



# **NVENC - NVIDIA Video Encoder API Reference Manual**

**May 2, 2017**

**Version 8.0**





# Contents

<b>1</b>	<b>Legal Notice</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Data Structure Index</b>	<b>5</b>
3.1	Data Structures . . . . .	5
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
4.1	NvEncodeAPI Data structures . . . . .	7
4.1.1	Define Documentation . . . . .	10
4.1.1.1	NV_ENC_PARAMS_RC_VBR_MINQP . . . . .	10
4.1.1.2	NV_ENC_PARAMS_RC_2_PASS_QUALITY . . . . .	10
4.1.1.3	NV_ENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP . . . . .	11
4.1.1.4	NV_ENC_PARAMS_RC_2_PASS_VBR . . . . .	11
4.1.1.5	NV_ENC_PARAMS_RC_CBR2 . . . . .	11
4.1.1.6	NV_ENC_CAPS_PARAM_VER . . . . .	11
4.1.1.7	NV_ENC_CREATE_INPUT_BUFFER_VER . . . . .	11
4.1.1.8	NV_ENC_CREATE_BITSTREAM_BUFFER_VER . . . . .	11
4.1.1.9	NV_ENC_CREATE_MV_BUFFER_VER . . . . .	11
4.1.1.10	NV_ENC_RC_PARAMS_VER . . . . .	11
4.1.1.11	NV_ENC_CONFIG_VER . . . . .	11
4.1.1.12	NV_ENC_INITIALIZE_PARAMS_VER . . . . .	11
4.1.1.13	NV_ENC_RECONFIGURE_PARAMS_VER . . . . .	11
4.1.1.14	NV_ENC_PRESET_CONFIG_VER . . . . .	12
4.1.1.15	NV_ENC_PIC_PARAMS_VER . . . . .	12
4.1.1.16	NV_ENC_MEONLY_PARAMS_VER . . . . .	12
4.1.1.17	NV_ENC_LOCK_BITSTREAM_VER . . . . .	12
4.1.1.18	NV_ENC_LOCK_INPUT_BUFFER_VER . . . . .	12

4.1.1.19	NV_ENC_MAP_INPUT_RESOURCE_VER	12
4.1.1.20	NV_ENC_REGISTER_RESOURCE_VER	12
4.1.1.21	NV_ENC_STAT_VER	12
4.1.1.22	NV_ENC_SEQUENCE_PARAM_PAYLOAD_VER	12
4.1.1.23	NV_ENC_EVENT_PARAMS_VER	12
4.1.1.24	NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS_VER	12
4.1.2	Enumeration Type Documentation	13
4.1.2.1	NV_ENC_PARAMS_FRAME_FIELD_MODE	13
4.1.2.2	NV_ENC_PARAMS_RC_MODE	13
4.1.2.3	NV_ENC_PIC_STRUCT	13
4.1.2.4	NV_ENC_PIC_TYPE	13
4.1.2.5	NV_ENC_MV_PRECISION	14
4.1.2.6	NV_ENC_BUFFER_FORMAT	14
4.1.2.7	NV_ENC_LEVEL	14
4.1.2.8	NVENCSTATUS	15
4.1.2.9	NV_ENC_PIC_FLAGS	16
4.1.2.10	NV_ENC_MEMORY_HEAP	16
4.1.2.11	NV_ENC_H264_ENTROPY_CODING_MODE	16
4.1.2.12	NV_ENC_H264_BDIRECT_MODE	17
4.1.2.13	NV_ENC_H264_FMO_MODE	17
4.1.2.14	NV_ENC_H264_ADAPTIVE_TRANSFORM_MODE	17
4.1.2.15	NV_ENC_STEREO_PACKING_MODE	17
4.1.2.16	NV_ENC_INPUT_RESOURCE_TYPE	18
4.1.2.17	NV_ENC_DEVICE_TYPE	18
4.1.2.18	NV_ENC_CAPS	18
4.1.2.19	NV_ENC_HEVC_CUSIZE	21
4.2	NvEncodeAPI Functions	22
4.2.1	Function Documentation	25
4.2.1.1	NvEncOpenEncodeSession	25
4.2.1.2	NvEncGetEncodeGUIDCount	25
4.2.1.3	NvEncGetEncodeGUIDs	25
4.2.1.4	NvEncGetEncodeProfileGUIDCount	26
4.2.1.5	NvEncGetEncodeProfileGUIDs	26
4.2.1.6	NvEncGetInputFormatCount	27
4.2.1.7	NvEncGetInputFormats	27
4.2.1.8	NvEncGetEncodeCaps	28
4.2.1.9	NvEncGetEncodePresetCount	28

4.2.1.10	NvEncGetEncodePresetGUIDs	29
4.2.1.11	NvEncGetEncodePresetConfig	29
4.2.1.12	NvEncInitializeEncoder	30
4.2.1.13	NvEncCreateInputBuffer	31
4.2.1.14	NvEncDestroyInputBuffer	32
4.2.1.15	NvEncCreateBitstreamBuffer	32
4.2.1.16	NvEncDestroyBitstreamBuffer	33
4.2.1.17	NvEncEncodePicture	33
4.2.1.18	NvEncLockBitstream	36
4.2.1.19	NvEncUnlockBitstream	36
4.2.1.20	NvEncLockInputBuffer	37
4.2.1.21	NvEncUnlockInputBuffer	37
4.2.1.22	NvEncGetEncodeStats	38
4.2.1.23	NvEncGetSequenceParams	38
4.2.1.24	NvEncRegisterAsyncEvent	39
4.2.1.25	NvEncUnregisterAsyncEvent	39
4.2.1.26	NvEncMapInputResource	40
4.2.1.27	NvEncUnmapInputResource	40
4.2.1.28	NvEncDestroyEncoder	41
4.2.1.29	NvEncInvalidateRefFrames	41
4.2.1.30	NvEncOpenEncodeSessionEx	42
4.2.1.31	NvEncRegisterResource	42
4.2.1.32	NvEncUnregisterResource	43
4.2.1.33	NvEncReconfigureEncoder	43
4.2.1.34	NvEncCreateMVBuffer	44
4.2.1.35	NvEncDestroyMVBuffer	44
4.2.1.36	NvEncRunMotionEstimationOnly	45
4.2.1.37	NvEncodeAPIGetMaxSupportedVersion	45
4.2.1.38	NvEncodeAPICreateInstance	46
<b>5</b>	<b>Data Structure Documentation</b>	<b>47</b>
5.1	GUID Struct Reference	47
5.1.1	Detailed Description	47
5.1.2	Field Documentation	47
5.1.2.1	Data1	47
5.1.2.2	Data2	47
5.1.2.3	Data3	47

5.1.2.4	Data4 . . . . .	47
5.2	NV_ENC_CAPS_PARAM Struct Reference . . . . .	48
5.2.1	Detailed Description . . . . .	48
5.2.2	Field Documentation . . . . .	48
5.2.2.1	version . . . . .	48
5.2.2.2	capsToQuery . . . . .	48
5.2.2.3	reserved . . . . .	48
5.3	NV_ENC_CODEC_CONFIG Union Reference . . . . .	49
5.3.1	Detailed Description . . . . .	49
5.3.2	Field Documentation . . . . .	49
5.3.2.1	h264Config . . . . .	49
5.3.2.2	hevcConfig . . . . .	49
5.3.2.3	h264MeOnlyConfig . . . . .	49
5.3.2.4	hevcMeOnlyConfig . . . . .	49
5.3.2.5	reserved . . . . .	49
5.4	NV_ENC_CODEC_PIC_PARAMS Union Reference . . . . .	50
5.4.1	Detailed Description . . . . .	50
5.4.2	Field Documentation . . . . .	50
5.4.2.1	h264PicParams . . . . .	50
5.4.2.2	hevcPicParams . . . . .	50
5.4.2.3	reserved . . . . .	50
5.5	NV_ENC_CONFIG Struct Reference . . . . .	51
5.5.1	Detailed Description . . . . .	51
5.5.2	Field Documentation . . . . .	51
5.5.2.1	version . . . . .	51
5.5.2.2	profileGUID . . . . .	51
5.5.2.3	gopLength . . . . .	51
5.5.2.4	frameIntervalP . . . . .	51
5.5.2.5	monoChromeEncoding . . . . .	51
5.5.2.6	frameFieldMode . . . . .	52
5.5.2.7	mvPrecision . . . . .	52
5.5.2.8	rcParams . . . . .	52
5.5.2.9	encodeCodecConfig . . . . .	52
5.5.2.10	reserved . . . . .	52
5.5.2.11	reserved2 . . . . .	52
5.6	NV_ENC_CONFIG_H264 Struct Reference . . . . .	53
5.6.1	Detailed Description . . . . .	54

5.6.2	Field Documentation	54
5.6.2.1	enableTemporalSVC	54
5.6.2.2	enableStereoMVC	54
5.6.2.3	hierarchicalPFrames	54
5.6.2.4	hierarchicalBFrames	54
5.6.2.5	outputBufferingPeriodSEI	54
5.6.2.6	outputPictureTimingSEI	54
5.6.2.7	outputAUD	54
5.6.2.8	disableSPSPPS	54
5.6.2.9	outputFramePackingSEI	54
5.6.2.10	outputRecoveryPointSEI	54
5.6.2.11	enableIntraRefresh	55
5.6.2.12	enableConstrainedEncoding	55
5.6.2.13	repeatSPSPPS	55
5.6.2.14	enableVFR	55
5.6.2.15	enableLTR	55
5.6.2.16	qpPrimeYZeroTransformBypassFlag	55
5.6.2.17	useConstrainedIntraPred	55
5.6.2.18	reservedBitFields	55
5.6.2.19	level	55
5.6.2.20	idrPeriod	56
5.6.2.21	separateColourPlaneFlag	56
5.6.2.22	disableDeblockingFilterIDC	56
5.6.2.23	numTemporalLayers	56
5.6.2.24	spsId	56
5.6.2.25	ppsId	56
5.6.2.26	adaptiveTransformMode	56
5.6.2.27	fmoMode	56
5.6.2.28	bdirectMode	56
5.6.2.29	entropyCodingMode	56
5.6.2.30	stereoMode	57
5.6.2.31	intraRefreshPeriod	57
5.6.2.32	intraRefreshCnt	57
5.6.2.33	maxNumRefFrames	57
5.6.2.34	sliceMode	57
5.6.2.35	sliceModeData	57
5.6.2.36	h264VUIParameters	57

5.6.2.37	ltrNumFrames	57
5.6.2.38	ltrTrustMode	58
5.6.2.39	chromaFormatIDC	58
5.6.2.40	maxTemporalLayers	58
5.6.2.41	reserved1	58
5.6.2.42	reserved2	58
5.7	NV_ENC_CONFIG_H264_MEONLY Struct Reference	59
5.7.1	Detailed Description	59
5.7.2	Field Documentation	59
5.7.2.1	disablePartition16x16	59
5.7.2.2	disablePartition8x16	59
5.7.2.3	disablePartition16x8	59
5.7.2.4	disablePartition8x8	59
5.7.2.5	disableIntraSearch	59
5.7.2.6	bStereoEnable	59
5.7.2.7	reserved	60
5.7.2.8	reserved1	60
5.7.2.9	reserved2	60
5.8	NV_ENC_CONFIG_H264_VUI_PARAMETERS Struct Reference	61
5.8.1	Detailed Description	61
5.8.2	Field Documentation	61
5.8.2.1	overscanInfoPresentFlag	61
5.8.2.2	overscanInfo	61
5.8.2.3	videoSignalTypePresentFlag	61
5.8.2.4	videoFormat	61
5.8.2.5	videoFullRangeFlag	61
5.8.2.6	colourDescriptionPresentFlag	62
5.8.2.7	colourPrimaries	62
5.8.2.8	transferCharacteristics	62
5.8.2.9	colourMatrix	62
5.8.2.10	chromaSampleLocationFlag	62
5.8.2.11	chromaSampleLocationTop	62
5.8.2.12	chromaSampleLocationBot	62
5.8.2.13	bitstreamRestrictionFlag	62
5.9	NV_ENC_CONFIG_HEVC Struct Reference	63
5.9.1	Detailed Description	63
5.9.2	Field Documentation	63



5.9.2.1	level	63
5.9.2.2	tier	64
5.9.2.3	minCUSize	64
5.9.2.4	maxCUSize	64
5.9.2.5	useConstrainedIntraPred	64
5.9.2.6	disableDeblockAcrossSliceBoundary	64
5.9.2.7	outputBufferingPeriodSEI	64
5.9.2.8	outputPictureTimingSEI	64
5.9.2.9	outputAUD	64
5.9.2.10	enableLTR	64
5.9.2.11	disableSPSPPS	64
5.9.2.12	repeatSPSPPS	65
5.9.2.13	enableIntraRefresh	65
5.9.2.14	chromaFormatIDC	65
5.9.2.15	pixelBitDepthMinus8	65
5.9.2.16	reserved	65
5.9.2.17	idrPeriod	65
5.9.2.18	intraRefreshPeriod	65
5.9.2.19	intraRefreshCnt	65
5.9.2.20	maxNumRefFramesInDPB	65
5.9.2.21	ltrNumFrames	65
5.9.2.22	vpsId	66
5.9.2.23	spsId	66
5.9.2.24	ppsId	66
5.9.2.25	sliceMode	66
5.9.2.26	sliceModeData	66
5.9.2.27	maxTemporalLayersMinus1	66
5.9.2.28	hevcVUIParameters	66
5.9.2.29	ltrTrustMode	66
5.9.2.30	reserved1	66
5.9.2.31	reserved2	66
5.10	NV_ENC_CONFIG_HEVC_MEONLY Struct Reference	67
5.10.1	Detailed Description	67
5.10.2	Field Documentation	67
5.10.2.1	reserved	67
5.10.2.2	reserved1	67
5.11	NV_ENC_CREATE_BITSTREAM_BUFFER Struct Reference	68

5.11.1 Detailed Description . . . . .	68
5.11.2 Field Documentation . . . . .	68
5.11.2.1 version . . . . .	68
5.11.2.2 size . . . . .	68
5.11.2.3 memoryHeap . . . . .	68
5.11.2.4 reserved . . . . .	68
5.11.2.5 bitstreamBuffer . . . . .	68
5.11.2.6 bitstreamBufferPtr . . . . .	68
5.11.2.7 reserved1 . . . . .	69
5.11.2.8 reserved2 . . . . .	69
5.12 NV_ENC_CREATE_INPUT_BUFFER Struct Reference . . . . .	70
5.12.1 Detailed Description . . . . .	70
5.12.2 Field Documentation . . . . .	70
5.12.2.1 version . . . . .	70
5.12.2.2 width . . . . .	70
5.12.2.3 height . . . . .	70
5.12.2.4 memoryHeap . . . . .	70
5.12.2.5 bufferFmt . . . . .	70
5.12.2.6 reserved . . . . .	70
5.12.2.7 inputBuffer . . . . .	71
5.12.2.8 pSysMemBuffer . . . . .	71
5.12.2.9 reserved1 . . . . .	71
5.12.2.10 reserved2 . . . . .	71
5.13 NV_ENC_CREATE_MV_BUFFER Struct Reference . . . . .	72
5.13.1 Detailed Description . . . . .	72
5.13.2 Field Documentation . . . . .	72
5.13.2.1 version . . . . .	72
5.13.2.2 mvBuffer . . . . .	72
5.13.2.3 reserved1 . . . . .	72
5.13.2.4 reserved2 . . . . .	72
5.14 NV_ENC_EVENT_PARAMS Struct Reference . . . . .	73
5.14.1 Detailed Description . . . . .	73
5.14.2 Field Documentation . . . . .	73
5.14.2.1 version . . . . .	73
5.14.2.2 reserved . . . . .	73
5.14.2.3 completionEvent . . . . .	73
5.14.2.4 reserved1 . . . . .	73

5.14.2.5 reserved2 . . . . .	73
5.15 NV_ENC_H264_MV_DATA Struct Reference . . . . .	74
5.15.1 Detailed Description . . . . .	74
5.15.2 Field Documentation . . . . .	74
5.15.2.1 mv . . . . .	74
5.15.2.2 mbType . . . . .	74
5.15.2.3 partitionType . . . . .	74
5.15.2.4 reserved . . . . .	74
5.16 NV_ENC_HEVC_MV_DATA Struct Reference . . . . .	75
5.16.1 Detailed Description . . . . .	75
5.16.2 Field Documentation . . . . .	75
5.16.2.1 mv . . . . .	75
5.16.2.2 cuType . . . . .	75
5.16.2.3 cuSize . . . . .	75
5.16.2.4 partitionMode . . . . .	75
5.16.2.5 lastCUInCTB . . . . .	75
5.17 NV_ENC_INITIALIZE_PARAMS Struct Reference . . . . .	76
5.17.1 Detailed Description . . . . .	76
5.17.2 Field Documentation . . . . .	76
5.17.2.1 version . . . . .	76
5.17.2.2 encodeGUID . . . . .	76
5.17.2.3 presetGUID . . . . .	77
5.17.2.4 encodeWidth . . . . .	77
5.17.2.5 encodeHeight . . . . .	77
5.17.2.6 darWidth . . . . .	77
5.17.2.7 darHeight . . . . .	77
5.17.2.8 frameRateNum . . . . .	77
5.17.2.9 frameRateDen . . . . .	77
5.17.2.10 enableEncodeAsync . . . . .	77
5.17.2.11 enablePTD . . . . .	77
5.17.2.12 reportSliceOffsets . . . . .	77
5.17.2.13 enableSubFrameWrite . . . . .	78
5.17.2.14 enableExternalMEHints . . . . .	78
5.17.2.15 enableMEOnlyMode . . . . .	78
5.17.2.16 enableWeightedPrediction . . . . .	78
5.17.2.17 reservedBitFields . . . . .	78
5.17.2.18 privDataSize . . . . .	78

5.17.2.19 privData . . . . .	78
5.17.2.20 encodeConfig . . . . .	78
5.17.2.21 maxEncodeWidth . . . . .	78
5.17.2.22 maxEncodeHeight . . . . .	79
5.17.2.23 maxMEHintCountsPerBlock . . . . .	79
5.17.2.24 reserved . . . . .	79
5.17.2.25 reserved2 . . . . .	79
5.18 NV_ENC_INPUT_RESOURCE_OPENGL_TEX Struct Reference . . . . .	80
5.18.1 Detailed Description . . . . .	80
5.18.2 Field Documentation . . . . .	80
5.18.2.1 texture . . . . .	80
5.18.2.2 target . . . . .	80
5.19 NV_ENC_LOCK_BITSTREAM Struct Reference . . . . .	81
5.19.1 Detailed Description . . . . .	81
5.19.2 Field Documentation . . . . .	81
5.19.2.1 version . . . . .	81
5.19.2.2 doNotWait . . . . .	81
5.19.2.3 ltrFrame . . . . .	81
5.19.2.4 reservedBitFields . . . . .	82
5.19.2.5 outputBitstream . . . . .	82
5.19.2.6 sliceOffsets . . . . .	82
5.19.2.7 frameIdx . . . . .	82
5.19.2.8 hwEncodeStatus . . . . .	82
5.19.2.9 numSlices . . . . .	82
5.19.2.10 bitstreamSizeInBytes . . . . .	82
5.19.2.11 outputTimeStamp . . . . .	82
5.19.2.12 outputDuration . . . . .	82
5.19.2.13 bitstreamBufferPtr . . . . .	82
5.19.2.14 pictureType . . . . .	82
5.19.2.15 pictureStruct . . . . .	83
5.19.2.16 frameAvgQP . . . . .	83
5.19.2.17 frameSatd . . . . .	83
5.19.2.18 ltrFrameIdx . . . . .	83
5.19.2.19 ltrFrameBitmap . . . . .	83
5.19.2.20 reserved . . . . .	83
5.19.2.21 reserved2 . . . . .	83
5.20 NV_ENC_LOCK_INPUT_BUFFER Struct Reference . . . . .	84

5.20.1 Detailed Description . . . . .	84
5.20.2 Field Documentation . . . . .	84
5.20.2.1 version . . . . .	84
5.20.2.2 doNotWait . . . . .	84
5.20.2.3 reservedBitFields . . . . .	84
5.20.2.4 inputBuffer . . . . .	84
5.20.2.5 bufferDataPtr . . . . .	84
5.20.2.6 pitch . . . . .	84
5.20.2.7 reserved1 . . . . .	85
5.20.2.8 reserved2 . . . . .	85
5.21 NV_ENC_MAP_INPUT_RESOURCE Struct Reference . . . . .	86
5.21.1 Detailed Description . . . . .	86
5.21.2 Field Documentation . . . . .	86
5.21.2.1 version . . . . .	86
5.21.2.2 subResourceIndex . . . . .	86
5.21.2.3 inputResource . . . . .	86
5.21.2.4 registeredResource . . . . .	86
5.21.2.5 mappedResource . . . . .	86
5.21.2.6 mappedBufferFmt . . . . .	86
5.21.2.7 reserved1 . . . . .	87
5.21.2.8 reserved2 . . . . .	87
5.22 NV_ENC_MEONLY_PARAMS Struct Reference . . . . .	88
5.22.1 Detailed Description . . . . .	88
5.22.2 Field Documentation . . . . .	88
5.22.2.1 version . . . . .	88
5.22.2.2 inputWidth . . . . .	88
5.22.2.3 inputHeight . . . . .	88
5.22.2.4 inputBuffer . . . . .	88
5.22.2.5 referenceFrame . . . . .	88
5.22.2.6 mvBuffer . . . . .	89
5.22.2.7 bufferFmt . . . . .	89
5.22.2.8 completionEvent . . . . .	89
5.22.2.9 viewID . . . . .	89
5.22.2.10 meHintCountsPerBlock . . . . .	89
5.22.2.11 meExternalHints . . . . .	89
5.22.2.12 reserved1 . . . . .	89
5.22.2.13 reserved2 . . . . .	89

5.23 NV_ENC_MVECTOR Struct Reference . . . . .	90
5.23.1 Detailed Description . . . . .	90
5.23.2 Field Documentation . . . . .	90
5.23.2.1 mvx . . . . .	90
5.23.2.2 mvy . . . . .	90
5.24 NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS Struct Reference . . . . .	91
5.24.1 Detailed Description . . . . .	91
5.24.2 Field Documentation . . . . .	91
5.24.2.1 version . . . . .	91
5.24.2.2 deviceType . . . . .	91
5.24.2.3 device . . . . .	91
5.24.2.4 reserved . . . . .	91
5.24.2.5 apiVersion . . . . .	91
5.24.2.6 reserved1 . . . . .	91
5.24.2.7 reserved2 . . . . .	92
5.25 NV_ENC_PIC_PARAMS Struct Reference . . . . .	93
5.25.1 Detailed Description . . . . .	93
5.25.2 Field Documentation . . . . .	93
5.25.2.1 version . . . . .	93
5.25.2.2 inputWidth . . . . .	93
5.25.2.3 inputHeight . . . . .	93
5.25.2.4 inputPitch . . . . .	94
5.25.2.5 encodePicFlags . . . . .	94
5.25.2.6 frameIdx . . . . .	94
5.25.2.7 inputTimeStamp . . . . .	94
5.25.2.8 inputDuration . . . . .	94
5.25.2.9 inputBuffer . . . . .	94
5.25.2.10 outputBitstream . . . . .	94
5.25.2.11 completionEvent . . . . .	94
5.25.2.12 bufferFmt . . . . .	94
5.25.2.13 pictureStruct . . . . .	94
5.25.2.14 pictureType . . . . .	94
5.25.2.15 codecPicParams . . . . .	95
5.25.2.16 meHintCountsPerBlock . . . . .	95
5.25.2.17 meExternalHints . . . . .	95
5.25.2.18 reserved1 . . . . .	95
5.25.2.19 reserved2 . . . . .	95

5.25.2.20 qpDeltaMap . . . . .	95
5.25.2.21 qpDeltaMapSize . . . . .	95
5.25.2.22 reservedBitFields . . . . .	95
5.25.2.23 meHintRefPicDist . . . . .	95
5.25.2.24 reserved3 . . . . .	96
5.25.2.25 reserved4 . . . . .	96
5.26 NV_ENC_PIC_PARAMS_H264 Struct Reference . . . . .	97
5.26.1 Detailed Description . . . . .	97
5.26.2 Field Documentation . . . . .	97
5.26.2.1 displayPOCSyntax . . . . .	97
5.26.2.2 reserved3 . . . . .	97
5.26.2.3 refPicFlag . . . . .	97
5.26.2.4 colourPlaneId . . . . .	98
5.26.2.5 forceIntraRefreshWithFrameCnt . . . . .	98
5.26.2.6 constrainedFrame . . . . .	98
5.26.2.7 sliceModeDataUpdate . . . . .	98
5.26.2.8 ltrMarkFrame . . . . .	98
5.26.2.9 ltrUseFrames . . . . .	98
5.26.2.10 reservedBitFields . . . . .	98
5.26.2.11 sliceTypeData . . . . .	98
5.26.2.12 sliceTypeArrayCnt . . . . .	98
5.26.2.13 seiPayloadArrayCnt . . . . .	98
5.26.2.14 seiPayloadArray . . . . .	98
5.26.2.15 sliceMode . . . . .	99
5.26.2.16 sliceModeData . . . . .	99
5.26.2.17 ltrMarkFrameIdx . . . . .	99
5.26.2.18 ltrUseFrameBitmap . . . . .	99
5.26.2.19 ltrUsageMode . . . . .	99
5.26.2.20 reserved . . . . .	99
5.26.2.21 reserved2 . . . . .	99
5.27 NV_ENC_PIC_PARAMS_HEVC Struct Reference . . . . .	100
5.27.1 Detailed Description . . . . .	100
5.27.2 Field Documentation . . . . .	100
5.27.2.1 displayPOCSyntax . . . . .	100
5.27.2.2 refPicFlag . . . . .	100
5.27.2.3 temporalId . . . . .	100
5.27.2.4 forceIntraRefreshWithFrameCnt . . . . .	101

5.27.2.5	constrainedFrame	101
5.27.2.6	sliceModeDataUpdate	101
5.27.2.7	ltrMarkFrame	101
5.27.2.8	ltrUseFrames	101
5.27.2.9	reservedBitFields	101
5.27.2.10	sliceTypeData	101
5.27.2.11	sliceTypeArrayCnt	101
5.27.2.12	sliceMode	101
5.27.2.13	sliceModeData	102
5.27.2.14	ltrMarkFrameIdx	102
5.27.2.15	ltrUseFrameBitmap	102
5.27.2.16	ltrUsageMode	102
5.27.2.17	seiPayloadArrayCnt	102
5.27.2.18	reserved	102
5.27.2.19	seiPayloadArray	102
5.27.2.20	reserved2	102
5.27.2.21	reserved3	102
5.28	NV_ENC_PRESET_CONFIG Struct Reference	103
5.28.1	Detailed Description	103
5.28.2	Field Documentation	103
5.28.2.1	version	103
5.28.2.2	presetCfg	103
5.28.2.3	reserved1	103
5.28.2.4	reserved2	103
5.29	NV_ENC_QP Struct Reference	104
5.29.1	Detailed Description	104
5.30	NV_ENC_RC_PARAMS Struct Reference	105
5.30.1	Detailed Description	105
5.30.2	Field Documentation	105
5.30.2.1	rateControlMode	105
5.30.2.2	constQP	105
5.30.2.3	averageBitRate	106
5.30.2.4	maxBitRate	106
5.30.2.5	vbvBufferSize	106
5.30.2.6	vbvInitialDelay	106
5.30.2.7	enableMinQP	106
5.30.2.8	enableMaxQP	106



5.30.2.9	enableInitialRCQP	106
5.30.2.10	enableAQ	106
5.30.2.11	enableExtQPDeltaMap	106
5.30.2.12	enableLookahead	106
5.30.2.13	disableIadapt	106
5.30.2.14	disableBadapt	107
5.30.2.15	enableTemporalAQ	107
5.30.2.16	zeroReorderDelay	107
5.30.2.17	enableNonRefP	107
5.30.2.18	strictGOPTarget	107
5.30.2.19	aqStrength	107
5.30.2.20	reservedBitFields	107
5.30.2.21	minQP	107
5.30.2.22	maxQP	107
5.30.2.23	initialRCQP	107
5.30.2.24	temporallayerIdxMask	107
5.30.2.25	temporalLayerQP	108
5.30.2.26	targetQuality	108
5.30.2.27	targetQualityLSB	108
5.30.2.28	lookaheadDepth	108
5.31	NV_ENC_RECONFIGURE_PARAMS Struct Reference	109
5.31.1	Detailed Description	109
5.31.2	Field Documentation	109
5.31.2.1	version	109
5.31.2.2	reInitEncodeParams	109
5.31.2.3	resetEncoder	109
5.31.2.4	forceIDR	109
5.32	NV_ENC_REGISTER_RESOURCE Struct Reference	110
5.32.1	Detailed Description	110
5.32.2	Field Documentation	110
5.32.2.1	version	110
5.32.2.2	resourceType	110
5.32.2.3	width	110
5.32.2.4	height	110
5.32.2.5	pitch	110
5.32.2.6	subResourceIndex	111
5.32.2.7	resourceToRegister	111

5.32.2.8	registeredResource . . . . .	111
5.32.2.9	bufferFormat . . . . .	111
5.32.2.10	reserved1 . . . . .	111
5.32.2.11	reserved2 . . . . .	111
5.33	NV_ENC_SEI_PAYLOAD Struct Reference . . . . .	112
5.33.1	Detailed Description . . . . .	112
5.33.2	Field Documentation . . . . .	112
5.33.2.1	payloadSize . . . . .	112
5.33.2.2	payloadType . . . . .	112
5.33.2.3	payload . . . . .	112
5.34	NV_ENC_SEQUENCE_PARAM_PAYLOAD Struct Reference . . . . .	113
5.34.1	Detailed Description . . . . .	113
5.34.2	Field Documentation . . . . .	113
5.34.2.1	version . . . . .	113
5.34.2.2	inBufferSize . . . . .	113
5.34.2.3	spsId . . . . .	113
5.34.2.4	ppsId . . . . .	113
5.34.2.5	spsppsBuffer . . . . .	113
5.34.2.6	outSPSPSPayloadSize . . . . .	113
5.34.2.7	reserved . . . . .	114
5.34.2.8	reserved2 . . . . .	114
5.35	NV_ENC_STAT Struct Reference . . . . .	115
5.35.1	Detailed Description . . . . .	115
5.35.2	Field Documentation . . . . .	115
5.35.2.1	version . . . . .	115
5.35.2.2	reserved . . . . .	115
5.35.2.3	outputBitStream . . . . .	115
5.35.2.4	bitStreamSize . . . . .	115
5.35.2.5	picType . . . . .	115
5.35.2.6	lastValidByteOffset . . . . .	115
5.35.2.7	sliceOffsets . . . . .	116
5.35.2.8	picIdx . . . . .	116
5.35.2.9	reserved1 . . . . .	116
5.35.2.10	reserved2 . . . . .	116
5.36	NV_ENCODE_API_FUNCTION_LIST Struct Reference . . . . .	117
5.36.1	Detailed Description . . . . .	117
5.36.2	Field Documentation . . . . .	118

5.36.2.1	version	118
5.36.2.2	reserved	118
5.36.2.3	nvEncOpenEncodeSession	118
5.36.2.4	nvEncGetEncodeGUIDCount	118
5.36.2.5	nvEncGetEncodeProfileGUIDCount	118
5.36.2.6	nvEncGetEncodeProfileGUIDs	118
5.36.2.7	nvEncGetEncodeGUIDs	118
5.36.2.8	nvEncGetInputFormatCount	118
5.36.2.9	nvEncGetInputFormats	118
5.36.2.10	nvEncGetEncodeCaps	118
5.36.2.11	nvEncGetEncodePresetCount	119
5.36.2.12	nvEncGetEncodePresetGUIDs	119
5.36.2.13	nvEncGetEncodePresetConfig	119
5.36.2.14	nvEncInitializeEncoder	119
5.36.2.15	nvEncCreateInputBuffer	119
5.36.2.16	nvEncDestroyInputBuffer	119
5.36.2.17	nvEncCreateBitstreamBuffer	119
5.36.2.18	nvEncDestroyBitstreamBuffer	119
5.36.2.19	nvEncEncodePicture	119
5.36.2.20	nvEncLockBitstream	119
5.36.2.21	nvEncUnlockBitstream	120
5.36.2.22	nvEncLockInputBuffer	120
5.36.2.23	nvEncUnlockInputBuffer	120
5.36.2.24	nvEncGetEncodeStats	120
5.36.2.25	nvEncGetSequenceParams	120
5.36.2.26	nvEncRegisterAsyncEvent	120
5.36.2.27	nvEncUnregisterAsyncEvent	120
5.36.2.28	nvEncMapInputResource	120
5.36.2.29	nvEncUnmapInputResource	120
5.36.2.30	nvEncDestroyEncoder	120
5.36.2.31	nvEncInvalidateRefFrames	121
5.36.2.32	nvEncOpenEncodeSessionEx	121
5.36.2.33	nvEncRegisterResource	121
5.36.2.34	nvEncUnregisterResource	121
5.36.2.35	nvEncReconfigureEncoder	121
5.36.2.36	nvEncCreateMVBuffer	121
5.36.2.37	nvEncDestroyMVBuffer	121

5.36.2.38	nvEncRunMotionEstimationOnly	121
5.36.2.39	reserved2	121
5.37	NVENC_EXTERNAL_ME_HINT Struct Reference	122
5.37.1	Detailed Description	122
5.37.2	Field Documentation	122
5.37.2.1	mvx	122
5.37.2.2	mvx	122
5.37.2.3	refidx	122
5.37.2.4	dir	122
5.37.2.5	partType	122
5.37.2.6	lastofPart	122
5.37.2.7	lastOfMB	123
5.38	NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE Struct Reference	124
5.38.1	Detailed Description	124
5.38.2	Field Documentation	124
5.38.2.1	numCandsPerBlk16x16	124
5.38.2.2	numCandsPerBlk16x8	124
5.38.2.3	numCandsPerBlk8x16	124
5.38.2.4	numCandsPerBlk8x8	124
5.38.2.5	reserved	124
5.38.2.6	reserved1	124
5.39	NVENC_RECT Struct Reference	125
5.39.1	Detailed Description	125
5.39.2	Field Documentation	125
5.39.2.1	left	125
5.39.2.2	top	125
5.39.2.3	right	125
5.39.2.4	bottom	125

# Chapter 1

## Legal Notice

**Copyright** (c) 2011-2017 NVIDIA Corporation. All rights reserved.

### **Notice**

This source code and/or documentation ("Licensed Deliverables") are subject to NVIDIA intellectual property rights under U.S. and international Copyright laws.

These Licensed Deliverables contained herein is PROPRIETARY and to NVIDIA and is being provided under the terms and conditions of a form of NVIDIA software license agreement by and between NVIDIA and Licensee ("License Agreement") or electronically accepted by Licensee. Notwithstanding any terms or conditions to the contrary in the License Agreement, reproduction or disclosure of the Licensed Deliverables to any third party without the express written consent of NVIDIA is prohibited.

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." WITHOUT EXPRESS OR IMPLIED WARRANTY OF ANY KIND. NVIDIA DISCLAIMS ALL WARRANTIES WITH REGARD TO THESE LICENSED DELIVERABLES, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE. NOTWITHSTANDING ANY TERMS OR CONDITIONS TO THE CONTRARY IN THE LICENSE AGREEMENT, IN NO EVENT SHALL NVIDIA BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THESE LICENSED DELIVERABLES.

Information furnished is believed to be accurate and reliable. However, NVIDIA assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties, which may result from its use. No License is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in the software are subject to change without notice. This publication supersedes and replaces all other information previously supplied.

NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

U.S. Government End Users. These Licensed Deliverables are a "commercial item" as that term is defined at 48 C.F.R. 2.101 (OCT \* 1995), consisting of "commercial computer software" and "commercial computer software documentation" as such terms are used in 48 C.F.R. 12.212 (SEPT 1995) and is provided to the U.S. Government only as a commercial end item. Consistent with 48 C.F.R.12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (JUNE 1995), all U.S. Government End Users acquire the Licensed Deliverables with only those rights set forth herein.

Any use of the Licensed Deliverables in individual and commercial software must include, in the user documentation and internal comments to the code, the above Disclaimer and U.S. Government End Users Notice.

**Trademarks**

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Microsoft, Windows, and the Windows logo are registered trademarks of Microsoft Corporation.

Other company and product names may be trademarks or registered trademarks of the respective companies with which they are associated.

# Chapter 2

## Module Index

### 2.1 Modules

Here is a list of all modules:

NvEncodeAPI Data structures . . . . .	<a href="#">7</a>
NvEncodeAPI Functions . . . . .	<a href="#">22</a>





## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

GUID	47
NV_ENC_CAPS_PARAM	48
NV_ENC_CODEC_CONFIG	49
NV_ENC_CODEC_PIC_PARAMS	50
NV_ENC_CONFIG	51
NV_ENC_CONFIG_H264	53
NV_ENC_CONFIG_H264_MEONLY	59
NV_ENC_CONFIG_H264_VUI_PARAMETERS	61
NV_ENC_CONFIG_HEVC	63
NV_ENC_CONFIG_HEVC_MEONLY	67
NV_ENC_CREATE_BITSTREAM_BUFFER	68
NV_ENC_CREATE_INPUT_BUFFER	70
NV_ENC_CREATE_MV_BUFFER	72
NV_ENC_EVENT_PARAMS	73
NV_ENC_H264_MV_DATA	74
NV_ENC_HEVC_MV_DATA	75
NV_ENC_INITIALIZE_PARAMS	76
NV_ENC_INPUT_RESOURCE_OPENGL_TEX	80
NV_ENC_LOCK_BITSTREAM	81
NV_ENC_LOCK_INPUT_BUFFER	84
NV_ENC_MAP_INPUT_RESOURCE	86
NV_ENC_MEONLY_PARAMS	88
NV_ENC_MVECTOR	90
NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS	91
NV_ENC_PIC_PARAMS	93
NV_ENC_PIC_PARAMS_H264	97
NV_ENC_PIC_PARAMS_HEVC	100
NV_ENC_PRESET_CONFIG	103
NV_ENC_QP	104
NV_ENC_RC_PARAMS	105
NV_ENC_RECONFIGURE_PARAMS	109
NV_ENC_REGISTER_RESOURCE	110
NV_ENC_SEI_PAYLOAD	112

NV_ENC_SEQUENCE_PARAM_PAYLOAD	113
NV_ENC_STAT	115
NV_ENCODE_API_FUNCTION_LIST	117
NVENC_EXTERNAL_ME_HINT	122
NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE	124
NVENC_RECT	125

# Chapter 4

## Module Documentation

### 4.1 NvEncodeAPI Data structures

#### Data Structures

- struct [GUID](#)
- struct [NVENC\\_RECT](#)
- struct [NV\\_ENC\\_CAPS\\_PARAM](#)
- struct [NV\\_ENC\\_CREATE\\_INPUT\\_BUFFER](#)
- struct [NV\\_ENC\\_CREATE\\_BITSTREAM\\_BUFFER](#)
- struct [NV\\_ENC\\_MVECTOR](#)
- struct [NV\\_ENC\\_H264\\_MV\\_DATA](#)
- struct [NV\\_ENC\\_HEVC\\_MV\\_DATA](#)
- struct [NV\\_ENC\\_CREATE\\_MV\\_BUFFER](#)
- struct [NV\\_ENC\\_QP](#)
- struct [NV\\_ENC\\_RC\\_PARAMS](#)
- struct [NV\\_ENC\\_CONFIG\\_H264\\_VUI\\_PARAMETERS](#)
- struct [NVENC\\_EXTERNAL\\_ME\\_HINT\\_COUNTS\\_PER\\_BLOCKTYPE](#)
- struct [NVENC\\_EXTERNAL\\_ME\\_HINT](#)
- struct [NV\\_ENC\\_CONFIG\\_H264](#)
- struct [NV\\_ENC\\_CONFIG\\_HEVC](#)
- struct [NV\\_ENC\\_CONFIG\\_H264\\_MEONLY](#)
- struct [NV\\_ENC\\_CONFIG\\_HEVC\\_MEONLY](#)
- union [NV\\_ENC\\_CODEC\\_CONFIG](#)
- struct [NV\\_ENC\\_CONFIG](#)
- struct [NV\\_ENC\\_INITIALIZE\\_PARAMS](#)
- struct [NV\\_ENC\\_RECONFIGURE\\_PARAMS](#)
- struct [NV\\_ENC\\_PRESET\\_CONFIG](#)
- struct [NV\\_ENC\\_SEI\\_PAYLOAD](#)
- struct [NV\\_ENC\\_PIC\\_PARAMS\\_H264](#)
- struct [NV\\_ENC\\_PIC\\_PARAMS\\_HEVC](#)
- union [NV\\_ENC\\_CODEC\\_PIC\\_PARAMS](#)
- struct [NV\\_ENC\\_PIC\\_PARAMS](#)
- struct [NV\\_ENC\\_MEONLY\\_PARAMS](#)
- struct [NV\\_ENC\\_LOCK\\_BITSTREAM](#)
- struct [NV\\_ENC\\_LOCK\\_INPUT\\_BUFFER](#)

- struct [NV\\_ENC\\_MAP\\_INPUT\\_RESOURCE](#)
- struct [NV\\_ENC\\_INPUT\\_RESOURCE\\_OPENGL\\_TEX](#)
- struct [NV\\_ENC\\_REGISTER\\_RESOURCE](#)
- struct [NV\\_ENC\\_STAT](#)
- struct [NV\\_ENC\\_SEQUENCE\\_PARAM\\_PAYLOAD](#)
- struct [NV\\_ENC\\_EVENT\\_PARAMS](#)
- struct [NV\\_ENC\\_OPEN\\_ENCODE\\_SESSION\\_EX\\_PARAMS](#)
- struct [NV\\_ENCODE\\_API\\_FUNCTION\\_LIST](#)

