame\_idx, [130](#)
  - is\_long\_term, [130](#)
  - surface, [130](#)
  - top\_is\_reference, [130](#)
- VdpStatus
  - Error Handling, [30](#)
- VdpTime
  - VdpPresentationQueue; Video presentation (display) object, [89](#)
- VdpVideoMixer
  - VdpVideoMixer; Video Post-processing and Compositing object, [78](#)
- VdpVideoMixer; Video Post-processing and Compositing object, [70](#)
  - VDP\_LAYER\_VERSION, [73](#)
  - VDP\_VIDEO\_MIXER\_ATTRIBUTE\_BACKGROUND\_COLOR, [73](#)
  - VDP\_VIDEO\_MIXER\_ATTRIBUTE\_CSC\_MATRIX, [73](#)
  - VDP\_VIDEO\_MIXER\_ATTRIBUTE\_LUMA\_KEY\_MAX\_LUMA, [73](#)
  - VDP\_VIDEO\_MIXER\_ATTRIBUTE\_LUMA\_KEY\_MIN\_LUMA, [74](#)
  - VDP\_VIDEO\_MIXER\_ATTRIBUTE\_NOISE\_REDUCTION\_LEVEL, [74](#)
  - VDP\_VIDEO\_MIXER\_ATTRIBUTE\_SHARPNESS\_LEVEL, [74](#)
  - VDP\_VIDEO\_MIXER\_ATTRIBUTE\_SKIP\_CHROMA\_DEINTERLACE, [74](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_DEINTERLACE\_TEMPORAL\_SPATIAL, [75](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_DEINTERLACE\_TEMPORAL, [74](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L1, [75](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L2, [75](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L3, [75](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L4, [75](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L5, [76](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L6, [76](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L7, [76](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L8, [76](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_HIGH\_QUALITY\_SCALING\_L9, [76](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_INVERSE\_TELECINE, [76](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_LUMA\_KEY, [76](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_NOISE\_REDUCTION, [77](#)
  - VDP\_VIDEO\_MIXER\_FEATURE\_SHARPNESS, [77](#)
  - VDP\_VIDEO\_MIXER\_PARAMETER\_CHROMA\_TYPE, [77](#)
  - VDP\_VIDEO\_MIXER\_PARAMETER\_LAYERS, [77](#)
  - VDP\_VIDEO\_MIXER\_PARAMETER\_VIDEO\_SURFACE\_HEIGHT, [77](#)
  - VDP\_VIDEO\_MIXER\_PARAMETER\_VIDEO\_SURFACE\_WIDTH, [78](#)
  - VDP\_VIDEO\_MIXER\_PICTURE\_STRUCTURE\_BOTTOM\_FIELD, [84](#)
  - VDP\_VIDEO\_MIXER\_PICTURE\_STRUCTURE\_FRAME, [84](#)
  - VDP\_VIDEO\_MIXER\_PICTURE\_STRUCTURE\_TOP\_FIELD, [84](#)
- VdpVideoMixer, [78](#)
- VdpVideoMixerAttribute, [78](#)
- VdpVideoMixerCreate, [78](#)
- VdpVideoMixerDestroy, [79](#)
- VdpVideoMixerFeature, [79](#)
- VdpVideoMixerGetAttributeValues, [79](#)
- VdpVideoMixerGetFeatureEnables, [80](#)
- VdpVideoMixerGetFeatureSupport, [80](#)
- VdpVideoMixerGetParameterValues, [80](#)
- VdpVideoMixerParameter, [81](#)
- VdpVideoMixerPictureStructure, [84](#)
- VdpVideoMixerQueryAttributeSupport, [81](#)
- VdpVideoMixerQueryAttributeValueRange, [81](#)
- VdpVideoMixerQueryFeatureSupport, [81](#)
- VdpVideoMixerQueryParameterSupport, [82](#)
- VdpVideoMixerQueryParameterValueRange, [82](#)
- VdpVideoMixerRender, [82](#)
- VdpVideoMixerSetAttributeValues, [84](#)
- VdpVideoMixerSetFeatureEnables, [84](#)
- VdpVideoMixerAttribute
  - VdpVideoMixer; Video Post-processing and Compositing object, [78](#)
- VdpVideoMixerCreate
  - VdpVideoMixer; Video Post-processing and Compositing object, [78](#)
- VdpVideoMixerDestroy
  - VdpVideoMixer; Video Post-processing and Com-

- positing object, [79](#)
- VdpVideoMixerFeature
  - VdpVideoMixer; Video Post-processing and Compositing object, [79](#)
- VdpVideoMixerGetAttributeValues
  - VdpVideoMixer; Video Post-processing and Compositing object, [79](#)
- VdpVideoMixerGetFeatureEnables