## Defines

- #define [NV\\_ENC\\_PARAMS\\_RC\\_VBR\\_MINQP](#) ([NV\\_ENC\\_PARAMS\\_RC\\_MODE](#))0x4
- #define [NV\\_ENC\\_PARAMS\\_RC\\_2\\_PASS\\_QUALITY](#) [NV\\_ENC\\_PARAMS\\_RC\\_CBR\\_LOWDELAY\\_HQ](#)
- #define [NV\\_ENC\\_PARAMS\\_RC\\_2\\_PASS\\_FRAMESIZE\\_CAP](#) [NV\\_ENC\\_PARAMS\\_RC\\_CBR\\_HQ](#)
- #define [NV\\_ENC\\_PARAMS\\_RC\\_2\\_PASS\\_VBR](#) [NV\\_ENC\\_PARAMS\\_RC\\_VBR\\_HQ](#)
- #define [NV\\_ENC\\_PARAMS\\_RC\\_CBR2](#) [NV\\_ENC\\_PARAMS\\_RC\\_CBR](#)
- #define [NV\\_ENC\\_CAPS\\_PARAM\\_VER](#) NVENC\_API\_STRUCT\_VERSION(1)
- #define [NV\\_ENC\\_CREATE\\_INPUT\\_BUFFER\\_VER](#) NVENC\_API\_STRUCT\_VERSION(1)
- #define [NV\\_ENC\\_CREATE\\_BITSTREAM\\_BUFFER\\_VER](#) NVENC\_API\_STRUCT\_VERSION(1)
- #define [NV\\_ENC\\_CREATE\\_MV\\_BUFFER\\_VER](#) NVENC\_API\_STRUCT\_VERSION(1)
- #define [NV\\_ENC\\_RC\\_PARAMS\\_VER](#) NVENC\_API\_STRUCT\_VERSION(1)
- #define [NV\\_ENC\\_CONFIG\\_VER](#) (NVENC\_API\_STRUCT\_VERSION(6) | ( 1<<31 ))
- #define [NV\\_ENC\\_INITIALIZE\\_PARAMS\\_VER](#) (NVENC\_API\_STRUCT\_VERSION(5) | ( 1<<31 ))
- #define [NV\\_ENC\\_RECONFIGURE\\_PARAMS\\_VER](#) (NVENC\_API\_STRUCT\_VERSION(1) | ( 1<<31 ))
- #define [NV\\_ENC\\_PRESET\\_CONFIG\\_VER](#) (NVENC\_API\_STRUCT\_VERSION(4) | ( 1<<31 ))
- #define [NV\\_ENC\\_PIC\\_PARAMS\\_VER](#) (NVENC\_API\_STRUCT\_VERSION(4) | ( 1<<31 ))
- #define [NV\\_ENC\\_MEONLY\\_PARAMS\\_VER](#) NVENC\_API\_STRUCT\_VERSION(3)
- #define [NV\\_ENC\\_LOCK\\_BITSTREAM\\_VER](#) NVENC\_API\_STRUCT\_VERSION(1)
- #define [NV\\_ENC\\_LOCK\\_INPUT\\_BUFFER\\_VER](#) NVENC\_API\_STRUCT\_VERSION(1)
- #define [NV\\_ENC\\_MAP\\_INPUT\\_RESOURCE\\_VER](#) NVENC\_API\_STRUCT\_VERSION(4)
- #define [NV\\_ENC\\_REGISTER\\_RESOURCE\\_VER](#) NVENC\_API\_STRUCT\_VERSION(3)
- #define [NV\\_ENC\\_STAT\\_VER](#) NVENC\_API\_STRUCT\_VERSION(1)
- #define [NV\\_ENC\\_SEQUENCE\\_PARAM\\_PAYLOAD\\_VER](#) NVENC\_API\_STRUCT\_VERSION(1)
- #define [NV\\_ENC\\_EVENT\\_PARAMS\\_VER](#) NVENC\_API\_STRUCT\_VERSION(1)
- #define [NV\\_ENC\\_OPEN\\_ENCODE\\_SESSION\\_EX\\_PARAMS\\_VER](#) NVENC\_API\_STRUCT\_VERSION(1)

## Enumerations

- enum [NV\\_ENC\\_PARAMS\\_FRAME\\_FIELD\\_MODE](#) { [NV\\_ENC\\_PARAMS\\_FRAME\\_FIELD\\_MODE\\_FRAME](#) = 0x01, [NV\\_ENC\\_PARAMS\\_FRAME\\_FIELD\\_MODE\\_FIELD](#) = 0x02, [NV\\_ENC\\_PARAMS\\_FRAME\\_FIELD\\_MODE\\_MBAFF](#) = 0x03 }
- enum [NV\\_ENC\\_PARAMS\\_RC\\_MODE](#) {  
[NV\\_ENC\\_PARAMS\\_RC\\_CONSTQP](#) = 0x0, [NV\\_ENC\\_PARAMS\\_RC\\_VBR](#) = 0x1, [NV\\_ENC\\_PARAMS\\_RC\\_CBR](#) = 0x2, [NV\\_ENC\\_PARAMS\\_RC\\_CBR\\_LOWDELAY\\_HQ](#) = 0x8,  
[NV\\_ENC\\_PARAMS\\_RC\\_CBR\\_HQ](#) = 0x10, [NV\\_ENC\\_PARAMS\\_RC\\_VBR\\_HQ](#) = 0x20 }
- enum [NV\\_ENC\\_PIC\\_STRUCT](#) { [NV\\_ENC\\_PIC\\_STRUCT\\_FRAME](#) = 0x01, [NV\\_ENC\\_PIC\\_STRUCT\\_FIELD\\_TOP\\_BOTTOM](#) = 0x02, [NV\\_ENC\\_PIC\\_STRUCT\\_FIELD\\_BOTTOM\\_TOP](#) = 0x03 }

- enum NV\_ENC\_PIC\_TYPE {  
 NV\_ENC\_PIC\_TYPE\_P = 0x0, NV\_ENC\_PIC\_TYPE\_B = 0x01, NV\_ENC\_PIC\_TYPE\_I = 0x02, NV\_ENC\_PIC\_TYPE\_IDR = 0x03,  
 NV\_ENC\_PIC\_TYPE\_BI = 0x04, NV\_ENC\_PIC\_TYPE\_SKIPPED = 0x05, NV\_ENC\_PIC\_TYPE\_INTRA\_REFRESH = 0x06, NV\_ENC\_PIC\_TYPE\_UNKNOWN = 0xFF }
- enum NV\_ENC\_MV\_PRECISION { NV\_ENC\_MV\_PRECISION\_DEFAULT = 0x0, NV\_ENC\_MV\_PRECISION\_FULL\_PEL = 0x01, NV\_ENC\_MV\_PRECISION\_HALF\_PEL = 0x02, NV\_ENC\_MV\_PRECISION\_QUARTER\_PEL = 0x03 }
- enum NV\_ENC\_BUFFER\_FORMAT {  
 NV\_ENC\_BUFFER\_FORMAT\_UNDEFINED = 0x00000000, NV\_ENC\_BUFFER\_FORMAT\_NV12 = 0x00000001, NV\_ENC\_BUFFER\_FORMAT\_YV12 = 0x00000010, NV\_ENC\_BUFFER\_FORMAT\_IYUV = 0x00000100,  
 NV\_ENC\_BUFFER\_FORMAT\_YUV444 = 0x00001000, NV\_ENC\_BUFFER\_FORMAT\_YUV420\_10BIT = 0x00010000, NV\_ENC\_BUFFER\_FORMAT\_YUV444\_10BIT = 0x00100000, NV\_ENC\_BUFFER\_FORMAT\_ARGB = 0x01000000,  
 NV\_ENC\_BUFFER\_FORMAT\_ARGB10 = 0x02000000, NV\_ENC\_BUFFER\_FORMAT\_AYUV = 0x04000000, NV\_ENC\_BUFFER\_FORMAT\_ABGR = 0x10000000, NV\_ENC\_BUFFER\_FORMAT\_ABGR10 = 0x20000000 }
- enum NV\_ENC\_LEVEL
- enum NVENCSTATUS {  
 NV\_ENC\_SUCCESS, NV\_ENC\_ERR\_NO\_ENCODE\_DEVICE, NV\_ENC\_ERR\_UNSUPPORTED\_DEVICE, NV\_ENC\_ERR\_INVALID\_ENCODERDEVICE,  
 NV\_ENC\_ERR\_INVALID\_DEVICE, NV\_ENC\_ERR\_DEVICE\_NOT\_EXIST, NV\_ENC\_ERR\_INVALID\_PTR, NV\_ENC\_ERR\_INVALID\_EVENT,  
 NV\_ENC\_ERR\_INVALID\_PARAM, NV\_ENC\_ERR\_INVALID\_CALL, NV\_ENC\_ERR\_OUT\_OF\_MEMORY, NV\_ENC\_ERR\_ENCODER\_NOT\_INITIALIZED,  
 NV\_ENC\_ERR\_UNSUPPORTED\_PARAM, NV\_ENC\_ERR\_LOCK\_BUSY, NV\_ENC\_ERR\_NOT\_ENOUGH\_BUFFER, NV\_ENC\_ERR\_INVALID\_VERSION,  
 NV\_ENC\_ERR\_MAP\_FAILED, NV\_ENC\_ERR\_NEED\_MORE\_INPUT, NV\_ENC\_ERR\_ENCODER\_BUSY, NV\_ENC\_ERR\_EVENT\_NOT\_REGISTERD,  
 NV\_ENC\_ERR\_GENERIC, NV\_ENC\_ERR\_INCOMPATIBLE\_CLIENT\_KEY, NV\_ENC\_ERR\_UNIMPLEMENTED, NV\_ENC\_ERR\_RESOURCE\_REGISTER\_FAILED,  
 NV\_ENC\_ERR\_RESOURCE\_NOT\_REGISTERED, NV\_ENC\_ERR\_RESOURCE\_NOT\_MAPPED }
- enum NV\_ENC\_PIC\_FLAGS { NV\_ENC\_PIC\_FLAG\_FORCEINTRA = 0x1, NV\_ENC\_PIC\_FLAG\_FORCEIDR = 0x2, NV\_ENC\_PIC\_FLAG\_OUTPUT\_SPSPPS = 0x4, NV\_ENC\_PIC\_FLAG\_EOS = 0x8 }
- enum NV\_ENC\_MEMORY\_HEAP { NV\_ENC\_MEMORY\_HEAP\_AUTOSELECT = 0, NV\_ENC\_MEMORY\_HEAP\_VID = 1, NV\_ENC\_MEMORY\_HEAP\_SYSMEM\_CACHED = 2, NV\_ENC\_MEMORY\_HEAP\_SYSMEM\_UNCACHED = 3 }
- enum NV\_ENC\_H264\_ENTROPY\_CODING\_MODE { NV\_ENC\_H264\_ENTROPY\_CODING\_MODE\_AUTOSELECT = 0x0, NV\_ENC\_H264\_ENTROPY\_CODING\_MODE\_CABAC = 0x1, NV\_ENC\_H264\_ENTROPY\_CODING\_MODE\_CAVLC = 0x2 }
- enum NV\_ENC\_H264\_BDIRECT\_MODE { NV\_ENC\_H264\_BDIRECT\_MODE\_AUTOSELECT = 0x0, NV\_ENC\_H264\_BDIRECT\_MODE\_DISABLE = 0x1, NV\_ENC\_H264\_BDIRECT\_MODE\_TEMPORAL = 0x2, NV\_ENC\_H264\_BDIRECT\_MODE\_SPATIAL = 0x3 }
- enum NV\_ENC\_H264\_FMO\_MODE { NV\_ENC\_H264\_FMO\_AUTOSELECT = 0x0, NV\_ENC\_H264\_FMO\_ENABLE = 0x1, NV\_ENC\_H264\_FMO\_DISABLE = 0x2 }
- enum NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_MODE { NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_AUTOSELECT = 0x0, NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_DISABLE = 0x1, NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_ENABLE = 0x2 }

- enum `NV_ENC_STEREO_PACKING_MODE` {  
`NV_ENC_STEREO_PACKING_MODE_NONE` = 0x0, `NV_ENC_STEREO_PACKING_MODE_-CHECKERBOARD` = 0x1, `NV_ENC_STEREO_PACKING_MODE_COLINTERLEAVE` = 0x2, `NV_ENC_STEREO_PACKING_MODE_ROWINTERLEAVE` = 0x3,  
`NV_ENC_STEREO_PACKING_MODE_SIDEBYSIDE` = 0x4, `NV_ENC_STEREO_PACKING_MODE_-TOPBOTTOM` = 0x5, `NV_ENC_STEREO_PACKING_MODE_FRAMESEQ` = 0x6 }
- enum `NV_ENC_INPUT_RESOURCE_TYPE` { `NV_ENC_INPUT_RESOURCE_TYPE_DIRECTX` = 0x0, `NV_ENC_INPUT_RESOURCE_TYPE_CUDADEVICEPTR` = 0x1, `NV_ENC_INPUT_RESOURCE_TYPE_CUDAARRAY` = 0x2, `NV_ENC_INPUT_RESOURCE_TYPE_OPENGL_TEX` = 0x3 }
- enum `NV_ENC_DEVICE_TYPE` { `NV_ENC_DEVICE_TYPE_DIRECTX` = 0x0, `NV_ENC_DEVICE_TYPE_CUDA` = 0x1, `NV_ENC_DEVICE_TYPE_OPENGL` = 0x2 }
- enum `NV_ENC_CAPS` {  
`NV_ENC_CAPS_NUM_MAX_BFRAMES`, `NV_ENC_CAPS_SUPPORTED_RATECONTROL_MODES`,  
`NV_ENC_CAPS_SUPPORT_FIELD_ENCODING`, `NV_ENC_CAPS_SUPPORT_MONOCHROME`,  
`NV_ENC_CAPS_SUPPORT_FMO`, `NV_ENC_CAPS_SUPPORT_QPELMV`, `NV_ENC_CAPS_SUPPORT_-BDIRECT_MODE`, `NV_ENC_CAPS_SUPPORT_CABAC`,  
`NV_ENC_CAPS_SUPPORT_ADAPTIVE_TRANSFORM`, `NV_ENC_CAPS_SUPPORT_RESERVED`,  
`NV_ENC_CAPS_NUM_MAX_TEMPORAL_LAYERS`, `NV_ENC_CAPS_SUPPORT_HIERARCHICAL_-PFRAMES`,  
`NV_ENC_CAPS_SUPPORT_HIERARCHICAL_BFRAMES`, `NV_ENC_CAPS_LEVEL_MAX`, `NV_ENC_-CAPS_LEVEL_MIN`, `NV_ENC_CAPS_SEPARATE_COLOUR_PLANE`,  
`NV_ENC_CAPS_WIDTH_MAX`, `NV_ENC_CAPS_HEIGHT_MAX`, `NV_ENC_CAPS_SUPPORT_-TEMPORAL_SVC`, `NV_ENC_CAPS_SUPPORT_DYN_RES_CHANGE`,  
`NV_ENC_CAPS_SUPPORT_DYN_BITRATE_CHANGE`, `NV_ENC_CAPS_SUPPORT_DYN_FORCE_-CONSTQP`, `NV_ENC_CAPS_SUPPORT_DYN_RC_MODE_CHANGE`, `NV_ENC_CAPS_SUPPORT_-SUBFRAME_READBACK`,  
`NV_ENC_CAPS_SUPPORT_CONSTRAINED_ENCODING`, `NV_ENC_CAPS_SUPPORT_INTRA_-REFRESH`, `NV_ENC_CAPS_SUPPORT_CUSTOM_VBV_BUF_SIZE`, `NV_ENC_CAPS_SUPPORT_-DYNAMIC_SLICE_MODE`,  
`NV_ENC_CAPS_SUPPORT_REF_PIC_INVALIDATION`, `NV_ENC_CAPS_PREPROC_SUPPORT`, `NV_-ENC_CAPS_ASYNC_ENCODE_SUPPORT`, `NV_ENC_CAPS_MB_NUM_MAX`,  
`NV_ENC_CAPS_MB_PER_SEC_MAX`, `NV_ENC_CAPS_SUPPORT_YUV444_ENCODE`, `NV_ENC_-CAPS_SUPPORT_LOSSLESS_ENCODE`, `NV_ENC_CAPS_SUPPORT_SAO`,  
`NV_ENC_CAPS_SUPPORT_MEONLY_MODE`, `NV_ENC_CAPS_SUPPORT_LOOKAHEAD`, `NV_ENC_-CAPS_SUPPORT_TEMPORAL_AQ`, `NV_ENC_CAPS_SUPPORT_10BIT_ENCODE`,  
`NV_ENC_CAPS_NUM_MAX_LTR_FRAMES`, `NV_ENC_CAPS_SUPPORT_WEIGHTED_PREDICTION`,  
`NV_ENC_CAPS_EXPOSED_COUNT` }
- enum `NV_ENC_HEVC_CUSIZE`

### 4.1.1 Define Documentation

#### 4.1.1.1 #define NV\_ENC\_PARAMS\_RC\_VBR\_MINQP (NV\_ENC\_PARAMS\_RC\_MODE)0x4

Deprecated

#### 4.1.1.2 #define NV\_ENC\_PARAMS\_RC\_2\_PASS\_QUALITY NV\_ENC\_PARAMS\_RC\_CBR\_-LOWDELAY\_HQ

Deprecated

**4.1.1.3 #define NV\_ENC\_PARAMS\_RC\_2\_PASS\_FRAME\_SIZE\_CAP NV\_ENC\_PARAMS\_RC\_CBR\_HQ**

Deprecated

**4.1.1.4 #define NV\_ENC\_PARAMS\_RC\_2\_PASS\_VBR NV\_ENC\_PARAMS\_RC\_VBR\_HQ**

Deprecated

**4.1.1.5 #define NV\_ENC\_PARAMS\_RC\_CBR2 NV\_ENC\_PARAMS\_RC\_CBR**

Deprecated

**4.1.1.6 #define NV\_ENC\_CAPS\_PARAM\_VER NVENCAPI\_STRUCT\_VERSION(1)**

[NV\\_ENC\\_CAPS\\_PARAM](#) struct version.

**4.1.1.7 #define NV\_ENC\_CREATE\_INPUT\_BUFFER\_VER NVENCAPI\_STRUCT\_VERSION(1)**

[NV\\_ENC\\_CREATE\\_INPUT\\_BUFFER](#) struct version.

**4.1.1.8 #define NV\_ENC\_CREATE\_BITSTREAM\_BUFFER\_VER NVENCAPI\_STRUCT\_VERSION(1)**

[NV\\_ENC\\_CREATE\\_BITSTREAM\\_BUFFER](#) struct version.

**4.1.1.9 #define NV\_ENC\_CREATE\_MV\_BUFFER\_VER NVENCAPI\_STRUCT\_VERSION(1)**

[NV\\_ENC\\_CREATE\\_MV\\_BUFFER](#) struct version

**4.1.1.10 #define NV\_ENC\_RC\_PARAMS\_VER NVENCAPI\_STRUCT\_VERSION(1)**

macro for constructing the version field of `_NV_ENC_RC_PARAMS`

**4.1.1.11 #define NV\_ENC\_CONFIG\_VER (NVENCAPI\_STRUCT\_VERSION(6) | (1 << 31))**

macro for constructing the version field of `_NV_ENC_CONFIG`

**4.1.1.12 #define NV\_ENC\_INITIALIZE\_PARAMS\_VER (NVENCAPI\_STRUCT\_VERSION(5) | (1 << 31))**

macro for constructing the version field of `_NV_ENC_INITIALIZE_PARAMS`

**4.1.1.13 #define NV\_ENC\_RECONFIGURE\_PARAMS\_VER (NVENCAPI\_STRUCT\_VERSION(1) | (1 << 31))**

macro for constructing the version field of `_NV_ENC_RECONFIGURE_PARAMS`

**4.1.1.14 #define NV\_ENC\_PRESET\_CONFIG\_VER (NVENC\_API\_STRUCT\_VERSION(4) | (1 < 31))**

macro for constructing the version field of \_NV\_ENC\_PRESET\_CONFIG

**4.1.1.15 #define NV\_ENC\_PIC\_PARAMS\_VER (NVENC\_API\_STRUCT\_VERSION(4) | (1 < 31))**

Macro for constructing the version field of \_NV\_ENC\_PIC\_PARAMS

**4.1.1.16 #define NV\_ENC\_MEONLY\_PARAMS\_VER NVENC\_API\_STRUCT\_VERSION(3)**

[NV\\_ENC\\_MEONLY\\_PARAMS](#) struct version

**4.1.1.17 #define NV\_ENC\_LOCK\_BITSTREAM\_VER NVENC\_API\_STRUCT\_VERSION(1)**

Macro for constructing the version field of \_NV\_ENC\_LOCK\_BITSTREAM

**4.1.1.18 #define NV\_ENC\_LOCK\_INPUT\_BUFFER\_VER NVENC\_API\_STRUCT\_VERSION(1)**

Macro for constructing the version field of \_NV\_ENC\_LOCK\_INPUT\_BUFFER

**4.1.1.19 #define NV\_ENC\_MAP\_INPUT\_RESOURCE\_VER NVENC\_API\_STRUCT\_VERSION(4)**

Macro for constructing the version field of \_NV\_ENC\_MAP\_INPUT\_RESOURCE

**4.1.1.20 #define NV\_ENC\_REGISTER\_RESOURCE\_VER NVENC\_API\_STRUCT\_VERSION(3)**

Macro for constructing the version field of \_NV\_ENC\_REGISTER\_RESOURCE

**4.1.1.21 #define NV\_ENC\_STAT\_VER NVENC\_API\_STRUCT\_VERSION(1)**

Macro for constructing the version field of \_NV\_ENC\_STAT

**4.1.1.22 #define NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD\_VER NVENC\_API\_STRUCT\_VERSION(1)**

Macro for constructing the version field of \_NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD

**4.1.1.23 #define NV\_ENC\_EVENT\_PARAMS\_VER NVENC\_API\_STRUCT\_VERSION(1)**

Macro for constructing the version field of \_NV\_ENC\_EVENT\_PARAMS

**4.1.1.24 #define NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_PARAMS\_VER NVENC\_API\_STRUCT\_VERSION(1)**

Macro for constructing the version field of \_NV\_ENC\_OPEN\_ENCODE\_SESSIONEX\_PARAMS



## 4.1.2 Enumeration Type Documentation

### 4.1.2.1 enum NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE

Input frame encode modes

**Enumerator:**

*NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE\_FRAME* Frame mode  
*NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE\_FIELD* Field mode  
*NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE\_MBAFF* MB adaptive frame/field

### 4.1.2.2 enum NV\_ENC\_PARAMS\_RC\_MODE

Rate Control Modes

**Enumerator:**

*NV\_ENC\_PARAMS\_RC\_CONSTQP* Constant QP mode  
*NV\_ENC\_PARAMS\_RC\_VBR* Variable bitrate mode  
*NV\_ENC\_PARAMS\_RC\_CBR* Constant bitrate mode  
*NV\_ENC\_PARAMS\_RC\_CBR\_LOWDELAY\_HQ* low-delay CBR, high quality  
*NV\_ENC\_PARAMS\_RC\_CBR\_HQ* CBR, high quality (slower)  
*NV\_ENC\_PARAMS\_RC\_VBR\_HQ* VBR, high quality (slower)

### 4.1.2.3 enum NV\_ENC\_PIC\_STRUCT

Input picture structure

**Enumerator:**

*NV\_ENC\_PIC\_STRUCT\_FRAME* Progressive frame  
*NV\_ENC\_PIC\_STRUCT\_FIELD\_TOP\_BOTTOM* Field encoding top field first  
*NV\_ENC\_PIC\_STRUCT\_FIELD\_BOTTOM\_TOP* Field encoding bottom field first

### 4.1.2.4 enum NV\_ENC\_PIC\_TYPE

Input picture type

**Enumerator:**

*NV\_ENC\_PIC\_TYPE\_P* Forward predicted  
*NV\_ENC\_PIC\_TYPE\_B* Bi-directionally predicted picture  
*NV\_ENC\_PIC\_TYPE\_I* Intra predicted picture  
*NV\_ENC\_PIC\_TYPE\_IDR* IDR picture  
*NV\_ENC\_PIC\_TYPE\_BI* Bi-directionally predicted with only Intra MBs  
*NV\_ENC\_PIC\_TYPE\_SKIPPED* Picture is skipped  
*NV\_ENC\_PIC\_TYPE\_INTRA\_REFRESH* First picture in intra refresh cycle  
*NV\_ENC\_PIC\_TYPE\_UNKNOWN* Picture type unknown

#### 4.1.2.5 enum NV\_ENC\_MV\_PRECISION

Motion vector precisions

##### Enumerator:

**NV\_ENC\_MV\_PRECISION\_DEFAULT** Driver selects QuarterPel motion vector precision by default  
**NV\_ENC\_MV\_PRECISION\_FULL\_PEL** FullPel motion vector precision  
**NV\_ENC\_MV\_PRECISION\_HALF\_PEL** HalfPel motion vector precision  
**NV\_ENC\_MV\_PRECISION\_QUARTER\_PEL** QuarterPel motion vector precision

#### 4.1.2.6 enum NV\_ENC\_BUFFER\_FORMAT

Input buffer formats

##### Enumerator:

**NV\_ENC\_BUFFER\_FORMAT\_UNDEFINED** Undefined buffer format  
**NV\_ENC\_BUFFER\_FORMAT\_NV12** Semi-Planar YUV [Y plane followed by interleaved UV plane]  
**NV\_ENC\_BUFFER\_FORMAT\_YV12** Planar YUV [Y plane followed by V and U planes]  
**NV\_ENC\_BUFFER\_FORMAT\_IYUV** Planar YUV [Y plane followed by U and V planes]  
**NV\_ENC\_BUFFER\_FORMAT\_YUV444** Planar YUV [Y plane followed by U and V planes]  
**NV\_ENC\_BUFFER\_FORMAT\_YUV420\_10BIT** 10 bit Semi-Planar YUV [Y plane followed by interleaved UV plane]. Each pixel of size 2 bytes. Most Significant 10 bits contain pixel data.  
**NV\_ENC\_BUFFER\_FORMAT\_YUV444\_10BIT** 10 bit Planar YUV444 [Y plane followed by U and V planes]. Each pixel of size 2 bytes. Most Significant 10 bits contain pixel data.  
**NV\_ENC\_BUFFER\_FORMAT\_ARGB** 8 bit Packed A8R8G8B8. This is a word-ordered format where a pixel is represented by a 32-bit word with B in the lowest 8 bits, G in the next 8 bits, R in the 8 bits after that and A in the highest 8 bits.  
**NV\_ENC\_BUFFER\_FORMAT\_ARGB10** 10 bit Packed A2R10G10B10. This is a word-ordered format where a pixel is represented by a 32-bit word with B in the lowest 10 bits, G in the next 10 bits, R in the 10 bits after that and A in the highest 2 bits.  
**NV\_ENC\_BUFFER\_FORMAT\_AYUV** 8 bit Packed A8Y8U8V8. This is a word-ordered format where a pixel is represented by a 32-bit word with V in the lowest 8 bits, U in the next 8 bits, Y in the 8 bits after that and A in the highest 8 bits.  
**NV\_ENC\_BUFFER\_FORMAT\_ABGR** 8 bit Packed A8B8G8R8. This is a word-ordered format where a pixel is represented by a 32-bit word with R in the lowest 8 bits, G in the next 8 bits, B in the 8 bits after that and A in the highest 8 bits.  
**NV\_ENC\_BUFFER\_FORMAT\_ABGR10** 10 bit Packed A2B10G10R10. This is a word-ordered format where a pixel is represented by a 32-bit word with R in the lowest 10 bits, G in the next 10 bits, B in the 10 bits after that and A in the highest 2 bits.

#### 4.1.2.7 enum NV\_ENC\_LEVEL

Encoding levels

## 4.1.2.8 enum NVENCSTATUS

## Error Codes

## Enumerator:

**NV\_ENC\_SUCCESS** This indicates that API call returned with no errors.

**NV\_ENC\_ERR\_NO\_ENCODE\_DEVICE** This indicates that no encode capable devices were detected.

**NV\_ENC\_ERR\_UNSUPPORTED\_DEVICE** This indicates that devices pass by the client is not supported.

**NV\_ENC\_ERR\_INVALID\_ENCODERDEVICE** This indicates that the encoder device supplied by the client is not valid.

**NV\_ENC\_ERR\_INVALID\_DEVICE** This indicates that device passed to the API call is invalid.

**NV\_ENC\_ERR\_DEVICE\_NOT\_EXIST** This indicates that device passed to the API call is no longer available and needs to be reinitialized. The clients need to destroy the current encoder session by freeing the allocated input output buffers and destroying the device and create a new encoding session.

**NV\_ENC\_ERR\_INVALID\_PTR** This indicates that one or more of the pointers passed to the API call is invalid.

**NV\_ENC\_ERR\_INVALID\_EVENT** This indicates that completion event passed in [NvEncEncodePicture\(\)](#) call is invalid.

**NV\_ENC\_ERR\_INVALID\_PARAM** This indicates that one or more of the parameter passed to the API call is invalid.

**NV\_ENC\_ERR\_INVALID\_CALL** This indicates that an API call was made in wrong sequence/order.

**NV\_ENC\_ERR\_OUT\_OF\_MEMORY** This indicates that the API call failed because it was unable to allocate enough memory to perform the requested operation.

**NV\_ENC\_ERR\_ENCODER\_NOT\_INITIALIZED** This indicates that the encoder has not been initialized with [NvEncInitializeEncoder\(\)](#) or that initialization has failed. The client cannot allocate input or output buffers or do any encoding related operation before successfully initializing the encoder.

**NV\_ENC\_ERR\_UNSUPPORTED\_PARAM** This indicates that an unsupported parameter was passed by the client.

**NV\_ENC\_ERR\_LOCK\_BUSY** This indicates that the [NvEncLockBitstream\(\)](#) failed to lock the output buffer. This happens when the client makes a non blocking lock call to access the output bitstream by passing [NV\\_ENC\\_LOCK\\_BITSTREAM::doNotWait](#) flag. This is not a fatal error and client should retry the same operation after few milliseconds.

**NV\_ENC\_ERR\_NOT\_ENOUGH\_BUFFER** This indicates that the size of the user buffer passed by the client is insufficient for the requested operation.

**NV\_ENC\_ERR\_INVALID\_VERSION** This indicates that an invalid struct version was used by the client.

**NV\_ENC\_ERR\_MAP\_FAILED** This indicates that [NvEncMapInputResource\(\)](#) API failed to map the client provided input resource.

**NV\_ENC\_ERR\_NEED\_MORE\_INPUT** This indicates encode driver requires more input buffers to produce an output bitstream. If this error is returned from [NvEncEncodePicture\(\)](#) API, this is not a fatal error. If the client is encoding with B frames then, [NvEncEncodePicture\(\)](#) API might be buffering the input frame for re-ordering.

A client operating in synchronous mode cannot call [NvEncLockBitstream\(\)](#) API on the output bitstream buffer if [NvEncEncodePicture\(\)](#) returned the **NV\_ENC\_ERR\_NEED\_MORE\_INPUT** error code. The client must continue providing input frames until encode driver returns **NV\_ENC\_SUCCESS**. After receiving **NV\_ENC\_SUCCESS** status the client can call [NvEncLockBitstream\(\)](#) API on the output buffers in the same order in which it has called [NvEncEncodePicture\(\)](#).

**NV\_ENC\_ERR\_ENCODER\_BUSY** This indicates that the HW encoder is busy encoding and is unable to encode the input. The client should call [NvEncEncodePicture\(\)](#) again after few milliseconds.

***NV\_ENC\_ERR\_EVENT\_NOT\_REGISTERD*** This indicates that the completion event passed in [NvEncEncodePicture\(\)](#) API has not been registered with encoder driver using [NvEncRegisterAsyncEvent\(\)](#).

***NV\_ENC\_ERR\_GENERIC*** This indicates that an unknown internal error has occurred.

***NV\_ENC\_ERR\_INCOMPATIBLE\_CLIENT\_KEY*** This indicates that the client is attempting to use a feature that is not available for the license type for the current system.

***NV\_ENC\_ERR\_UNIMPLEMENTED*** This indicates that the client is attempting to use a feature that is not implemented for the current version.

***NV\_ENC\_ERR\_RESOURCE\_REGISTER\_FAILED*** This indicates that the [NvEncRegisterResource](#) API failed to register the resource.

***NV\_ENC\_ERR\_RESOURCE\_NOT\_REGISTERED*** This indicates that the client is attempting to unregister a resource that has not been successfully registered.

***NV\_ENC\_ERR\_RESOURCE\_NOT\_MAPPED*** This indicates that the client is attempting to unmap a resource that has not been successfully mapped.

#### 4.1.2.9 enum NV\_ENC\_PIC\_FLAGS

Encode Picture encode flags.

##### Enumerator:

***NV\_ENC\_PIC\_FLAG\_FORCEINTRA*** Encode the current picture as an Intra picture

***NV\_ENC\_PIC\_FLAG\_FORCEIDR*** Encode the current picture as an IDR picture. This flag is only valid when Picture type decision is taken by the Encoder [`_NV_ENC_INITIALIZE_PARAMS::enablePTD == 1`].

***NV\_ENC\_PIC\_FLAG\_OUTPUT\_SPSPPS*** Write the sequence and picture header in encoded bitstream of the current picture

***NV\_ENC\_PIC\_FLAG\_EOS*** Indicates end of the input stream

#### 4.1.2.10 enum NV\_ENC\_MEMORY\_HEAP

Memory heap to allocate input and output buffers.

##### Enumerator:

***NV\_ENC\_MEMORY\_HEAP\_AUTOSELECT*** Memory heap to be decided by the encoder driver based on the usage

***NV\_ENC\_MEMORY\_HEAP\_VID*** Memory heap is in local video memory

***NV\_ENC\_MEMORY\_HEAP\_SYSMEM\_CACHED*** Memory heap is in cached system memory

***NV\_ENC\_MEMORY\_HEAP\_SYSMEM\_UNCACHED*** Memory heap is in uncached system memory

#### 4.1.2.11 enum NV\_ENC\_H264\_ENTROPY\_CODING\_MODE

H.264 entropy coding modes.

##### Enumerator:

***NV\_ENC\_H264\_ENTROPY\_CODING\_MODE\_AUTOSELECT*** Entropy coding mode is auto selected by the encoder driver

***NV\_ENC\_H264\_ENTROPY\_CODING\_MODE\_CABAC*** Entropy coding mode is CABAC

***NV\_ENC\_H264\_ENTROPY\_CODING\_MODE\_CAVLC*** Entropy coding mode is CAVLC

#### 4.1.2.12 enum NV\_ENC\_H264\_BDIRECT\_MODE

H.264 specific Bdirect modes

**Enumerator:**

*NV\_ENC\_H264\_BDIRECT\_MODE\_AUTOSELECT* BDirect mode is auto selected by the encoder driver  
*NV\_ENC\_H264\_BDIRECT\_MODE\_DISABLE* Disable BDirect mode  
*NV\_ENC\_H264\_BDIRECT\_MODE\_TEMPORAL* Temporal BDirect mode  
*NV\_ENC\_H264\_BDIRECT\_MODE\_SPATIAL* Spatial BDirect mode

#### 4.1.2.13 enum NV\_ENC\_H264\_FMO\_MODE

H.264 specific FMO usage

**Enumerator:**

*NV\_ENC\_H264\_FMO\_AUTOSELECT* FMO usage is auto selected by the encoder driver  
*NV\_ENC\_H264\_FMO\_ENABLE* Enable FMO  
*NV\_ENC\_H264\_FMO\_DISABLE* Disble FMO

#### 4.1.2.14 enum NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_MODE

H.264 specific Adaptive Transform modes

**Enumerator:**

*NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_AUTOSELECT* Adaptive Transform 8x8 mode is auto selected by the encoder driver  
*NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_DISABLE* Adaptive Transform 8x8 mode disabled  
*NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_ENABLE* Adaptive Transform 8x8 mode should be used

#### 4.1.2.15 enum NV\_ENC\_STEREO\_PACKING\_MODE

Stereo frame packing modes.

**Enumerator:**

*NV\_ENC\_STEREO\_PACKING\_MODE\_NONE* No Stereo packing required  
*NV\_ENC\_STEREO\_PACKING\_MODE\_CHECKERBOARD* Checkerboard mode for packing stereo frames  
*NV\_ENC\_STEREO\_PACKING\_MODE\_COLINTERLEAVE* Column Interleave mode for packing stereo frames  
*NV\_ENC\_STEREO\_PACKING\_MODE\_ROWINTERLEAVE* Row Interleave mode for packing stereo frames  
  
*NV\_ENC\_STEREO\_PACKING\_MODE\_SIDEBYSIDE* Side-by-side mode for packing stereo frames  
*NV\_ENC\_STEREO\_PACKING\_MODE\_TOPBOTTOM* Top-Bottom mode for packing stereo frames  
*NV\_ENC\_STEREO\_PACKING\_MODE\_FRAMESEQ* Frame Sequential mode for packing stereo frames

#### 4.1.2.16 enum NV\_ENC\_INPUT\_RESOURCE\_TYPE

Input Resource type

##### Enumerator:

- NV\_ENC\_INPUT\_RESOURCE\_TYPE\_DIRECTX* input resource type is a directx9 surface
- NV\_ENC\_INPUT\_RESOURCE\_TYPE\_CUDADEVICEPTR* input resource type is a cuda device pointer surface
- NV\_ENC\_INPUT\_RESOURCE\_TYPE\_CUDAARRAY* input resource type is a cuda array surface
- NV\_ENC\_INPUT\_RESOURCE\_TYPE\_OPENGL\_TEX* input resource type is an OpenGL texture

#### 4.1.2.17 enum NV\_ENC\_DEVICE\_TYPE

Encoder Device type

##### Enumerator:

- NV\_ENC\_DEVICE\_TYPE\_DIRECTX* encode device type is a directx9 device
- NV\_ENC\_DEVICE\_TYPE\_CUDA* encode device type is a cuda device
- NV\_ENC\_DEVICE\_TYPE\_OPENGL* encode device type is an OpenGL device. Use of this device type is supported only on Linux

#### 4.1.2.18 enum NV\_ENC\_CAPS

Encoder capabilities enumeration.

##### Enumerator:

- NV\_ENC\_CAPS\_NUM\_MAX\_BFRAMES* Maximum number of B-Frames supported.
- NV\_ENC\_CAPS\_SUPPORTED\_RATECONTROL\_MODES* Rate control modes supported.  
The API return value is a bitmask of the values in NV\_ENC\_PARAMS\_RC\_MODE.
- NV\_ENC\_CAPS\_SUPPORT\_FIELD\_ENCODING* Indicates HW support for field mode encoding.
  - 0 : Interlaced mode encoding is not supported.
  - 1 : Interlaced field mode encoding is supported.
  - 2 : Interlaced frame encoding and field mode encoding are both supported.
- NV\_ENC\_CAPS\_SUPPORT\_MONOCHROME* Indicates HW support for monochrome mode encoding.
  - 0 : Monochrome mode not supported.
  - 1 : Monochrome mode supported.
- NV\_ENC\_CAPS\_SUPPORT\_FMO* Indicates HW support for FMO.
  - 0 : FMO not supported.
  - 1 : FMO supported.
- NV\_ENC\_CAPS\_SUPPORT\_QPELMV* Indicates HW capability for Quarter pel motion estimation.
  - 0 : QuarterPel Motion Estimation not supported.
  - 1 : QuarterPel Motion Estimation supported.
- NV\_ENC\_CAPS\_SUPPORT\_BDIRECT\_MODE* H.264 specific. Indicates HW support for BDirect modes.
  - 0 : BDirect mode encoding not supported.
  - 1 : BDirect mode encoding supported.

- NV\_ENC\_CAPS\_SUPPORT\_CABAC*** H264 specific. Indicates HW support for CABAC entropy coding mode.  
 0 : CABAC entropy coding not supported.  
 1 : CABAC entropy coding supported.
- NV\_ENC\_CAPS\_SUPPORT\_ADAPTIVE\_TRANSFORM*** Indicates HW support for Adaptive Transform.  
 0 : Adaptive Transform not supported.  
 1 : Adaptive Transform supported.
- NV\_ENC\_CAPS\_SUPPORT\_RESERVED*** Reserved enum field.
- NV\_ENC\_CAPS\_NUM\_MAX\_TEMPORAL\_LAYERS*** Indicates HW support for encoding Temporal layers.  
 0 : Encoding Temporal layers not supported.  
 1 : Encoding Temporal layers supported.
- NV\_ENC\_CAPS\_SUPPORT\_HIERARCHICAL\_PFRAMES*** Indicates HW support for Hierarchical P frames.  
 0 : Hierarchical P frames not supported.  
 1 : Hierarchical P frames supported.
- NV\_ENC\_CAPS\_SUPPORT\_HIERARCHICAL\_BFRAMES*** Indicates HW support for Hierarchical B frames.  
 0 : Hierarchical B frames not supported.  
 1 : Hierarchical B frames supported.
- NV\_ENC\_CAPS\_LEVEL\_MAX*** Maximum Encoding level supported (See [NV\\_ENC\\_LEVEL](#) for details).
- NV\_ENC\_CAPS\_LEVEL\_MIN*** Minimum Encoding level supported (See [NV\\_ENC\\_LEVEL](#) for details).
- NV\_ENC\_CAPS\_SEPARATE\_COLOUR\_PLANE*** Indicates HW support for separate colour plane encoding.  
 0 : Separate colour plane encoding not supported.  
 1 : Separate colour plane encoding supported.
- NV\_ENC\_CAPS\_WIDTH\_MAX*** Maximum output width supported.
- NV\_ENC\_CAPS\_HEIGHT\_MAX*** Maximum output height supported.
- NV\_ENC\_CAPS\_SUPPORT\_TEMPORAL\_SVC*** Indicates Temporal Scalability Support.  
 0 : Temporal SVC encoding not supported.  
 1 : Temporal SVC encoding supported.
- NV\_ENC\_CAPS\_SUPPORT\_DYN\_RES\_CHANGE*** Indicates Dynamic Encode Resolution Change Support. Support added from NvEncodeAPI version 2.0.  
 0 : Dynamic Encode Resolution Change not supported.  
 1 : Dynamic Encode Resolution Change supported.
- NV\_ENC\_CAPS\_SUPPORT\_DYN\_BITRATE\_CHANGE*** Indicates Dynamic Encode Bitrate Change Support. Support added from NvEncodeAPI version 2.0.  
 0 : Dynamic Encode bitrate change not supported.  
 1 : Dynamic Encode bitrate change supported.
- NV\_ENC\_CAPS\_SUPPORT\_DYN\_FORCE\_CONSTQP*** Indicates Forcing Constant QP On The Fly Support. Support added from NvEncodeAPI version 2.0.  
 0 : Forcing constant QP on the fly not supported.  
 1 : Forcing constant QP on the fly supported.
- NV\_ENC\_CAPS\_SUPPORT\_DYN\_RCMODE\_CHANGE*** Indicates Dynamic rate control mode Change Support.  
 0 : Dynamic rate control mode change not supported.  
 1 : Dynamic rate control mode change supported.

- NV\_ENC\_CAPS\_SUPPORT\_SUBFRAME\_READBACK*** Indicates Subframe readback support for slice-based encoding.  
 0 : Subframe readback not supported.  
 1 : Subframe readback supported.
- NV\_ENC\_CAPS\_SUPPORT\_CONSTRAINED\_ENCODING*** Indicates Constrained Encoding mode support. Support added from NvEncodeAPI version 2.0.  
 0 : Constrained encoding mode not supported.  
 1 : Constrained encoding mode supported. If this mode is supported client can enable this during initialisation. Client can then force a picture to be coded as constrained picture where each slice in a constrained picture will have `constrained_intra_pred_flag` set to 1 and `disable_deblocking_filter_idc` will be set to 2 and prediction vectors for inter macroblocks in each slice will be restricted to the slice region.
- NV\_ENC\_CAPS\_SUPPORT\_INTRA\_REFRESH*** Indicates Intra Refresh Mode Support. Support added from NvEncodeAPI version 2.0.  
 0 : Intra Refresh Mode not supported.  
 1 : Intra Refresh Mode supported.
- NV\_ENC\_CAPS\_SUPPORT\_CUSTOM\_VBV\_BUF\_SIZE*** Indicates Custom VBV Buffer Size support. It can be used for capping frame size. Support added from NvEncodeAPI version 2.0.  
 0 : Custom VBV buffer size specification from client, not supported.  
 1 : Custom VBV buffer size specification from client, supported.
- NV\_ENC\_CAPS\_SUPPORT\_DYNAMIC\_SLICE\_MODE*** Indicates Dynamic Slice Mode Support. Support added from NvEncodeAPI version 2.0.  
 0 : Dynamic Slice Mode not supported.  
 1 : Dynamic Slice Mode supported.
- NV\_ENC\_CAPS\_SUPPORT\_REF\_PIC\_INVALIDATION*** Indicates Reference Picture Invalidation Support. Support added from NvEncodeAPI version 2.0.  
 0 : Reference Picture Invalidation not supported.  
 1 : Reference Picture Invalidation supported.
- NV\_ENC\_CAPS\_PREPROC\_SUPPORT*** Indicates support for PreProcessing. The API return value is a bit-mask of the values defined in `NV_ENC_PREPROC_FLAGS`
- NV\_ENC\_CAPS\_ASYNC\_ENCODE\_SUPPORT*** Indicates support Async mode.  
 0 : Async Encode mode not supported.  
 1 : Async Encode mode supported.
- NV\_ENC\_CAPS\_MB\_NUM\_MAX*** Maximum MBs per frame supported.
- NV\_ENC\_CAPS\_MB\_PER\_SEC\_MAX*** Maximum aggregate throughput in MBs per sec.
- NV\_ENC\_CAPS\_SUPPORT\_YUV444\_ENCODE*** Indicates HW support for YUV444 mode encoding.  
 0 : YUV444 mode encoding not supported.  
 1 : YUV444 mode encoding supported.
- NV\_ENC\_CAPS\_SUPPORT\_LOSSLESS\_ENCODE*** Indicates HW support for lossless encoding.  
 0 : lossless encoding not supported.  
 1 : lossless encoding supported.
- NV\_ENC\_CAPS\_SUPPORT\_SAO*** Indicates HW support for Sample Adaptive Offset.  
 0 : SAO not supported.  
 1 : SAO encoding supported.
- NV\_ENC\_CAPS\_SUPPORT\_MEONLY\_MODE*** Indicates HW support for MEOnly Mode.  
 0 : MEOnly Mode not supported.  
 1 : MEOnly Mode supported for I and P frames.  
 2 : MEOnly Mode supported for I, P and B frames.



***NV\_ENC\_CAPS\_SUPPORT\_LOOKAHEAD*** Indicates HW support for lookahead encoding (enableLookahead=1).

0 : Lookahead not supported.

1 : Lookahead supported.

***NV\_ENC\_CAPS\_SUPPORT\_TEMPORAL\_AQ*** Indicates HW support for temporal AQ encoding (enableTemporalAQ=1).

0 : Temporal AQ not supported.

1 : Temporal AQ supported.

***NV\_ENC\_CAPS\_SUPPORT\_10BIT\_ENCODE*** Indicates HW support for 10 bit encoding.

0 : 10 bit encoding not supported.

1 : 10 bit encoding supported.

***NV\_ENC\_CAPS\_NUM\_MAX\_LTR\_FRAMES*** Maximum number of Long Term Reference frames supported

***NV\_ENC\_CAPS\_SUPPORT\_WEIGHTED\_PREDICTION*** Indicates HW support for Weighted Prediction.

0 : Weighted Prediction not supported.

1 : Weighted Prediction supported.

***NV\_ENC\_CAPS\_EXPOSED\_COUNT*** Reserved - Not to be used by clients.

#### 4.1.2.19 enum NV\_ENC\_HEVC\_CUSIZE

HEVC CU SIZE

## 4.2 NvEncodeAPI Functions

### Functions

- **NVENCSTATUS** NVENCAPI **NvEncOpenEncodeSession** (void \*device, uint32\_t deviceType, void \*\*encoder)  
*Opens an encoding session.*
- **NVENCSTATUS** NVENCAPI **NvEncGetEncodeGUIDCount** (void \*encoder, uint32\_t \*encodeGUIDCount)  
*Retrieves the number of supported encode GUIDs.*
- **NVENCSTATUS** NVENCAPI **NvEncGetEncodeGUIDs** (void \*encoder, GUID \*GUIDs, uint32\_t guidArraySize, uint32\_t \*GUIDCount)  
*Retrieves an array of supported encoder codec GUIDs.*
- **NVENCSTATUS** NVENCAPI **NvEncGetEncodeProfileGUIDCount** (void \*encoder, GUID encodeGUID, uint32\_t \*encodeProfileGUIDCount)  
*Retrieves the number of supported profile GUIDs.*
- **NVENCSTATUS** NVENCAPI **NvEncGetEncodeProfileGUIDs** (void \*encoder, GUID encodeGUID, GUID \*profileGUIDs, uint32\_t guidArraySize, uint32\_t \*GUIDCount)  
*Retrieves an array of supported encode profile GUIDs.*
- **NVENCSTATUS** NVENCAPI **NvEncGetInputFormatCount** (void \*encoder, GUID encodeGUID, uint32\_t \*inputFmtCount)  
*Retrieve the number of supported Input formats.*
- **NVENCSTATUS** NVENCAPI **NvEncGetInputFormats** (void \*encoder, GUID encodeGUID, NV\_ENC\_BUFFER\_FORMAT \*inputFmts, uint32\_t inputFmtArraySize, uint32\_t \*inputFmtCount)  
*Retrieves an array of supported Input formats.*
- **NVENCSTATUS** NVENCAPI **NvEncGetEncodeCaps** (void \*encoder, GUID encodeGUID, NV\_ENC\_CAPS\_PARAM \*capsParam, int \*capsVal)  
*Retrieves the capability value for a specified encoder attribute.*
- **NVENCSTATUS** NVENCAPI **NvEncGetEncodePresetCount** (void \*encoder, GUID encodeGUID, uint32\_t \*encodePresetGUIDCount)  
*Retrieves the number of supported preset GUIDs.*
- **NVENCSTATUS** NVENCAPI **NvEncGetEncodePresetGUIDs** (void \*encoder, GUID encodeGUID, GUID \*presetGUIDs, uint32\_t guidArraySize, uint32\_t \*encodePresetGUIDCount)  
*Receives an array of supported encoder preset GUIDs.*
- **NVENCSTATUS** NVENCAPI **NvEncGetEncodePresetConfig** (void \*encoder, GUID encodeGUID, GUID presetGUID, NV\_ENC\_PRESET\_CONFIG \*presetConfig)  
*Returns a preset config structure supported for given preset GUID.*
- **NVENCSTATUS** NVENCAPI **NvEncInitializeEncoder** (void \*encoder, NV\_ENC\_INITIALIZE\_PARAMS \*createEncodeParams)  
*Initialize the encoder.*

- **NVENCSTATUS** NVENCAPI **NvEncCreateInputBuffer** (void \*encoder, NV\_ENC\_CREATE\_INPUT\_BUFFER \*createInputBufferParams)  
*Allocates Input buffer.*
- **NVENCSTATUS** NVENCAPI **NvEncDestroyInputBuffer** (void \*encoder, NV\_ENC\_INPUT\_PTR inputBuffer)  
*Release an input buffers.*
- **NVENCSTATUS** NVENCAPI **NvEncCreateBitstreamBuffer** (void \*encoder, NV\_ENC\_CREATE\_BITSTREAM\_BUFFER \*createBitstreamBufferParams)  
*Allocates an output bitstream buffer.*
- **NVENCSTATUS** NVENCAPI **NvEncDestroyBitstreamBuffer** (void \*encoder, NV\_ENC\_OUTPUT\_PTR bitstreamBuffer)  
*Release a bitstream buffer.*
- **NVENCSTATUS** NVENCAPI **NvEncEncodePicture** (void \*encoder, NV\_ENC\_PIC\_PARAMS \*encodePicParams)  
*Submit an input picture for encoding.*
- **NVENCSTATUS** NVENCAPI **NvEncLockBitstream** (void \*encoder, NV\_ENC\_LOCK\_BITSTREAM \*lockBitstreamBufferParams)  
*Lock output bitstream buffer.*
- **NVENCSTATUS** NVENCAPI **NvEncUnlockBitstream** (void \*encoder, NV\_ENC\_OUTPUT\_PTR bitstreamBuffer)  
*Unlock the output bitstream buffer.*
- **NVENCSTATUS** NVENCAPI **NvEncLockInputBuffer** (void \*encoder, NV\_ENC\_LOCK\_INPUT\_BUFFER \*lockInputBufferParams)  
*Locks an input buffer.*
- **NVENCSTATUS** NVENCAPI **NvEncUnlockInputBuffer** (void \*encoder, NV\_ENC\_INPUT\_PTR inputBuffer)  
*Unlocks the input buffer.*
- **NVENCSTATUS** NVENCAPI **NvEncGetEncodeStats** (void \*encoder, NV\_ENC\_STAT \*encodeStats)  
*Get encoding statistics.*
- **NVENCSTATUS** NVENCAPI **NvEncGetSequenceParams** (void \*encoder, NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD \*sequenceParamPayload)  
*Get encoded sequence and picture header.*
- **NVENCSTATUS** NVENCAPI **NvEncRegisterAsyncEvent** (void \*encoder, NV\_ENC\_EVENT\_PARAMS \*eventParams)  
*Register event for notification to encoding completion.*
- **NVENCSTATUS** NVENCAPI **NvEncUnregisterAsyncEvent** (void \*encoder, NV\_ENC\_EVENT\_PARAMS \*eventParams)  
*Unregister completion event.*

- **NVENCSTATUS** NVENCAPI **NvEncMapInputResource** (void \*encoder, NV\_ENC\_MAP\_INPUT\_RESOURCE \*mapInputResParams)  
*Map an externally created input resource pointer for encoding.*
- **NVENCSTATUS** NVENCAPI **NvEncUnmapInputResource** (void \*encoder, NV\_ENC\_INPUT\_PTR mapped-InputBuffer)  
*UnMaps a NV\_ENC\_INPUT\_PTR which was mapped for encoding.*
- **NVENCSTATUS** NVENCAPI **NvEncDestroyEncoder** (void \*encoder)  
*Destroy Encoding Session.*
- **NVENCSTATUS** NVENCAPI **NvEncInvalidateRefFrames** (void \*encoder, uint64\_t invalidRefFrameTimeStamp)  
*Invalidate reference frames.*
- **NVENCSTATUS** NVENCAPI **NvEncOpenEncodeSessionEx** (NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_PARAMS \*openSessionExParams, void \*\*encoder)  
*Opens an encoding session.*
- **NVENCSTATUS** NVENCAPI **NvEncRegisterResource** (void \*encoder, NV\_ENC\_REGISTER\_RESOURCE \*registerResParams)  
*Registers a resource with the Nvidia Video Encoder Interface.*
- **NVENCSTATUS** NVENCAPI **NvEncUnregisterResource** (void \*encoder, NV\_ENC\_REGISTERED\_PTR registeredResource)  
*Unregisters a resource previously registered with the Nvidia Video Encoder Interface.*
- **NVENCSTATUS** NVENCAPI **NvEncReconfigureEncoder** (void \*encoder, NV\_ENC\_RECONFIGURE\_PARAMS \*reInitEncodeParams)  
*Reconfigure an existing encoding session.*
- **NVENCSTATUS** NVENCAPI **NvEncCreateMVBuffer** (void \*encoder, NV\_ENC\_CREATE\_MV\_BUFFER \*createMVBufferParams)  
*Allocates output MV buffer for ME only mode.*
- **NVENCSTATUS** NVENCAPI **NvEncDestroyMVBuffer** (void \*encoder, NV\_ENC\_OUTPUT\_PTR mvBuffer)  
*Release an output MV buffer for ME only mode.*
- **NVENCSTATUS** NVENCAPI **NvEncRunMotionEstimationOnly** (void \*encoder, NV\_ENC\_MEONLY\_PARAMS \*meOnlyParams)  
*Submit an input picture and reference frame for motion estimation in ME only mode.*
- **NVENCSTATUS** NVENCAPI **NvEncodeAPIGetMaxSupportedVersion** (uint32\_t \*version)  
*Get the largest NvEncodeAPI version supported by the driver.*
- **NVENCSTATUS** NVENCAPI **NvEncodeAPICreateInstance** (NV\_ENCODE\_API\_FUNCTION\_LIST \*functionList)

### 4.2.1 Function Documentation

#### 4.2.1.1 NVENCSTATUS NVENCAPI NvEncOpenEncodeSession (void \* *device*, uint32\_t *deviceType*, void \*\* *encoder*)

Deprecated.

Returns:

[NV\\_ENC\\_ERR\\_INVALID\\_CALL](#)

#### 4.2.1.2 NVENCSTATUS NVENCAPI NvEncGetEncodeGUIDCount (void \* *encoder*, uint32\_t \* *encodeGUIDCount*)

The function returns the number of codec guids supported by the NvEncodeAPI interface.

Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- *encodeGUIDCount* Number of supported encode GUIDs.

Returns:

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.3 NVENCSTATUS NVENCAPI NvEncGetEncodeGUIDs (void \* *encoder*, GUID \* *GUIDs*, uint32\_t *guidArraySize*, uint32\_t \* *GUIDCount*)

The function returns an array of codec guids supported by the NvEncodeAPI interface. The client must allocate an array where the NvEncodeAPI interface can fill the supported guids and pass the pointer in \**GUIDs* parameter. The size of the array can be determined by using [NvEncGetEncodeGUIDCount\(\)](#) API. The Nvidia Encoding interface returns the number of codec guids it has actually filled in the guid array in the *GUIDCount* parameter.

Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *guidArraySize* Number of GUIDs to retrieved. Should be set to the number retrieved using [NvEncGetEncodeGUIDCount](#).
- *GUIDs* Array of supported Encode GUIDs.
- *GUIDCount* Number of supported Encode GUIDs.

Returns:

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)

[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.4 NVENCSTATUS NVENCAPI [NvEncGetEncodeProfileGUIDCount](#) (void \* *encoder*, GUID *encodeGUID*, uint32\_t \* *encodeProfileGUIDCount*)

The function returns the number of profile GUIDs supported for a given codec. The client must first enumerate the codec guides supported by the NvEncodeAPI interface. After determining the codec guid, it can query the NvEncodeAPI interface to determine the number of profile guides supported for a particular codec guid.

##### Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *encodeGUID* The codec guid for which the profile guides are being enumerated.
- *encodeProfileGUIDCount* Number of encode profiles supported for the given encodeGUID.

##### Returns:

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.5 NVENCSTATUS NVENCAPI [NvEncGetEncodeProfileGUIDs](#) (void \* *encoder*, GUID *encodeGUID*, GUID \* *profileGUIDs*, uint32\_t *guidArraySize*, uint32\_t \* *GUIDCount*)

The function returns an array of supported profile guides for a particular codec guid. The client must allocate an array where the NvEncodeAPI interface can populate the profile guides. The client can determine the array size using [NvEncGetEncodeProfileGUIDCount\(\)](#) API. The client must also validate that the NvEncodeAPI interface supports the GUID the client wants to pass as encodeGUID parameter.

##### Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *encodeGUID* The encode guid whose profile guides are being enumerated.
- ← *guidArraySize* Number of GUIDs to be retrieved. Should be set to the number retrieved using [NvEncGetEncodeProfileGUIDCount](#).
- *profileGUIDs* Array of supported Encode Profile GUIDs
- *GUIDCount* Number of valid encode profile GUIDs in *profileGUIDs* array.

##### Returns:

[NV\\_ENC\\_SUCCESS](#)

[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.6 NVENCSTATUS NVENCAPI NvEncGetInputFormatCount (void \* *encoder*, GUID *encodeGUID*, uint32\_t \* *inputFmtCount*)

The function returns the number of supported input formats. The client must query the NvEncodeAPI interface to determine the supported input formats before creating the input surfaces.

##### Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *encodeGUID* Encode [GUID](#), corresponding to which the number of supported input formats is to be retrieved.
- *inputFmtCount* Number of input formats supported for specified Encode [GUID](#).

##### Returns:

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.7 NVENCSTATUS NVENCAPI NvEncGetInputFormats (void \* *encoder*, GUID *encodeGUID*, NV\_ENC\_BUFFER\_FORMAT \* *inputFmts*, uint32\_t *inputFmtArraySize*, uint32\_t \* *inputFmtCount*)

Returns an array of supported input formats. The client must use the input format to create input surface using [NvEncCreateInputBuffer\(\)](#) API.

##### Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *encodeGUID* Encode [GUID](#), corresponding to which the number of supported input formats is to be retrieved.
- ← *inputFmtArraySize* Size input format count array passed in *inputFmts*.
- *inputFmts* Array of input formats supported for this Encode [GUID](#).
- *inputFmtCount* The number of valid input format types returned by the NvEncodeAPI interface in *inputFmts* array.

##### Returns:

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)

[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.8 NVENCSTATUS NVENCAPI NvEncGetEncodeCaps (void \* *encoder*, GUID *encodeGUID*, NV\_ENC\_CAPS\_PARAM \* *capsParam*, int \* *capsVal*)

The function returns the capability value for a given encoder attribute. The client must validate the encodeGUID using [NvEncGetEncodeGUIDs\(\)](#) API before calling this function. The encoder attribute being queried are enumerated in [NV\\_ENC\\_CAPS\\_PARAM](#) enum.

##### Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *encodeGUID* Encode [GUID](#), corresponding to which the capability attribute is to be retrieved.
- ← *capsParam* Used to specify attribute being queried. Refer [NV\\_ENC\\_CAPS\\_PARAM](#) for more details.
- *capsVal* The value corresponding to the capability attribute being queried.

##### Returns:

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.9 NVENCSTATUS NVENCAPI NvEncGetEncodePresetCount (void \* *encoder*, GUID *encodeGUID*, uint32\_t \* *encodePresetGUIDCount*)

The function returns the number of preset GUIDs available for a given codec. The client must validate the codec guid using [NvEncGetEncodeGUIDs\(\)](#) API before calling this function.

##### Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *encodeGUID* Encode [GUID](#), corresponding to which the number of supported presets is to be retrieved.
- *encodePresetGUIDCount* Receives the number of supported preset GUIDs.

##### Returns:

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)



#### 4.2.1.10 NVENCSTATUS NVENCAPI NvEncGetEncodePresetGUIDs (void \* *encoder*, GUID *encodeGUID*, GUID \* *presetGUIDs*, uint32\_t *guidArraySize*, uint32\_t \* *encodePresetGUIDCount*)

The function returns an array of encode preset guides available for a given codec. The client can directly use one of the preset guides based upon the use case or target device. The preset guide chosen can be directly used in `NV_ENC_INITIALIZE_PARAMS::presetGUID` parameter to `NvEncEncodePicture()` API. Alternately client can also use the preset guide to retrieve the encoding config parameters being used by NvEncodeAPI interface for that given preset, using `NvEncGetEncodePresetConfig()` API. It can then modify preset config parameters as per its use case and send it to NvEncodeAPI interface as part of `NV_ENC_INITIALIZE_PARAMS::encodeConfig` parameter for `NvEncInitializeEncoder()` API.

##### Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *encodeGUID* Encode GUID, corresponding to which the list of supported presets is to be retrieved.
- ← *guidArraySize* Size of array of preset guides passed in preset GUIDs
- *presetGUIDs* Array of supported Encode preset GUIDs from the NvEncodeAPI interface to client.
- *encodePresetGUIDCount* Receives the number of preset GUIDs returned by the NvEncodeAPI interface.

##### Returns:

NV\_ENC\_SUCCESS  
 NV\_ENC\_ERR\_INVALID\_PTR  
 NV\_ENC\_ERR\_INVALID\_ENCODERDEVICE  
 NV\_ENC\_ERR\_DEVICE\_NOT\_EXIST  
 NV\_ENC\_ERR\_UNSUPPORTED\_PARAM  
 NV\_ENC\_ERR\_OUT\_OF\_MEMORY  
 NV\_ENC\_ERR\_INVALID\_PARAM  
 NV\_ENC\_ERR\_GENERIC

#### 4.2.1.11 NVENCSTATUS NVENCAPI NvEncGetEncodePresetConfig (void \* *encoder*, GUID *encodeGUID*, GUID *presetGUID*, NV\_ENC\_PRESET\_CONFIG \* *presetConfig*)

The function returns a preset config structure for a given preset guide. Before using this function the client must enumerate the preset guides available for a given codec. The preset config structure can be modified by the client depending upon its use case and can be then used to initialize the encoder using `NvEncInitializeEncoder()` API. The client can use this function only if it wants to modify the NvEncodeAPI preset configuration, otherwise it can directly use the preset guide.

##### Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *encodeGUID* Encode GUID, corresponding to which the list of supported presets is to be retrieved.
- ← *presetGUID* Preset GUID, corresponding to which the Encoding configurations is to be retrieved.
- *presetConfig* The requested Preset Encoder Attribute set. Refer `_NV_ENC_CONFIG` for more details.

##### Returns:

NV\_ENC\_SUCCESS  
 NV\_ENC\_ERR\_INVALID\_PTR  
 NV\_ENC\_ERR\_INVALID\_ENCODERDEVICE  
 NV\_ENC\_ERR\_DEVICE\_NOT\_EXIST

```

NV_ENC_ERR_UNSUPPORTED_PARAM
NV_ENC_ERR_OUT_OF_MEMORY
NV_ENC_ERR_INVALID_PARAM
NV_ENC_ERR_INVALID_VERSION
NV_ENC_ERR_GENERIC

```

#### 4.2.1.12 NVENCSTATUS NVENCAPI NvEncInitializeEncoder (void \* *encoder*, NV\_ENC\_INITIALIZE\_PARAMS \* *createEncodeParams*)

This API must be used to initialize the encoder. The initialization parameter is passed using *\*createEncodeParams*. The client must send the following fields of the `_NV_ENC_INITIALIZE_PARAMS` structure with a valid value.

- `NV_ENC_INITIALIZE_PARAMS::encodeGUID`
- `NV_ENC_INITIALIZE_PARAMS::encodeWidth`
- `NV_ENC_INITIALIZE_PARAMS::encodeHeight`

The client can pass a preset guid directly to the NvEncodeAPI interface using `NV_ENC_INITIALIZE_PARAMS::presetGUID` field. If the client doesn't pass `NV_ENC_INITIALIZE_PARAMS::encodeConfig` structure, the codec specific parameters will be selected based on the preset guid. The preset guid must have been validated by the client using `NvEncGetEncodePresetGUIDs()` API. If the client passes a custom `_NV_ENC_CONFIG` structure through `NV_ENC_INITIALIZE_PARAMS::encodeConfig`, it will override the codec specific parameters based on the preset guid. It is recommended that even if the client passes a custom config, it should also send a preset guid. In this case, the preset guid passed by the client will not override any of the custom config parameters programmed by the client, it is only used as a hint by the NvEncodeAPI interface to determine certain encoder parameters which are not exposed to the client.

There are two modes of operation for the encoder namely:

- Asynchronous mode
- Synchronous mode

The client can select asynchronous or synchronous mode by setting the `enableEncodeAsync` field in `_NV_ENC_INITIALIZE_PARAMS` to 1 or 0 respectively.

##### Asynchronous mode of operation:

The Asynchronous mode can be enabled by setting `NV_ENC_INITIALIZE_PARAMS::enableEncodeAsync` to 1. The client operating in asynchronous mode must allocate completion event object for each output buffer and pass the completion event object in the `NvEncEncodePicture()` API. The client can create another thread and wait on the event object to be signalled by NvEncodeAPI interface on completion of the encoding process for the output frame. This should unblock the main thread from submitting work to the encoder. When the event is signalled the client can call NvEncodeAPI interfaces to copy the bitstream data using `NvEncLockBitstream()` API. This is the preferred mode of operation.

NOTE: Asynchronous mode is not supported on Linux.

##### Synchronous mode of operation:

The client can select synchronous mode by setting `NV_ENC_INITIALIZE_PARAMS::enableEncodeAsync` to 0. The client working in synchronous mode can work in a single threaded or multi threaded mode. The client

need not allocate any event objects. The client can only lock the bitstream data after NvEncodeAPI interface has returned `NV_ENC_SUCCESS` from encode picture. The NvEncodeAPI interface can return `NV_ENC_ERR_NEED_MORE_INPUT` error code from `NvEncEncodePicture()` API. The client must not lock the output buffer in such case but should send the next frame for encoding. The client must keep on calling `NvEncEncodePicture()` API until it returns `NV_ENC_SUCCESS`.

The client must always lock the bitstream data in order in which it has submitted. This is true for both asynchronous and synchronous mode.

#### Picture type decision:

If the client is taking the picture type decision and it must disable the picture type decision module in NvEncodeAPI by setting `NV_ENC_INITIALIZE_PARAMS::enablePTD` to 0. In this case the client is required to send the picture in encoding order to NvEncodeAPI by doing the re-ordering for B frames.

If the client doesn't want to take the picture type decision it can enable picture type decision module in the NvEncodeAPI interface by setting `NV_ENC_INITIALIZE_PARAMS::enablePTD` to 1 and send the input pictures in display order.

#### Parameters:

← *encoder* Pointer to the NvEncodeAPI interface.

← *createEncodeParams* Refer `_NV_ENC_INITIALIZE_PARAMS` for details.

#### Returns:

`NV_ENC_SUCCESS`  
`NV_ENC_ERR_INVALID_PTR`  
`NV_ENC_ERR_INVALID_ENCODERDEVICE`  
`NV_ENC_ERR_DEVICE_NOT_EXIST`  
`NV_ENC_ERR_UNSUPPORTED_PARAM`  
`NV_ENC_ERR_OUT_OF_MEMORY`  
`NV_ENC_ERR_INVALID_PARAM`  
`NV_ENC_ERR_INVALID_VERSION`  
`NV_ENC_ERR_GENERIC`

#### 4.2.1.13 NVENCSTATUS NVENCAPI NvEncCreateInputBuffer (void \* *encoder*, NV\_ENC\_CREATE\_INPUT\_BUFFER \* *createInputBufferParams*)

This function is used to allocate an input buffer. The client must enumerate the input buffer format before allocating the input buffer resources. The `NV_ENC_INPUT_PTR` returned by the NvEncodeAPI interface in the `NV_ENC_CREATE_INPUT_BUFFER::inputBuffer` field can be directly used in `NvEncEncodePicture()` API. The number of input buffers to be allocated by the client must be at least 4 more than the number of B frames being used for encoding.

#### Parameters:

← *encoder* Pointer to the NvEncodeAPI interface.

↔ *createInputBufferParams* Pointer to the `NV_ENC_CREATE_INPUT_BUFFER` structure.

#### Returns:

`NV_ENC_SUCCESS`  
`NV_ENC_ERR_INVALID_PTR`  
`NV_ENC_ERR_INVALID_ENCODERDEVICE`  
`NV_ENC_ERR_DEVICE_NOT_EXIST`  
`NV_ENC_ERR_UNSUPPORTED_PARAM`

[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.14 NVENCSTATUS NVENCAPI NvEncDestroyInputBuffer (void \* *encoder*, NV\_ENC\_INPUT\_PTR *inputBuffer*)

This function is used to free an input buffer. If the client has allocated any input buffer using [NvEncCreateInputBuffer\(\)](#) API, it must free those input buffers by calling this function. The client must release the input buffers before destroying the encoder using [NvEncDestroyEncoder\(\)](#) API.

##### Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *inputBuffer* Pointer to the input buffer to be released.

##### Returns:

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.15 NVENCSTATUS NVENCAPI NvEncCreateBitstreamBuffer (void \* *encoder*, NV\_ENC\_CREATE\_BITSTREAM\_BUFFER \* *createBitstreamBufferParams*)

This function is used to allocate an output bitstream buffer and returns a NV\_ENC\_OUTPUT\_PTR to bitstream buffer to the client in the [NV\\_ENC\\_CREATE\\_BITSTREAM\\_BUFFER::bitstreamBuffer](#) field. The client can only call this function after the encoder session has been initialized using [NvEncInitializeEncoder\(\)](#) API. The minimum number of output buffers allocated by the client must be at least 4 more than the number of B B frames being used for encoding. The client can only access the output bitstream data by locking the *bitstreamBuffer* using the [NvEncLockBitstream\(\)](#) function.

##### Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ↔ *createBitstreamBufferParams* Pointer [NV\\_ENC\\_CREATE\\_BITSTREAM\\_BUFFER](#) for details.

##### Returns:

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)

[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.16 NVENCSTATUS NVENCAPI NvEncDestroyBitstreamBuffer (void \* *encoder*, NV\_ENC\_OUTPUT\_PTR *bitstreamBuffer*)

This function is used to release the output bitstream buffer allocated using the [NvEncCreateBitstreamBuffer\(\)](#) function. The client must release the output bitstreamBuffer using this function before destroying the encoder session.

##### Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *bitstreamBuffer* Pointer to the bitstream buffer being released.

##### Returns:

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.17 NVENCSTATUS NVENCAPI NvEncEncodePicture (void \* *encoder*, NV\_ENC\_PIC\_PARAMS \* *encodePicParams*)

This function is used to submit an input picture buffer for encoding. The encoding parameters are passed using \*encodePicParams which is a pointer to the NV\_ENC\_PIC\_PARAMS structure.

If the client has set [NV\\_ENC\\_INITIALIZE\\_PARAMS::enablePTD](#) to 0, then it must send a valid value for the following fields.

- [NV\\_ENC\\_PIC\\_PARAMS::pictureType](#)
- [NV\\_ENC\\_PIC\\_PARAMS\\_H264::displayPOCSyntax](#) (H264 only)
- [NV\\_ENC\\_PIC\\_PARAMS\\_H264::frameNumSyntax](#)(H264 only)
- [NV\\_ENC\\_PIC\\_PARAMS\\_H264::refPicFlag](#)(H264 only)

##### Asynchronous Encoding

If the client has enabled asynchronous mode of encoding by setting [NV\\_ENC\\_INITIALIZE\\_PARAMS::enableEncodeAsync](#) to 1 in the [NvEncInitializeEncoder\(\)](#) API ,then the client must send a valid [NV\\_ENC\\_PIC\\_PARAMS::completionEvent](#). Incase of asynchronous mode of operation, client can queue the [NvEncEncodePicture\(\)](#) API commands from the main thread and then queue output buffers to be processed to a secondary worker thread. Before the locking the output buffers in the secondary thread , the client must wait on

[NV\\_ENC\\_PIC\\_PARAMS::completionEvent](#) it has queued in [NvEncEncodePicture\(\)](#) API call. The client must always process completion event and the output buffer in the same order in which they have been submitted for encoding. The [NvEncodeAPI](#) interface is responsible for any re-ordering required for B frames and will always ensure that encoded bitstream data is written in the same order in which output buffer is submitted.

The below example shows how asynchronous encoding in case of 1 B frames

-----  
 Suppose the client allocated 4 input buffers(I1,I2..), 4 output buffers(O1,O2..) and 4 completion events(E1, E2, ...). The [NvEncodeAPI](#) interface will need to keep a copy of the input buffers for re-ordering and it allocates following internal buffers (NvI1, NvI2...). These internal buffers are managed by [NvEncodeAPI](#) and the client is not responsible for the allocating or freeing the memory of the internal buffers.

a) The client main thread will queue the following encode frame calls.  
 Note the picture type is unknown to the client, the decision is being taken by [NvEncodeAPI](#) interface. The client should pass [:\\_NV\\_ENC\\_PIC\\_PARAMS](#) parameter consisting of allocated input buffer, output buffer and output events in successive [::NvEncEncodePicture\(\)](#) API calls along with other required encode picture params.

For example:

1st EncodePicture parameters - (I1, O1, E1)  
 2nd EncodePicture parameters - (I2, O2, E2)  
 3rd EncodePicture parameters - (I3, O3, E3)

b) [NvEncodeAPI](#) SW will receive the following encode Commands from the client. The left side shows input from client in the form (Input buffer, Output Buffer, Output Event). The right hand side shows a possible picture type decision take by the [NvEncodeAPI](#) interface.

(I1, O1, E1) ---P1 Frame  
 (I2, O2, E2) ---B2 Frame  
 (I3, O3, E3) ---P3 Frame

c) [NvEncodeAPI](#) interface will make a copy of the input buffers to its internal buffers for re-ordering. These copies are done as part of [nvEncEncodePicture](#) function call from the client and [NvEncodeAPI](#) interface is responsible for synchronization of copy operation with the actual encoding operation.

I1 --> NvI1  
 I2 --> NvI2  
 I3 --> NvI3

d) After returning from [::NvEncEncodePicture\(\)](#) call, the client must queue the output bitstream processing work to the secondary thread. The output bitstream processing for asynchronous mode consist of first waiting on completion event(E1, E2..) and then locking the output bitstream buffer(O1, O2..) for reading the encoded data. The work queued to the secondary thread by the client is in the following order

(I1, O1, E1)  
 (I2, O2, E2)  
 (I3, O3, E3)

Note they are in the same order in which client calls [::NvEncEncodePicture\(\)](#) API in \p step a).

e) [NvEncodeAPI](#) interface will do the re-ordering such that Encoder HW will receive the following encode commands:

(NvI1, O1, E1) ---P1 Frame  
 (NvI3, O2, E2) ---P3 Frame  
 (NvI2, O3, E3) ---B2 frame

f) After the encoding operations are completed, the events will be signalled by [NvEncodeAPI](#) interface in the following order :

(O1, E1) ---P1 Frame ,output bitstream copied to O1 and event E1 signalled.  
 (O2, E2) ---P3 Frame ,output bitstream copied to O2 and event E2 signalled.  
 (O3, E3) ---B2 Frame ,output bitstream copied to O3 and event E3 signalled.

g) The client must lock the bitstream data using [::NvEncLockBitstream\(\)](#) API in the order O1,O2,O3 to read the encoded data, after waiting for the events to be signalled in the same order i.e E1, E2 and E3. The output processing is done in the secondary thread in the following order:  
 Waits on E1, copies encoded bitstream from O1

Waits on E2, copies encoded bitstream from O2  
 Waits on E3, copies encoded bitstream from O3

- Note the client will receive the events signalling and output buffer in the same order in which they have submitted for encoding.
- Note the LockBitstream will have picture type field which will notify the output picture type to the clients.
- Note the input, output buffer and the output completion event are free to be reused once NvEncodeAPI interfaced has signalled the event and the client has copied the data from the output buffer.

### Synchronous Encoding

The client can enable synchronous mode of encoding by setting `NV_ENC_INITIALIZE_PARAMS::enableEncodeAsync` to 0 in `NvEncInitializeEncoder()` API. The NvEncodeAPI interface may return `NV_ENC_ERR_NEED_MORE_INPUT` error code for some `NvEncEncodePicture()` API calls when `NV_ENC_INITIALIZE_PARAMS::enablePTD` is set to 1, but the client must not treat it as a fatal error. The NvEncodeAPI interface might not be able to submit an input picture buffer for encoding immediately due to re-ordering for B frames. The NvEncodeAPI interface cannot submit the input picture which is decided to be encoded as B frame as it waits for backward reference from temporally subsequent frames. This input picture is buffered internally and waits for more input picture to arrive. The client must not call `NvEncLockBitstream()` API on the output buffers whose `NvEncEncodePicture()` API returns `NV_ENC_ERR_NEED_MORE_INPUT`. The client must wait for the NvEncodeAPI interface to return `NV_ENC_SUCCESS` before locking the output bitstreams to read the encoded bitstream data. The following example explains the scenario with synchronous encoding with 2 B frames.

The below example shows how synchronous encoding works in case of 1 B frames

-----  
 Suppose the client allocated 4 input buffers(I1,I2..), 4 output buffers(O1,O2..) and 4 completion events(E1, E2, ...). The NvEncodeAPI interface will need to keep a copy of the input buffers for re-ordering and it allocates following internal buffers (NvI1, NvI2...). These internal buffers are managed by NvEncodeAPI and the client is not responsible for the allocating or freeing the memory of the internal buffers.

The client calls `::NvEncEncodePicture()` API with input buffer I1 and output buffer O1. The NvEncodeAPI decides to encode I1 as P frame and submits it to encoder HW and returns `::NV_ENC_SUCCESS`. The client can now read the encoded data by locking the output O1 by calling `NvEncLockBitstream` API.

The client calls `::NvEncEncodePicture()` API with input buffer I2 and output buffer O2. The NvEncodeAPI decides to encode I2 as B frame and buffers I2 by copying it to internal buffer and returns `::NV_ENC_ERR_NEED_MORE_INPUT`. The error is not fatal and it notifies client that it cannot read the encoded data by locking the output O2 by calling `::NvEncLockBitstream()` API without submitting more work to the NvEncodeAPI interface.

The client calls `::NvEncEncodePicture()` with input buffer I3 and output buffer O3. The NvEncodeAPI decides to encode I3 as P frame and it first submits I3 for encoding which will be used as backward reference frame for I2. The NvEncodeAPI then submits I2 for encoding and returns `::NV_ENC_SUCCESS`. Both the submission are part of the same `::NvEncEncodePicture()` function call. The client can now read the encoded data for both the frames by locking the output O2 followed by O3 ,by calling `::NvEncLockBitstream()` API.

The client must always lock the output in the same order in which it has submitted to receive the encoded bitstream in correct encoding order.

### Parameters:

- ← **encoder** Pointer to the NvEncodeAPI interface.
- ↔ **encodePicParams** Pointer to the `_NV_ENC_PIC_PARAMS` structure.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_BUSY](#)  
[NV\\_ENC\\_ERR\\_NEED\\_MORE\\_INPUT](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.18 NVENCSTATUS NVENCAPI NvEncLockBitstream (void \* *encoder*, NV\_ENC\_LOCK\_BITSTREAM \* *lockBitstreamBufferParams*)

This function is used to lock the bitstream buffer to read the encoded data. The client can only access the encoded data by calling this function. The pointer to client accessible encoded data is returned in the [NV\\_ENC\\_LOCK\\_BITSTREAM::bitstreamBufferPtr](#) field. The size of the encoded data in the output buffer is returned in the [NV\\_ENC\\_LOCK\\_BITSTREAM::bitstreamSizeInBytes](#). The NvEncodeAPI interface also returns the output picture type and picture structure of the encoded frame in [NV\\_ENC\\_LOCK\\_BITSTREAM::pictureType](#) and [NV\\_ENC\\_LOCK\\_BITSTREAM::pictureStruct](#) fields respectively. If the client has set [NV\\_ENC\\_LOCK\\_BITSTREAM::doNotWait](#) to 1, the function might return [NV\\_ENC\\_ERR\\_LOCK\\_BUSY](#) if client is operating in synchronous mode. This is not a fatal failure if [NV\\_ENC\\_LOCK\\_BITSTREAM::doNotWait](#) is set to 1. In the above case the client can retry the function after few milliseconds.

**Parameters:**

← *encoder* Pointer to the NvEncodeAPI interface.  
 ↔ *lockBitstreamBufferParams* Pointer to the [\\_NV\\_ENC\\_LOCK\\_BITSTREAM](#) structure.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_LOCK\\_BUSY](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.19 NVENCSTATUS NVENCAPI NvEncUnlockBitstream (void \* *encoder*, NV\_ENC\_OUTPUT\_PTR *bitstreamBuffer*)

This function is used to unlock the output bitstream buffer after the client has read the encoded data from output buffer. The client must call this function to unlock the output buffer which it has previously locked using [NvEncLockBitstream\(\)](#) function. Using a locked bitstream buffer in [NvEncEncodePicture\(\)](#) API will cause the function to fail.



**Parameters:**

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ↔ *bitstreamBuffer* bitstream buffer pointer being unlocked

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.20 NVENCSTATUS NVENCAPI NvEncLockInputBuffer (void \* *encoder*, NV\_ENC\_LOCK\_INPUT\_BUFFER \* *lockInputBufferParams*)

This function is used to lock the input buffer to load the uncompressed YUV pixel data into input buffer memory. The client must pass the NV\_ENC\_INPUT\_PTR it had previously allocated using [NvEncCreateInputBuffer\(\)](#) in the NV\_ENC\_LOCK\_INPUT\_BUFFER::inputBuffer field. The NvEncodeAPI interface returns pointer to client accessible input buffer memory in NV\_ENC\_LOCK\_INPUT\_BUFFER::bufferDataPtr field.

**Parameters:**

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ↔ *lockInputBufferParams* Pointer to the \_NV\_ENC\_LOCK\_INPUT\_BUFFER structure

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_LOCK\\_BUSY](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.21 NVENCSTATUS NVENCAPI NvEncUnlockInputBuffer (void \* *encoder*, NV\_ENC\_INPUT\_PTR *inputBuffer*)

This function is used to unlock the input buffer memory previously locked for uploading YUV pixel data. The input buffer must be unlocked before being used again for encoding, otherwise NvEncodeAPI will fail the [NvEncEncodePicture\(\)](#)

**Parameters:**

- ← *encoder* Pointer to the NvEncodeAPI interface.

← *inputBuffer* Pointer to the input buffer that is being unlocked.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.22 NVENCSTATUS NVENCAPI NvEncGetEncodeStats (void \* *encoder*, NV\_ENC\_STAT \* *encodeStats*)

This function is used to retrieve the encoding statistics. This API is not supported when encode device type is CUDA.

**Parameters:**

← *encoder* Pointer to the NvEncodeAPI interface.  
 ↔ *encodeStats* Pointer to the `_NV_ENC_STAT` structure.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.23 NVENCSTATUS NVENCAPI NvEncGetSequenceParams (void \* *encoder*, NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD \* *sequenceParamPayload*)

This function can be used to retrieve the sequence and picture header out of band. The client must call this function only after the encoder has been initialized using [NvEncInitializeEncoder\(\)](#) function. The client must allocate the memory where the NvEncodeAPI interface can copy the bitstream header and pass the pointer to the memory in `NV_ENC_SEQUENCE_PARAM_PAYLOAD::spsppsBuffer`. The size of buffer is passed in the field `NV_ENC_SEQUENCE_PARAM_PAYLOAD::inBufferSize`. The NvEncodeAPI interface will copy the bitstream header payload and returns the actual size of the bitstream header in the field `NV_ENC_SEQUENCE_PARAM_PAYLOAD::outSPSPSPayloadSize`. The client must call [NvEncGetSequenceParams\(\)](#) function from the same thread which is being used to call [NvEncEncodePicture\(\)](#) function.

**Parameters:**

← *encoder* Pointer to the NvEncodeAPI interface.

↔ *sequenceParamPayload* Pointer to the `_NV_ENC_SEQUENCE_PARAM_PAYLOAD` structure.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

**4.2.1.24 NVENCSTATUS NVENCAPI NvEncRegisterAsyncEvent (void \* *encoder*, NV\_ENC\_EVENT\_PARAMS \* *eventParams*)**

This function is used to register the completion event with NvEncodeAPI interface. The event is required when the client has configured the encoder to work in asynchronous mode. In this mode the client needs to send a completion event with every output buffer. The NvEncodeAPI interface will signal the completion of the encoding process using this event. Only after the event is signalled the client can get the encoded data using [NvEncLockBitstream\(\)](#) function.

**Parameters:**

← *encoder* Pointer to the NvEncodeAPI interface.  
 ← *eventParams* Pointer to the `_NV_ENC_EVENT_PARAMS` structure.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

**4.2.1.25 NVENCSTATUS NVENCAPI NvEncUnregisterAsyncEvent (void \* *encoder*, NV\_ENC\_EVENT\_PARAMS \* *eventParams*)**

This function is used to unregister completion event which has been previously registered using [NvEncRegisterAsyncEvent\(\)](#) function. The client must unregister all events before destroying the encoder using [NvEncDestroyEncoder\(\)](#) function.