  - VdpVideoMixer; Video Post-processing and Compositing object, [80](#)
- VdpVideoMixerGetFeatureSupport
  - VdpVideoMixer; Video Post-processing and Compositing object, [80](#)
- VdpVideoMixerGetParameterValues
  - VdpVideoMixer; Video Post-processing and Compositing object, [80](#)
- VdpVideoMixerParameter
  - VdpVideoMixer; Video Post-processing and Compositing object, [81](#)
- VdpVideoMixerPictureStructure
  - VdpVideoMixer; Video Post-processing and Compositing object, [84](#)
- VdpVideoMixerQueryAttributeSupport
  - VdpVideoMixer; Video Post-processing and Compositing object, [81](#)
- VdpVideoMixerQueryAttributeValueRange
  - VdpVideoMixer; Video Post-processing and Compositing object, [81](#)
- VdpVideoMixerQueryFeatureSupport
  - VdpVideoMixer; Video Post-processing and Compositing object, [81](#)
- VdpVideoMixerQueryParameterSupport
  - VdpVideoMixer; Video Post-processing and Compositing object, [82](#)
- VdpVideoMixerQueryParameterValueRange
  - VdpVideoMixer; Video Post-processing and Compositing object, [82](#)
- VdpVideoMixerRender
  - VdpVideoMixer; Video Post-processing and Compositing object, [82](#)
- VdpVideoMixerSetAttributeValues
  - VdpVideoMixer; Video Post-processing and Compositing object, [84](#)
- VdpVideoMixerSetFeatureEnables
  - VdpVideoMixer; Video Post-processing and Compositing object, [84](#)
- VdpVideoSurface
  - VdpVideoSurface; Video Surface object, [39](#)
- VdpVideoSurface; Video Surface object, [38](#)
  - VdpVideoSurface, [39](#)
  - VdpVideoSurfaceCreate, [39](#)
  - VdpVideoSurfaceDestroy, [40](#)
  - VdpVideoSurfaceGetBitsYCbCr, [40](#)
  - VdpVideoSurfaceGetParameters, [40](#)
  - VdpVideoSurfacePutBitsYCbCr, [41](#)
  - VdpVideoSurfaceQueryCapabilities, [41](#)
  - VdpVideoSurfaceQueryGetPutBitsYCbCr↔Capabilities, [41](#)
- VdpVideoSurfaceCreate
  - VdpVideoSurface; Video Surface object, [39](#)
- VdpVideoSurfaceDestroy
  - VdpVideoSurface; Video Surface object, [40](#)
- VdpVideoSurfaceGetBitsYCbCr
  - VdpVideoSurface; Video Surface object, [40](#)
- VdpVideoSurfaceGetParameters
  - VdpVideoSurface; Video Surface object, [40](#)
- VdpVideoSurfacePutBitsYCbCr
  - VdpVideoSurface; Video Surface object, [41](#)
- VdpVideoSurfaceQueryCapabilities
  - VdpVideoSurface; Video Surface object, [41](#)
- VdpVideoSurfaceQueryGetPutBitsYCbCrCapabilities
  - VdpVideoSurface; Video Surface object, [41](#)
- VdpYCbCrFormat
  - Miscellaneous Types, [28](#)
- vdpa/vdpau.h, [131](#)
- vdpa/vdpau\_x11.h, [143](#)
- Versioning, [32](#)
  - VDPAU\_INTERFACE\_VERSION, [32](#)
  - VDPAU\_VERSION, [32](#)
  - VdpGetApiVersion, [33](#)
  - VdpGetInformationString, [33](#)
- vop\_coding\_type
  - VdpPictureInfoMPEG4Part2, [122](#)
- vop\_fcode\_backward
  - VdpPictureInfoMPEG4Part2, [122](#)
- vop\_fcode\_forward
  - VdpPictureInfoMPEG4Part2, [122](#)
- vop\_time\_increment\_resolution
  - VdpPictureInfoMPEG4Part2, [122](#)
- vstransform
  - VdpPictureInfoVC1, [126](#)
- weighted\_bipred\_flag
  - VdpPictureInfoHEVC, [119](#)
- weighted\_bipred\_idc
  - VdpPictureInfoH264, [109](#)
- weighted\_pred\_flag
  - VdpPictureInfoH264, [109](#)
  - VdpPictureInfoHEVC, [119](#)
- Window System Integration Layer, [98](#)
- x
  - VdpPoint, [127](#)
- x0
  - VdpRect, [129](#)
- x1
  - VdpRect, [129](#)
- X11 Window System Integration Layer, [99](#)
  - VDP\_FUNC\_ID\_PRESENTATION\_QUEUE\_TARGET\_CREATE\_X11, [100](#)
  - RGET\_CREATE\_X11, [100](#)
  - vdpa\_device\_create\_x11, [101](#)
  - VdpDeviceCreateX11, [100](#)
  - VdpPresentationQueueTargetCreateX11, [100](#)
- y
  - VdpPoint, [127](#)
- y0

VdpRect, [129](#)  
y1  
VdpRect, [129](#)