**Parameters:**

← *encoder* Pointer to the NvEncodeAPI interface.  
 ← *eventParams* Pointer to the `_NV_ENC_EVENT_PARAMS` structure.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.26 NVENCSTATUS NVENCAPI [NvEncMapInputResource](#) (void \* *encoder*, NV\_ENC\_MAP\_INPUT\_RESOURCE \* *mapInputResParams*)

Maps an externally allocated input resource [using and returns a NV\_ENC\_INPUT\_PTR which can be used for encoding in the [NvEncEncodePicture\(\)](#) function. The mapped resource is returned in the field NV\_ENC\_MAP\_INPUT\_RESOURCE::outputResourcePtr. The NvEncodeAPI interface also returns the buffer format of the mapped resource in the field NV\_ENC\_MAP\_INPUT\_RESOURCE::outbufferFmt. This function provides synchronization guarantee that any graphics or compute work submitted on the input buffer is completed before the buffer is used for encoding. The client should not access any input buffer while they are mapped by the encoder.

**Parameters:**

← *encoder* Pointer to the NvEncodeAPI interface.  
 ↔ *mapInputResParams* Pointer to the NV\_ENC\_MAP\_INPUT\_RESOURCE structure.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_RESOURCE\\_NOT\\_REGISTERED](#)  
[NV\\_ENC\\_ERR\\_MAP\\_FAILED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.27 NVENCSTATUS NVENCAPI [NvEncUnmapInputResource](#) (void \* *encoder*, NV\_ENC\_INPUT\_PTR *mappedInputBuffer*)

UnMaps an input buffer which was previously mapped using [NvEncMapInputResource\(\)](#) API. The mapping created using [NvEncMapInputResource\(\)](#) should be invalidated using this API before the external resource is destroyed by the client. The client must unmap the buffer after [NvEncLockBitstream\(\)](#) API returns successfully for encode work submitted using the mapped input buffer.

**Parameters:**

← *encoder* Pointer to the NvEncodeAPI interface.

← *mappedInputBuffer* Pointer to the NV\_ENC\_INPUT\_PTR

**Returns:**

NV\_ENC\_SUCCESS  
 NV\_ENC\_ERR\_INVALID\_PTR  
 NV\_ENC\_ERR\_INVALID\_ENCODERDEVICE  
 NV\_ENC\_ERR\_DEVICE\_NOT\_EXIST  
 NV\_ENC\_ERR\_UNSUPPORTED\_PARAM  
 NV\_ENC\_ERR\_OUT\_OF\_MEMORY  
 NV\_ENC\_ERR\_INVALID\_VERSION  
 NV\_ENC\_ERR\_INVALID\_PARAM  
 NV\_ENC\_ERR\_ENCODER\_NOT\_INITIALIZED  
 NV\_ENC\_ERR\_RESOURCE\_NOT\_REGISTERED  
 NV\_ENC\_ERR\_RESOURCE\_NOT\_MAPPED  
 NV\_ENC\_ERR\_GENERIC

#### 4.2.1.28 NVENCSTATUS NVENCAPI NvEncDestroyEncoder (void \* *encoder*)

Destroys the encoder session previously created using [NvEncOpenEncodeSession\(\)](#) function. The client must flush the encoder before freeing any resources. In order to flush the encoder the client must pass a NULL encode picture packet and either wait for the [NvEncEncodePicture\(\)](#) function to return in synchronous mode or wait for the flush event to be signaled by the encoder in asynchronous mode. The client must free all the input and output resources created using the NvEncodeAPI interface before destroying the encoder. If the client is operating in asynchronous mode, it must also unregister the completion events previously registered.

**Parameters:**

← *encoder* Pointer to the NvEncodeAPI interface.

**Returns:**

NV\_ENC\_SUCCESS  
 NV\_ENC\_ERR\_INVALID\_PTR  
 NV\_ENC\_ERR\_INVALID\_ENCODERDEVICE  
 NV\_ENC\_ERR\_DEVICE\_NOT\_EXIST  
 NV\_ENC\_ERR\_UNSUPPORTED\_PARAM  
 NV\_ENC\_ERR\_OUT\_OF\_MEMORY  
 NV\_ENC\_ERR\_INVALID\_PARAM  
 NV\_ENC\_ERR\_GENERIC

#### 4.2.1.29 NVENCSTATUS NVENCAPI NvEncInvalidateRefFrames (void \* *encoder*, uint64\_t *invalidRefFrameTimeStamp*)

Invalidates reference frame based on the time stamp provided by the client. The encoder marks any reference frames or any frames which have been reconstructed using the corrupt frame as invalid for motion estimation and uses older reference frames for motion estimation. The encoded forces the current frame to be encoded as an intra frame if no reference frames are left after invalidation process. This is useful for low latency application for error resiliency. The client is recommended to set [NV\\_ENC\\_CONFIG\\_H264::maxNumRefFrames](#) to a large value so that encoder can keep a backup of older reference frames in the DPB and can use them for motion estimation when the newer reference frames have been invalidated. This API can be called multiple times.

**Parameters:**

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *invalidRefFrameTimeStamp* Timestamp of the invalid reference frames which needs to be invalidated.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.30 NVENCSTATUS NVENCAPI NvEncOpenEncodeSessionEx (NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_PARAMS \* *openSessionExParams*, void \*\* *encoder*)

Opens an encoding session and returns a pointer to the encoder interface in the *\*\*encoder* parameter. The client should start encoding process by calling this API first. The client must pass a pointer to IDirect3DDevice9 device or CUDA context in the *\*device* parameter. For the OpenGL interface, *device* must be NULL. An OpenGL context must be current when calling all NvEncodeAPI functions. If the creation of encoder session fails, the client must call [NvEncDestroyEncoder](#) API before exiting.

**Parameters:**

- ← *openSessionExParams* Pointer to a [NV\\_ENC\\_OPEN\\_ENCODE\\_SESSION\\_EX\\_PARAMS](#) structure.
- *encoder* Encode Session pointer to the NvEncodeAPI interface.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_NO\\_ENCODE\\_DEVICE](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_DEVICE](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_DEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

#### 4.2.1.31 NVENCSTATUS NVENCAPI NvEncRegisterResource (void \* *encoder*, NV\_ENC\_REGISTER\_RESOURCE \* *registerResParams*)

Registers a resource with the Nvidia Video Encoder Interface for book keeping. The client is expected to pass the registered resource handle as well, while calling [NvEncMapInputResource](#) API.

**Parameters:**

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *registerResParams* Pointer to a [\\_NV\\_ENC\\_REGISTER\\_RESOURCE](#) structure

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_RESOURCE\\_REGISTER\\_FAILED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)  
[NV\\_ENC\\_ERR\\_UNIMPLEMENTED](#)

**4.2.1.32 NVENCSTATUS NVENCAPI NvEncUnregisterResource (void \* *encoder*,  
 NV\_ENC\_REGISTERED\_PTR *registeredResource*)**

Unregisters a resource previously registered with the Nvidia Video Encoder Interface. The client is expected to unregister any resource that it has registered with the Nvidia Video Encoder Interface before destroying the resource.

**Parameters:**

← *encoder* Pointer to the NvEncodeAPI interface.  
 ← *registeredResource* The registered resource pointer that was returned in [NvEncRegisterResource](#).

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_ENCODER\\_NOT\\_INITIALIZED](#)  
[NV\\_ENC\\_ERR\\_RESOURCE\\_NOT\\_REGISTERED](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)  
[NV\\_ENC\\_ERR\\_UNIMPLEMENTED](#)

**4.2.1.33 NVENCSTATUS NVENCAPI NvEncReconfigureEncoder (void \* *encoder*,  
 NV\_ENC\_RECONFIGURE\_PARAMS \* *reInitEncodeParams*)**

Reconfigure an existing encoding session. The client should call this API to change/reconfigure the parameter passed during NvEncInitializeEncoder API call. Currently Reconfiguration of following are not supported. Change in GOP structure. Change in sync-Async mode. Change in MaxWidth & MaxHeight. Change in PTDmode.

Resolution change is possible only if maxEncodeWidth & maxEncodeHeight of [NV\\_ENC\\_INITIALIZE\\_PARAMS](#) is set while creating encoder session.

**Parameters:**

← *encoder* Pointer to the NvEncodeAPI interface.

← *reInitEncodeParams* Pointer to a [NV\\_ENC\\_RECONFIGURE\\_PARAMS](#) structure.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_NO\\_ENCODE\\_DEVICE](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_DEVICE](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_DEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

**4.2.1.34 NVENCSTATUS NVENCAPI NvEncCreateMVBuffer (void \* *encoder*,  
 NV\_ENC\_CREATE\_MV\_BUFFER \* *createMVBufferParams*)**

This function is used to allocate an output MV buffer. The size of the mvBuffer is dependent on the frame height and width of the last [NvEncCreateInputBuffer\(\)](#) call. The NV\_ENC\_OUTPUT\_PTR returned by the NvEncodeAPI interface in the [NV\\_ENC\\_CREATE\\_MV\\_BUFFER::mvBuffer](#) field should be used in [NvEncRunMotionEstimationOnly\(\)](#) API. Client must lock [NV\\_ENC\\_CREATE\\_MV\\_BUFFER::mvBuffer](#) using [NvEncLockBitstream\(\)](#) API to get the motion vector data.

**Parameters:**

← *encoder* Pointer to the NvEncodeAPI interface.  
 ↔ *createMVBufferParams* Pointer to the [NV\\_ENC\\_CREATE\\_MV\\_BUFFER](#) structure.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_ENCODERDEVICE](#)  
[NV\\_ENC\\_ERR\\_DEVICE\\_NOT\\_EXIST](#)  
[NV\\_ENC\\_ERR\\_UNSUPPORTED\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_OUT\\_OF\\_MEMORY](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_PARAM](#)  
[NV\\_ENC\\_ERR\\_INVALID\\_VERSION](#)  
[NV\\_ENC\\_ERR\\_GENERIC](#)

**4.2.1.35 NVENCSTATUS NVENCAPI NvEncDestroyMVBuffer (void \* *encoder*, NV\_ENC\_OUTPUT\_PTR  
*mvBuffer*)**

This function is used to release the output MV buffer allocated using the [NvEncCreateMVBuffer\(\)](#) function. The client must release the output mvBuffer using this function before destroying the encoder session.

**Parameters:**

← *encoder* Pointer to the NvEncodeAPI interface.  
 ← *mvBuffer* Pointer to the mvBuffer being released.

**Returns:**

[NV\\_ENC\\_SUCCESS](#)



NV\_ENC\_ERR\_INVALID\_PTR  
 NV\_ENC\_ERR\_INVALID\_ENCODERDEVICE  
 NV\_ENC\_ERR\_DEVICE\_NOT\_EXIST  
 NV\_ENC\_ERR\_UNSUPPORTED\_PARAM  
 NV\_ENC\_ERR\_OUT\_OF\_MEMORY  
 NV\_ENC\_ERR\_INVALID\_PARAM  
 NV\_ENC\_ERR\_INVALID\_VERSION  
 NV\_ENC\_ERR\_ENCODER\_NOT\_INITIALIZED  
 NV\_ENC\_ERR\_GENERIC

#### 4.2.1.36 NVENCSTATUS NVENCAPI NvEncRunMotionEstimationOnly (void \* *encoder*, NV\_ENC\_MEONLY\_PARAMS \* *meOnlyParams*)

This function is used to submit the input frame and reference frame for motion estimation. The ME parameters are passed using *\*meOnlyParams* which is a pointer to `_NV_ENC_MEONLY_PARAMS` structure. Client must lock `NV_ENC_CREATE_MV_BUFFER::mvBuffer` using `NvEncLockBitstream()` API to get the motion vector data. to get motion vector data.

##### Parameters:

- ← *encoder* Pointer to the NvEncodeAPI interface.
- ← *meOnlyParams* Pointer to the `_NV_ENC_MEONLY_PARAMS` structure.

##### Returns:

NV\_ENC\_SUCCESS  
 NV\_ENC\_ERR\_INVALID\_PTR  
 NV\_ENC\_ERR\_INVALID\_ENCODERDEVICE  
 NV\_ENC\_ERR\_DEVICE\_NOT\_EXIST  
 NV\_ENC\_ERR\_UNSUPPORTED\_PARAM  
 NV\_ENC\_ERR\_OUT\_OF\_MEMORY  
 NV\_ENC\_ERR\_INVALID\_PARAM  
 NV\_ENC\_ERR\_INVALID\_VERSION  
 NV\_ENC\_ERR\_NEED\_MORE\_INPUT  
 NV\_ENC\_ERR\_ENCODER\_NOT\_INITIALIZED  
 NV\_ENC\_ERR\_GENERIC

#### 4.2.1.37 NVENCSTATUS NVENCAPI NvEncodeAPIGetMaxSupportedVersion (uint32\_t \* *version*)

This function can be used by clients to determine if the driver supports the NvEncodeAPI header the application was compiled with.

##### Parameters:

- *version* Pointer to the requested value. The 4 least significant bits in the returned indicate the minor version and the rest of the bits indicate the major version of the largest supported version.

##### Returns:

NV\_ENC\_SUCCESS  
 NV\_ENC\_ERR\_INVALID\_PTR

#### 4.2.1.38 NVENCSTATUS NVENCAPI NvEncodeAPICreateInstance (NV\_ENCODE\_API\_FUNCTION\_LIST \**functionList*)

Entry Point to the NvEncodeAPI interface.

Creates an instance of the NvEncodeAPI interface, and populates the pFunctionList with function pointers to the API routines implemented by the NvEncodeAPI interface.

**Parameters:**

→ *functionList*

**Returns:**

[NV\\_ENC\\_SUCCESS](#) [NV\\_ENC\\_ERR\\_INVALID\\_PTR](#)

# Chapter 5

## Data Structure Documentation

### 5.1 GUID Struct Reference

```
#include <nvEncodeAPI.h>
```

#### Data Fields

- uint32\_t [Data1](#)
- uint16\_t [Data2](#)
- uint16\_t [Data3](#)
- uint8\_t [Data4](#) [8]

#### 5.1.1 Detailed Description

[GUID](#) Abstracts the [GUID](#) structure for non-windows platforms.

#### 5.1.2 Field Documentation

##### 5.1.2.1 uint32\_t GUID::Data1

[in]: Specifies the first 8 hexadecimal digits of the [GUID](#).

##### 5.1.2.2 uint16\_t GUID::Data2

[in]: Specifies the first group of 4 hexadecimal digits.

##### 5.1.2.3 uint16\_t GUID::Data3

[in]: Specifies the second group of 4 hexadecimal digits.

##### 5.1.2.4 uint8\_t GUID::Data4[8]

[in]: Array of 8 bytes. The first 2 bytes contain the third group of 4 hexadecimal digits. The remaining 6 bytes contain the final 12 hexadecimal digits.

## 5.2 NV\_ENC\_CAPS\_PARAM Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- `uint32_t` [version](#)
- `NV_ENC_CAPS` [capsToQuery](#)
- `uint32_t` [reserved](#) [62]

### 5.2.1 Detailed Description

Input struct for querying Encoding capabilities.

### 5.2.2 Field Documentation

#### 5.2.2.1 `uint32_t NV_ENC_CAPS_PARAM::version`

[in]: Struct version. Must be set to [NV\\_ENC\\_CAPS\\_PARAM\\_VER](#)

#### 5.2.2.2 `NV_ENC_CAPS NV_ENC_CAPS_PARAM::capsToQuery`

[in]: Specifies the encode capability to be queried. Client should pass a member for [NV\\_ENC\\_CAPS](#) enum.

#### 5.2.2.3 `uint32_t NV_ENC_CAPS_PARAM::reserved[62]`

[in]: Reserved and must be set to 0

## 5.3 NV\_ENC\_CODEC\_CONFIG Union Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- [NV\\_ENC\\_CONFIG\\_H264](#) `h264Config`
- [NV\\_ENC\\_CONFIG\\_HEVC](#) `hevcConfig`
- [NV\\_ENC\\_CONFIG\\_H264\\_MEONLY](#) `h264MeOnlyConfig`
- [NV\\_ENC\\_CONFIG\\_HEVC\\_MEONLY](#) `hevcMeOnlyConfig`
- `uint32_t` [reserved](#) [320]

### 5.3.1 Detailed Description

`_NV_ENC_CODEC_CONFIG` Codec-specific encoder configuration parameters to be set during initialization.

### 5.3.2 Field Documentation

#### 5.3.2.1 NV\_ENC\_CONFIG\_H264 NV\_ENC\_CODEC\_CONFIG::h264Config

[in]: Specifies the H.264-specific encoder configuration.

#### 5.3.2.2 NV\_ENC\_CONFIG\_HEVC NV\_ENC\_CODEC\_CONFIG::hevcConfig

[in]: Specifies the HEVC-specific encoder configuration.

#### 5.3.2.3 NV\_ENC\_CONFIG\_H264\_MEONLY NV\_ENC\_CODEC\_CONFIG::h264MeOnlyConfig

[in]: Specifies the H.264-specific ME only encoder configuration.

#### 5.3.2.4 NV\_ENC\_CONFIG\_HEVC\_MEONLY NV\_ENC\_CODEC\_CONFIG::hevcMeOnlyConfig

[in]: Specifies the HEVC-specific ME only encoder configuration.

#### 5.3.2.5 `uint32_t` NV\_ENC\_CODEC\_CONFIG::reserved[320]

[in]: Reserved and must be set to 0

## 5.4 NV\_ENC\_CODEC\_PIC\_PARAMS Union Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- [NV\\_ENC\\_PIC\\_PARAMS\\_H264](#) `h264PicParams`
- [NV\\_ENC\\_PIC\\_PARAMS\\_HEVC](#) `hevcPicParams`
- `uint32_t` [reserved](#) [256]

### 5.4.1 Detailed Description

Codec specific per-picture encoding parameters.

### 5.4.2 Field Documentation

#### 5.4.2.1 NV\_ENC\_PIC\_PARAMS\_H264 NV\_ENC\_CODEC\_PIC\_PARAMS::h264PicParams

[in]: H264 encode picture params.

#### 5.4.2.2 NV\_ENC\_PIC\_PARAMS\_HEVC NV\_ENC\_CODEC\_PIC\_PARAMS::hevcPicParams

[in]: HEVC encode picture params.

#### 5.4.2.3 `uint32_t` NV\_ENC\_CODEC\_PIC\_PARAMS::reserved[256]

[in]: Reserved and must be set to 0.

## 5.5 NV\_ENC\_CONFIG Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- [uint32\\_t version](#)
- [GUID profileGUID](#)
- [uint32\\_t gopLength](#)
- [int32\\_t frameIntervalP](#)
- [uint32\\_t monoChromeEncoding](#)
- [NV\\_ENC\\_PARAMS\\_FRAME\\_FIELD\\_MODE frameFieldMode](#)
- [NV\\_ENC\\_MV\\_PRECISION mvPrecision](#)
- [NV\\_ENC\\_RC\\_PARAMS rcParams](#)
- [NV\\_ENC\\_CODEC\\_CONFIG encodeCodecConfig](#)
- [uint32\\_t reserved](#) [278]
- [void \\* reserved2](#) [64]

### 5.5.1 Detailed Description

`_NV_ENC_CONFIG` Encoder configuration parameters to be set during initialization.

### 5.5.2 Field Documentation

#### 5.5.2.1 `uint32_t NV_ENC_CONFIG::version`

[in]: Struct version. Must be set to [NV\\_ENC\\_CONFIG\\_VER](#).

#### 5.5.2.2 `GUID NV_ENC_CONFIG::profileGUID`

[in]: Specifies the codec profile guid. If client specifies `NV_ENC_CODEC_PROFILE_AUTOSELECT_GUID` the `NvEncodeAPI` interface will select the appropriate codec profile.

#### 5.5.2.3 `uint32_t NV_ENC_CONFIG::gopLength`

[in]: Specifies the number of pictures in one GOP. Low latency application client can set `goplength` to `NVENC_INFINITE_GOPLength` so that keyframes are not inserted automatically.

#### 5.5.2.4 `int32_t NV_ENC_CONFIG::frameIntervalP`

[in]: Specifies the GOP pattern as follows: `frameIntervalP` = 0: I, 1: IPP, 2: IBP, 3: IBBP If `goplength` is set to `NVENC_INFINITE_GOPLength` `frameIntervalP` should be set to 1.

#### 5.5.2.5 `uint32_t NV_ENC_CONFIG::monoChromeEncoding`

[in]: Set this to 1 to enable monochrome encoding for this session.

**5.5.2.6 NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE NV\_ENC\_CONFIG::frameFieldMode**

[in]: Specifies the frame/field mode. Check support for field encoding using [NV\\_ENC\\_CAPS\\_SUPPORT\\_FIELD\\_ENCODING](#) caps. Using a frameFieldMode other than NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE\_FRAME for RGB input is not supported.

**5.5.2.7 NV\_ENC\_MV\_PRECISION NV\_ENC\_CONFIG::mvPrecision**

[in]: Specifies the desired motion vector prediction precision.

**5.5.2.8 NV\_ENC\_RC\_PARAMS NV\_ENC\_CONFIG::rcParams**

[in]: Specifies the rate control parameters for the current encoding session.

**5.5.2.9 NV\_ENC\_CODEC\_CONFIG NV\_ENC\_CONFIG::encodeCodecConfig**

[in]: Specifies the codec specific config parameters through this union.

**5.5.2.10 uint32\_t NV\_ENC\_CONFIG::reserved[278]**

[in]: Reserved and must be set to 0

**5.5.2.11 void\* NV\_ENC\_CONFIG::reserved2[64]**

[in]: Reserved and must be set to NULL



## 5.6 NV\_ENC\_CONFIG\_H264 Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [enableTemporalSVC](#):1
- uint32\_t [enableStereoMVC](#):1
- uint32\_t [hierarchicalPFrames](#):1
- uint32\_t [hierarchicalBFrames](#):1
- uint32\_t [outputBufferingPeriodSEI](#):1
- uint32\_t [outputPictureTimingSEI](#):1
- uint32\_t [outputAUD](#):1
- uint32\_t [disableSPSPPS](#):1
- uint32\_t [outputFramePackingSEI](#):1
- uint32\_t [outputRecoveryPointSEI](#):1
- uint32\_t [enableIntraRefresh](#):1
- uint32\_t [enableConstrainedEncoding](#):1
- uint32\_t [repeatSPSPPS](#):1
- uint32\_t [enableVFR](#):1
- uint32\_t [enableLTR](#):1
- uint32\_t [qpPrimeYZeroTransformBypassFlag](#):1
- uint32\_t [useConstrainedIntraPred](#):1
- uint32\_t [reservedBitFields](#):15
- uint32\_t [level](#)
- uint32\_t [idrPeriod](#)
- uint32\_t [separateColourPlaneFlag](#)
- uint32\_t [disableDeblockingFilterIDC](#)
- uint32\_t [numTemporalLayers](#)
- uint32\_t [spsId](#)
- uint32\_t [ppsId](#)
- [NV\\_ENC\\_H264\\_ADAPTIVE\\_TRANSFORM\\_MODE](#) adaptiveTransformMode
- [NV\\_ENC\\_H264\\_FMO\\_MODE](#) fmoMode
- [NV\\_ENC\\_H264\\_BDIRECT\\_MODE](#) bdirectMode
- [NV\\_ENC\\_H264\\_ENTROPY\\_CODING\\_MODE](#) entropyCodingMode
- [NV\\_ENC\\_STEREO\\_PACKING\\_MODE](#) stereoMode
- uint32\_t [intraRefreshPeriod](#)
- uint32\_t [intraRefreshCnt](#)
- uint32\_t [maxNumRefFrames](#)
- uint32\_t [sliceMode](#)
- uint32\_t [sliceModeData](#)
- [NV\\_ENC\\_CONFIG\\_H264\\_VUI\\_PARAMETERS](#) h264VUIParameters
- uint32\_t [ltrNumFrames](#)
- uint32\_t [ltrTrustMode](#)
- uint32\_t [chromaFormatIDC](#)
- uint32\_t [maxTemporalLayers](#)
- uint32\_t [reserved1](#) [270]
- void \* [reserved2](#) [64]

## 5.6.1 Detailed Description

\_NV\_ENC\_CONFIG\_H264 H264 encoder configuration parameters

## 5.6.2 Field Documentation

### 5.6.2.1 uint32\_t NV\_ENC\_CONFIG\_H264::enableTemporalSVC

[in]: Set to 1 to enable SVC temporal

### 5.6.2.2 uint32\_t NV\_ENC\_CONFIG\_H264::enableStereoMVC

[in]: Set to 1 to enable stereo MVC

### 5.6.2.3 uint32\_t NV\_ENC\_CONFIG\_H264::hierarchicalPFrames

[in]: Set to 1 to enable hierarchical PFrames

### 5.6.2.4 uint32\_t NV\_ENC\_CONFIG\_H264::hierarchicalBFrames

[in]: Set to 1 to enable hierarchical BFrames

### 5.6.2.5 uint32\_t NV\_ENC\_CONFIG\_H264::outputBufferingPeriodSEI

[in]: Set to 1 to write SEI buffering period syntax in the bitstream

### 5.6.2.6 uint32\_t NV\_ENC\_CONFIG\_H264::outputPictureTimingSEI

[in]: Set to 1 to write SEI picture timing syntax in the bitstream. When set for following rateControlMode : NV\_ENC\_PARAMS\_RC\_CBR, NV\_ENC\_PARAMS\_RC\_CBR\_LOWDELAY\_HQ, NV\_ENC\_PARAMS\_RC\_CBR\_HQ, filler data is inserted if needed to achieve hrd bitrate

### 5.6.2.7 uint32\_t NV\_ENC\_CONFIG\_H264::outputAUD

[in]: Set to 1 to write access unit delimiter syntax in bitstream

### 5.6.2.8 uint32\_t NV\_ENC\_CONFIG\_H264::disableSPSPPS

[in]: Set to 1 to disable writing of Sequence and Picture parameter info in bitstream

### 5.6.2.9 uint32\_t NV\_ENC\_CONFIG\_H264::outputFramePackingSEI

[in]: Set to 1 to enable writing of frame packing arrangement SEI messages to bitstream

### 5.6.2.10 uint32\_t NV\_ENC\_CONFIG\_H264::outputRecoveryPointSEI

[in]: Set to 1 to enable writing of recovery point SEI message

**5.6.2.11 uint32\_t NV\_ENC\_CONFIG\_H264::enableIntraRefresh**

[in]: Set to 1 to enable gradual decoder refresh or intra refresh. If the GOP structure uses B frames this will be ignored

**5.6.2.12 uint32\_t NV\_ENC\_CONFIG\_H264::enableConstrainedEncoding**

[in]: Set this to 1 to enable constrainedFrame encoding where each slice in the constrained picture is independent of other slices. Check support for constrained encoding using [NV\\_ENC\\_CAPS\\_SUPPORT\\_CONSTRAINED\\_ENCODING](#) caps.

**5.6.2.13 uint32\_t NV\_ENC\_CONFIG\_H264::repeatSPSPS**

[in]: Set to 1 to enable writing of Sequence and Picture parameter for every IDR frame

**5.6.2.14 uint32\_t NV\_ENC\_CONFIG\_H264::enableVFR**

[in]: Set to 1 to enable variable frame rate.

**5.6.2.15 uint32\_t NV\_ENC\_CONFIG\_H264::enableLTR**

[in]: Set to 1 to enable LTR (Long Term Reference) frame support. LTR can be used in two modes: "LTR Trust" mode and "LTR Per Picture" mode. LTR Trust mode: In this mode, ltrNumFrames pictures after IDR are automatically marked as LTR. This mode is enabled by setting ltrTrustMode = 1. Use of LTR Trust mode is strongly discouraged as this mode may be deprecated in future. LTR Per Picture mode: In this mode, client can control whether the current picture should be marked as LTR. Enable this mode by setting ltrTrustMode = 0 and ltrMarkFrame = 1 for the picture to be marked as LTR. This is the preferred mode for using LTR. Note that LTRs are not supported if encoding session is configured with B-frames

**5.6.2.16 uint32\_t NV\_ENC\_CONFIG\_H264::qpPrimeYZeroTransformBypassFlag**

[in]: To enable lossless encode set this to 1, set QP to 0 and RC\_mode to NV\_ENC\_PARAMS\_RC\_CONSTQP and profile to HIGH\_444\_PREDICTIVE\_PROFILE. Check support for lossless encoding using [NV\\_ENC\\_CAPS\\_SUPPORT\\_LOSSLESS\\_ENCODE](#) caps.

**5.6.2.17 uint32\_t NV\_ENC\_CONFIG\_H264::useConstrainedIntraPred**

[in]: Set 1 to enable constrained intra prediction.

**5.6.2.18 uint32\_t NV\_ENC\_CONFIG\_H264::reservedBitFields**

[in]: Reserved bitfields and must be set to 0

**5.6.2.19 uint32\_t NV\_ENC\_CONFIG\_H264::level**

[in]: Specifies the encoding level. Client is recommended to set this to NV\_ENC\_LEVEL\_AUTOSELECT in order to enable the NvEncodeAPI interface to select the correct level.

**5.6.2.20 uint32\_t NV\_ENC\_CONFIG\_H264::idrPeriod**

[in]: Specifies the IDR interval. If not set, this is made equal to gopLength in NV\_ENC\_CONFIG. Low latency application client can set IDR interval to NVENC\_INFINITE\_GOPLength so that IDR frames are not inserted automatically.

**5.6.2.21 uint32\_t NV\_ENC\_CONFIG\_H264::separateColourPlaneFlag**

[in]: Set to 1 to enable 4:4:4 separate colour planes

**5.6.2.22 uint32\_t NV\_ENC\_CONFIG\_H264::disableDeblockingFilterIDC**

[in]: Specifies the deblocking filter mode. Permissible value range: [0,2]

**5.6.2.23 uint32\_t NV\_ENC\_CONFIG\_H264::numTemporalLayers**

[in]: Specifies max temporal layers to be used for hierarchical coding. Valid value range is [1, [NV\\_ENC\\_CAPS\\_NUM\\_MAX\\_TEMPORAL\\_LAYERS](#)]

**5.6.2.24 uint32\_t NV\_ENC\_CONFIG\_H264::spsId**

[in]: Specifies the SPS id of the sequence header

**5.6.2.25 uint32\_t NV\_ENC\_CONFIG\_H264::ppsId**

[in]: Specifies the PPS id of the picture header

**5.6.2.26 NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_MODE NV\_ENC\_CONFIG\_H264::adaptiveTransformMode**

[in]: Specifies the AdaptiveTransform Mode. Check support for AdaptiveTransform mode using [NV\\_ENC\\_CAPS\\_SUPPORT\\_ADAPTIVE\\_TRANSFORM](#) caps.

**5.6.2.27 NV\_ENC\_H264\_FMO\_MODE NV\_ENC\_CONFIG\_H264::fmoMode**

[in]: Specified the FMO Mode. Check support for FMO using [NV\\_ENC\\_CAPS\\_SUPPORT\\_FMO](#) caps.

**5.6.2.28 NV\_ENC\_H264\_BDIRECT\_MODE NV\_ENC\_CONFIG\_H264::bdirectMode**

[in]: Specifies the BDirect mode. Check support for BDirect mode using [NV\\_ENC\\_CAPS\\_SUPPORT\\_BDIRECT\\_MODE](#) caps.

**5.6.2.29 NV\_ENC\_H264\_ENTROPY\_CODING\_MODE NV\_ENC\_CONFIG\_H264::entropyCodingMode**

[in]: Specifies the entropy coding mode. Check support for CABAC mode using [NV\\_ENC\\_CAPS\\_SUPPORT\\_CABAC](#) caps.

**5.6.2.30 NV\_ENC\_STEREO\_PACKING\_MODE NV\_ENC\_CONFIG\_H264::stereoMode**

[in]: Specifies the stereo frame packing mode which is to be signalled in frame packing arrangement SEI

**5.6.2.31 uint32\_t NV\_ENC\_CONFIG\_H264::intraRefreshPeriod**

[in]: Specifies the interval between successive intra refresh if enableIntraRefresh is set. Requires enableIntraRefresh to be set. Will be disabled if [NV\\_ENC\\_CONFIG::gopLength](#) is not set to NVENC\_INFINITE\_GOPLength.

**5.6.2.32 uint32\_t NV\_ENC\_CONFIG\_H264::intraRefreshCnt**

[in]: Specifies the length of intra refresh in number of frames for periodic intra refresh. This value should be smaller than intraRefreshPeriod

**5.6.2.33 uint32\_t NV\_ENC\_CONFIG\_H264::maxNumRefFrames**

[in]: Specifies the DPB size used for encoding. Setting it to 0 will let driver use the default dpb size. The low latency application which wants to invalidate reference frame as an error resilience tool is recommended to use a large DPB size so that the encoder can keep old reference frames which can be used if recent frames are invalidated.

**5.6.2.34 uint32\_t NV\_ENC\_CONFIG\_H264::sliceMode**

[in]: This parameter in conjunction with sliceModeData specifies the way in which the picture is divided into slices sliceMode = 0 MB based slices, sliceMode = 1 Byte based slices, sliceMode = 2 MB row based slices, sliceMode = 3, numSlices in Picture When forceIntraRefreshWithFrameCnt is set it will have priority over sliceMode setting When sliceMode == 0 and sliceModeData == 0 whole picture will be coded with one slice

**5.6.2.35 uint32\_t NV\_ENC\_CONFIG\_H264::sliceModeData**

[in]: Specifies the parameter needed for sliceMode. For: sliceMode = 0, sliceModeData specifies # of MBs in each slice (except last slice) sliceMode = 1, sliceModeData specifies maximum # of bytes in each slice (except last slice) sliceMode = 2, sliceModeData specifies # of MB rows in each slice (except last slice) sliceMode = 3, sliceModeData specifies number of slices in the picture. Driver will divide picture into slices optimally

**5.6.2.36 NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS NV\_ENC\_CONFIG\_H264::h264VUIParameters**

[in]: Specifies the H264 video usability info parameters

**5.6.2.37 uint32\_t NV\_ENC\_CONFIG\_H264::ltrNumFrames**

[in]: Specifies the number of LTR frames. This parameter has different meaning in two LTR modes. In "LTR Trust" mode (ltrTrustMode = 1), encoder will mark the first ltrNumFrames base layer reference frames within each IDR interval as LTR. In "LTR Per Picture" mode (ltrTrustMode = 0 and ltrMarkFrame = 1), ltrNumFrames specifies maximum number of LTR frames in DPB.

**5.6.2.38 uint32\_t NV\_ENC\_CONFIG\_H264::ltrTrustMode**

[in]: Specifies the LTR operating mode. See comments near [NV\\_ENC\\_CONFIG\\_H264::enableLTR](#) for description of the two modes. Set to 1 to use "LTR Trust" mode of LTR operation. Clients are discouraged to use "LTR Trust" mode as this mode may be deprecated in future releases. Set to 0 when using "LTR Per Picture" mode of LTR operation.

**5.6.2.39 uint32\_t NV\_ENC\_CONFIG\_H264::chromaFormatIDC**

[in]: Specifies the chroma format. Should be set to 1 for yuv420 input, 3 for yuv444 input. Check support for YUV444 encoding using [NV\\_ENC\\_CAPS\\_SUPPORT\\_YUV444\\_ENCODE](#) caps.

**5.6.2.40 uint32\_t NV\_ENC\_CONFIG\_H264::maxTemporalLayers**

[in]: Specifies the max temporal layer used for hierarchical coding.

**5.6.2.41 uint32\_t NV\_ENC\_CONFIG\_H264::reserved1[270]**

[in]: Reserved and must be set to 0

**5.6.2.42 void\* NV\_ENC\_CONFIG\_H264::reserved2[64]**

[in]: Reserved and must be set to NULL

## 5.7 NV\_ENC\_CONFIG\_H264\_MEONLY Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [disablePartition16x16](#):1
- uint32\_t [disablePartition8x16](#):1
- uint32\_t [disablePartition16x8](#):1
- uint32\_t [disablePartition8x8](#):1
- uint32\_t [disableIntraSearch](#):1
- uint32\_t [bStereoEnable](#):1
- uint32\_t [reserved](#):26
- uint32\_t [reserved1](#) [255]
- void \* [reserved2](#) [64]

### 5.7.1 Detailed Description

`_NV_ENC_CONFIG_H264_MEONLY` H264 encoder configuration parameters for ME only Mode

### 5.7.2 Field Documentation

#### 5.7.2.1 uint32\_t NV\_ENC\_CONFIG\_H264\_MEONLY::disablePartition16x16

[in]: Disable MotionEstimation on 16x16 blocks

#### 5.7.2.2 uint32\_t NV\_ENC\_CONFIG\_H264\_MEONLY::disablePartition8x16

[in]: Disable MotionEstimation on 8x16 blocks

#### 5.7.2.3 uint32\_t NV\_ENC\_CONFIG\_H264\_MEONLY::disablePartition16x8

[in]: Disable MotionEstimation on 16x8 blocks

#### 5.7.2.4 uint32\_t NV\_ENC\_CONFIG\_H264\_MEONLY::disablePartition8x8

[in]: Disable MotionEstimation on 8x8 blocks

#### 5.7.2.5 uint32\_t NV\_ENC\_CONFIG\_H264\_MEONLY::disableIntraSearch

[in]: Disable Intra search during MotionEstimation

#### 5.7.2.6 uint32\_t NV\_ENC\_CONFIG\_H264\_MEONLY::bStereoEnable

[in]: Enable Stereo Mode for Motion Estimation where each view is independently executed

**5.7.2.7 uint32\_t NV\_ENC\_CONFIG\_H264\_MEONLY::reserved**

[in]: Reserved and must be set to 0

**5.7.2.8 uint32\_t NV\_ENC\_CONFIG\_H264\_MEONLY::reserved1[255]**

[in]: Reserved and must be set to 0

**5.7.2.9 void\* NV\_ENC\_CONFIG\_H264\_MEONLY::reserved2[64]**

[in]: Reserved and must be set to NULL



## 5.8 NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [overscanInfoPresentFlag](#)
- uint32\_t [overscanInfo](#)
- uint32\_t [videoSignalTypePresentFlag](#)
- uint32\_t [videoFormat](#)
- uint32\_t [videoFullRangeFlag](#)
- uint32\_t [colourDescriptionPresentFlag](#)
- uint32\_t [colourPrimaries](#)
- uint32\_t [transferCharacteristics](#)
- uint32\_t [colourMatrix](#)
- uint32\_t [chromaSampleLocationFlag](#)
- uint32\_t [chromaSampleLocationTop](#)
- uint32\_t [chromaSampleLocationBot](#)
- uint32\_t [bitstreamRestrictionFlag](#)

### 5.8.1 Detailed Description

\_NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS H264 Video Usability Info parameters

### 5.8.2 Field Documentation

#### 5.8.2.1 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::overscanInfoPresentFlag

[in]: if set to 1 , it specifies that the overscanInfo is present

#### 5.8.2.2 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::overscanInfo

[in]: Specifies the overscan info(as defined in Annex E of the ITU-T Specification).

#### 5.8.2.3 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::videoSignalTypePresentFlag

[in]: If set to 1, it specifies that the videoFormat, videoFullRangeFlag and colourDescriptionPresentFlag are present.

#### 5.8.2.4 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::videoFormat

[in]: Specifies the source video format(as defined in Annex E of the ITU-T Specification).

#### 5.8.2.5 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::videoFullRangeFlag

[in]: Specifies the output range of the luma and chroma samples(as defined in Annex E of the ITU-T Specification).

**5.8.2.6 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::colourDescriptionPresentFlag**

[in]: If set to 1, it specifies that the colourPrimaries, transferCharacteristics and colourMatrix are present.

**5.8.2.7 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::colourPrimaries**

[in]: Specifies color primaries for converting to RGB(as defined in Annex E of the ITU-T Specification)

**5.8.2.8 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::transferCharacteristics**

[in]: Specifies the opto-electronic transfer characteristics to use (as defined in Annex E of the ITU-T Specification)

**5.8.2.9 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::colourMatrix**

[in]: Specifies the matrix coefficients used in deriving the luma and chroma from the RGB primaries (as defined in Annex E of the ITU-T Specification).

**5.8.2.10 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::chromaSampleLocationFlag**

[in]: if set to 1 , it specifies that the chromaSampleLocationTop and chromaSampleLocationBot are present.

**5.8.2.11 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::chromaSampleLocationTop**

[in]: Specifies the chroma sample location for top field(as defined in Annex E of the ITU-T Specification)

**5.8.2.12 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::chromaSampleLocationBot**

[in]: Specifies the chroma sample location for bottom field(as defined in Annex E of the ITU-T Specification)

**5.8.2.13 uint32\_t NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS::bitstreamRestrictionFlag**

[in]: if set to 1, it specifies the bitstream restriction parameters are present in the bitstream.

## 5.9 NV\_ENC\_CONFIG\_HEVC Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [level](#)
- uint32\_t [tier](#)
- [NV\\_ENC\\_HEVC\\_CUSIZE](#) minCUsSize
- [NV\\_ENC\\_HEVC\\_CUSIZE](#) maxCUsSize
- uint32\_t [useConstrainedIntraPred](#):1
- uint32\_t [disableDeblockAcrossSliceBoundary](#):1
- uint32\_t [outputBufferingPeriodSEI](#):1
- uint32\_t [outputPictureTimingSEI](#):1
- uint32\_t [outputAUD](#):1
- uint32\_t [enableLTR](#):1
- uint32\_t [disableSPSPPS](#):1
- uint32\_t [repeatSPSPPS](#):1
- uint32\_t [enableIntraRefresh](#):1
- uint32\_t [chromaFormatIDC](#):2
- uint32\_t [pixelBitDepthMinus8](#):3
- uint32\_t [reserved](#):18
- uint32\_t [idrPeriod](#)
- uint32\_t [intraRefreshPeriod](#)
- uint32\_t [intraRefreshCnt](#)
- uint32\_t [maxNumRefFramesInDPB](#)
- uint32\_t [ltrNumFrames](#)
- uint32\_t [vpsId](#)
- uint32\_t [spsId](#)
- uint32\_t [ppsId](#)
- uint32\_t [sliceMode](#)
- uint32\_t [sliceModeData](#)
- uint32\_t [maxTemporalLayersMinus1](#)
- [NV\\_ENC\\_CONFIG\\_HEVC\\_VUI\\_PARAMETERS](#) hevcVUIParameters
- uint32\_t [ltrTrustMode](#)
- uint32\_t [reserved1](#) [217]
- void \* [reserved2](#) [64]

### 5.9.1 Detailed Description

`_NV_ENC_CONFIG_HEVC` HEVC encoder configuration parameters to be set during initialization.

### 5.9.2 Field Documentation

#### 5.9.2.1 uint32\_t NV\_ENC\_CONFIG\_HEVC::level

[in]: Specifies the level of the encoded bitstream.

### 5.9.2.2 `uint32_t NV_ENC_CONFIG_HEVC::tier`

[in]: Specifies the level tier of the encoded bitstream.

### 5.9.2.3 `NV_ENC_HEVC_CUSIZE NV_ENC_CONFIG_HEVC::minCUsSize`

[in]: Specifies the minimum size of luma coding unit.

### 5.9.2.4 `NV_ENC_HEVC_CUSIZE NV_ENC_CONFIG_HEVC::maxCUsSize`

[in]: Specifies the maximum size of luma coding unit. Currently NVENC SDK only supports maxCUsSize equal to NV\_ENC\_HEVC\_CUSIZE\_32x32.

### 5.9.2.5 `uint32_t NV_ENC_CONFIG_HEVC::useConstrainedIntraPred`

[in]: Set 1 to enable constrained intra prediction.

### 5.9.2.6 `uint32_t NV_ENC_CONFIG_HEVC::disableDeblockAcrossSliceBoundary`

[in]: Set 1 to disable in loop filtering across slice boundary.

### 5.9.2.7 `uint32_t NV_ENC_CONFIG_HEVC::outputBufferingPeriodSEI`

[in]: Set 1 to write SEI buffering period syntax in the bitstream

### 5.9.2.8 `uint32_t NV_ENC_CONFIG_HEVC::outputPictureTimingSEI`

[in]: Set 1 to write SEI picture timing syntax in the bitstream

### 5.9.2.9 `uint32_t NV_ENC_CONFIG_HEVC::outputAUD`

[in]: Set 1 to write Access Unit Delimiter syntax.

### 5.9.2.10 `uint32_t NV_ENC_CONFIG_HEVC::enableLTR`

[in]: Set to 1 to enable LTR (Long Term Reference) frame support. LTR can be used in two modes: "LTR Trust" mode and "LTR Per Picture" mode. LTR Trust mode: In this mode, ltrNumFrames pictures after IDR are automatically marked as LTR. This mode is enabled by setting ltrTrustMode = 1. Use of LTR Trust mode is strongly discouraged as this mode may be deprecated in future releases. LTR Per Picture mode: In this mode, client can control whether the current picture should be marked as LTR. Enable this mode by setting ltrTrustMode = 0 and ltrMarkFrame = 1 for the picture to be marked as LTR. This is the preferred mode for using LTR. Note that LTRs are not supported if encoding session is configured with B-frames

### 5.9.2.11 `uint32_t NV_ENC_CONFIG_HEVC::disableSPSPPS`

[in]: Set 1 to disable VPS,SPS and PPS signalling in the bitstream.

**5.9.2.12 uint32\_t NV\_ENC\_CONFIG\_HEVC::repeatSPSPPS**

[in]: Set 1 to output VPS,SPS and PPS for every IDR frame.

**5.9.2.13 uint32\_t NV\_ENC\_CONFIG\_HEVC::enableIntraRefresh**

[in]: Set 1 to enable gradual decoder refresh or intra refresh. If the GOP structure uses B frames this will be ignored

**5.9.2.14 uint32\_t NV\_ENC\_CONFIG\_HEVC::chromaFormatIDC**

[in]: Specifies the chroma format. Should be set to 1 for yuv420 input, 3 for yuv444 input.

**5.9.2.15 uint32\_t NV\_ENC\_CONFIG\_HEVC::pixelBitDepthMinus8**

[in]: Specifies pixel bit depth minus 8. Should be set to 0 for 8 bit input, 2 for 10 bit input.

**5.9.2.16 uint32\_t NV\_ENC\_CONFIG\_HEVC::reserved**

[in]: Reserved bitfields.

**5.9.2.17 uint32\_t NV\_ENC\_CONFIG\_HEVC::idrPeriod**

[in]: Specifies the IDR interval. If not set, this is made equal to gopLength in NV\_ENC\_CONFIG. Low latency application client can set IDR interval to NVENC\_INFINITE\_GOPLength so that IDR frames are not inserted automatically.

**5.9.2.18 uint32\_t NV\_ENC\_CONFIG\_HEVC::intraRefreshPeriod**

[in]: Specifies the interval between successive intra refresh if enableIntraRefresh is set. Requires enableIntraRefresh to be set. Will be disabled if [NV\\_ENC\\_CONFIG::gopLength](#) is not set to NVENC\_INFINITE\_GOPLength.

**5.9.2.19 uint32\_t NV\_ENC\_CONFIG\_HEVC::intraRefreshCnt**

[in]: Specifies the length of intra refresh in number of frames for periodic intra refresh. This value should be smaller than intraRefreshPeriod

**5.9.2.20 uint32\_t NV\_ENC\_CONFIG\_HEVC::maxNumRefFramesInDPB**

[in]: Specifies the maximum number of references frames in the DPB.

**5.9.2.21 uint32\_t NV\_ENC\_CONFIG\_HEVC::ltrNumFrames**

[in]: This parameter has different meaning in two LTR modes. In "LTR Trust" mode (ltrTrustMode = 1), encoder will mark the first ltrNumFrames base layer reference frames within each IDR interval as LTR. In "LTR Per Picture" mode (ltrTrustMode = 0 and ltrMarkFrame = 1), ltrNumFrames specifies maximum number of LTR frames in DPB.

**5.9.2.22 uint32\_t NV\_ENC\_CONFIG\_HEVC::vpsId**

[in]: Specifies the VPS id of the video parameter set

**5.9.2.23 uint32\_t NV\_ENC\_CONFIG\_HEVC::spsId**

[in]: Specifies the SPS id of the sequence header

**5.9.2.24 uint32\_t NV\_ENC\_CONFIG\_HEVC::ppsId**

[in]: Specifies the PPS id of the picture header

**5.9.2.25 uint32\_t NV\_ENC\_CONFIG\_HEVC::sliceMode**

[in]: This parameter in conjunction with sliceModeData specifies the way in which the picture is divided into slices sliceMode = 0 CTU based slices, sliceMode = 1 Byte based slices, sliceMode = 2 CTU row based slices, sliceMode = 3, numSlices in Picture When sliceMode == 0 and sliceModeData == 0 whole picture will be coded with one slice

**5.9.2.26 uint32\_t NV\_ENC\_CONFIG\_HEVC::sliceModeData**

[in]: Specifies the parameter needed for sliceMode. For: sliceMode = 0, sliceModeData specifies # of CTUs in each slice (except last slice) sliceMode = 1, sliceModeData specifies maximum # of bytes in each slice (except last slice) sliceMode = 2, sliceModeData specifies # of CTU rows in each slice (except last slice) sliceMode = 3, sliceModeData specifies number of slices in the picture. Driver will divide picture into slices optimally

**5.9.2.27 uint32\_t NV\_ENC\_CONFIG\_HEVC::maxTemporalLayersMinus1**

[in]: Specifies the max temporal layer used for hierarchical coding.

**5.9.2.28 NV\_ENC\_CONFIG\_HEVC\_VUI\_PARAMETERS NV\_ENC\_CONFIG\_HEVC::hevcVUIParameters**

[in]: Specifies the HEVC video usability info parameters

**5.9.2.29 uint32\_t NV\_ENC\_CONFIG\_HEVC::ltrTrustMode**

[in]: Specifies the LTR operating mode. See comments near [NV\\_ENC\\_CONFIG\\_HEVC::enableLTR](#) for description of the two modes. Set to 1 to use "LTR Trust" mode of LTR operation. Clients are discouraged to use "LTR Trust" mode as this mode may be deprecated in future releases. Set to 0 when using "LTR Per Picture" mode of LTR operation.

**5.9.2.30 uint32\_t NV\_ENC\_CONFIG\_HEVC::reserved1[217]**

[in]: Reserved and must be set to 0.

**5.9.2.31 void\* NV\_ENC\_CONFIG\_HEVC::reserved2[64]**

[in]: Reserved and must be set to NULL

## 5.10 NV\_ENC\_CONFIG\_HEVC\_MEONLY Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [reserved](#) [256]
- void \* [reserved1](#) [64]

### 5.10.1 Detailed Description

\_NV\_ENC\_CONFIG\_HEVC\_MEONLY HEVC encoder configuration parameters for ME only Mode

### 5.10.2 Field Documentation

#### 5.10.2.1 uint32\_t NV\_ENC\_CONFIG\_HEVC\_MEONLY::reserved[256]

[in]: Reserved and must be set to 0

#### 5.10.2.2 void\* NV\_ENC\_CONFIG\_HEVC\_MEONLY::reserved1[64]

[in]: Reserved and must be set to NULL

## 5.11 NV\_ENC\_CREATE\_BITSTREAM\_BUFFER Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- `uint32_t` [version](#)
- `uint32_t` [size](#)
- `NV_ENC_MEMORY_HEAP` [memoryHeap](#)
- `uint32_t` [reserved](#)
- `NV_ENC_OUTPUT_PTR` [bitstreamBuffer](#)
- `void *` [bitstreamBufferPtr](#)
- `uint32_t` [reserved1](#) [58]
- `void *` [reserved2](#) [64]

### 5.11.1 Detailed Description

Creation parameters for output bitstream buffer.

### 5.11.2 Field Documentation

#### 5.11.2.1 `uint32_t NV_ENC_CREATE_BITSTREAM_BUFFER::version`

[in]: Struct version. Must be set to [NV\\_ENC\\_CREATE\\_BITSTREAM\\_BUFFER\\_VER](#)

#### 5.11.2.2 `uint32_t NV_ENC_CREATE_BITSTREAM_BUFFER::size`

[in]: Deprecated. Do not use

#### 5.11.2.3 `NV_ENC_MEMORY_HEAP NV_ENC_CREATE_BITSTREAM_BUFFER::memoryHeap`

[in]: Deprecated. Do not use

#### 5.11.2.4 `uint32_t NV_ENC_CREATE_BITSTREAM_BUFFER::reserved`

[in]: Reserved and must be set to 0

#### 5.11.2.5 `NV_ENC_OUTPUT_PTR NV_ENC_CREATE_BITSTREAM_BUFFER::bitstreamBuffer`

[out]: Pointer to the output bitstream buffer

#### 5.11.2.6 `void* NV_ENC_CREATE_BITSTREAM_BUFFER::bitstreamBufferPtr`

[out]: Reserved and should not be used



**5.11.2.7 uint32\_t NV\_ENC\_CREATE\_BITSTREAM\_BUFFER::reserved1[58]**

[in]: Reserved and should be set to 0

**5.11.2.8 void\* NV\_ENC\_CREATE\_BITSTREAM\_BUFFER::reserved2[64]**

[in]: Reserved and should be set to NULL

## 5.12 NV\_ENC\_CREATE\_INPUT\_BUFFER Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [version](#)
- uint32\_t [width](#)
- uint32\_t [height](#)
- NV\_ENC\_MEMORY\_HEAP [memoryHeap](#)
- NV\_ENC\_BUFFER\_FORMAT [bufferFmt](#)
- uint32\_t [reserved](#)
- NV\_ENC\_INPUT\_PTR [inputBuffer](#)
- void \* [pSysMemBuffer](#)
- uint32\_t [reserved1](#) [57]
- void \* [reserved2](#) [63]

### 5.12.1 Detailed Description

Creation parameters for input buffer.

### 5.12.2 Field Documentation

#### 5.12.2.1 uint32\_t NV\_ENC\_CREATE\_INPUT\_BUFFER::version

[in]: Struct version. Must be set to [NV\\_ENC\\_CREATE\\_INPUT\\_BUFFER\\_VER](#)

#### 5.12.2.2 uint32\_t NV\_ENC\_CREATE\_INPUT\_BUFFER::width

[in]: Input buffer width

#### 5.12.2.3 uint32\_t NV\_ENC\_CREATE\_INPUT\_BUFFER::height

[in]: Input buffer width

#### 5.12.2.4 NV\_ENC\_MEMORY\_HEAP NV\_ENC\_CREATE\_INPUT\_BUFFER::memoryHeap

[in]: Deprecated. Do not use

#### 5.12.2.5 NV\_ENC\_BUFFER\_FORMAT NV\_ENC\_CREATE\_INPUT\_BUFFER::bufferFmt

[in]: Input buffer format

#### 5.12.2.6 uint32\_t NV\_ENC\_CREATE\_INPUT\_BUFFER::reserved

[in]: Reserved and must be set to 0

**5.12.2.7 NV\_ENC\_INPUT\_PTR NV\_ENC\_CREATE\_INPUT\_BUFFER::inputBuffer**

[out]: Pointer to input buffer

**5.12.2.8 void\* NV\_ENC\_CREATE\_INPUT\_BUFFER::pSysMemBuffer**

[in]: Pointer to existing system buffer

**5.12.2.9 uint32\_t NV\_ENC\_CREATE\_INPUT\_BUFFER::reserved1[57]**

[in]: Reserved and must be set to 0

**5.12.2.10 void\* NV\_ENC\_CREATE\_INPUT\_BUFFER::reserved2[63]**

[in]: Reserved and must be set to NULL

## 5.13 NV\_ENC\_CREATE\_MV\_BUFFER Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [version](#)
- NV\_ENC\_OUTPUT\_PTR [mvBuffer](#)
- uint32\_t [reserved1](#) [255]
- void \* [reserved2](#) [63]

### 5.13.1 Detailed Description

Creation parameters for output motion vector buffer for ME only mode.

### 5.13.2 Field Documentation

#### 5.13.2.1 uint32\_t NV\_ENC\_CREATE\_MV\_BUFFER::version

[in]: Struct version. Must be set to NV\_ENC\_CREATE\_MV\_BUFFER\_VER

#### 5.13.2.2 NV\_ENC\_OUTPUT\_PTR NV\_ENC\_CREATE\_MV\_BUFFER::mvBuffer

[out]: Pointer to the output motion vector buffer

#### 5.13.2.3 uint32\_t NV\_ENC\_CREATE\_MV\_BUFFER::reserved1[255]

[in]: Reserved and should be set to 0

#### 5.13.2.4 void\* NV\_ENC\_CREATE\_MV\_BUFFER::reserved2[63]

[in]: Reserved and should be set to NULL

## 5.14 NV\_ENC\_EVENT\_PARAMS Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [version](#)
- uint32\_t [reserved](#)
- void \* [completionEvent](#)
- uint32\_t [reserved1](#) [253]
- void \* [reserved2](#) [64]

#### 5.14.1 Detailed Description

Event registration/unregistration parameters.

#### 5.14.2 Field Documentation

##### 5.14.2.1 uint32\_t NV\_ENC\_EVENT\_PARAMS::version

[in]: Struct version. Must be set to [NV\\_ENC\\_EVENT\\_PARAMS\\_VER](#).

##### 5.14.2.2 uint32\_t NV\_ENC\_EVENT\_PARAMS::reserved

[in]: Reserved and must be set to 0

##### 5.14.2.3 void\* NV\_ENC\_EVENT\_PARAMS::completionEvent

[in]: Handle to event to be registered/unregistered with the NvEncodeAPI interface.

##### 5.14.2.4 uint32\_t NV\_ENC\_EVENT\_PARAMS::reserved1[253]

[in]: Reserved and must be set to 0

##### 5.14.2.5 void\* NV\_ENC\_EVENT\_PARAMS::reserved2[64]

[in]: Reserved and must be set to NULL

## 5.15 NV\_ENC\_H264\_MV\_DATA Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- [NV\\_ENC\\_MVECTOR mv](#) [4]
- [uint8\\_t mbType](#)
- [uint8\\_t partitionType](#)
- [uint16\\_t reserved](#)

### 5.15.1 Detailed Description

Motion vector structure per macroblock for H264 motion estimation.

### 5.15.2 Field Documentation

#### 5.15.2.1 NV\_ENC\_MVECTOR NV\_ENC\_H264\_MV\_DATA::mv[4]

up to 4 vectors for 8x8 partition

#### 5.15.2.2 uint8\_t NV\_ENC\_H264\_MV\_DATA::mbType

0 (I), 1 (P), 2 (IPCM), 3 (B)

#### 5.15.2.3 uint8\_t NV\_ENC\_H264\_MV\_DATA::partitionType

Specifies the block partition type. 0:16x16, 1:8x8, 2:16x8, 3:8x16

#### 5.15.2.4 uint16\_t NV\_ENC\_H264\_MV\_DATA::reserved

reserved padding for alignment

## 5.16 NV\_ENC\_HEVC\_MV\_DATA Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- [NV\\_ENC\\_MVECTOR mv](#) [4]
- [uint8\\_t cuType](#)
- [uint8\\_t cuSize](#)
- [uint8\\_t partitionMode](#)
- [uint8\\_t lastCUInCTB](#)

### 5.16.1 Detailed Description

Motion vector structure per CU for HEVC motion estimation.

### 5.16.2 Field Documentation

#### 5.16.2.1 NV\_ENC\_MVECTOR NV\_ENC\_HEVC\_MV\_DATA::mv[4]

up to 4 vectors within a CU

#### 5.16.2.2 uint8\_t NV\_ENC\_HEVC\_MV\_DATA::cuType

0 (I), 1(P), 2 (Skip)

#### 5.16.2.3 uint8\_t NV\_ENC\_HEVC\_MV\_DATA::cuSize

0: 8x8, 1: 16x16, 2: 32x32, 3: 64x64

#### 5.16.2.4 uint8\_t NV\_ENC\_HEVC\_MV\_DATA::partitionMode

The CU partition mode 0 (2Nx2N), 1 (2NxN), 2(Nx2N), 3 (NxN), 4 (2NxnU), 5 (2NxnD), 6(nLx2N), 7 (nRx2N)

#### 5.16.2.5 uint8\_t NV\_ENC\_HEVC\_MV\_DATA::lastCUInCTB

Marker to separate CUs in the current CTB from CUs in the next CTB

## 5.17 NV\_ENC\_INITIALIZE\_PARAMS Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [version](#)
- GUID [encodeGUID](#)
- GUID [presetGUID](#)
- uint32\_t [encodeWidth](#)
- uint32\_t [encodeHeight](#)
- uint32\_t [darWidth](#)
- uint32\_t [darHeight](#)
- uint32\_t [frameRateNum](#)
- uint32\_t [frameRateDen](#)
- uint32\_t [enableEncodeAsync](#)
- uint32\_t [enablePTD](#)
- uint32\_t [reportSliceOffsets](#):1
- uint32\_t [enableSubFrameWrite](#):1
- uint32\_t [enableExternalMEHints](#):1
- uint32\_t [enableMEOnlyMode](#):1
- uint32\_t [enableWeightedPrediction](#):1
- uint32\_t [reservedBitFields](#):27
- uint32\_t [privDataSize](#)
- void \* [privData](#)
- NV\_ENC\_CONFIG \* [encodeConfig](#)
- uint32\_t [maxEncodeWidth](#)
- uint32\_t [maxEncodeHeight](#)
- NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE [maxMEHintCountsPerBlock](#) [2]
- uint32\_t [reserved](#) [289]
- void \* [reserved2](#) [64]

### 5.17.1 Detailed Description

[\\_NV\\_ENC\\_INITIALIZE\\_PARAMS](#) Encode Session Initialization parameters.

### 5.17.2 Field Documentation

#### 5.17.2.1 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::version

[in]: Struct version. Must be set to [NV\\_ENC\\_INITIALIZE\\_PARAMS\\_VER](#).

#### 5.17.2.2 GUID NV\_ENC\_INITIALIZE\_PARAMS::encodeGUID

[in]: Specifies the Encode [GUID](#) for which the encoder is being created. [NvEncInitializeEncoder\(\)](#) API will fail if this is not set, or set to unsupported value.



### 5.17.2.3 GUID NV\_ENC\_INITIALIZE\_PARAMS::presetGUID

[in]: Specifies the preset for encoding. If the preset [GUID](#) is set then , the preset configuration will be applied before any other parameter.

### 5.17.2.4 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::encodeWidth

[in]: Specifies the encode width. If not set [NvEncInitializeEncoder\(\)](#) API will fail.

### 5.17.2.5 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::encodeHeight

[in]: Specifies the encode height. If not set [NvEncInitializeEncoder\(\)](#) API will fail.

### 5.17.2.6 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::darWidth

[in]: Specifies the display aspect ratio Width.

### 5.17.2.7 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::darHeight

[in]: Specifies the display aspect ratio height.

### 5.17.2.8 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::frameRateNum

[in]: Specifies the numerator for frame rate used for encoding in frames per second ( Frame rate = frameRateNum / frameRateDen ).

### 5.17.2.9 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::frameRateDen

[in]: Specifies the denominator for frame rate used for encoding in frames per second ( Frame rate = frameRateNum / frameRateDen ).

### 5.17.2.10 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::enableEncodeAsync

[in]: Set this to 1 to enable asynchronous mode and is expected to use events to get picture completion notification.

### 5.17.2.11 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::enablePTD

[in]: Set this to 1 to enable the Picture Type Decision is be taken by the NvEncodeAPI interface.

### 5.17.2.12 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::reportSliceOffsets

[in]: Set this to 1 to enable reporting slice offsets in `_NV_ENC_LOCK_BITSTREAM`. [NV\\_ENC\\_INITIALIZE\\_PARAMS::enableEncodeAsync](#) must be set to 0 to use this feature. Client must set this to 0 if [NV\\_ENC\\_CONFIG\\_H264::sliceMode](#) is 1 on Kepler GPUs

**5.17.2.13 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::enableSubFrameWrite**

[in]: Set this to 1 to write out available bitstream to memory at subframe intervals

**5.17.2.14 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::enableExternalMEHints**

[in]: Set to 1 to enable external ME hints for the current frame. For [NV\\_ENC\\_INITIALIZE\\_PARAMS::enablePTD=1](#) with B frames, programming L1 hints is optional for B frames since Client doesn't know internal GOP structure. [NV\\_ENC\\_PICTURE\\_PARAMS::meHintRefPicDist](#) should preferably be set with enablePTD=1.

**5.17.2.15 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::enableMEOnlyMode**

[in]: Set to 1 to enable ME Only Mode .

**5.17.2.16 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::enableWeightedPrediction**

[in]: Set this to 1 to enable weighted prediction. Not supported if encode session is configured for B-Frames( 'frameIntervalP' in [NV\\_ENC\\_CONFIG](#) is greater than 1).

**5.17.2.17 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::reservedBitFields**

[in]: Reserved bitfields and must be set to 0

**5.17.2.18 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::privDataSize**

[in]: Reserved private data buffer size and must be set to 0

**5.17.2.19 void\* NV\_ENC\_INITIALIZE\_PARAMS::privData**

[in]: Reserved private data buffer and must be set to NULL

**5.17.2.20 NV\_ENC\_CONFIG\* NV\_ENC\_INITIALIZE\_PARAMS::encodeConfig**

[in]: Specifies the advanced codec specific structure. If client has sent a valid codec config structure, it will override parameters set by the [NV\\_ENC\\_INITIALIZE\\_PARAMS::presetGUID](#) parameter. If set to NULL the NvEncodeAPI interface will use the [NV\\_ENC\\_INITIALIZE\\_PARAMS::presetGUID](#) to set the codec specific parameters. Client can also optionally query the NvEncodeAPI interface to get codec specific parameters for a presetGUID using [NvEncGetEncodePresetConfig\(\)](#) API. It can then modify (if required) some of the codec config parameters and send down a custom config structure as part of [\\_NV\\_ENC\\_INITIALIZE\\_PARAMS](#). Even in this case client is recommended to pass the same preset guid it has used in [NvEncGetEncodePresetConfig\(\)](#) API to query the config structure; as [NV\\_ENC\\_INITIALIZE\\_PARAMS::presetGUID](#). This will not override the custom config structure but will be used to determine other Encoder HW specific parameters not exposed in the API.

**5.17.2.21 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::maxEncodeWidth**

[in]: Maximum encode width to be used for current Encode session. Client should allocate output buffers according to this dimension for dynamic resolution change. If set to 0, Encoder will not allow dynamic resolution change.

**5.17.2.22 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::maxEncodeHeight**

[in]: Maximum encode height to be allowed for current Encode session. Client should allocate output buffers according to this dimension for dynamic resolution change. If set to 0, Encode will not allow dynamic resolution change.

**5.17.2.23 NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE NV\_ENC\_INITIALIZE\_PARAMS::maxMEHintCountsPerBlock[2]**

[in]: If Client wants to pass external motion vectors in [NV\\_ENC\\_PICTURE\\_PARAMS::meExternalHints](#) buffer it must specify the maximum number of hint candidates per block per direction for the encode session. The [NV\\_ENC\\_INITIALIZE\\_PARAMS::maxMEHintCountsPerBlock\[0\]](#) is for L0 predictors and [NV\\_ENC\\_INITIALIZE\\_PARAMS::maxMEHintCountsPerBlock\[1\]](#) is for L1 predictors. This client must also set [NV\\_ENC\\_INITIALIZE\\_PARAMS::enableExternalMEHints](#) to 1.

**5.17.2.24 uint32\_t NV\_ENC\_INITIALIZE\_PARAMS::reserved[289]**

[in]: Reserved and must be set to 0

**5.17.2.25 void\* NV\_ENC\_INITIALIZE\_PARAMS::reserved2[64]**

[in]: Reserved and must be set to NULL

## 5.18 NV\_ENC\_INPUT\_RESOURCE\_OPENGL\_TEX Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- `uint32_t texture`
- `uint32_t target`

### 5.18.1 Detailed Description

`_NV_ENC_INPUT_RESOURCE_OPENGL_TEX` `NV_ENC_REGISTER_RESOURCE::resourceToRegister` must be a pointer to a variable of this type, when `NV_ENC_REGISTER_RESOURCE::resourceType` is `NV_ENC_INPUT_RESOURCE_TYPE_OPENGL_TEX`

### 5.18.2 Field Documentation

#### 5.18.2.1 `uint32_t NV_ENC_INPUT_RESOURCE_OPENGL_TEX::texture`

[in]: The name of the texture to be used.

#### 5.18.2.2 `uint32_t NV_ENC_INPUT_RESOURCE_OPENGL_TEX::target`

[in]: Accepted values are `GL_TEXTURE_RECTANGLE` and `GL_TEXTURE_2D`.

## 5.19 NV\_ENC\_LOCK\_BITSTREAM Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [version](#)
- uint32\_t [doNotWait](#):1
- uint32\_t [ltrFrame](#):1
- uint32\_t [reservedBitFields](#):30
- void \* [outputBitstream](#)
- uint32\_t \* [sliceOffsets](#)
- uint32\_t [frameIdx](#)
- uint32\_t [hwEncodeStatus](#)
- uint32\_t [numSlices](#)
- uint32\_t [bitstreamSizeInBytes](#)
- uint64\_t [outputTimeStamp](#)
- uint64\_t [outputDuration](#)
- void \* [bitstreamBufferPtr](#)
- [NV\\_ENC\\_PIC\\_TYPE](#) [pictureType](#)
- [NV\\_ENC\\_PIC\\_STRUCT](#) [pictureStruct](#)
- uint32\_t [frameAvgQP](#)
- uint32\_t [frameSatd](#)
- uint32\_t [ltrFrameIdx](#)
- uint32\_t [ltrFrameBitmap](#)
- uint32\_t [reserved](#) [236]
- void \* [reserved2](#) [64]

### 5.19.1 Detailed Description

[\\_NV\\_ENC\\_LOCK\\_BITSTREAM](#) Bitstream buffer lock parameters.

### 5.19.2 Field Documentation

#### 5.19.2.1 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::version

[in]: Struct version. Must be set to [NV\\_ENC\\_LOCK\\_BITSTREAM\\_VER](#).

#### 5.19.2.2 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::doNotWait

[in]: If this flag is set, the NvEncodeAPI interface will return buffer pointer even if operation is not completed. If not set, the call will block until operation completes.

#### 5.19.2.3 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::ltrFrame

[out]: Flag indicating this frame is marked as LTR frame

**5.19.2.4 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::reservedBitFields**

[in]: Reserved bit fields and must be set to 0

**5.19.2.5 void\* NV\_ENC\_LOCK\_BITSTREAM::outputBitstream**

[in]: Pointer to the bitstream buffer being locked.

**5.19.2.6 uint32\_t\* NV\_ENC\_LOCK\_BITSTREAM::sliceOffsets**

[in,out]: Array which receives the slice offsets. This is not supported if [NV\\_ENC\\_CONFIG\\_H264::sliceMode](#) is 1 on Kepler GPUs. Array size must be equal to size of frame in MBs.

**5.19.2.7 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::frameIdx**

[out]: Frame no. for which the bitstream is being retrieved.

**5.19.2.8 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::hwEncodeStatus**

[out]: The NvEncodeAPI interface status for the locked picture.

**5.19.2.9 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::numSlices**

[out]: Number of slices in the encoded picture. Will be reported only if [NV\\_ENC\\_INITIALIZE\\_PARAMS::reportSliceOffsets](#) set to 1.

**5.19.2.10 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::bitstreamSizeInBytes**

[out]: Actual number of bytes generated and copied to the memory pointed by bitstreamBufferPtr.

**5.19.2.11 uint64\_t NV\_ENC\_LOCK\_BITSTREAM::outputTimeStamp**

[out]: Presentation timestamp associated with the encoded output.

**5.19.2.12 uint64\_t NV\_ENC\_LOCK\_BITSTREAM::outputDuration**

[out]: Presentation duration associates with the encoded output.

**5.19.2.13 void\* NV\_ENC\_LOCK\_BITSTREAM::bitstreamBufferPtr**

[out]: Pointer to the generated output bitstream. For MEOnly mode [\\_NV\\_ENC\\_LOCK\\_BITSTREAM::bitstreamBufferPtr](#) should be typecast to [NV\\_ENC\\_H264\\_MV\\_DATA](#)/[NV\\_ENC\\_HEVC\\_MV\\_DATA](#) pointer respectively for H264/HEVC

**5.19.2.14 NV\_ENC\_PICTURE\_TYPE NV\_ENC\_LOCK\_BITSTREAM::pictureType**

[out]: Picture type of the encoded picture.

**5.19.2.15 NV\_ENC\_PIC\_STRUCT NV\_ENC\_LOCK\_BITSTREAM::pictureStruct**

[out]: Structure of the generated output picture.

**5.19.2.16 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::frameAvgQP**

[out]: Average QP of the frame.

**5.19.2.17 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::frameSatd**

[out]: Total SATD cost for whole frame.

**5.19.2.18 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::ltrFrameIdx**

[out]: Frame index associated with this LTR frame.

**5.19.2.19 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::ltrFrameBitmap**

[out]: Bitmap of LTR frames indices which were used for encoding this frame. Value of 0 if no LTR frames were used.

**5.19.2.20 uint32\_t NV\_ENC\_LOCK\_BITSTREAM::reserved[236]**

[in]: Reserved and must be set to 0

**5.19.2.21 void\* NV\_ENC\_LOCK\_BITSTREAM::reserved2[64]**

[in]: Reserved and must be set to NULL

## 5.20 NV\_ENC\_LOCK\_INPUT\_BUFFER Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [version](#)
- uint32\_t [doNotWait](#):1
- uint32\_t [reservedBitFields](#):31
- NV\_ENC\_INPUT\_PTR [inputBuffer](#)
- void \* [bufferDataPtr](#)
- uint32\_t [pitch](#)
- uint32\_t [reserved1](#) [251]
- void \* [reserved2](#) [64]

### 5.20.1 Detailed Description

[\\_NV\\_ENC\\_LOCK\\_INPUT\\_BUFFER](#) Uncompressed Input Buffer lock parameters.

### 5.20.2 Field Documentation

#### 5.20.2.1 uint32\_t NV\_ENC\_LOCK\_INPUT\_BUFFER::version

[in]: Struct version. Must be set to [NV\\_ENC\\_LOCK\\_INPUT\\_BUFFER\\_VER](#).

#### 5.20.2.2 uint32\_t NV\_ENC\_LOCK\_INPUT\_BUFFER::doNotWait

[in]: Set to 1 to make [NvEncLockInputBuffer\(\)](#) a unblocking call. If the encoding is not completed, driver will return [NV\\_ENC\\_ERR\\_ENCODER\\_BUSY](#) error code.

#### 5.20.2.3 uint32\_t NV\_ENC\_LOCK\_INPUT\_BUFFER::reservedBitFields

[in]: Reserved bitfields and must be set to 0

#### 5.20.2.4 NV\_ENC\_INPUT\_PTR NV\_ENC\_LOCK\_INPUT\_BUFFER::inputBuffer

[in]: Pointer to the input buffer to be locked, client should pass the pointer obtained from [NvEncCreateInputBuffer\(\)](#) or [NvEncMapInputResource](#) API.

#### 5.20.2.5 void\* NV\_ENC\_LOCK\_INPUT\_BUFFER::bufferDataPtr

[out]: Pointed to the locked input buffer data. Client can only access input buffer using the `bufferDataPtr`.

#### 5.20.2.6 uint32\_t NV\_ENC\_LOCK\_INPUT\_BUFFER::pitch

[out]: Pitch of the locked input buffer.



**5.20.2.7 uint32\_t NV\_ENC\_LOCK\_INPUT\_BUFFER::reserved1[251]**

[in]: Reserved and must be set to 0

**5.20.2.8 void\* NV\_ENC\_LOCK\_INPUT\_BUFFER::reserved2[64]**

[in]: Reserved and must be set to NULL

## 5.21 NV\_ENC\_MAP\_INPUT\_RESOURCE Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [version](#)
- uint32\_t [subResourceIndex](#)
- void \* [inputResource](#)
- NV\_ENC\_REGISTERED\_PTR [registeredResource](#)
- NV\_ENC\_INPUT\_PTR [mappedResource](#)
- NV\_ENC\_BUFFER\_FORMAT [mappedBufferFmt](#)
- uint32\_t [reserved1](#) [251]
- void \* [reserved2](#) [63]

### 5.21.1 Detailed Description

`_NV_ENC_MAP_INPUT_RESOURCE` Map an input resource to a Nvidia Encoder Input Buffer

### 5.21.2 Field Documentation

#### 5.21.2.1 uint32\_t NV\_ENC\_MAP\_INPUT\_RESOURCE::version

[in]: Struct version. Must be set to [NV\\_ENC\\_MAP\\_INPUT\\_RESOURCE\\_VER](#).

#### 5.21.2.2 uint32\_t NV\_ENC\_MAP\_INPUT\_RESOURCE::subResourceIndex

[in]: Deprecated. Do not use.

#### 5.21.2.3 void\* NV\_ENC\_MAP\_INPUT\_RESOURCE::inputResource

[in]: Deprecated. Do not use.

#### 5.21.2.4 NV\_ENC\_REGISTERED\_PTR NV\_ENC\_MAP\_INPUT\_RESOURCE::registeredResource

[in]: The Registered resource handle obtained by calling `NvEncRegisterInputResource`.

#### 5.21.2.5 NV\_ENC\_INPUT\_PTR NV\_ENC\_MAP\_INPUT\_RESOURCE::mappedResource

[out]: Mapped pointer corresponding to the `registeredResource`. This pointer must be used in [NV\\_ENC\\_PICTURE\\_PARAMS::inputBuffer](#) parameter in `NvEncEncodePicture()` API.

#### 5.21.2.6 NV\_ENC\_BUFFER\_FORMAT NV\_ENC\_MAP\_INPUT\_RESOURCE::mappedBufferFmt

[out]: Buffer format of the `outputResource`. This buffer format must be used in [NV\\_ENC\\_PICTURE\\_PARAMS::bufferFmt](#) if client using the above mapped resource pointer.

**5.21.2.7 uint32\_t NV\_ENC\_MAP\_INPUT\_RESOURCE::reserved1[251]**

[in]: Reserved and must be set to 0.

**5.21.2.8 void\* NV\_ENC\_MAP\_INPUT\_RESOURCE::reserved2[63]**

[in]: Reserved and must be set to NULL

## 5.22 NV\_ENC\_MEONLY\_PARAMS Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [version](#)
- uint32\_t [inputWidth](#)
- uint32\_t [inputHeight](#)
- NV\_ENC\_INPUT\_PTR [inputBuffer](#)
- NV\_ENC\_INPUT\_PTR [referenceFrame](#)
- NV\_ENC\_OUTPUT\_PTR [mvBuffer](#)
- NV\_ENC\_BUFFER\_FORMAT [bufferFmt](#)
- void \* [completionEvent](#)
- uint32\_t [viewID](#)
- NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE [meHintCountsPerBlock](#) [2]
- NVENC\_EXTERNAL\_ME\_HINT \* [meExternalHints](#)
- uint32\_t [reserved1](#) [243]
- void \* [reserved2](#) [59]

### 5.22.1 Detailed Description

\_NV\_ENC\_MEONLY\_PARAMS MEOnly parameters that need to be sent on a per motion estimation basis. [NV\\_ENC\\_MEONLY\\_PARAMS::meExternalHints](#) is supported for H264 only.

### 5.22.2 Field Documentation

#### 5.22.2.1 uint32\_t NV\_ENC\_MEONLY\_PARAMS::version

[in]: Struct version. Must be set to NV\_ENC\_MEONLY\_PARAMS\_VER.

#### 5.22.2.2 uint32\_t NV\_ENC\_MEONLY\_PARAMS::inputWidth

[in]: Specifies the input buffer width

#### 5.22.2.3 uint32\_t NV\_ENC\_MEONLY\_PARAMS::inputHeight

[in]: Specifies the input buffer height

#### 5.22.2.4 NV\_ENC\_INPUT\_PTR NV\_ENC\_MEONLY\_PARAMS::inputBuffer

[in]: Specifies the input buffer pointer. Client must use a pointer obtained from [NvEncCreateInputBuffer\(\)](#) or [NvEncMapInputResource\(\)](#) APIs.

#### 5.22.2.5 NV\_ENC\_INPUT\_PTR NV\_ENC\_MEONLY\_PARAMS::referenceFrame

[in]: Specifies the reference frame pointer

**5.22.2.6 NV\_ENC\_OUTPUT\_PTR NV\_ENC\_MEONLY\_PARAMS::mvBuffer**

[in]: Specifies the pointer to motion vector data buffer allocated by NvEncCreateMVBuffer. Client must lock mvBuffer using [NvEncLockBitstream\(\)](#) API to get the motion vector data.

**5.22.2.7 NV\_ENC\_BUFFER\_FORMAT NV\_ENC\_MEONLY\_PARAMS::bufferFmt**

[in]: Specifies the input buffer format.

**5.22.2.8 void\* NV\_ENC\_MEONLY\_PARAMS::completionEvent**

[in]: Specifies an event to be signalled on completion of motion estimation of this Frame [only if operating in Asynchronous mode]. Each output buffer should be associated with a distinct event pointer.

**5.22.2.9 uint32\_t NV\_ENC\_MEONLY\_PARAMS::viewID**

[in]: Specifies left,right viewID if [NV\\_ENC\\_CONFIG\\_H264\\_MEONLY::bStereoEnable](#) is set. viewID can be 0,1 if bStereoEnable is set, 0 otherwise.

**5.22.2.10 NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE NV\_ENC\_MEONLY\_PARAMS::meHintCountsPerBlock[2]**

[in]: Specifies the number of hint candidates per block for the current frame. meHintCountsPerBlock[0] is for L0 predictors. The candidate count in [NV\\_ENC\\_PIC\\_PARAMS::meHintCountsPerBlock\[lx\]](#) must never exceed [NV\\_ENC\\_INITIALIZE\\_PARAMS::maxMEHintCountsPerBlock\[lx\]](#) provided during encoder initialization.

**5.22.2.11 NVENC\_EXTERNAL\_ME\_HINT\* NV\_ENC\_MEONLY\_PARAMS::meExternalHints**

[in]: Specifies the pointer to ME external hints for the current frame. The size of ME hint buffer should be equal to number of macroblocks \* the total number of candidates per macroblock. The total number of candidates per MB per direction =  $1 * \text{meHintCountsPerBlock}[Lx].\text{numCandsPerBlk}16 \times 16 + 2 * \text{meHintCountsPerBlock}[Lx].\text{numCandsPerBlk}16 \times 8 + 2 * \text{meHintCountsPerBlock}[Lx].\text{numCandsPerBlk}8 \times 8 + 4 * \text{meHintCountsPerBlock}[Lx].\text{numCandsPerBlk}8 \times 8$ . For frames using bidirectional ME, the total number of candidates for single macroblock is sum of total number of candidates per MB for each direction (L0 and L1)

**5.22.2.12 uint32\_t NV\_ENC\_MEONLY\_PARAMS::reserved1[243]**

[in]: Reserved and must be set to 0

**5.22.2.13 void\* NV\_ENC\_MEONLY\_PARAMS::reserved2[59]**

[in]: Reserved and must be set to NULL

## 5.23 NV\_ENC\_MVECTOR Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- `int16_t mvx`
- `int16_t mvy`

### 5.23.1 Detailed Description

Structs needed for ME only mode.

### 5.23.2 Field Documentation

#### 5.23.2.1 `int16_t NV_ENC_MVECTOR::mvx`

the x component of MV in qpel units

#### 5.23.2.2 `int16_t NV_ENC_MVECTOR::mvy`

the y component of MV in qpel units

## 5.24 NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_PARAMS Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- `uint32_t` [version](#)
- `NV_ENC_DEVICE_TYPE` [deviceType](#)
- `void *` [device](#)
- `void *` [reserved](#)
- `uint32_t` [apiVersion](#)
- `uint32_t` [reserved1](#) [253]
- `void *` [reserved2](#) [64]

### 5.24.1 Detailed Description

Encoder Session Creation parameters

### 5.24.2 Field Documentation

#### 5.24.2.1 `uint32_t NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS::version`

[in]: Struct version. Must be set to [NV\\_ENC\\_OPEN\\_ENCODE\\_SESSION\\_EX\\_PARAMS\\_VER](#).

#### 5.24.2.2 `NV_ENC_DEVICE_TYPE NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS::deviceType`

[in]: Specified the device Type

#### 5.24.2.3 `void* NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS::device`

[in]: Pointer to client device.

#### 5.24.2.4 `void* NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS::reserved`

[in]: Reserved and must be set to 0.

#### 5.24.2.5 `uint32_t NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS::apiVersion`

[in]: API version. Should be set to NVENCAPI\_VERSION.

#### 5.24.2.6 `uint32_t NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS::reserved1[253]`

[in]: Reserved and must be set to 0

**5.24.2.7 void\* NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_PARAMS::reserved2[64]**

[in]: Reserved and must be set to NULL



## 5.25 NV\_ENC\_PIC\_PARAMS Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [version](#)
- uint32\_t [inputWidth](#)
- uint32\_t [inputHeight](#)
- uint32\_t [inputPitch](#)
- uint32\_t [encodePicFlags](#)
- uint32\_t [frameIdx](#)
- uint64\_t [inputTimeStamp](#)
- uint64\_t [inputDuration](#)
- NV\_ENC\_INPUT\_PTR [inputBuffer](#)
- NV\_ENC\_OUTPUT\_PTR [outputBitstream](#)
- void \* [completionEvent](#)
- NV\_ENC\_BUFFER\_FORMAT [bufferFmt](#)
- NV\_ENC\_PIC\_STRUCT [pictureStruct](#)
- NV\_ENC\_PIC\_TYPE [pictureType](#)
- NV\_ENC\_CODEC\_PIC\_PARAMS [codecPicParams](#)
- NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE [meHintCountsPerBlock](#) [2]
- NVENC\_EXTERNAL\_ME\_HINT \* [meExternalHints](#)
- uint32\_t [reserved1](#) [6]
- void \* [reserved2](#) [2]
- int8\_t \* [qpDeltaMap](#)
- uint32\_t [qpDeltaMapSize](#)
- uint32\_t [reservedBitFields](#)
- uint16\_t [meHintRefPicDist](#) [2]
- uint32\_t [reserved3](#) [286]
- void \* [reserved4](#) [60]

### 5.25.1 Detailed Description

\_NV\_ENC\_PIC\_PARAMS Encoding parameters that need to be sent on a per frame basis.

### 5.25.2 Field Documentation

#### 5.25.2.1 uint32\_t NV\_ENC\_PIC\_PARAMS::version

[in]: Struct version. Must be set to [NV\\_ENC\\_PIC\\_PARAMS\\_VER](#).

#### 5.25.2.2 uint32\_t NV\_ENC\_PIC\_PARAMS::inputWidth

[in]: Specifies the input buffer width

#### 5.25.2.3 uint32\_t NV\_ENC\_PIC\_PARAMS::inputHeight

[in]: Specifies the input buffer height

**5.25.2.4 uint32\_t NV\_ENC\_PIC\_PARAMS::inputPitch**

[in]: Specifies the input buffer pitch. If pitch value is not known, set this to inputWidth.

**5.25.2.5 uint32\_t NV\_ENC\_PIC\_PARAMS::encodePicFlags**

[in]: Specifies bit-wise OR'ed encode pic flags. See [NV\\_ENC\\_PIC\\_FLAGS](#) enum.

**5.25.2.6 uint32\_t NV\_ENC\_PIC\_PARAMS::frameIdx**

[in]: Specifies the frame index associated with the input frame [optional].

**5.25.2.7 uint64\_t NV\_ENC\_PIC\_PARAMS::inputTimeStamp**

[in]: Specifies presentation timestamp associated with the input picture.

**5.25.2.8 uint64\_t NV\_ENC\_PIC\_PARAMS::inputDuration**

[in]: Specifies duration of the input picture

**5.25.2.9 NV\_ENC\_INPUT\_PTR NV\_ENC\_PIC\_PARAMS::inputBuffer**

[in]: Specifies the input buffer pointer. Client must use a pointer obtained from [NvEncCreateInputBuffer\(\)](#) or [NvEncMapInputResource\(\)](#) APIs.

**5.25.2.10 NV\_ENC\_OUTPUT\_PTR NV\_ENC\_PIC\_PARAMS::outputBitstream**

[in]: Specifies the pointer to output buffer. Client should use a pointer obtained from [NvEncCreateBitstreamBuffer\(\)](#) API.

**5.25.2.11 void\* NV\_ENC\_PIC\_PARAMS::completionEvent**

[in]: Specifies an event to be signalled on completion of encoding of this Frame [only if operating in Asynchronous mode]. Each output buffer should be associated with a distinct event pointer.

**5.25.2.12 NV\_ENC\_BUFFER\_FORMAT NV\_ENC\_PIC\_PARAMS::bufferFmt**

[in]: Specifies the input buffer format.

**5.25.2.13 NV\_ENC\_PIC\_STRUCT NV\_ENC\_PIC\_PARAMS::pictureStruct**

[in]: Specifies structure of the input picture.

**5.25.2.14 NV\_ENC\_PIC\_TYPE NV\_ENC\_PIC\_PARAMS::pictureType**

[in]: Specifies input picture type. Client required to be set explicitly by the client if the client has not set NV\_ENC\_INITIALIZE\_PARAMS::enablePTD to 1 while calling NvInitializeEncoder.

**5.25.2.15 NV\_ENC\_CODEC\_PIC\_PARAMS NV\_ENC\_PIC\_PARAMS::codecPicParams**

[in]: Specifies the codec specific per-picture encoding parameters.

**5.25.2.16 NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE NV\_ENC\_PIC\_PARAMS::meHintCountsPerBlock[2]**

[in]: Specifies the number of hint candidates per block per direction for the current frame. meHintCountsPerBlock[0] is for L0 predictors and meHintCountsPerBlock[1] is for L1 predictors. The candidate count in [NV\\_ENC\\_PIC\\_PARAMS::meHintCountsPerBlock\[lx\]](#) must never exceed [NV\\_ENC\\_INITIALIZE\\_PARAMS::maxMEHintCountsPerBlock\[lx\]](#) provided during encoder initialization.

**5.25.2.17 NVENC\_EXTERNAL\_ME\_HINT\* NV\_ENC\_PIC\_PARAMS::meExternalHints**

[in]: Specifies the pointer to ME external hints for the current frame. The size of ME hint buffer should be equal to number of macroblocks \* the total number of candidates per macroblock. The total number of candidates per MB per direction =  $1 * \text{meHintCountsPerBlock}[Lx].\text{numCandsPerBlk16x16} + 2 * \text{meHintCountsPerBlock}[Lx].\text{numCandsPerBlk8x8} + 4 * \text{meHintCountsPerBlock}[Lx].\text{numCandsPerBlk4x4}$ . For frames using bidirectional ME, the total number of candidates for single macroblock is sum of total number of candidates per MB for each direction (L0 and L1)

**5.25.2.18 uint32\_t NV\_ENC\_PIC\_PARAMS::reserved1[6]**

[in]: Reserved and must be set to 0

**5.25.2.19 void\* NV\_ENC\_PIC\_PARAMS::reserved2[2]**

[in]: Reserved and must be set to NULL

**5.25.2.20 int8\_t\* NV\_ENC\_PIC\_PARAMS::qpDeltaMap**

[in]: Specifies the pointer to signed byte array containing QP delta value per MB in raster scan order in the current picture. This QP modifier is applied on top of the QP chosen by rate control.

**5.25.2.21 uint32\_t NV\_ENC\_PIC\_PARAMS::qpDeltaMapSize**

[in]: Specifies the size in bytes of qpDeltaMap surface allocated by client and pointed to by [NV\\_ENC\\_PIC\\_PARAMS::qpDeltaMap](#). Surface (array) should be picWidthInMbs \* picHeightInMbs

**5.25.2.22 uint32\_t NV\_ENC\_PIC\_PARAMS::reservedBitFields**

[in]: Reserved bitfields and must be set to 0

**5.25.2.23 uint16\_t NV\_ENC\_PIC\_PARAMS::meHintRefPicDist[2]**

[in]: Specifies temporal distance for reference picture ([NVENC\\_EXTERNAL\\_ME\\_HINT::refIdx](#) = 0) used during external ME with [NV\\_ENC\\_INITIALIZE\\_PARAMS::enablePTD](#) = 1. meHintRefPicDist[0] is for L0 hints and meHintRefPicDist[1] is for L1 hints. If not set, will internally infer distance of 1. Ignored for [NV\\_ENC\\_INITIALIZE\\_PARAMS::enablePTD](#) = 0

**5.25.2.24   uint32\_t NV\_ENC\_PIC\_PARAMS::reserved3[286]**

[in]: Reserved and must be set to 0

**5.25.2.25   void\* NV\_ENC\_PIC\_PARAMS::reserved4[60]**

[in]: Reserved and must be set to NULL

## 5.26 NV\_ENC\_PIC\_PARAMS\_H264 Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [displayPOCSyntax](#)
- uint32\_t [reserved3](#)
- uint32\_t [refPicFlag](#)
- uint32\_t [colourPlaneId](#)
- uint32\_t [forceIntraRefreshWithFrameCnt](#)
- uint32\_t [constrainedFrame](#):1
- uint32\_t [sliceModeDataUpdate](#):1
- uint32\_t [ltrMarkFrame](#):1
- uint32\_t [ltrUseFrames](#):1
- uint32\_t [reservedBitFields](#):28
- uint8\_t \* [sliceTypeData](#)
- uint32\_t [sliceTypeArrayCnt](#)
- uint32\_t [seiPayloadArrayCnt](#)
- NV\_ENC\_SEI\_PAYLOAD \* [seiPayloadArray](#)
- uint32\_t [sliceMode](#)
- uint32\_t [sliceModeData](#)
- uint32\_t [ltrMarkFrameIdx](#)
- uint32\_t [ltrUseFrameBitmap](#)
- uint32\_t [ltrUsageMode](#)
- uint32\_t [reserved](#) [243]
- void \* [reserved2](#) [62]

### 5.26.1 Detailed Description

\_NV\_ENC\_PIC\_PARAMS\_H264 H264 specific enc pic params. sent on a per frame basis.

### 5.26.2 Field Documentation

#### 5.26.2.1 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::displayPOCSyntax

[in]: Specifies the display POC syntax This is required to be set if client is handling the picture type decision.

#### 5.26.2.2 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::reserved3

[in]: Reserved and must be set to 0

#### 5.26.2.3 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::refPicFlag

[in]: Set to 1 for a reference picture. This is ignored if [NV\\_ENC\\_INITIALIZE\\_PARAMS::enablePTD](#) is set to 1.

**5.26.2.4 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::colourPlaneId**

[in]: Specifies the colour plane ID associated with the current input.

**5.26.2.5 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::forceIntraRefreshWithFrameCnt**

[in]: Forces an intra refresh with duration equal to intraRefreshFrameCnt. When outputRecoveryPointSEI is set this is value is used for recovery\_frame\_cnt in recovery point SEI message forceIntraRefreshWithFrameCnt cannot be used if B frames are used in the GOP structure specified

**5.26.2.6 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::constrainedFrame**

[in]: Set to 1 if client wants to encode this frame with each slice completely independent of other slices in the frame. NV\_ENC\_INITIALIZE\_PARAMS::enableConstrainedEncoding should be set to 1

**5.26.2.7 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::sliceModeDataUpdate**

[in]: Set to 1 if client wants to change the sliceModeData field to specify new sliceSize Parameter When forceIntraRefreshWithFrameCnt is set it will have priority over sliceMode setting

**5.26.2.8 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::ltrMarkFrame**

[in]: Set to 1 if client wants to mark this frame as LTR

**5.26.2.9 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::ltrUseFrames**

[in]: Set to 1 if client allows encoding this frame using the LTR frames specified in ltrFrameBitmap

**5.26.2.10 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::reservedBitFields**

[in]: Reserved bit fields and must be set to 0

**5.26.2.11 uint8\_t\* NV\_ENC\_PIC\_PARAMS\_H264::sliceTypeData**

[in]: Deprecated.

**5.26.2.12 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::sliceTypeArrayCnt**

[in]: Deprecated.

**5.26.2.13 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::seiPayloadArrayCnt**

[in]: Specifies the number of elements allocated in seiPayloadArray array.

**5.26.2.14 NV\_ENC\_SEI\_PAYLOAD\* NV\_ENC\_PIC\_PARAMS\_H264::seiPayloadArray**

[in]: Array of SEI payloads which will be inserted for this frame.

**5.26.2.15 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::sliceMode**

[in]: This parameter in conjunction with sliceModeData specifies the way in which the picture is divided into slices sliceMode = 0 MB based slices, sliceMode = 1 Byte based slices, sliceMode = 2 MB row based slices, sliceMode = 3, numSlices in Picture When forceIntraRefreshWithFrameCnt is set it will have priority over sliceMode setting When sliceMode == 0 and sliceModeData == 0 whole picture will be coded with one slice

**5.26.2.16 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::sliceModeData**

[in]: Specifies the parameter needed for sliceMode. For: sliceMode = 0, sliceModeData specifies # of MBs in each slice (except last slice) sliceMode = 1, sliceModeData specifies maximum # of bytes in each slice (except last slice) sliceMode = 2, sliceModeData specifies # of MB rows in each slice (except last slice) sliceMode = 3, sliceModeData specifies number of slices in the picture. Driver will divide picture into slices optimally

**5.26.2.17 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::ltrMarkFrameIdx**

[in]: Specifies the long term referenceframe index to use for marking this frame as LTR.

**5.26.2.18 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::ltrUseFrameBitmap**

[in]: Specifies the the associated bitmap of LTR frame indices to use when encoding this frame.

**5.26.2.19 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::ltrUsageMode**

[in]: Not supported. Reserved for future use and must be set to 0.

**5.26.2.20 uint32\_t NV\_ENC\_PIC\_PARAMS\_H264::reserved[243]**

[in]: Reserved and must be set to 0.

**5.26.2.21 void\* NV\_ENC\_PIC\_PARAMS\_H264::reserved2[62]**

[in]: Reserved and must be set to NULL.

## 5.27 NV\_ENC\_PIC\_PARAMS\_HEVC Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [displayPOCSyntax](#)
- uint32\_t [refPicFlag](#)
- uint32\_t [temporalId](#)
- uint32\_t [forceIntraRefreshWithFrameCnt](#)
- uint32\_t [constrainedFrame](#):1
- uint32\_t [sliceModeDataUpdate](#):1
- uint32\_t [ltrMarkFrame](#):1
- uint32\_t [ltrUseFrames](#):1
- uint32\_t [reservedBitFields](#):28
- uint8\_t \* [sliceTypeData](#)
- uint32\_t [sliceTypeArrayCnt](#)
- uint32\_t [sliceMode](#)
- uint32\_t [sliceModeData](#)
- uint32\_t [ltrMarkFrameIdx](#)
- uint32\_t [ltrUseFrameBitmap](#)
- uint32\_t [ltrUsageMode](#)
- uint32\_t [seiPayloadArrayCnt](#)
- uint32\_t [reserved](#)
- [NV\\_ENC\\_SEI\\_PAYLOAD](#) \* [seiPayloadArray](#)
- uint32\_t [reserved2](#) [244]
- void \* [reserved3](#) [61]

### 5.27.1 Detailed Description

[\\_NV\\_ENC\\_PIC\\_PARAMS\\_HEVC](#) HEVC specific enc pic params. sent on a per frame basis.

### 5.27.2 Field Documentation

#### 5.27.2.1 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::displayPOCSyntax

[in]: Specifies the display POC syntax This is required to be set if client is handling the picture type decision.

#### 5.27.2.2 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::refPicFlag

[in]: Set to 1 for a reference picture. This is ignored if [NV\\_ENC\\_INITIALIZE\\_PARAMS::enablePTD](#) is set to 1.

#### 5.27.2.3 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::temporalId

[in]: Specifies the temporal id of the picture



**5.27.2.4 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::forceIntraRefreshWithFrameCnt**

[in]: Forces an intra refresh with duration equal to intraRefreshFrameCnt. When outputRecoveryPointSEI is set this is value is used for recovery\_frame\_cnt in recovery point SEI message forceIntraRefreshWithFrameCnt cannot be used if B frames are used in the GOP structure specified

**5.27.2.5 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::constrainedFrame**

[in]: Set to 1 if client wants to encode this frame with each slice completely independent of other slices in the frame. NV\_ENC\_INITIALIZE\_PARAMS::enableConstrainedEncoding should be set to 1

**5.27.2.6 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::sliceModeDataUpdate**

[in]: Set to 1 if client wants to change the sliceModeData field to specify new sliceSize Parameter When forceIntraRefreshWithFrameCnt is set it will have priority over sliceMode setting

**5.27.2.7 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::ltrMarkFrame**

[in]: Set to 1 if client wants to mark this frame as LTR

**5.27.2.8 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::ltrUseFrames**

[in]: Set to 1 if client allows encoding this frame using the LTR frames specified in ltrFrameBitmap

**5.27.2.9 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::reservedBitFields**

[in]: Reserved bit fields and must be set to 0

**5.27.2.10 uint8\_t\* NV\_ENC\_PIC\_PARAMS\_HEVC::sliceTypeData**

[in]: Array which specifies the slice type used to force intra slice for a particular slice. Currently supported only for [NV\\_ENC\\_CONFIG\\_H264::sliceMode == 3](#). Client should allocate array of size sliceModeData where sliceModeData is specified in field of \_NV\_ENC\_CONFIG\_H264 Array element with index n corresponds to nth slice. To force a particular slice to intra client should set corresponding array element to NV\_ENC\_SLICE\_TYPE\_I all other array elements should be set to NV\_ENC\_SLICE\_TYPE\_DEFAULT

**5.27.2.11 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::sliceTypeArrayCnt**

[in]: Client should set this to the number of elements allocated in sliceTypeData array. If sliceTypeData is NULL then this should be set to 0

**5.27.2.12 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::sliceMode**

[in]: This parameter in conjunction with sliceModeData specifies the way in which the picture is divided into slices sliceMode = 0 CTU based slices, sliceMode = 1 Byte based slices, sliceMode = 2 CTU row based slices, sliceMode = 3, numSlices in Picture When forceIntraRefreshWithFrameCnt is set it will have priority over sliceMode setting When sliceMode == 0 and sliceModeData == 0 whole picture will be coded with one slice

**5.27.2.13 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::sliceModeData**

[in]: Specifies the parameter needed for sliceMode. For: sliceMode = 0, sliceModeData specifies # of CTUs in each slice (except last slice) sliceMode = 1, sliceModeData specifies maximum # of bytes in each slice (except last slice) sliceMode = 2, sliceModeData specifies # of CTU rows in each slice (except last slice) sliceMode = 3, sliceModeData specifies number of slices in the picture. Driver will divide picture into slices optimally

**5.27.2.14 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::ltrMarkFrameIdx**

[in]: Specifies the long term reference frame index to use for marking this frame as LTR.

**5.27.2.15 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::ltrUseFrameBitmap**

[in]: Specifies the associated bitmap of LTR frame indices to use when encoding this frame.

**5.27.2.16 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::ltrUsageMode**

[in]: Not supported. Reserved for future use and must be set to 0.

**5.27.2.17 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::seiPayloadArrayCnt**

[in]: Specifies the number of elements allocated in seiPayloadArray array.

**5.27.2.18 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::reserved**

[in]: Reserved and must be set to 0.

**5.27.2.19 NV\_ENC\_SEI\_PAYLOAD\* NV\_ENC\_PIC\_PARAMS\_HEVC::seiPayloadArray**

[in]: Array of SEI payloads which will be inserted for this frame.

**5.27.2.20 uint32\_t NV\_ENC\_PIC\_PARAMS\_HEVC::reserved2[244]**

[in]: Reserved and must be set to 0.

**5.27.2.21 void\* NV\_ENC\_PIC\_PARAMS\_HEVC::reserved3[61]**

[in]: Reserved and must be set to NULL.

## 5.28 NV\_ENC\_PRESET\_CONFIG Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- `uint32_t` [version](#)
- `NV_ENC_CONFIG` [presetCfg](#)
- `uint32_t` [reserved1](#) [255]
- `void *` [reserved2](#) [64]

### 5.28.1 Detailed Description

`_NV_ENC_PRESET_CONFIG` Encoder preset config

### 5.28.2 Field Documentation

#### 5.28.2.1 `uint32_t NV_ENC_PRESET_CONFIG::version`

[in]: Struct version. Must be set to [NV\\_ENC\\_PRESET\\_CONFIG\\_VER](#).

#### 5.28.2.2 `NV_ENC_CONFIG NV_ENC_PRESET_CONFIG::presetCfg`

[out]: preset config returned by the Nvidia Video Encoder interface.

#### 5.28.2.3 `uint32_t NV_ENC_PRESET_CONFIG::reserved1[255]`

[in]: Reserved and must be set to 0

#### 5.28.2.4 `void* NV_ENC_PRESET_CONFIG::reserved2[64]`

[in]: Reserved and must be set to NULL

## 5.29 NV\_ENC\_QP Struct Reference

```
#include <nvEncodeAPI.h>
```

### 5.29.1 Detailed Description

QP value for frames

## 5.30 NV\_ENC\_RC\_PARAMS Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- [NV\\_ENC\\_PARAMS\\_RC\\_MODE](#) rateControlMode
- [NV\\_ENC\\_QP](#) constQP
- [uint32\\_t](#) averageBitRate
- [uint32\\_t](#) maxBitRate
- [uint32\\_t](#) vbvBufferSize
- [uint32\\_t](#) vbvInitialDelay
- [uint32\\_t](#) enableMinQP:1
- [uint32\\_t](#) enableMaxQP:1
- [uint32\\_t](#) enableInitialRCQP:1
- [uint32\\_t](#) enableAQ:1
- [uint32\\_t](#) enableExtQPDeltaMap:1
- [uint32\\_t](#) enableLookahead:1
- [uint32\\_t](#) disableIadapt:1
- [uint32\\_t](#) disableBadapt:1
- [uint32\\_t](#) enableTemporalAQ:1
- [uint32\\_t](#) zeroReorderDelay:1
- [uint32\\_t](#) enableNonRefP:1
- [uint32\\_t](#) strictGOPTarget:1
- [uint32\\_t](#) aqStrength:4
- [uint32\\_t](#) reservedBitFields:16
- [NV\\_ENC\\_QP](#) minQP
- [NV\\_ENC\\_QP](#) maxQP
- [NV\\_ENC\\_QP](#) initialRCQP
- [uint32\\_t](#) temporallayerIdxMask
- [uint8\\_t](#) temporalLayerQP [8]
- [uint8\\_t](#) targetQuality
- [uint8\\_t](#) targetQualityLSB
- [uint16\\_t](#) lookaheadDepth

### 5.30.1 Detailed Description

Rate Control Configuration Paramters

### 5.30.2 Field Documentation

#### 5.30.2.1 NV\_ENC\_PARAMS\_RC\_MODE NV\_ENC\_RC\_PARAMS::rateControlMode

[in]: Specifies the rate control mode. Check support for various rate control modes using [NV\\_ENC\\_CAPS\\_SUPPORTED\\_RATECONTROL\\_MODES](#) caps.

#### 5.30.2.2 NV\_ENC\_QP NV\_ENC\_RC\_PARAMS::constQP

[in]: Specifies the initial QP to be used for encoding, these values would be used for all frames if in Constant QP mode.

**5.30.2.3 uint32\_t NV\_ENC\_RC\_PARAMS::averageBitRate**

[in]: Specifies the average bitrate(in bits/sec) used for encoding.

**5.30.2.4 uint32\_t NV\_ENC\_RC\_PARAMS::maxBitRate**

[in]: Specifies the maximum bitrate for the encoded output. This is used for VBR and ignored for CBR mode.

**5.30.2.5 uint32\_t NV\_ENC\_RC\_PARAMS::vbvBufferSize**

[in]: Specifies the VBV(HRD) buffer size. in bits. Set 0 to use the default VBV buffer size.

**5.30.2.6 uint32\_t NV\_ENC\_RC\_PARAMS::vbvInitialDelay**

[in]: Specifies the VBV(HRD) initial delay in bits. Set 0 to use the default VBV initial delay .

**5.30.2.7 uint32\_t NV\_ENC\_RC\_PARAMS::enableMinQP**

[in]: Set this to 1 if minimum QP used for rate control.

**5.30.2.8 uint32\_t NV\_ENC\_RC\_PARAMS::enableMaxQP**

[in]: Set this to 1 if maximum QP used for rate control.

**5.30.2.9 uint32\_t NV\_ENC\_RC\_PARAMS::enableInitialRCQP**

[in]: Set this to 1 if user supplied initial QP is used for rate control.

**5.30.2.10 uint32\_t NV\_ENC\_RC\_PARAMS::enableAQ**

[in]: Set this to 1 to enable adaptive quantization (Spatial).

**5.30.2.11 uint32\_t NV\_ENC\_RC\_PARAMS::enableExtQPDeltaMap**

[in]: Set this to 1 to enable additional QP modifier for each MB supplied by client though signed byte array pointed to by [NV\\_ENC\\_PIC\\_PARAMS::qpDeltaMap](#) (Not Supported when AQ(Spatial/Temporal) is enabled)

**5.30.2.12 uint32\_t NV\_ENC\_RC\_PARAMS::enableLookahead**

[in]: Set this to 1 to enable lookahead with depth <lookaheadDepth> (if lookahead is enabled, input frames must remain available to the encoder until encode completion)

**5.30.2.13 uint32\_t NV\_ENC\_RC\_PARAMS::disableIadapt**

[in]: Set this to 1 to disable adaptive I-frame insertion at scene cuts (only has an effect when lookahead is enabled)

**5.30.2.14 uint32\_t NV\_ENC\_RC\_PARAMS::disableBadapt**

[in]: Set this to 1 to disable adaptive B-frame decision (only has an effect when lookahead is enabled)

**5.30.2.15 uint32\_t NV\_ENC\_RC\_PARAMS::enableTemporalAQ**

[in]: Set this to 1 to enable temporal AQ for H.264

**5.30.2.16 uint32\_t NV\_ENC\_RC\_PARAMS::zeroReorderDelay**

[in]: Set this to 1 to indicate zero latency operation (no reordering delay, num\_reorder\_frames=0)

**5.30.2.17 uint32\_t NV\_ENC\_RC\_PARAMS::enableNonRefP**

[in]: Set this to 1 to enable automatic insertion of non-reference P-frames (no effect if enablePTD=0)

**5.30.2.18 uint32\_t NV\_ENC\_RC\_PARAMS::strictGOPTarget**

[in]: Set this to 1 to minimize GOP-to-GOP rate fluctuations

**5.30.2.19 uint32\_t NV\_ENC\_RC\_PARAMS::aqStrength**

[in]: When AQ (Spatial) is enabled (i.e. [NV\\_ENC\\_RC\\_PARAMS::enableAQ](#) is set), this field is used to specify AQ strength. AQ strength scale is from 1 (low) - 15 (aggressive). If not set, strength is autoselected by driver.

**5.30.2.20 uint32\_t NV\_ENC\_RC\_PARAMS::reservedBitFields**

[in]: Reserved bitfields and must be set to 0

**5.30.2.21 NV\_ENC\_QP NV\_ENC\_RC\_PARAMS::minQP**

[in]: Specifies the minimum QP used for rate control. Client must set NV\_ENC\_CONFIG::enableMinQP to 1.

**5.30.2.22 NV\_ENC\_QP NV\_ENC\_RC\_PARAMS::maxQP**

[in]: Specifies the maximum QP used for rate control. Client must set NV\_ENC\_CONFIG::enableMaxQP to 1.

**5.30.2.23 NV\_ENC\_QP NV\_ENC\_RC\_PARAMS::initialRCQP**

[in]: Specifies the initial QP used for rate control. Client must set NV\_ENC\_CONFIG::enableInitialRCQP to 1.

**5.30.2.24 uint32\_t NV\_ENC\_RC\_PARAMS::temporallayerIdxMask**

[in]: Specifies the temporal layers (as a bitmask) whose QPs have changed. Valid max bitmask is  $[2^{\text{NV\_ENC\_CAPS\_NUM\_MAX\_TEMPORAL\_LAYERS}} - 1]$

**5.30.2.25    `uint8_t NV_ENC_RC_PARAMS::temporalLayerQP[8]`**

[in]: Specifies the temporal layer QPs used for rate control. Temporal layer index is used as the array index

**5.30.2.26    `uint8_t NV_ENC_RC_PARAMS::targetQuality`**

[in]: Target CQ (Constant Quality) level for VBR mode (range 0-51 with 0-automatic)

**5.30.2.27    `uint8_t NV_ENC_RC_PARAMS::targetQualityLSB`**

[in]: Fractional part of target quality (as 8.8 fixed point format)

**5.30.2.28    `uint16_t NV_ENC_RC_PARAMS::lookaheadDepth`**

[in]: Maximum depth of lookahead with range 0-32 (only used if enableLookahead=1)



## 5.31 NV\_ENC\_RECONFIGURE\_PARAMS Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- [uint32\\_t version](#)
- [NV\\_ENC\\_INITIALIZE\\_PARAMS reInitEncodeParams](#)
- [uint32\\_t resetEncoder:1](#)
- [uint32\\_t forceIDR:1](#)

### 5.31.1 Detailed Description

`_NV_ENC_RECONFIGURE_PARAMS` Encode Session Reconfigured parameters.

### 5.31.2 Field Documentation

#### 5.31.2.1 `uint32_t NV_ENC_RECONFIGURE_PARAMS::version`

[in]: Struct version. Must be set to [NV\\_ENC\\_RECONFIGURE\\_PARAMS\\_VER](#).

#### 5.31.2.2 `NV_ENC_INITIALIZE_PARAMS NV_ENC_RECONFIGURE_PARAMS::reInitEncodeParams`

[in]: Encoder session re-initialization parameters.

#### 5.31.2.3 `uint32_t NV_ENC_RECONFIGURE_PARAMS::resetEncoder`

[in]: This resets the rate control states and other internal encoder states. This should be used only with an IDR frame. If [NV\\_ENC\\_INITIALIZE\\_PARAMS::enablePTD](#) is set to 1, encoder will force the frame type to IDR

#### 5.31.2.4 `uint32_t NV_ENC_RECONFIGURE_PARAMS::forceIDR`

[in]: Encode the current picture as an IDR picture. This flag is only valid when Picture type decision is taken by the Encoder [`_NV_ENC_INITIALIZE_PARAMS::enablePTD == 1`].

## 5.32 NV\_ENC\_REGISTER\_RESOURCE Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- `uint32_t` [version](#)
- `NV_ENC_INPUT_RESOURCE_TYPE` [resourceType](#)
- `uint32_t` [width](#)
- `uint32_t` [height](#)
- `uint32_t` [pitch](#)
- `uint32_t` [subResourceIndex](#)
- `void *` [resourceToRegister](#)
- `NV_ENC_REGISTERED_PTR` [registeredResource](#)
- `NV_ENC_BUFFER_FORMAT` [bufferFormat](#)
- `uint32_t` [reserved1](#) [248]
- `void *` [reserved2](#) [62]

### 5.32.1 Detailed Description

`_NV_ENC_REGISTER_RESOURCE` Register a resource for future use with the Nvidia Video Encoder Interface.

### 5.32.2 Field Documentation

#### 5.32.2.1 `uint32_t NV_ENC_REGISTER_RESOURCE::version`

[in]: Struct version. Must be set to [NV\\_ENC\\_REGISTER\\_RESOURCE\\_VER](#).

#### 5.32.2.2 `NV_ENC_INPUT_RESOURCE_TYPE NV_ENC_REGISTER_RESOURCE::resourceType`

[in]: Specifies the type of resource to be registered. Supported values are [NV\\_ENC\\_INPUT\\_RESOURCE\\_TYPE\\_DIRECTX](#), [NV\\_ENC\\_INPUT\\_RESOURCE\\_TYPE\\_CUDADEVICEPTR](#), [NV\\_ENC\\_INPUT\\_RESOURCE\\_TYPE\\_OPENGL\\_TEX](#)

#### 5.32.2.3 `uint32_t NV_ENC_REGISTER_RESOURCE::width`

[in]: Input buffer Width.

#### 5.32.2.4 `uint32_t NV_ENC_REGISTER_RESOURCE::height`

[in]: Input buffer Height.

#### 5.32.2.5 `uint32_t NV_ENC_REGISTER_RESOURCE::pitch`

[in]: Input buffer Pitch.

**5.32.2.6 uint32\_t NV\_ENC\_REGISTER\_RESOURCE::subResourceIndex**

[in]: Subresource Index of the DirectX resource to be registered. Should be set to 0 for other interfaces.

**5.32.2.7 void\* NV\_ENC\_REGISTER\_RESOURCE::resourceToRegister**

[in]: Handle to the resource that is being registered.

**5.32.2.8 NV\_ENC\_REGISTERED\_PTR NV\_ENC\_REGISTER\_RESOURCE::registeredResource**

[out]: Registered resource handle. This should be used in future interactions with the Nvidia Video Encoder Interface.

**5.32.2.9 NV\_ENC\_BUFFER\_FORMAT NV\_ENC\_REGISTER\_RESOURCE::bufferFormat**

[in]: Buffer format of resource to be registered.

**5.32.2.10 uint32\_t NV\_ENC\_REGISTER\_RESOURCE::reserved1[248]**

[in]: Reserved and must be set to 0.

**5.32.2.11 void\* NV\_ENC\_REGISTER\_RESOURCE::reserved2[62]**

[in]: Reserved and must be set to NULL.

## 5.33 NV\_ENC\_SEI\_PAYLOAD Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- `uint32_t` [payloadSize](#)
- `uint32_t` [payloadType](#)
- `uint8_t *` [payload](#)

### 5.33.1 Detailed Description

`_NV_ENC_SEI_PAYLOAD` User SEI message

### 5.33.2 Field Documentation

#### 5.33.2.1 `uint32_t NV_ENC_SEI_PAYLOAD::payloadSize`

[in] SEI payload size in bytes. SEI payload must be byte aligned, as described in Annex D

#### 5.33.2.2 `uint32_t NV_ENC_SEI_PAYLOAD::payloadType`

[in] SEI payload types and syntax can be found in Annex D of the H.264 Specification.

#### 5.33.2.3 `uint8_t* NV_ENC_SEI_PAYLOAD::payload`

[in] pointer to user data

## 5.34 NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [version](#)
- uint32\_t [inBufferSize](#)
- uint32\_t [spsId](#)
- uint32\_t [ppsId](#)
- void \* [spsppsBuffer](#)
- uint32\_t \* [outSPSPSPayloadSize](#)
- uint32\_t [reserved](#) [250]
- void \* [reserved2](#) [64]

### 5.34.1 Detailed Description

\_NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD Sequence and picture paramaters payload.

### 5.34.2 Field Documentation

#### 5.34.2.1 uint32\_t NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD::version

[in]: Struct version. Must be set to [NV\\_ENC\\_INITIALIZE\\_PARAMS\\_VER](#).

#### 5.34.2.2 uint32\_t NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD::inBufferSize

[in]: Specifies the size of the spsppsBuffer provied by the client

#### 5.34.2.3 uint32\_t NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD::spsId

[in]: Specifies the SPS id to be used in sequence header. Default value is 0.

#### 5.34.2.4 uint32\_t NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD::ppsId

[in]: Specifies the PPS id to be used in picture header. Default value is 0.

#### 5.34.2.5 void\* NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD::spsppsBuffer

[in]: Specifies bitstream header pointer of size [NV\\_ENC\\_SEQUENCE\\_PARAM\\_PAYLOAD::inBufferSize](#). It is the client's responsibility to manage this memory.

#### 5.34.2.6 uint32\_t\* NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD::outSPSPSPayloadSize

[out]: Size of the sequence and picture header in bytes written by the NvEncodeAPI interface to the SPSPSPBuffer.

**5.34.2.7    `uint32_t NV_ENC_SEQUENCE_PARAM_PAYLOAD::reserved[250]`**

[in]: Reserved and must be set to 0

**5.34.2.8    `void* NV_ENC_SEQUENCE_PARAM_PAYLOAD::reserved2[64]`**

[in]: Reserved and must be set to NULL

## 5.35 NV\_ENC\_STAT Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [version](#)
- uint32\_t [reserved](#)
- NV\_ENC\_OUTPUT\_PTR [outputBitStream](#)
- uint32\_t [bitStreamSize](#)
- uint32\_t [picType](#)
- uint32\_t [lastValidByteOffset](#)
- uint32\_t [sliceOffsets](#) [16]
- uint32\_t [picIdx](#)
- uint32\_t [reserved1](#) [233]
- void \* [reserved2](#) [64]

### 5.35.1 Detailed Description

\_NV\_ENC\_STAT Encode Stats structure.

### 5.35.2 Field Documentation

#### 5.35.2.1 uint32\_t NV\_ENC\_STAT::version

[in]: Struct version. Must be set to [NV\\_ENC\\_STAT\\_VER](#).

#### 5.35.2.2 uint32\_t NV\_ENC\_STAT::reserved

[in]: Reserved and must be set to 0

#### 5.35.2.3 NV\_ENC\_OUTPUT\_PTR NV\_ENC\_STAT::outputBitStream

[out]: Specifies the pointer to output bitstream.

#### 5.35.2.4 uint32\_t NV\_ENC\_STAT::bitStreamSize

[out]: Size of generated bitstream in bytes.

#### 5.35.2.5 uint32\_t NV\_ENC\_STAT::picType

[out]: Picture type of encoded picture. See [NV\\_ENC\\_PIC\\_TYPE](#).

#### 5.35.2.6 uint32\_t NV\_ENC\_STAT::lastValidByteOffset

[out]: Offset of last valid bytes of completed bitstream

**5.35.2.7    uint32\_t NV\_ENC\_STAT::sliceOffsets[16]**

[out]: Offsets of each slice

**5.35.2.8    uint32\_t NV\_ENC\_STAT::picIdx**

[out]: Picture number

**5.35.2.9    uint32\_t NV\_ENC\_STAT::reserved1[233]**

[in]: Reserved and must be set to 0

**5.35.2.10   void\* NV\_ENC\_STAT::reserved2[64]**

[in]: Reserved and must be set to NULL



## 5.36 NV\_ENCODE\_API\_FUNCTION\_LIST Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [version](#)
- uint32\_t [reserved](#)
- PNVENCODESESSION [nvEncOpenEncodeSession](#)
- PNVEGETENCODEGUIDCOUNT [nvEncGetEncodeGUIDCount](#)
- PNVEGETENCODEPRESETCOUNT [nvEncGetEncodeProfileGUIDCount](#)
- PNVEGETENCODEPRESETGUIDS [nvEncGetEncodeProfileGUIDs](#)
- PNVEGETENCODEGUIDS [nvEncGetEncodeGUIDs](#)
- PNVEGETINPUTFORMATCOUNT [nvEncGetInputFormatCount](#)
- PNVEGETINPUTFORMATS [nvEncGetInputFormats](#)
- PNVEGETENCODECAPS [nvEncGetEncodeCaps](#)
- PNVEGETENCODEPRESETCOUNT [nvEncGetEncodePresetCount](#)
- PNVEGETENCODEPRESETGUIDS [nvEncGetEncodePresetGUIDs](#)
- PNVEGETENCODEPRESETCONFIG [nvEncGetEncodePresetConfig](#)
- PNVEINITIALIZEENCODER [nvEncInitializeEncoder](#)
- PNVECCREATEINPUTBUFFER [nvEncCreateInputBuffer](#)
- PNVECDestroyINPUTBUFFER [nvEncDestroyInputBuffer](#)
- PNVECCREATEBITSTREAMBUFFER [nvEncCreateBitstreamBuffer](#)
- PNVECDestroyBITSTREAMBUFFER [nvEncDestroyBitstreamBuffer](#)
- PNVEENCODEPICTURE [nvEncEncodePicture](#)
- PNVELOCKBITSTREAM [nvEncLockBitstream](#)
- PNVEUNLOCKBITSTREAM [nvEncUnlockBitstream](#)
- PNVELOCKINPUTBUFFER [nvEncLockInputBuffer](#)
- PNVEUNLOCKINPUTBUFFER [nvEncUnlockInputBuffer](#)
- PNVEGETENCODESTATS [nvEncGetEncodeStats](#)
- PNVEGETSEQUENCEPARAMS [nvEncGetSequenceParams](#)
- PNVEREGISTERASYNC EVENT [nvEncRegisterAsyncEvent](#)
- PNVEUNREGISTERASYNC EVENT [nvEncUnregisterAsyncEvent](#)
- PNVECMAPINPUTRESOURCE [nvEncMapInputResource](#)
- PNVEUNMAPINPUTRESOURCE [nvEncUnmapInputResource](#)
- PNVECDestroyENCODER [nvEncDestroyEncoder](#)
- PNVEINVALIDATEREF FRAMES [nvEncInvalidateRefFrames](#)
- PNVEOPENENCODESESSIONEX [nvEncOpenEncodeSessionEx](#)
- PNVEREGISTERRESOURCE [nvEncRegisterResource](#)
- PNVEUNREGISTERRESOURCE [nvEncUnregisterResource](#)
- PNVECRECONFIGUREENCODER [nvEncReconfigureEncoder](#)
- PNVECCREATEMVBUFFER [nvEncCreateMVBuffer](#)
- PNVECDestroyMVBUFFER [nvEncDestroyMVBuffer](#)
- PNVECRUNMOTIONESTIMATIONONLY [nvEncRunMotionEstimationOnly](#)
- void \* [reserved2](#) [281]

### 5.36.1 Detailed Description

[NV\\_ENCODE\\_API\\_FUNCTION\\_LIST](#)

## 5.36.2 Field Documentation

### 5.36.2.1 uint32\_t NV\_ENCODE\_API\_FUNCTION\_LIST::version

[in]: Client should pass NV\_ENCODE\_API\_FUNCTION\_LIST\_VER.

### 5.36.2.2 uint32\_t NV\_ENCODE\_API\_FUNCTION\_LIST::reserved

[in]: Reserved and should be set to 0.

### 5.36.2.3 PNVEOPENENCODESESSION NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncOpenEncodeSession

[out]: Client should access [NvEncOpenEncodeSession\(\)](#) API through this pointer.

### 5.36.2.4 PNVEGETENCODEGUIDCOUNT NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncGetEncodeGUIDCount

[out]: Client should access [NvEncGetEncodeGUIDCount\(\)](#) API through this pointer.

### 5.36.2.5 PNVEGETENCODEPRESETCOUNT NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncGetEncodeProfileGUIDCount

[out]: Client should access [NvEncGetEncodeProfileGUIDCount\(\)](#) API through this pointer.

### 5.36.2.6 PNVEGETENCODEPRESETGUIDS NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncGetEncodeProfileGUIDs

[out]: Client should access [NvEncGetEncodeProfileGUIDs\(\)](#) API through this pointer.

### 5.36.2.7 PNVEGETENCODEGUIDS NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncGetEncodeGUIDs

[out]: Client should access [NvEncGetEncodeGUIDs\(\)](#) API through this pointer.

### 5.36.2.8 PNVEGETINPUTFORMATCOUNT NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncGetInputFormatCount

[out]: Client should access [NvEncGetInputFormatCount\(\)](#) API through this pointer.

### 5.36.2.9 PNVEGETINPUTFORMATS NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncGetInputFormats

[out]: Client should access [NvEncGetInputFormats\(\)](#) API through this pointer.

### 5.36.2.10 PNVEGETENCODECAPS NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncGetEncodeCaps

[out]: Client should access [NvEncGetEncodeCaps\(\)](#) API through this pointer.

**5.36.2.11 PNVE NCGETENCODEPRESETCOUNT NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncGetEncodePresetCount**

[out]: Client should access [NvEncGetEncodePresetCount\(\)](#) API through this pointer.

**5.36.2.12 PNVE NCGETENCODEPRESETGUIDS NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncGetEncodePresetGUIDs**

[out]: Client should access [NvEncGetEncodePresetGUIDs\(\)](#) API through this pointer.

**5.36.2.13 PNVE NCGETENCODEPRESETCONFIG NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncGetEncodePresetConfig**

[out]: Client should access [NvEncGetEncodePresetConfig\(\)](#) API through this pointer.

**5.36.2.14 PNVE NCINITIALIZEENCODER NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncInitializeEncoder**

[out]: Client should access [NvEncInitializeEncoder\(\)](#) API through this pointer.

**5.36.2.15 PNVE NC CREATEINPUTBUFFER NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncCreateInputBuffer**

[out]: Client should access [NvEncCreateInputBuffer\(\)](#) API through this pointer.

**5.36.2.16 PNVE NCDESTROYINPUTBUFFER NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncDestroyInputBuffer**

[out]: Client should access [NvEncDestroyInputBuffer\(\)](#) API through this pointer.

**5.36.2.17 PNVE NC CREATEBITSTREAMBUFFER NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncCreateBitstreamBuffer**

[out]: Client should access [NvEncCreateBitstreamBuffer\(\)](#) API through this pointer.

**5.36.2.18 PNVE NCDESTROYBITSTREAMBUFFER NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncDestroyBitstreamBuffer**

[out]: Client should access [NvEncDestroyBitstreamBuffer\(\)](#) API through this pointer.

**5.36.2.19 PNVE NC ENCODEPICTURE NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncEncodePicture**

[out]: Client should access [NvEncEncodePicture\(\)](#) API through this pointer.

**5.36.2.20 PNVE NC LOCKBITSTREAM NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncLockBitstream**

[out]: Client should access [NvEncLockBitstream\(\)](#) API through this pointer.

**5.36.2.21 PNVENCUNLOCKBITSTREAM NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncUnlockBitstream**

[out]: Client should access [NvEncUnlockBitstream\(\)](#) API through this pointer.

**5.36.2.22 PNVENCLOCKINPUTBUFFER NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncLockInputBuffer**

[out]: Client should access [NvEncLockInputBuffer\(\)](#) API through this pointer.

**5.36.2.23 PNVENCUNLOCKINPUTBUFFER NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncUnlockInputBuffer**

[out]: Client should access [NvEncUnlockInputBuffer\(\)](#) API through this pointer.

**5.36.2.24 PNVENCGETENCODESTATS NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncGetEncodeStats**

[out]: Client should access [NvEncGetEncodeStats\(\)](#) API through this pointer.

**5.36.2.25 PNVENCGETSEQUENCEPARAMS NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncGetSequenceParams**

[out]: Client should access [NvEncGetSequenceParams\(\)](#) API through this pointer.

**5.36.2.26 PNVENCREGISTERASYNCEVENT NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncRegisterAsyncEvent**

[out]: Client should access [NvEncRegisterAsyncEvent\(\)](#) API through this pointer.

**5.36.2.27 PNVENCUNREGISTERASYNCEVENT NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncUnregisterAsyncEvent**

[out]: Client should access [NvEncUnregisterAsyncEvent\(\)](#) API through this pointer.

**5.36.2.28 PNVENCMAPINPUTRESOURCE NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncMapInputResource**

[out]: Client should access [NvEncMapInputResource\(\)](#) API through this pointer.

**5.36.2.29 PNVENCUNMAPINPUTRESOURCE NV\_ENCODE\_API\_FUNCTION\_-  
LIST::nvEncUnmapInputResource**

[out]: Client should access [NvEncUnmapInputResource\(\)](#) API through this pointer.

**5.36.2.30 PNVENCDESTROYENCODER NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncDestroyEncoder**

[out]: Client should access [NvEncDestroyEncoder\(\)](#) API through this pointer.

**5.36.2.31 PNVENCINVALIDATEREFFRAMES NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncInvalidateRefFrames**

[out]: Client should access [NvEncInvalidateRefFrames\(\)](#) API through this pointer.

**5.36.2.32 PNVENCOPENENCODESESSIONEX NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncOpenEncodeSessionEx**

[out]: Client should access [NvEncOpenEncodeSession\(\)](#) API through this pointer.

**5.36.2.33 PNVENCREGISTERRESOURCE NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncRegisterResource**

[out]: Client should access [NvEncRegisterResource\(\)](#) API through this pointer.

**5.36.2.34 PNVENCUNREGISTERRESOURCE NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncUnregisterResource**

[out]: Client should access [NvEncUnregisterResource\(\)](#) API through this pointer.

**5.36.2.35 PNVENCRECONFIGUREENCODER NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncReconfigureEncoder**

[out]: Client should access [NvEncReconfigureEncoder\(\)](#) API through this pointer.

**5.36.2.36 PNVENC\_CREATE\_MV\_BUFFER NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncCreateMVBuffer**

[out]: Client should access [NvEncCreateMVBuffer](#) API through this pointer.

**5.36.2.37 PNVENC\_DESTROY\_MV\_BUFFER NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncDestroyMVBuffer**

[out]: Client should access [NvEncDestroyMVBuffer](#) API through this pointer.

**5.36.2.38 PNVENC\_RUN\_MOTION\_ESTIMATION\_ONLY NV\_ENCODE\_API\_FUNCTION\_LIST::nvEncRunMotionEstimationOnly**

[out]: Client should access [NvEncRunMotionEstimationOnly](#) API through this pointer.

**5.36.2.39 void\* NV\_ENCODE\_API\_FUNCTION\_LIST::reserved2[281]**

[in]: Reserved and must be set to NULL

## 5.37 NVENC\_EXTERNAL\_ME\_HINT Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- `int32_t mvx`: 12
- `int32_t mvy`: 10
- `int32_t refidx`: 5
- `int32_t dir`: 1
- `int32_t partType`: 2
- `int32_t lastofPart`: 1
- `int32_t lastOfMB`: 1

### 5.37.1 Detailed Description

`_NVENC_EXTERNAL_ME_HINT` External Motion Vector hint structure.

### 5.37.2 Field Documentation

#### 5.37.2.1 `int32_t NVENC_EXTERNAL_ME_HINT::mvx`

[in]: Specifies the x component of integer pixel MV (relative to current MB) S12.0.

#### 5.37.2.2 `int32_t NVENC_EXTERNAL_ME_HINT::mvy`

[in]: Specifies the y component of integer pixel MV (relative to current MB) S10.0 .

#### 5.37.2.3 `int32_t NVENC_EXTERNAL_ME_HINT::refidx`

[in]: Specifies the reference index (31=invalid). Current we support only 1 reference frame per direction for external hints, so `refidx` must be 0.

#### 5.37.2.4 `int32_t NVENC_EXTERNAL_ME_HINT::dir`

[in]: Specifies the direction of motion estimation . 0=L0 1=L1.

#### 5.37.2.5 `int32_t NVENC_EXTERNAL_ME_HINT::partType`

[in]: Specifies the block partition type.0=16x16 1=16x8 2=8x16 3=8x8 (blocks in partition must be consecutive).

#### 5.37.2.6 `int32_t NVENC_EXTERNAL_ME_HINT::lastofPart`

[in]: Set to 1 for the last MV of (sub) partition

**5.37.2.7 int32\_t NVENC\_EXTERNAL\_ME\_HINT::lastOfMB**

[in]: Set to 1 for the last MV of macroblock.

## 5.38 NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- uint32\_t [numCandsPerBlk16x16](#): 4
- uint32\_t [numCandsPerBlk16x8](#): 4
- uint32\_t [numCandsPerBlk8x16](#): 4
- uint32\_t [numCandsPerBlk8x8](#): 4
- uint32\_t [reserved](#): 16
- uint32\_t [reserved1](#) [3]

### 5.38.1 Detailed Description

`_NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE` External motion vector hint counts per block type. H264 supports multiple hint while HEVC supports one hint for each valid candidate.

### 5.38.2 Field Documentation

#### 5.38.2.1 uint32\_t NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE::numCandsPerBlk16x16

[in]: Supported for H264,HEVC.It Specifies the number of candidates per 16x16 block.

#### 5.38.2.2 uint32\_t NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE::numCandsPerBlk16x8

[in]: Supported for H264 only.Specifies the number of candidates per 16x8 block.

#### 5.38.2.3 uint32\_t NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE::numCandsPerBlk8x16

[in]: Supported for H264 only.Specifies the number of candidates per 8x16 block.

#### 5.38.2.4 uint32\_t NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE::numCandsPerBlk8x8

[in]: Supported for H264,HEVC.Specifies the number of candidates per 8x8 block.

#### 5.38.2.5 uint32\_t NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE::reserved

[in]: Reserved for padding.

#### 5.38.2.6 uint32\_t NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE::reserved1[3]

[in]: Reserved for future use.



## 5.39 NVENC\_RECT Struct Reference

```
#include <nvEncodeAPI.h>
```

### Data Fields

- [uint32\\_t left](#)
- [uint32\\_t top](#)
- [uint32\\_t right](#)
- [uint32\\_t bottom](#)

### 5.39.1 Detailed Description

`_NVENC_RECT` Defines a Rectangle. Used in `NV_ENC_PREPROCESS_FRAME`.

### 5.39.2 Field Documentation

#### 5.39.2.1 `uint32_t NVENC_RECT::left`

[in]: X coordinate of the upper left corner of rectangular area to be specified.

#### 5.39.2.2 `uint32_t NVENC_RECT::top`

[in]: Y coordinate of the upper left corner of the rectangular area to be specified.

#### 5.39.2.3 `uint32_t NVENC_RECT::right`

[in]: X coordinate of the bottom right corner of the rectangular area to be specified.

#### 5.39.2.4 `uint32_t NVENC_RECT::bottom`

[in]: Y coordinate of the bottom right corner of the rectangular area to be specified.

# Index

- adaptiveTransformMode
  - NV\_ENC\_CONFIG\_H264, [56](#)
- apiVersion
  - NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_PARAMS, [91](#)
- aqStrength
  - NV\_ENC\_RC\_PARAMS, [107](#)
- averageBitRate
  - NV\_ENC\_RC\_PARAMS, [105](#)
- bdirectMode
  - NV\_ENC\_CONFIG\_H264, [56](#)
- bitstreamBuffer
  - NV\_ENC\_CREATE\_BITSTREAM\_BUFFER, [68](#)
- bitstreamBufferPtr
  - NV\_ENC\_CREATE\_BITSTREAM\_BUFFER, [68](#)
  - NV\_ENC\_LOCK\_BITSTREAM, [82](#)
- bitstreamRestrictionFlag
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, [62](#)
- bitStreamSize
  - NV\_ENC\_STAT, [115](#)
- bitstreamSizeInBytes
  - NV\_ENC\_LOCK\_BITSTREAM, [82](#)
- bottom
  - NVENC\_RECT, [125](#)
- bStereoEnable
  - NV\_ENC\_CONFIG\_H264\_MEONLY, [59](#)
- bufferDataPtr
  - NV\_ENC\_LOCK\_INPUT\_BUFFER, [84](#)
- bufferFmt
  - NV\_ENC\_CREATE\_INPUT\_BUFFER, [70](#)
  - NV\_ENC\_MEONLY\_PARAMS, [89](#)
  - NV\_ENC\_PIC\_PARAMS, [94](#)
- bufferFormat
  - NV\_ENC\_REGISTER\_RESOURCE, [111](#)
- capsToQuery
  - NV\_ENC\_CAPS\_PARAM, [48](#)
- chromaFormatIDC
  - NV\_ENC\_CONFIG\_H264, [58](#)
  - NV\_ENC\_CONFIG\_HEVC, [65](#)
- chromaSampleLocationBot
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, [62](#)
- chromaSampleLocationFlag
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, [62](#)
- chromaSampleLocationTop
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, [62](#)
- codecPicParams
  - NV\_ENC\_PIC\_PARAMS, [94](#)
- colourDescriptionPresentFlag
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, [61](#)
- colourMatrix
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, [62](#)
- colourPlaneId
  - NV\_ENC\_PIC\_PARAMS\_H264, [97](#)
- colourPrimaries
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, [62](#)
- completionEvent
  - NV\_ENC\_EVENT\_PARAMS, [73](#)
  - NV\_ENC\_MEONLY\_PARAMS, [89](#)
  - NV\_ENC\_PIC\_PARAMS, [94](#)
- constQP
  - NV\_ENC\_RC\_PARAMS, [105](#)
- constrainedFrame
  - NV\_ENC\_PIC\_PARAMS\_H264, [98](#)
  - NV\_ENC\_PIC\_PARAMS\_HEVC, [101](#)
- cuSize
  - NV\_ENC\_HEVC\_MV\_DATA, [75](#)
- cuType
  - NV\_ENC\_HEVC\_MV\_DATA, [75](#)
- darHeight
  - NV\_ENC\_INITIALIZE\_PARAMS, [77](#)
- darWidth
  - NV\_ENC\_INITIALIZE\_PARAMS, [77](#)
- Data1
  - GUID, [47](#)
- Data2
  - GUID, [47](#)
- Data3
  - GUID, [47](#)
- Data4
  - GUID, [47](#)

- device
  - NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_PARAMS, 91
- deviceType
  - NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_PARAMS, 91
- dir
  - NVENC\_EXTERNAL\_ME\_HINT, 122
- disableBadapt
  - NV\_ENC\_RC\_PARAMS, 106
- disableDeblockAcrossSliceBoundary
  - NV\_ENC\_CONFIG\_HEVC, 64
- disableDeblockingFilterIDC
  - NV\_ENC\_CONFIG\_H264, 56
- disableIadapt
  - NV\_ENC\_RC\_PARAMS, 106
- disableIntraSearch
  - NV\_ENC\_CONFIG\_H264\_MEONLY, 59
- disablePartition16x16
  - NV\_ENC\_CONFIG\_H264\_MEONLY, 59
- disablePartition16x8
  - NV\_ENC\_CONFIG\_H264\_MEONLY, 59
- disablePartition8x16
  - NV\_ENC\_CONFIG\_H264\_MEONLY, 59
- disablePartition8x8
  - NV\_ENC\_CONFIG\_H264\_MEONLY, 59
- disableSPSPPS
  - NV\_ENC\_CONFIG\_H264, 54
  - NV\_ENC\_CONFIG\_HEVC, 64
- displayPOCSyntax
  - NV\_ENC\_PIC\_PARAMS\_H264, 97
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 100
- doNotWait
  - NV\_ENC\_LOCK\_BITSTREAM, 81
  - NV\_ENC\_LOCK\_INPUT\_BUFFER, 84
- enableAQ
  - NV\_ENC\_RC\_PARAMS, 106
- enableConstrainedEncoding
  - NV\_ENC\_CONFIG\_H264, 55
- enableEncodeAsync
  - NV\_ENC\_INITIALIZE\_PARAMS, 77
- enableExternalMEHints
  - NV\_ENC\_INITIALIZE\_PARAMS, 78
- enableExtQPDeltaMap
  - NV\_ENC\_RC\_PARAMS, 106
- enableInitialRCQP
  - NV\_ENC\_RC\_PARAMS, 106
- enableIntraRefresh
  - NV\_ENC\_CONFIG\_H264, 54
  - NV\_ENC\_CONFIG\_HEVC, 65
- enableLookahead
  - NV\_ENC\_RC\_PARAMS, 106
- enableLTR
  - NV\_ENC\_CONFIG\_H264, 55
  - NV\_ENC\_CONFIG\_HEVC, 64
- enableMaxQP
  - NV\_ENC\_RC\_PARAMS, 106
- enableMEOnlyMode
  - NV\_ENC\_INITIALIZE\_PARAMS, 78
- enableMinQP
  - NV\_ENC\_RC\_PARAMS, 106
- enableNonRefP
  - NV\_ENC\_RC\_PARAMS, 107
- enablePTD
  - NV\_ENC\_INITIALIZE\_PARAMS, 77
- enableStereoMVC
  - NV\_ENC\_CONFIG\_H264, 54
- enableSubFrameWrite
  - NV\_ENC\_INITIALIZE\_PARAMS, 77
- enableTemporalAQ
  - NV\_ENC\_RC\_PARAMS, 107
- enableTemporalSVC
  - NV\_ENC\_CONFIG\_H264, 54
- enableVFR
  - NV\_ENC\_CONFIG\_H264, 55
- enableWeightedPrediction
  - NV\_ENC\_INITIALIZE\_PARAMS, 78
- ENCODE\_FUNC
  - NvEncCreateBitstreamBuffer, 32
  - NvEncCreateInputBuffer, 31
  - NvEncCreateMVBuffer, 44
  - NvEncDestroyBitstreamBuffer, 33
  - NvEncDestroyEncoder, 41
  - NvEncDestroyInputBuffer, 32
  - NvEncDestroyMVBuffer, 44
  - NvEncEncodePicture, 33
  - NvEncGetEncodeCaps, 28
  - NvEncGetEncodeGUIDCount, 25
  - NvEncGetEncodeGUIDs, 25
  - NvEncGetEncodePresetConfig, 29
  - NvEncGetEncodePresetCount, 28
  - NvEncGetEncodePresetGUIDs, 28
  - NvEncGetEncodeProfileGUIDCount, 26
  - NvEncGetEncodeProfileGUIDs, 26
  - NvEncGetEncodeStats, 38
  - NvEncGetInputFormatCount, 27
  - NvEncGetInputFormats, 27
  - NvEncGetSequenceParams, 38
  - NvEncInitializeEncoder, 30
  - NvEncInvalidateRefFrames, 41
  - NvEncLockBitstream, 36
  - NvEncLockInputBuffer, 37
  - NvEncMapInputResource, 40
  - NvEncodeAPICreateInstance, 45
  - NvEncodeAPIGetMaxSupportedVersion, 45
  - NvEncOpenEncodeSession, 25
  - NvEncOpenEncodeSessionEx, 42

- NvEncReconfigureEncoder, 43
- NvEncRegisterAsyncEvent, 39
- NvEncRegisterResource, 42
- NvEncRunMotionEstimationOnly, 45
- NvEncUnlockBitstream, 36
- NvEncUnlockInputBuffer, 37
- NvEncUnmapInputResource, 40
- NvEncUnregisterAsyncEvent, 39
- NvEncUnregisterResource, 43
- encodeCodecConfig
  - NV\_ENC\_CONFIG, 52
- encodeConfig
  - NV\_ENC\_INITIALIZE\_PARAMS, 78
- encodeGUID
  - NV\_ENC\_INITIALIZE\_PARAMS, 76
- encodeHeight
  - NV\_ENC\_INITIALIZE\_PARAMS, 77
- encodePicFlags
  - NV\_ENC\_PIC\_PARAMS, 94
- ENCODER\_STRUCTURE
  - NV\_ENC\_BUFFER\_FORMAT\_ABGR, 14
  - NV\_ENC\_BUFFER\_FORMAT\_ABGR10, 14
  - NV\_ENC\_BUFFER\_FORMAT\_ARGB, 14
  - NV\_ENC\_BUFFER\_FORMAT\_ARGB10, 14
  - NV\_ENC\_BUFFER\_FORMAT\_AYUV, 14
  - NV\_ENC\_BUFFER\_FORMAT\_IYUV, 14
  - NV\_ENC\_BUFFER\_FORMAT\_NV12, 14
  - NV\_ENC\_BUFFER\_FORMAT\_UNDEFINED, 14
  - NV\_ENC\_BUFFER\_FORMAT\_YUV420\_10BIT, 14
  - NV\_ENC\_BUFFER\_FORMAT\_YUV444, 14
  - NV\_ENC\_BUFFER\_FORMAT\_YUV444\_10BIT, 14
  - NV\_ENC\_BUFFER\_FORMAT\_YV12, 14
  - NV\_ENC\_CAPS\_ASYNC\_ENCODE\_SUPPORT, 20
  - NV\_ENC\_CAPS\_EXPOSED\_COUNT, 21
  - NV\_ENC\_CAPS\_HEIGHT\_MAX, 19
  - NV\_ENC\_CAPS\_LEVEL\_MAX, 19
  - NV\_ENC\_CAPS\_LEVEL\_MIN, 19
  - NV\_ENC\_CAPS\_MB\_NUM\_MAX, 20
  - NV\_ENC\_CAPS\_MB\_PER\_SEC\_MAX, 20
  - NV\_ENC\_CAPS\_NUM\_MAX\_BFRAMES, 18
  - NV\_ENC\_CAPS\_NUM\_MAX\_LTR\_FRAMES, 21
  - NV\_ENC\_CAPS\_NUM\_MAX\_TEMPORAL\_LAYERS, 19
  - NV\_ENC\_CAPS\_PREPROC\_SUPPORT, 20
  - NV\_ENC\_CAPS\_SEPARATE\_COLOUR\_PLANE, 19
  - NV\_ENC\_CAPS\_SUPPORT\_10BIT\_ENCODE, 21
  - NV\_ENC\_CAPS\_SUPPORT\_ADAPTIVE\_TRANSFORM, 19
  - NV\_ENC\_CAPS\_SUPPORT\_BDIRECT\_MODE, 18
  - NV\_ENC\_CAPS\_SUPPORT\_CABAC, 18
  - NV\_ENC\_CAPS\_SUPPORT\_CONSTRAINED\_ENCODING, 20
  - NV\_ENC\_CAPS\_SUPPORT\_CUSTOM\_VBV\_BUF\_SIZE, 20
  - NV\_ENC\_CAPS\_SUPPORT\_DYN\_BITRATE\_CHANGE, 19
  - NV\_ENC\_CAPS\_SUPPORT\_DYN\_FORCE\_CONSTQP, 19
  - NV\_ENC\_CAPS\_SUPPORT\_DYN\_RC\_MODE\_CHANGE, 19
  - NV\_ENC\_CAPS\_SUPPORT\_DYN\_RES\_CHANGE, 19
  - NV\_ENC\_CAPS\_SUPPORT\_DYNAMIC\_SLICE\_MODE, 20
  - NV\_ENC\_CAPS\_SUPPORT\_FIELD\_ENCODING, 18
  - NV\_ENC\_CAPS\_SUPPORT\_FMO, 18
  - NV\_ENC\_CAPS\_SUPPORT\_HIERARCHICAL\_BFRAMES, 19
  - NV\_ENC\_CAPS\_SUPPORT\_HIERARCHICAL\_PFRAMES, 19
  - NV\_ENC\_CAPS\_SUPPORT\_INTRA\_REFRESH, 20
  - NV\_ENC\_CAPS\_SUPPORT\_LOOKAHEAD, 20
  - NV\_ENC\_CAPS\_SUPPORT\_LOSSLESS\_ENCODE, 20
  - NV\_ENC\_CAPS\_SUPPORT\_MEONLY\_MODE, 20
  - NV\_ENC\_CAPS\_SUPPORT\_MONOCHROME, 18
  - NV\_ENC\_CAPS\_SUPPORT\_QPELMV, 18
  - NV\_ENC\_CAPS\_SUPPORT\_REF\_PIC\_INVALIDATION, 20
  - NV\_ENC\_CAPS\_SUPPORT\_RESERVED, 19
  - NV\_ENC\_CAPS\_SUPPORT\_SAO, 20
  - NV\_ENC\_CAPS\_SUPPORT\_SUBFRAME\_READBACK, 19
  - NV\_ENC\_CAPS\_SUPPORT\_TEMPORAL\_AQ, 21
  - NV\_ENC\_CAPS\_SUPPORT\_TEMPORAL\_SVC, 19
  - NV\_ENC\_CAPS\_SUPPORT\_WEIGHTED\_PREDICTION, 21
  - NV\_ENC\_CAPS\_SUPPORT\_YUV444\_ENCODE, 20
  - NV\_ENC\_CAPS\_SUPPORTED\_RATECONTROL\_MODES, 18
  - NV\_ENC\_CAPS\_WIDTH\_MAX, 19
  - NV\_ENC\_DEVICE\_TYPE\_CUDA, 18
  - NV\_ENC\_DEVICE\_TYPE\_DIRECTX, 18
  - NV\_ENC\_DEVICE\_TYPE\_OPENGL, 18
  - NV\_ENC\_ERR\_DEVICE\_NOT\_EXIST, 15
  - NV\_ENC\_ERR\_ENCODER\_BUSY, 15

NV\_ENC\_ERR\_ENCODER\_NOT\_INITIALIZED, 15

NV\_ENC\_ERR\_EVENT\_NOT\_REGISTERD, 15

NV\_ENC\_ERR\_GENERIC, 16

NV\_ENC\_ERR\_INCOMPATIBLE\_CLIENT\_KEY, 16

NV\_ENC\_ERR\_INVALID\_CALL, 15

NV\_ENC\_ERR\_INVALID\_DEVICE, 15

NV\_ENC\_ERR\_INVALID\_ENCODERDEVICE, 15

NV\_ENC\_ERR\_INVALID\_EVENT, 15

NV\_ENC\_ERR\_INVALID\_PARAM, 15

NV\_ENC\_ERR\_INVALID\_PTR, 15

NV\_ENC\_ERR\_INVALID\_VERSION, 15

NV\_ENC\_ERR\_LOCK\_BUSY, 15

NV\_ENC\_ERR\_MAP\_FAILED, 15

NV\_ENC\_ERR\_NEED\_MORE\_INPUT, 15

NV\_ENC\_ERR\_NO\_ENCODE\_DEVICE, 15

NV\_ENC\_ERR\_NOT\_ENOUGH\_BUFFER, 15

NV\_ENC\_ERR\_OUT\_OF\_MEMORY, 15

NV\_ENC\_ERR\_RESOURCE\_NOT\_MAPPED, 16

NV\_ENC\_ERR\_RESOURCE\_NOT\_REGISTERED, 16

NV\_ENC\_ERR\_RESOURCE\_REGISTER\_FAILED, 16

NV\_ENC\_ERR\_UNIMPLEMENTED, 16

NV\_ENC\_ERR\_UNSUPPORTED\_DEVICE, 15

NV\_ENC\_ERR\_UNSUPPORTED\_PARAM, 15

NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_AUTOSELECT, 17

NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_DISABLE, 17

NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_ENABLE, 17

NV\_ENC\_H264\_BDIRECT\_MODE\_AUTOSELECT, 17

NV\_ENC\_H264\_BDIRECT\_MODE\_DISABLE, 17

NV\_ENC\_H264\_BDIRECT\_MODE\_SPATIAL, 17

NV\_ENC\_H264\_BDIRECT\_MODE\_TEMPORAL, 17

NV\_ENC\_H264\_ENTROPY\_CODING\_MODE\_AUTOSELECT, 16

NV\_ENC\_H264\_ENTROPY\_CODING\_MODE\_CABAC, 16

NV\_ENC\_H264\_ENTROPY\_CODING\_MODE\_CAVLC, 16

NV\_ENC\_H264\_FMO\_AUTOSELECT, 17

NV\_ENC\_H264\_FMO\_DISABLE, 17

NV\_ENC\_H264\_FMO\_ENABLE, 17

NV\_ENC\_INPUT\_RESOURCE\_TYPE\_CUDAARRAY, 18

NV\_ENC\_INPUT\_RESOURCE\_TYPE\_CUDADEVICEPTR, 18

NV\_ENC\_INPUT\_RESOURCE\_TYPE\_DIRECTX, 18

NV\_ENC\_INPUT\_RESOURCE\_TYPE\_OPENGL\_TEX, 18

NV\_ENC\_MEMORY\_HEAP\_AUTOSELECT, 16

NV\_ENC\_MEMORY\_HEAP\_SYSMEM\_CACHED, 16

NV\_ENC\_MEMORY\_HEAP\_SYSMEM\_UNCACHED, 16

NV\_ENC\_MEMORY\_HEAP\_VID, 16

NV\_ENC\_MV\_PRECISION\_DEFAULT, 14

NV\_ENC\_MV\_PRECISION\_FULL\_PEL, 14

NV\_ENC\_MV\_PRECISION\_HALF\_PEL, 14

NV\_ENC\_MV\_PRECISION\_QUARTER\_PEL, 14

NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE\_FIELD, 13

NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE\_FRAME, 13

NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE\_MBAFF, 13

NV\_ENC\_PARAMS\_RC\_CBR, 13

NV\_ENC\_PARAMS\_RC\_CBR\_HQ, 13

NV\_ENC\_PARAMS\_RC\_CBR\_LOWDELAY\_HQ, 13

NV\_ENC\_PARAMS\_RC\_CONSTQP, 13

NV\_ENC\_PARAMS\_RC\_VBR, 13

NV\_ENC\_PARAMS\_RC\_VBR\_HQ, 13

NV\_ENC\_PIC\_FLAG\_EOS, 16

NV\_ENC\_PIC\_FLAG\_FORCEIDR, 16

NV\_ENC\_PIC\_FLAG\_FORCEINTRA, 16

NV\_ENC\_PIC\_FLAG\_OUTPUT\_SPSPPS, 16

NV\_ENC\_PIC\_STRUCT\_FIELD\_BOTTOM\_TOP, 13

NV\_ENC\_PIC\_STRUCT\_FIELD\_TOP\_BOTTOM, 13

NV\_ENC\_PIC\_STRUCT\_FRAME, 13

NV\_ENC\_PIC\_TYPE\_B, 13

NV\_ENC\_PIC\_TYPE\_BI, 13

NV\_ENC\_PIC\_TYPE\_I, 13

NV\_ENC\_PIC\_TYPE\_IDR, 13

NV\_ENC\_PIC\_TYPE\_INTRA\_REFRESH, 13

NV\_ENC\_PIC\_TYPE\_P, 13

NV\_ENC\_PIC\_TYPE\_SKIPPED, 13

NV\_ENC\_PIC\_TYPE\_UNKNOWN, 13

NV\_ENC\_STEREO\_PACKING\_MODE\_CHECKERBOARD, 17

NV\_ENC\_STEREO\_PACKING\_MODE\_COLINTERLEAVE, 17

NV\_ENC\_STEREO\_PACKING\_MODE\_FRAMESEQ, 17

NV\_ENC\_STEREO\_PACKING\_MODE\_NONE, 17

NV\_ENC\_STEREO\_PACKING\_MODE\_ROWINTERLEAVE, 17

- NV\_ENC\_STEREO\_PACKING\_MODE\_-  
SIDE BY SIDE, 17
- NV\_ENC\_STEREO\_PACKING\_MODE\_-  
TOP BOTTOM, 17
- NV\_ENC\_SUCCESS, 15
- ENCODER\_STRUCTURE
  - NV\_ENC\_BUFFER\_FORMAT, 14
  - NV\_ENC\_CAPS, 18
  - NV\_ENC\_CAPS\_PARAM\_VER, 11
  - NV\_ENC\_CONFIG\_VER, 11
  - NV\_ENC\_CREATE\_BITSTREAM\_BUFFER\_-  
VER, 11
  - NV\_ENC\_CREATE\_INPUT\_BUFFER\_VER, 11
  - NV\_ENC\_CREATE\_MV\_BUFFER\_VER, 11
  - NV\_ENC\_DEVICE\_TYPE, 18
  - NV\_ENC\_EVENT\_PARAMS\_VER, 12
  - NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_-  
MODE, 17
  - NV\_ENC\_H264\_BDIRECT\_MODE, 16
  - NV\_ENC\_H264\_ENTROPY\_CODING\_MODE,  
16
  - NV\_ENC\_H264\_FMO\_MODE, 17
  - NV\_ENC\_HEVC\_CUSIZE, 21
  - NV\_ENC\_INITIALIZE\_PARAMS\_VER, 11
  - NV\_ENC\_INPUT\_RESOURCE\_TYPE, 17
  - NV\_ENC\_LEVEL, 14
  - NV\_ENC\_LOCK\_BITSTREAM\_VER, 12
  - NV\_ENC\_LOCK\_INPUT\_BUFFER\_VER, 12
  - NV\_ENC\_MAP\_INPUT\_RESOURCE\_VER, 12
  - NV\_ENC\_MEMORY\_HEAP, 16
  - NV\_ENC\_MEONLY\_PARAMS\_VER, 12
  - NV\_ENC\_MV\_PRECISION, 13
  - NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_-  
PARAMS\_VER, 12
  - NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE, 13
  - NV\_ENC\_PARAMS\_RC\_2\_PASS\_FRAMESIZE\_-  
CAP, 10
  - NV\_ENC\_PARAMS\_RC\_2\_PASS\_QUALITY, 10
  - NV\_ENC\_PARAMS\_RC\_2\_PASS\_VBR, 11
  - NV\_ENC\_PARAMS\_RC\_CBR2, 11
  - NV\_ENC\_PARAMS\_RC\_MODE, 13
  - NV\_ENC\_PARAMS\_RC\_VBR\_MINQP, 10
  - NV\_ENC\_PIC\_FLAGS, 16
  - NV\_ENC\_PIC\_PARAMS\_VER, 12
  - NV\_ENC\_PIC\_STRUCT, 13
  - NV\_ENC\_PIC\_TYPE, 13
  - NV\_ENC\_PRESET\_CONFIG\_VER, 11
  - NV\_ENC\_RC\_PARAMS\_VER, 11
  - NV\_ENC\_RECONFIGURE\_PARAMS\_VER, 11
  - NV\_ENC\_REGISTER\_RESOURCE\_VER, 12
  - NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD\_-  
VER, 12
  - NV\_ENC\_STAT\_VER, 12
  - NV\_ENC\_STEREO\_PACKING\_MODE, 17
  - NVENCSTATUS, 14
  - encodeWidth
    - NV\_ENC\_INITIALIZE\_PARAMS, 77
  - entropyCodingMode
    - NV\_ENC\_CONFIG\_H264, 56
  - fmoMode
    - NV\_ENC\_CONFIG\_H264, 56
  - forceIDR
    - NV\_ENC\_RECONFIGURE\_PARAMS, 109
  - forceIntraRefreshWithFrameCnt
    - NV\_ENC\_PIC\_PARAMS\_H264, 98
    - NV\_ENC\_PIC\_PARAMS\_HEVC, 100
  - frameAvgQP
    - NV\_ENC\_LOCK\_BITSTREAM, 83
  - frameFieldMode
    - NV\_ENC\_CONFIG, 51
  - frameIdx
    - NV\_ENC\_LOCK\_BITSTREAM, 82
    - NV\_ENC\_PIC\_PARAMS, 94
  - frameIntervalP
    - NV\_ENC\_CONFIG, 51
  - frameRateDen
    - NV\_ENC\_INITIALIZE\_PARAMS, 77
  - frameRateNum
    - NV\_ENC\_INITIALIZE\_PARAMS, 77
  - frameSatd
    - NV\_ENC\_LOCK\_BITSTREAM, 83
  - gopLength
    - NV\_ENC\_CONFIG, 51
  - GUID, 47
    - Data1, 47
    - Data2, 47
    - Data3, 47
    - Data4, 47
  - h264Config
    - NV\_ENC\_CODEC\_CONFIG, 49
  - h264MeOnlyConfig
    - NV\_ENC\_CODEC\_CONFIG, 49
  - h264PicParams
    - NV\_ENC\_CODEC\_PIC\_PARAMS, 50
  - h264VUIParameters
    - NV\_ENC\_CONFIG\_H264, 57
  - height
    - NV\_ENC\_CREATE\_INPUT\_BUFFER, 70
    - NV\_ENC\_REGISTER\_RESOURCE, 110
  - hevcConfig
    - NV\_ENC\_CODEC\_CONFIG, 49
  - hevcMeOnlyConfig
    - NV\_ENC\_CODEC\_CONFIG, 49
  - hevcPicParams
    - NV\_ENC\_CODEC\_PIC\_PARAMS, 50
  - hevcVUIParameters

- NV\_ENC\_CONFIG\_HEVC, 66
- hierarchicalBFrames
  - NV\_ENC\_CONFIG\_H264, 54
- hierarchicalPFrames
  - NV\_ENC\_CONFIG\_H264, 54
- hwEncodeStatus
  - NV\_ENC\_LOCK\_BITSTREAM, 82
- idrPeriod
  - NV\_ENC\_CONFIG\_H264, 55
  - NV\_ENC\_CONFIG\_HEVC, 65
- inBufferSize
  - NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD, 113
- initialRCQP
  - NV\_ENC\_RC\_PARAMS, 107
- inputBuffer
  - NV\_ENC\_CREATE\_INPUT\_BUFFER, 70
  - NV\_ENC\_LOCK\_INPUT\_BUFFER, 84
  - NV\_ENC\_MEONLY\_PARAMS, 88
  - NV\_ENC\_PIC\_PARAMS, 94
- inputDuration
  - NV\_ENC\_PIC\_PARAMS, 94
- inputHeight
  - NV\_ENC\_MEONLY\_PARAMS, 88
  - NV\_ENC\_PIC\_PARAMS, 93
- inputPitch
  - NV\_ENC\_PIC\_PARAMS, 93
- inputResource
  - NV\_ENC\_MAP\_INPUT\_RESOURCE, 86
- inputTimeStamp
  - NV\_ENC\_PIC\_PARAMS, 94
- inputWidth
  - NV\_ENC\_MEONLY\_PARAMS, 88
  - NV\_ENC\_PIC\_PARAMS, 93
- intraRefreshCnt
  - NV\_ENC\_CONFIG\_H264, 57
  - NV\_ENC\_CONFIG\_HEVC, 65
- intraRefreshPeriod
  - NV\_ENC\_CONFIG\_H264, 57
  - NV\_ENC\_CONFIG\_HEVC, 65
- lastCUInCTB
  - NV\_ENC\_HEVC\_MV\_DATA, 75
- lastOfMB
  - NVENC\_EXTERNAL\_ME\_HINT, 122
- lastofPart
  - NVENC\_EXTERNAL\_ME\_HINT, 122
- lastValidByteOffset
  - NV\_ENC\_STAT, 115
- left
  - NVENC\_RECT, 125
- level
  - NV\_ENC\_CONFIG\_H264, 55
  - NV\_ENC\_CONFIG\_HEVC, 63
- lookaheadDepth
  - NV\_ENC\_RC\_PARAMS, 108
- ltrFrame
  - NV\_ENC\_LOCK\_BITSTREAM, 81
- ltrFrameBitmap
  - NV\_ENC\_LOCK\_BITSTREAM, 83
- ltrFrameIdx
  - NV\_ENC\_LOCK\_BITSTREAM, 83
- ltrMarkFrame
  - NV\_ENC\_PIC\_PARAMS\_H264, 98
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 101
- ltrMarkFrameIdx
  - NV\_ENC\_PIC\_PARAMS\_H264, 99
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 102
- ltrNumFrames
  - NV\_ENC\_CONFIG\_H264, 57
  - NV\_ENC\_CONFIG\_HEVC, 65
- ltrTrustMode
  - NV\_ENC\_CONFIG\_H264, 57
  - NV\_ENC\_CONFIG\_HEVC, 66
- ltrUsageMode
  - NV\_ENC\_PIC\_PARAMS\_H264, 99
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 102
- ltrUseFrameBitmap
  - NV\_ENC\_PIC\_PARAMS\_H264, 99
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 102
- ltrUseFrames
  - NV\_ENC\_PIC\_PARAMS\_H264, 98
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 101
- mappedBufferFmt
  - NV\_ENC\_MAP\_INPUT\_RESOURCE, 86
- mappedResource
  - NV\_ENC\_MAP\_INPUT\_RESOURCE, 86
- maxBitRate
  - NV\_ENC\_RC\_PARAMS, 106
- maxCUSize
  - NV\_ENC\_CONFIG\_HEVC, 64
- maxEncodeHeight
  - NV\_ENC\_INITIALIZE\_PARAMS, 78
- maxEncodeWidth
  - NV\_ENC\_INITIALIZE\_PARAMS, 78
- maxMEHintCountsPerBlock
  - NV\_ENC\_INITIALIZE\_PARAMS, 79
- maxNumRefFrames
  - NV\_ENC\_CONFIG\_H264, 57
- maxNumRefFramesInDPB
  - NV\_ENC\_CONFIG\_HEVC, 65
- maxQP
  - NV\_ENC\_RC\_PARAMS, 107
- maxTemporalLayers
  - NV\_ENC\_CONFIG\_H264, 58
- maxTemporalLayersMinus1
  - NV\_ENC\_CONFIG\_HEVC, 66



- mbType
  - NV\_ENC\_H264\_MV\_DATA, 74
- meExternalHints
  - NV\_ENC\_MEONLY\_PARAMS, 89
  - NV\_ENC\_PIC\_PARAMS, 95
- meHintCountsPerBlock
  - NV\_ENC\_MEONLY\_PARAMS, 89
  - NV\_ENC\_PIC\_PARAMS, 95
- meHintRefPicDist
  - NV\_ENC\_PIC\_PARAMS, 95
- memoryHeap
  - NV\_ENC\_CREATE\_BITSTREAM\_BUFFER, 68
  - NV\_ENC\_CREATE\_INPUT\_BUFFER, 70
- minCUSize
  - NV\_ENC\_CONFIG\_HEVC, 64
- minQP
  - NV\_ENC\_RC\_PARAMS, 107
- monoChromeEncoding
  - NV\_ENC\_CONFIG, 51
- mv
  - NV\_ENC\_H264\_MV\_DATA, 74
  - NV\_ENC\_HEVC\_MV\_DATA, 75
- mvBuffer
  - NV\_ENC\_CREATE\_MV\_BUFFER, 72
  - NV\_ENC\_MEONLY\_PARAMS, 88
- mvPrecision
  - NV\_ENC\_CONFIG, 52
- mvx
  - NV\_ENC\_MVECTOR, 90
  - NVENC\_EXTERNAL\_ME\_HINT, 122
- mvy
  - NV\_ENC\_MVECTOR, 90
  - NVENC\_EXTERNAL\_ME\_HINT, 122
- numCandsPerBlk16x16
  - NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_-  
PER\_BLOCKTYPE, 124
- numCandsPerBlk16x8
  - NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_-  
PER\_BLOCKTYPE, 124
- numCandsPerBlk8x16
  - NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_-  
PER\_BLOCKTYPE, 124
- numCandsPerBlk8x8
  - NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_-  
PER\_BLOCKTYPE, 124
- numSlices
  - NV\_ENC\_LOCK\_BITSTREAM, 82
- numTemporalLayers
  - NV\_ENC\_CONFIG\_H264, 56
- NV\_ENC\_BUFFER\_FORMAT\_ABGR
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_BUFFER\_FORMAT\_ABGR10
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_BUFFER\_FORMAT\_ARGB
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_BUFFER\_FORMAT\_ARGB10
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_BUFFER\_FORMAT\_AYUV
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_BUFFER\_FORMAT\_IYUV
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_BUFFER\_FORMAT\_NV12
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_BUFFER\_FORMAT\_UNDEFINED
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_BUFFER\_FORMAT\_YUV420\_10BIT
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_BUFFER\_FORMAT\_YUV444
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_BUFFER\_FORMAT\_YUV444\_10BIT
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_BUFFER\_FORMAT\_YV12
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_CAPS\_ASYNC\_ENCODE\_SUPPORT
  - ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_EXPOSED\_COUNT
  - ENCODER\_STRUCTURE, 21
- NV\_ENC\_CAPS\_HEIGHT\_MAX
  - ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_LEVEL\_MAX
  - ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_LEVEL\_MIN
  - ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_MB\_NUM\_MAX
  - ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_MB\_PER\_SEC\_MAX
  - ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_NUM\_MAX\_BFRAMES
  - ENCODER\_STRUCTURE, 18
- NV\_ENC\_CAPS\_NUM\_MAX\_LTR\_FRAMES
  - ENCODER\_STRUCTURE, 21
- NV\_ENC\_CAPS\_NUM\_MAX\_TEMPORAL\_LAYERS
  - ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_PREPROC\_SUPPORT
  - ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_SEPARATE\_COLOUR\_PLANE
  - ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_SUPPORT\_10BIT\_ENCODE
  - ENCODER\_STRUCTURE, 21
- NV\_ENC\_CAPS\_SUPPORT\_ADAPTIVE\_-  
TRANSFORM
  - ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_SUPPORT\_BDIRECT\_MODE
  - ENCODER\_STRUCTURE, 18
- NV\_ENC\_CAPS\_SUPPORT\_CABAC
  - ENCODER\_STRUCTURE, 18



- NV\_ENC\_CAPS\_SUPPORT\_CONSTRAINED\_-  
ENCODING  
ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_SUPPORT\_CUSTOM\_VBV\_BUF\_-  
SIZE  
ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_SUPPORT\_DYN\_BITRATE\_-  
CHANGE  
ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_SUPPORT\_DYN\_FORCE\_-  
CONSTQP  
ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_SUPPORT\_DYN\_RC\_MODE\_-  
CHANGE  
ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_SUPPORT\_DYN\_RES\_CHANGE  
ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_SUPPORT\_DYNAMIC\_SLICE\_-  
MODE  
ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_SUPPORT\_FIELD\_ENCODING  
ENCODER\_STRUCTURE, 18
- NV\_ENC\_CAPS\_SUPPORT\_FMO  
ENCODER\_STRUCTURE, 18
- NV\_ENC\_CAPS\_SUPPORT\_HIERARCHICAL\_-  
BFRAMES  
ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_SUPPORT\_HIERARCHICAL\_-  
PFRAMES  
ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_SUPPORT\_INTRA\_REFRESH  
ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_SUPPORT\_LOOKAHEAD  
ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_SUPPORT\_LOSSLESS\_ENCODE  
ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_SUPPORT\_MEONLY\_MODE  
ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_SUPPORT\_MONOCHROME  
ENCODER\_STRUCTURE, 18
- NV\_ENC\_CAPS\_SUPPORT\_QPELMV  
ENCODER\_STRUCTURE, 18
- NV\_ENC\_CAPS\_SUPPORT\_REF\_PIC\_-  
INVALIDATION  
ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_SUPPORT\_RESERVED  
ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_SUPPORT\_SAO  
ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_SUPPORT\_SUBFRAME\_-  
READBACK  
ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_SUPPORT\_TEMPORAL\_AQ  
ENCODER\_STRUCTURE, 21
- NV\_ENC\_CAPS\_SUPPORT\_TEMPORAL\_SVC  
ENCODER\_STRUCTURE, 19
- NV\_ENC\_CAPS\_SUPPORT\_WEIGHTED\_-  
PREDICTION  
ENCODER\_STRUCTURE, 21
- NV\_ENC\_CAPS\_SUPPORT\_YUV444\_ENCODE  
ENCODER\_STRUCTURE, 20
- NV\_ENC\_CAPS\_SUPPORTED\_RATECONTROL\_-  
MODES  
ENCODER\_STRUCTURE, 18
- NV\_ENC\_CAPS\_WIDTH\_MAX  
ENCODER\_STRUCTURE, 19
- NV\_ENC\_DEVICE\_TYPE\_CUDA  
ENCODER\_STRUCTURE, 18
- NV\_ENC\_DEVICE\_TYPE\_DIRECTX  
ENCODER\_STRUCTURE, 18
- NV\_ENC\_DEVICE\_TYPE\_OPENGL  
ENCODER\_STRUCTURE, 18
- NV\_ENC\_ERR\_DEVICE\_NOT\_EXIST  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_ENCODER\_BUSY  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_ENCODER\_NOT\_INITIALIZED  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_EVENT\_NOT\_REGISTERD  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_GENERIC  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_ERR\_INCOMPATIBLE\_CLIENT\_KEY  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_ERR\_INVALID\_CALL  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_INVALID\_DEVICE  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_INVALID\_ENCODERDEVICE  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_INVALID\_EVENT  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_INVALID\_PARAM  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_INVALID\_PTR  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_INVALID\_VERSION  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_LOCK\_BUSY  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_MAP\_FAILED  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_NEED\_MORE\_INPUT  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_NO\_ENCODE\_DEVICE  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_NOT\_ENOUGH\_BUFFER  
ENCODER\_STRUCTURE, 15

- NV\_ENC\_ERR\_OUT\_OF\_MEMORY  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_RESOURCE\_NOT\_MAPPED  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_ERR\_RESOURCE\_NOT\_REGISTERED  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_ERR\_RESOURCE\_REGISTER\_FAILED  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_ERR\_UNIMPLEMENTED  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_ERR\_UNSUPPORTED\_DEVICE  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_ERR\_UNSUPPORTED\_PARAM  
ENCODER\_STRUCTURE, 15
- NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_-  
AUTOSELECT  
ENCODER\_STRUCTURE, 17
- NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_-  
DISABLE  
ENCODER\_STRUCTURE, 17
- NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_-  
ENABLE  
ENCODER\_STRUCTURE, 17
- NV\_ENC\_H264\_BDIRECT\_MODE\_AUTOSELECT  
ENCODER\_STRUCTURE, 17
- NV\_ENC\_H264\_BDIRECT\_MODE\_DISABLE  
ENCODER\_STRUCTURE, 17
- NV\_ENC\_H264\_BDIRECT\_MODE\_SPATIAL  
ENCODER\_STRUCTURE, 17
- NV\_ENC\_H264\_BDIRECT\_MODE\_TEMPORAL  
ENCODER\_STRUCTURE, 17
- NV\_ENC\_H264\_ENTROPY\_CODING\_MODE\_-  
AUTOSELECT  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_H264\_ENTROPY\_CODING\_MODE\_-  
CABAC  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_H264\_ENTROPY\_CODING\_MODE\_-  
CAVLC  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_H264\_FMO\_AUTOSELECT  
ENCODER\_STRUCTURE, 17
- NV\_ENC\_H264\_FMO\_DISABLE  
ENCODER\_STRUCTURE, 17
- NV\_ENC\_H264\_FMO\_ENABLE  
ENCODER\_STRUCTURE, 17
- NV\_ENC\_INPUT\_RESOURCE\_TYPE\_CUDAARRAY  
ENCODER\_STRUCTURE, 18
- NV\_ENC\_INPUT\_RESOURCE\_TYPE\_-  
CUDADEVICEPTR  
ENCODER\_STRUCTURE, 18
- NV\_ENC\_INPUT\_RESOURCE\_TYPE\_DIRECTX  
ENCODER\_STRUCTURE, 18
- NV\_ENC\_INPUT\_RESOURCE\_TYPE\_OPENGL\_-  
TEX  
ENCODER\_STRUCTURE, 18
- NV\_ENC\_MEMORY\_HEAP\_AUTOSELECT  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_MEMORY\_HEAP\_SYMMEM\_CACHED  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_MEMORY\_HEAP\_SYMMEM\_UNCACHED  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_MEMORY\_HEAP\_VID  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_MV\_PRECISION\_DEFAULT  
ENCODER\_STRUCTURE, 14
- NV\_ENC\_MV\_PRECISION\_FULL\_PEL  
ENCODER\_STRUCTURE, 14
- NV\_ENC\_MV\_PRECISION\_HALF\_PEL  
ENCODER\_STRUCTURE, 14
- NV\_ENC\_MV\_PRECISION\_QUARTER\_PEL  
ENCODER\_STRUCTURE, 14
- NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE\_FIELD  
ENCODER\_STRUCTURE, 13
- NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE\_-  
FRAME  
ENCODER\_STRUCTURE, 13
- NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE\_-  
MBAFF  
ENCODER\_STRUCTURE, 13
- NV\_ENC\_PARAMS\_RC\_CBR  
ENCODER\_STRUCTURE, 13
- NV\_ENC\_PARAMS\_RC\_CBR\_HQ  
ENCODER\_STRUCTURE, 13
- NV\_ENC\_PARAMS\_RC\_CBR\_LOWDELAY\_HQ  
ENCODER\_STRUCTURE, 13
- NV\_ENC\_PARAMS\_RC\_CONSTQP  
ENCODER\_STRUCTURE, 13
- NV\_ENC\_PARAMS\_RC\_VBR  
ENCODER\_STRUCTURE, 13
- NV\_ENC\_PARAMS\_RC\_VBR\_HQ  
ENCODER\_STRUCTURE, 13
- NV\_ENC\_PIC\_FLAG\_EOS  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_PIC\_FLAG\_FORCEIDR  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_PIC\_FLAG\_FORCEINTRA  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_PIC\_FLAG\_OUTPUT\_SPSPPS  
ENCODER\_STRUCTURE, 16
- NV\_ENC\_PIC\_STRUCT\_FIELD\_BOTTOM\_TOP  
ENCODER\_STRUCTURE, 13
- NV\_ENC\_PIC\_STRUCT\_FIELD\_TOP\_BOTTOM  
ENCODER\_STRUCTURE, 13
- NV\_ENC\_PIC\_STRUCT\_FRAME  
ENCODER\_STRUCTURE, 13
- NV\_ENC\_PIC\_TYPE\_B

- ENCODER\_STRUCTURE, [13](#)
- NV\_ENC\_PIC\_TYPE\_BI
  - ENCODER\_STRUCTURE, [13](#)
- NV\_ENC\_PIC\_TYPE\_I
  - ENCODER\_STRUCTURE, [13](#)
- NV\_ENC\_PIC\_TYPE\_IDR
  - ENCODER\_STRUCTURE, [13](#)
- NV\_ENC\_PIC\_TYPE\_INTRA\_REFRESH
  - ENCODER\_STRUCTURE, [13](#)
- NV\_ENC\_PIC\_TYPE\_P
  - ENCODER\_STRUCTURE, [13](#)
- NV\_ENC\_PIC\_TYPE\_SKIPPED
  - ENCODER\_STRUCTURE, [13](#)
- NV\_ENC\_PIC\_TYPE\_UNKNOWN
  - ENCODER\_STRUCTURE, [13](#)
- NV\_ENC\_STEREO\_PACKING\_MODE\_-CHECKERBOARD
  - ENCODER\_STRUCTURE, [17](#)
- NV\_ENC\_STEREO\_PACKING\_MODE\_-COLINTERLEAVE
  - ENCODER\_STRUCTURE, [17](#)
- NV\_ENC\_STEREO\_PACKING\_MODE\_FRAMESEQ
  - ENCODER\_STRUCTURE, [17](#)
- NV\_ENC\_STEREO\_PACKING\_MODE\_NONE
  - ENCODER\_STRUCTURE, [17](#)
- NV\_ENC\_STEREO\_PACKING\_MODE\_-ROWINTERLEAVE
  - ENCODER\_STRUCTURE, [17](#)
- NV\_ENC\_STEREO\_PACKING\_MODE\_SIDE BYSIDE
  - ENCODER\_STRUCTURE, [17](#)
- NV\_ENC\_STEREO\_PACKING\_MODE\_-TOPBOTTOM
  - ENCODER\_STRUCTURE, [17](#)
- NV\_ENC\_SUCCESS
  - ENCODER\_STRUCTURE, [15](#)
- NV\_ENC\_BUFFER\_FORMAT
  - ENCODER\_STRUCTURE, [14](#)
- NV\_ENC\_CAPS
  - ENCODER\_STRUCTURE, [18](#)
- NV\_ENC\_CAPS\_PARAM, [48](#)
  - capsToQuery, [48](#)
  - reserved, [48](#)
  - version, [48](#)
- NV\_ENC\_CAPS\_PARAM\_VER
  - ENCODER\_STRUCTURE, [11](#)
- NV\_ENC\_CODEC\_CONFIG, [49](#)
  - h264Config, [49](#)
  - h264MeOnlyConfig, [49](#)
  - hevcConfig, [49](#)
  - hevcMeOnlyConfig, [49](#)
  - reserved, [49](#)
- NV\_ENC\_CODEC\_PIC\_PARAMS, [50](#)
  - h264PicParams, [50](#)
  - hevcPicParams, [50](#)
  - reserved, [50](#)
- NV\_ENC\_CONFIG, [51](#)
  - encodeCodecConfig, [52](#)
  - frameFieldMode, [51](#)
  - frameIntervalP, [51](#)
  - gopLength, [51](#)
  - monoChromeEncoding, [51](#)
  - mvPrecision, [52](#)
  - profileGUID, [51](#)
  - rcParams, [52](#)
  - reserved, [52](#)
  - reserved2, [52](#)
  - version, [51](#)
- NV\_ENC\_CONFIG\_H264, [53](#)
  - adaptiveTransformMode, [56](#)
  - bdirectMode, [56](#)
  - chromaFormatIDC, [58](#)
  - disableDeblockingFilterIDC, [56](#)
  - disableSPSPPS, [54](#)
  - enableConstrainedEncoding, [55](#)
  - enableIntraRefresh, [54](#)
  - enableLTR, [55](#)
  - enableStereoMVC, [54](#)
  - enableTemporalSVC, [54](#)
  - enableVFR, [55](#)
  - entropyCodingMode, [56](#)
  - fmoMode, [56](#)
  - h264VUIParameters, [57](#)
  - hierarchicalBFrames, [54](#)
  - hierarchicalPFrames, [54](#)
  - idrPeriod, [55](#)
  - intraRefreshCnt, [57](#)
  - intraRefreshPeriod, [57](#)
  - level, [55](#)
  - ltrNumFrames, [57](#)
  - ltrTrustMode, [57](#)
  - maxNumRefFrames, [57](#)
  - maxTemporalLayers, [58](#)
  - numTemporalLayers, [56](#)
  - outputAUD, [54](#)
  - outputBufferingPeriodSEI, [54](#)
  - outputFramePackingSEI, [54](#)
  - outputPictureTimingSEI, [54](#)
  - outputRecoveryPointSEI, [54](#)
  - ppsId, [56](#)
  - qpPrimeYZeroTransformBypassFlag, [55](#)
  - repeatSPSPPS, [55](#)
  - reserved1, [58](#)
  - reserved2, [58](#)
  - reservedBitFields, [55](#)
  - separateColourPlaneFlag, [56](#)
  - sliceMode, [57](#)
  - sliceModeData, [57](#)
  - spsId, [56](#)

- stereoMode, 56
- useConstrainedIntraPred, 55
- NV\_ENC\_CONFIG\_H264\_MEONLY, 59
  - bStereoEnable, 59
  - disableIntraSearch, 59
  - disablePartition16x16, 59
  - disablePartition16x8, 59
  - disablePartition8x16, 59
  - disablePartition8x8, 59
  - reserved, 59
  - reserved1, 60
  - reserved2, 60
- NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, 61
  - bitstreamRestrictionFlag, 62
  - chromaSampleLocationBot, 62
  - chromaSampleLocationFlag, 62
  - chromaSampleLocationTop, 62
  - colourDescriptionPresentFlag, 61
  - colourMatrix, 62
  - colourPrimaries, 62
  - overscanInfo, 61
  - overscanInfoPresentFlag, 61
  - transferCharacteristics, 62
  - videoFormat, 61
  - videoFullRangeFlag, 61
  - videoSignalTypePresentFlag, 61
- NV\_ENC\_CONFIG\_HEVC, 63
  - chromaFormatIDC, 65
  - disableDeblockAcrossSliceBoundary, 64
  - disableSPSPPS, 64
  - enableIntraRefresh, 65
  - enableLTR, 64
  - hevcVUIParameters, 66
  - idrPeriod, 65
  - intraRefreshCnt, 65
  - intraRefreshPeriod, 65
  - level, 63
  - ltrNumFrames, 65
  - ltrTrustMode, 66
  - maxCUSize, 64
  - maxNumRefFramesInDPB, 65
  - maxTemporalLayersMinus1, 66
  - minCUSize, 64
  - outputAUD, 64
  - outputBufferingPeriodSEI, 64
  - outputPictureTimingSEI, 64
  - pixelBitDepthMinus8, 65
  - ppsId, 66
  - repeatSPSPPS, 64
  - reserved, 65
  - reserved1, 66
  - reserved2, 66
  - sliceMode, 66
  - sliceModeData, 66
  - spsId, 66
  - tier, 63
  - useConstrainedIntraPred, 64
  - vpsId, 65
- NV\_ENC\_CONFIG\_HEVC\_MEONLY, 67
  - reserved, 67
  - reserved1, 67
- NV\_ENC\_CONFIG\_VER
  - ENCODER\_STRUCTURE, 11
- NV\_ENC\_CREATE\_BITSTREAM\_BUFFER, 68
  - bitstreamBuffer, 68
  - bitstreamBufferPtr, 68
  - memoryHeap, 68
  - reserved, 68
  - reserved1, 68
  - reserved2, 69
  - size, 68
  - version, 68
- NV\_ENC\_CREATE\_BITSTREAM\_BUFFER\_VER
  - ENCODER\_STRUCTURE, 11
- NV\_ENC\_CREATE\_INPUT\_BUFFER, 70
  - bufferFmt, 70
  - height, 70
  - inputBuffer, 70
  - memoryHeap, 70
  - pSysMemBuffer, 71
  - reserved, 70
  - reserved1, 71
  - reserved2, 71
  - version, 70
  - width, 70
- NV\_ENC\_CREATE\_INPUT\_BUFFER\_VER
  - ENCODER\_STRUCTURE, 11
- NV\_ENC\_CREATE\_MV\_BUFFER, 72
  - mvBuffer, 72
  - reserved1, 72
  - reserved2, 72
  - version, 72
- NV\_ENC\_CREATE\_MV\_BUFFER\_VER
  - ENCODER\_STRUCTURE, 11
- NV\_ENC\_DEVICE\_TYPE
  - ENCODER\_STRUCTURE, 18
- NV\_ENC\_EVENT\_PARAMS, 73
  - completionEvent, 73
  - reserved, 73
  - reserved1, 73
  - reserved2, 73
  - version, 73
- NV\_ENC\_EVENT\_PARAMS\_VER
  - ENCODER\_STRUCTURE, 12
- NV\_ENC\_H264\_ADAPTIVE\_TRANSFORM\_MODE
  - ENCODER\_STRUCTURE, 17
- NV\_ENC\_H264\_BDIRECT\_MODE
  - ENCODER\_STRUCTURE, 16

- NV\_ENC\_H264\_ENTROPY\_CODING\_MODE
  - ENCODER\_STRUCTURE, 16
- NV\_ENC\_H264\_FMO\_MODE
  - ENCODER\_STRUCTURE, 17
- NV\_ENC\_H264\_MV\_DATA, 74
  - mbType, 74
  - mv, 74
  - partitionType, 74
  - reserved, 74
- NV\_ENC\_HEVC\_CUSIZE
  - ENCODER\_STRUCTURE, 21
- NV\_ENC\_HEVC\_MV\_DATA, 75
  - cuSize, 75
  - cuType, 75
  - lastCUInCTB, 75
  - mv, 75
  - partitionMode, 75
- NV\_ENC\_INITIALIZE\_PARAMS, 76
  - darHeight, 77
  - darWidth, 77
  - enableEncodeAsync, 77
  - enableExternalMEHints, 78
  - enableMEOnlyMode, 78
  - enablePTD, 77
  - enableSubFrameWrite, 77
  - enableWeightedPrediction, 78
  - encodeConfig, 78
  - encodeGUID, 76
  - encodeHeight, 77
  - encodeWidth, 77
  - frameRateDen, 77
  - frameRateNum, 77
  - maxEncodeHeight, 78
  - maxEncodeWidth, 78
  - maxMEHintCountsPerBlock, 79
  - presetGUID, 76
  - privData, 78
  - privDataSize, 78
  - reportSliceOffsets, 77
  - reserved, 79
  - reserved2, 79
  - reservedBitFields, 78
  - version, 76
- NV\_ENC\_INITIALIZE\_PARAMS\_VER
  - ENCODER\_STRUCTURE, 11
- NV\_ENC\_INPUT\_RESOURCE\_OPENGL\_TEX, 80
  - target, 80
  - texture, 80
- NV\_ENC\_INPUT\_RESOURCE\_TYPE
  - ENCODER\_STRUCTURE, 17
- NV\_ENC\_LEVEL
  - ENCODER\_STRUCTURE, 14
- NV\_ENC\_LOCK\_BITSTREAM, 81
  - bitstreamBufferPtr, 82
  - bitstreamSizeInBytes, 82
  - doNotWait, 81
  - frameAvgQP, 83
  - frameIdx, 82
  - frameSatd, 83
  - hwEncodeStatus, 82
  - ltrFrame, 81
  - ltrFrameBitmap, 83
  - ltrFrameIdx, 83
  - numSlices, 82
  - outputBitstream, 82
  - outputDuration, 82
  - outputTimeStamp, 82
  - pictureStruct, 82
  - pictureType, 82
  - reserved, 83
  - reserved2, 83
  - reservedBitFields, 81
  - sliceOffsets, 82
  - version, 81
- NV\_ENC\_LOCK\_BITSTREAM\_VER
  - ENCODER\_STRUCTURE, 12
- NV\_ENC\_LOCK\_INPUT\_BUFFER, 84
  - bufferDataPtr, 84
  - doNotWait, 84
  - inputBuffer, 84
  - pitch, 84
  - reserved1, 84
  - reserved2, 85
  - reservedBitFields, 84
  - version, 84
- NV\_ENC\_LOCK\_INPUT\_BUFFER\_VER
  - ENCODER\_STRUCTURE, 12
- NV\_ENC\_MAP\_INPUT\_RESOURCE, 86
  - inputResource, 86
  - mappedBufferFmt, 86
  - mappedResource, 86
  - registeredResource, 86
  - reserved1, 86
  - reserved2, 87
  - subResourceIndex, 86
  - version, 86
- NV\_ENC\_MAP\_INPUT\_RESOURCE\_VER
  - ENCODER\_STRUCTURE, 12
- NV\_ENC\_MEMORY\_HEAP
  - ENCODER\_STRUCTURE, 16
- NV\_ENC\_MEONLY\_PARAMS, 88
  - bufferFmt, 89
  - completionEvent, 89
  - inputBuffer, 88
  - inputHeight, 88
  - inputWidth, 88
  - meExternalHints, 89
  - meHintCountsPerBlock, 89

- mvBuffer, 88
- referenceFrame, 88
- reserved1, 89
- reserved2, 89
- version, 88
- viewID, 89
- NV\_ENC\_MEONLY\_PARAMS\_VER
  - ENCODER\_STRUCTURE, 12
- NV\_ENC\_MV\_PRECISION
  - ENCODER\_STRUCTURE, 13
- NV\_ENC\_MVECTOR, 90
  - mvx, 90
  - mvy, 90
- NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_PARAMS, 91
  - apiVersion, 91
  - device, 91
  - deviceType, 91
  - reserved, 91
  - reserved1, 91
  - reserved2, 91
  - version, 91
- NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_PARAMS\_VER
  - ENCODER\_STRUCTURE, 12
- NV\_ENC\_PARAMS\_FRAME\_FIELD\_MODE
  - ENCODER\_STRUCTURE, 13
- NV\_ENC\_PARAMS\_RC\_2\_PASS\_FRAMESIZE\_CAP
  - ENCODER\_STRUCTURE, 10
- NV\_ENC\_PARAMS\_RC\_2\_PASS\_QUALITY
  - ENCODER\_STRUCTURE, 10
- NV\_ENC\_PARAMS\_RC\_2\_PASS\_VBR
  - ENCODER\_STRUCTURE, 11
- NV\_ENC\_PARAMS\_RC\_CBR2
  - ENCODER\_STRUCTURE, 11
- NV\_ENC\_PARAMS\_RC\_MODE
  - ENCODER\_STRUCTURE, 13
- NV\_ENC\_PARAMS\_RC\_VBR\_MINQP
  - ENCODER\_STRUCTURE, 10
- NV\_ENC\_PIC\_FLAGS
  - ENCODER\_STRUCTURE, 16
- NV\_ENC\_PIC\_PARAMS, 93
  - bufferFmt, 94
  - codecPicParams, 94
  - completionEvent, 94
  - encodePicFlags, 94
  - frameIdx, 94
  - inputBuffer, 94
  - inputDuration, 94
  - inputHeight, 93
  - inputPitch, 93
  - inputTimeStamp, 94
  - inputWidth, 93
  - meExternalHints, 95
  - meHintCountsPerBlock, 95
  - meHintRefPicDist, 95
  - outputBitstream, 94
  - pictureStruct, 94
  - pictureType, 94
  - qpDeltaMap, 95
  - qpDeltaMapSize, 95
  - reserved1, 95
  - reserved2, 95
  - reserved3, 95
  - reserved4, 96
  - reservedBitFields, 95
  - version, 93
- NV\_ENC\_PIC\_PARAMS\_H264, 97
  - colourPlaneId, 97
  - constrainedFrame, 98
  - displayPOCSyntax, 97
  - forceIntraRefreshWithFrameCnt, 98
  - ltrMarkFrame, 98
  - ltrMarkFrameIdx, 99
  - ltrUsageMode, 99
  - ltrUseFrameBitmap, 99
  - ltrUseFrames, 98
  - refPicFlag, 97
  - reserved, 99
  - reserved2, 99
  - reserved3, 97
  - reservedBitFields, 98
  - seiPayloadArray, 98
  - seiPayloadArrayCnt, 98
  - sliceMode, 98
  - sliceModeData, 99
  - sliceModeDataUpdate, 98
  - sliceTypeArrayCnt, 98
  - sliceTypeData, 98
- NV\_ENC\_PIC\_PARAMS\_HEVC, 100
  - constrainedFrame, 101
  - displayPOCSyntax, 100
  - forceIntraRefreshWithFrameCnt, 100
  - ltrMarkFrame, 101
  - ltrMarkFrameIdx, 102
  - ltrUsageMode, 102
  - ltrUseFrameBitmap, 102
  - ltrUseFrames, 101
  - refPicFlag, 100
  - reserved, 102
  - reserved2, 102
  - reserved3, 102
  - reservedBitFields, 101
  - seiPayloadArray, 102
  - seiPayloadArrayCnt, 102
  - sliceMode, 101
  - sliceModeData, 101
  - sliceModeDataUpdate, 101

- sliceTypeArrayCnt, 101
- sliceTypeData, 101
- temporalId, 100
- NV\_ENC\_PIC\_PARAMS\_VER
  - ENCODER\_STRUCTURE, 12
- NV\_ENC\_PIC\_STRUCT
  - ENCODER\_STRUCTURE, 13
- NV\_ENC\_PIC\_TYPE
  - ENCODER\_STRUCTURE, 13
- NV\_ENC\_PRESET\_CONFIG, 103
  - presetCfg, 103
  - reserved1, 103
  - reserved2, 103
  - version, 103
- NV\_ENC\_PRESET\_CONFIG\_VER
  - ENCODER\_STRUCTURE, 11
- NV\_ENC\_QP, 104
- NV\_ENC\_RC\_PARAMS, 105
  - aqStrength, 107
  - averageBitRate, 105
  - constQP, 105
  - disableBadapt, 106
  - disableIadapt, 106
  - enableAQ, 106
  - enableExtQPDeltaMap, 106
  - enableInitialRCQP, 106
  - enableLookahead, 106
  - enableMaxQP, 106
  - enableMinQP, 106
  - enableNonRefP, 107
  - enableTemporalAQ, 107
  - initialRCQP, 107
  - lookaheadDepth, 108
  - maxBitRate, 106
  - maxQP, 107
  - minQP, 107
  - rateControlMode, 105
  - reservedBitFields, 107
  - strictGOPTarget, 107
  - targetQuality, 108
  - targetQualityLSB, 108
  - temporalLayerIdxMask, 107
  - temporalLayerQP, 107
  - vbvBufferSize, 106
  - vbvInitialDelay, 106
  - zeroReorderDelay, 107
- NV\_ENC\_RC\_PARAMS\_VER
  - ENCODER\_STRUCTURE, 11
- NV\_ENC\_RECONFIGURE\_PARAMS, 109
  - forceIDR, 109
  - reInitEncodeParams, 109
  - resetEncoder, 109
  - version, 109
- NV\_ENC\_RECONFIGURE\_PARAMS\_VER
  - ENCODER\_STRUCTURE, 11
- NV\_ENC\_REGISTER\_RESOURCE, 110
  - bufferFormat, 111
  - height, 110
  - pitch, 110
  - registeredResource, 111
  - reserved1, 111
  - reserved2, 111
  - resourceToRegister, 111
  - resourceType, 110
  - subResourceIndex, 110
  - version, 110
  - width, 110
- NV\_ENC\_REGISTER\_RESOURCE\_VER
  - ENCODER\_STRUCTURE, 12
- NV\_ENC\_SEI\_PAYLOAD, 112
  - payload, 112
  - payloadSize, 112
  - payloadType, 112
- NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD, 113
  - inBufferSize, 113
  - outSPSPSPayloadSize, 113
  - ppsId, 113
  - reserved, 113
  - reserved2, 114
  - spsId, 113
  - spsppsBuffer, 113
  - version, 113
- NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD\_VER
  - ENCODER\_STRUCTURE, 12
- NV\_ENC\_STAT, 115
  - bitStreamSize, 115
  - lastValidByteOffset, 115
  - outputBitStream, 115
  - picIdx, 116
  - picType, 115
  - reserved, 115
  - reserved1, 116
  - reserved2, 116
  - sliceOffsets, 115
  - version, 115
- NV\_ENC\_STAT\_VER
  - ENCODER\_STRUCTURE, 12
- NV\_ENC\_STEREO\_PACKING\_MODE
  - ENCODER\_STRUCTURE, 17
- NV\_ENCODE\_API\_FUNCTION\_LIST, 117
  - nvEncCreateBitstreamBuffer, 119
  - nvEncCreateInputBuffer, 119
  - nvEncCreateMVBuffer, 121
  - nvEncDestroyBitstreamBuffer, 119
  - nvEncDestroyEncoder, 120
  - nvEncDestroyInputBuffer, 119
  - nvEncDestroyMVBuffer, 121
  - nvEncEncodePicture, 119



- nvEncGetEncodeCaps, 118
- nvEncGetEncodeGUIDCount, 118
- nvEncGetEncodeGUIDs, 118
- nvEncGetEncodePresetConfig, 119
- nvEncGetEncodePresetCount, 118
- nvEncGetEncodePresetGUIDs, 119
- nvEncGetEncodeProfileGUIDCount, 118
- nvEncGetEncodeProfileGUIDs, 118
- nvEncGetEncodeStats, 120
- nvEncGetInputFormatCount, 118
- nvEncGetInputFormats, 118
- nvEncGetSequenceParams, 120
- nvEncInitializeEncoder, 119
- nvEncInvalidateRefFrames, 120
- nvEncLockBitstream, 119
- nvEncLockInputBuffer, 120
- nvEncMapInputResource, 120
- nvEncOpenEncodeSession, 118
- nvEncOpenEncodeSessionEx, 121
- nvEncReconfigureEncoder, 121
- nvEncRegisterAsyncEvent, 120
- nvEncRegisterResource, 121
- nvEncRunMotionEstimationOnly, 121
- nvEncUnlockBitstream, 119
- nvEncUnlockInputBuffer, 120
- nvEncUnmapInputResource, 120
- nvEncUnregisterAsyncEvent, 120
- nvEncUnregisterResource, 121
- reserved, 118
- reserved2, 121
- version, 118
- NVENC\_EXTERNAL\_ME\_HINT, 122
  - dir, 122
  - lastOfMB, 122
  - lastofPart, 122
  - mvx, 122
  - mvx, 122
  - partType, 122
  - refidx, 122
- NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_PER\_BLOCKTYPE, 124
  - numCandsPerBlk16x16, 124
  - numCandsPerBlk16x8, 124
  - numCandsPerBlk8x16, 124
  - numCandsPerBlk8x8, 124
  - reserved, 124
  - reserved1, 124
- NVENC\_RECT, 125
  - bottom, 125
  - left, 125
  - right, 125
  - top, 125
- NvEncCreateBitstreamBuffer
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 119
- NvEncCreateInputBuffer
  - ENCODE\_FUNC, 31
- nvEncCreateInputBuffer
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 119
- NvEncCreateMVBuffer
  - ENCODE\_FUNC, 44
- nvEncCreateMVBuffer
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 121
- NvEncDestroyBitstreamBuffer
  - ENCODE\_FUNC, 33
- nvEncDestroyBitstreamBuffer
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 119
- NvEncDestroyEncoder
  - ENCODE\_FUNC, 41
- nvEncDestroyEncoder
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 120
- NvEncDestroyInputBuffer
  - ENCODE\_FUNC, 32
- nvEncDestroyInputBuffer
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 119
- NvEncDestroyMVBuffer
  - ENCODE\_FUNC, 44
- nvEncDestroyMVBuffer
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 121
- NvEncEncodePicture
  - ENCODE\_FUNC, 33
- nvEncEncodePicture
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 119
- NvEncGetEncodeCaps
  - ENCODE\_FUNC, 28
- nvEncGetEncodeCaps
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 118
- NvEncGetEncodeGUIDCount
  - ENCODE\_FUNC, 25
- nvEncGetEncodeGUIDCount
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 118
- NvEncGetEncodeGUIDs
  - ENCODE\_FUNC, 25
- nvEncGetEncodeGUIDs
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 118
- NvEncGetEncodePresetConfig
  - ENCODE\_FUNC, 29
- nvEncGetEncodePresetConfig
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 119
- NvEncGetEncodePresetCount
  - ENCODE\_FUNC, 28
- nvEncGetEncodePresetCount
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 118
- NvEncGetEncodePresetGUIDs
  - ENCODE\_FUNC, 28
- nvEncGetEncodePresetGUIDs
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 119



- NvEncGetEncodeProfileGUIDCount  
ENCODE\_FUNC, 26
- nvEncGetEncodeProfileGUIDCount  
NV\_ENCODE\_API\_FUNCTION\_LIST, 118
- NvEncGetEncodeProfileGUIDs  
ENCODE\_FUNC, 26
- nvEncGetEncodeProfileGUIDs  
NV\_ENCODE\_API\_FUNCTION\_LIST, 118
- NvEncGetEncodeStats  
ENCODE\_FUNC, 38
- nvEncGetEncodeStats  
NV\_ENCODE\_API\_FUNCTION\_LIST, 120
- NvEncGetInputFormatCount  
ENCODE\_FUNC, 27
- nvEncGetInputFormatCount  
NV\_ENCODE\_API\_FUNCTION\_LIST, 118
- NvEncGetInputFormats  
ENCODE\_FUNC, 27
- nvEncGetInputFormats  
NV\_ENCODE\_API\_FUNCTION\_LIST, 118
- NvEncGetSequenceParams  
ENCODE\_FUNC, 38
- nvEncGetSequenceParams  
NV\_ENCODE\_API\_FUNCTION\_LIST, 120
- NvEncInitializeEncoder  
ENCODE\_FUNC, 30
- nvEncInitializeEncoder  
NV\_ENCODE\_API\_FUNCTION\_LIST, 119
- NvEncInvalidateRefFrames  
ENCODE\_FUNC, 41
- nvEncInvalidateRefFrames  
NV\_ENCODE\_API\_FUNCTION\_LIST, 120
- NvEncLockBitstream  
ENCODE\_FUNC, 36
- nvEncLockBitstream  
NV\_ENCODE\_API\_FUNCTION\_LIST, 119
- NvEncLockInputBuffer  
ENCODE\_FUNC, 37
- nvEncLockInputBuffer  
NV\_ENCODE\_API\_FUNCTION\_LIST, 120
- NvEncMapInputResource  
ENCODE\_FUNC, 40
- nvEncMapInputResource  
NV\_ENCODE\_API\_FUNCTION\_LIST, 120
- NvEncodeAPI Data structures, 7
- NvEncodeAPI Functions, 22
- NvEncodeAPICreateInstance  
ENCODE\_FUNC, 45
- NvEncodeAPIGetMaxSupportedVersion  
ENCODE\_FUNC, 45
- NvEncOpenEncodeSession  
ENCODE\_FUNC, 25
- nvEncOpenEncodeSession  
NV\_ENCODE\_API\_FUNCTION\_LIST, 118
- NvEncOpenEncodeSessionEx  
ENCODE\_FUNC, 42
- nvEncOpenEncodeSessionEx  
NV\_ENCODE\_API\_FUNCTION\_LIST, 121
- NvEncReconfigureEncoder  
ENCODE\_FUNC, 43
- nvEncReconfigureEncoder  
NV\_ENCODE\_API\_FUNCTION\_LIST, 121
- NvEncRegisterAsyncEvent  
ENCODE\_FUNC, 39
- nvEncRegisterAsyncEvent  
NV\_ENCODE\_API\_FUNCTION\_LIST, 120
- NvEncRegisterResource  
ENCODE\_FUNC, 42
- nvEncRegisterResource  
NV\_ENCODE\_API\_FUNCTION\_LIST, 121
- NvEncRunMotionEstimationOnly  
ENCODE\_FUNC, 45
- nvEncRunMotionEstimationOnly  
NV\_ENCODE\_API\_FUNCTION\_LIST, 121
- NVENCSTATUS  
ENCODER\_STRUCTURE, 14
- NvEncUnlockBitstream  
ENCODE\_FUNC, 36
- nvEncUnlockBitstream  
NV\_ENCODE\_API\_FUNCTION\_LIST, 119
- NvEncUnlockInputBuffer  
ENCODE\_FUNC, 37
- nvEncUnlockInputBuffer  
NV\_ENCODE\_API\_FUNCTION\_LIST, 120
- NvEncUnmapInputResource  
ENCODE\_FUNC, 40
- nvEncUnmapInputResource  
NV\_ENCODE\_API\_FUNCTION\_LIST, 120
- NvEncUnregisterAsyncEvent  
ENCODE\_FUNC, 39
- nvEncUnregisterAsyncEvent  
NV\_ENCODE\_API\_FUNCTION\_LIST, 120
- NvEncUnregisterResource  
ENCODE\_FUNC, 43
- nvEncUnregisterResource  
NV\_ENCODE\_API\_FUNCTION\_LIST, 121
- outputAUD  
NV\_ENC\_CONFIG\_H264, 54  
NV\_ENC\_CONFIG\_HEVC, 64
- outputBitStream  
NV\_ENC\_STAT, 115
- outputBitstream  
NV\_ENC\_LOCK\_BITSTREAM, 82  
NV\_ENC\_PIC\_PARAMS, 94
- outputBufferingPeriodSEI  
NV\_ENC\_CONFIG\_H264, 54  
NV\_ENC\_CONFIG\_HEVC, 64

- outputDuration
  - NV\_ENC\_LOCK\_BITSTREAM, 82
- outputFramePackingSEI
  - NV\_ENC\_CONFIG\_H264, 54
- outputPictureTimingSEI
  - NV\_ENC\_CONFIG\_H264, 54
  - NV\_ENC\_CONFIG\_HEVC, 64
- outputRecoveryPointSEI
  - NV\_ENC\_CONFIG\_H264, 54
- outputTimeStamp
  - NV\_ENC\_LOCK\_BITSTREAM, 82
- outSPSPSPayloadSize
  - NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD, 113
- overscanInfo
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, 61
- overscanInfoPresentFlag
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, 61
- partitionMode
  - NV\_ENC\_HEVC\_MV\_DATA, 75
- partitionType
  - NV\_ENC\_H264\_MV\_DATA, 74
- partType
  - NVENC\_EXTERNAL\_ME\_HINT, 122
- payload
  - NV\_ENC\_SEI\_PAYLOAD, 112
- payloadSize
  - NV\_ENC\_SEI\_PAYLOAD, 112
- payloadType
  - NV\_ENC\_SEI\_PAYLOAD, 112
- picIdx
  - NV\_ENC\_STAT, 116
- pictureStruct
  - NV\_ENC\_LOCK\_BITSTREAM, 82
  - NV\_ENC\_PIC\_PARAMS, 94
- pictureType
  - NV\_ENC\_LOCK\_BITSTREAM, 82
  - NV\_ENC\_PIC\_PARAMS, 94
- picType
  - NV\_ENC\_STAT, 115
- pitch
  - NV\_ENC\_LOCK\_INPUT\_BUFFER, 84
  - NV\_ENC\_REGISTER\_RESOURCE, 110
- pixelBitDepthMinus8
  - NV\_ENC\_CONFIG\_HEVC, 65
- ppsId
  - NV\_ENC\_CONFIG\_H264, 56
  - NV\_ENC\_CONFIG\_HEVC, 66
  - NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD, 113
- presetCfg
  - NV\_ENC\_PRESET\_CONFIG, 103
- presetGUID
  - NV\_ENC\_INITIALIZE\_PARAMS, 76
- privData
  - NV\_ENC\_INITIALIZE\_PARAMS, 78
- privDataSize
  - NV\_ENC\_INITIALIZE\_PARAMS, 78
- profileGUID
  - NV\_ENC\_CONFIG, 51
- pSysMemBuffer
  - NV\_ENC\_CREATE\_INPUT\_BUFFER, 71
- qpDeltaMap
  - NV\_ENC\_PIC\_PARAMS, 95
- qpDeltaMapSize
  - NV\_ENC\_PIC\_PARAMS, 95
- qpPrimeYZeroTransformBypassFlag
  - NV\_ENC\_CONFIG\_H264, 55
- rateControlMode
  - NV\_ENC\_RC\_PARAMS, 105
- rcParams
  - NV\_ENC\_CONFIG, 52
- referenceFrame
  - NV\_ENC\_MEONLY\_PARAMS, 88
- refidx
  - NVENC\_EXTERNAL\_ME\_HINT, 122
- refPicFlag
  - NV\_ENC\_PIC\_PARAMS\_H264, 97
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 100
- registeredResource
  - NV\_ENC\_MAP\_INPUT\_RESOURCE, 86
  - NV\_ENC\_REGISTER\_RESOURCE, 111
- reInitEncodeParams
  - NV\_ENC\_RECONFIGURE\_PARAMS, 109
- repeatSPSPPS
  - NV\_ENC\_CONFIG\_H264, 55
  - NV\_ENC\_CONFIG\_HEVC, 64
- reportSliceOffsets
  - NV\_ENC\_INITIALIZE\_PARAMS, 77
- reserved
  - NV\_ENC\_CAPS\_PARAM, 48
  - NV\_ENC\_CODEC\_CONFIG, 49
  - NV\_ENC\_CODEC\_PIC\_PARAMS, 50
  - NV\_ENC\_CONFIG, 52
  - NV\_ENC\_CONFIG\_H264\_MEONLY, 59
  - NV\_ENC\_CONFIG\_HEVC, 65
  - NV\_ENC\_CONFIG\_HEVC\_MEONLY, 67
  - NV\_ENC\_CREATE\_BITSTREAM\_BUFFER, 68
  - NV\_ENC\_CREATE\_INPUT\_BUFFER, 70
  - NV\_ENC\_EVENT\_PARAMS, 73
  - NV\_ENC\_H264\_MV\_DATA, 74
  - NV\_ENC\_INITIALIZE\_PARAMS, 79
  - NV\_ENC\_LOCK\_BITSTREAM, 83
  - NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_PARAMS, 91

- NV\_ENC\_PIC\_PARAMS\_H264, 99
- NV\_ENC\_PIC\_PARAMS\_HEVC, 102
- NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD, 113
- NV\_ENC\_STAT, 115
- NV\_ENCODE\_API\_FUNCTION\_LIST, 118
- NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_-  
PER\_BLOCKTYPE, 124
- reserved1
  - NV\_ENC\_CONFIG\_H264, 58
  - NV\_ENC\_CONFIG\_H264\_MEONLY, 60
  - NV\_ENC\_CONFIG\_HEVC, 66
  - NV\_ENC\_CONFIG\_HEVC\_MEONLY, 67
  - NV\_ENC\_CREATE\_BITSTREAM\_BUFFER, 68
  - NV\_ENC\_CREATE\_INPUT\_BUFFER, 71
  - NV\_ENC\_CREATE\_MV\_BUFFER, 72
  - NV\_ENC\_EVENT\_PARAMS, 73
  - NV\_ENC\_LOCK\_INPUT\_BUFFER, 84
  - NV\_ENC\_MAP\_INPUT\_RESOURCE, 86
  - NV\_ENC\_MEONLY\_PARAMS, 89
  - NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_-  
PARAMS, 91
  - NV\_ENC\_PIC\_PARAMS, 95
  - NV\_ENC\_PRESET\_CONFIG, 103
  - NV\_ENC\_REGISTER\_RESOURCE, 111
  - NV\_ENC\_STAT, 116
  - NVENC\_EXTERNAL\_ME\_HINT\_COUNTS\_-  
PER\_BLOCKTYPE, 124
- reserved2
  - NV\_ENC\_CONFIG, 52
  - NV\_ENC\_CONFIG\_H264, 58
  - NV\_ENC\_CONFIG\_H264\_MEONLY, 60
  - NV\_ENC\_CONFIG\_HEVC, 66
  - NV\_ENC\_CREATE\_BITSTREAM\_BUFFER, 69
  - NV\_ENC\_CREATE\_INPUT\_BUFFER, 71
  - NV\_ENC\_CREATE\_MV\_BUFFER, 72
  - NV\_ENC\_EVENT\_PARAMS, 73
  - NV\_ENC\_INITIALIZE\_PARAMS, 79
  - NV\_ENC\_LOCK\_BITSTREAM, 83
  - NV\_ENC\_LOCK\_INPUT\_BUFFER, 85
  - NV\_ENC\_MAP\_INPUT\_RESOURCE, 87
  - NV\_ENC\_MEONLY\_PARAMS, 89
  - NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_-  
PARAMS, 91
  - NV\_ENC\_PIC\_PARAMS, 95
  - NV\_ENC\_PIC\_PARAMS\_H264, 99
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 102
  - NV\_ENC\_PRESET\_CONFIG, 103
  - NV\_ENC\_REGISTER\_RESOURCE, 111
  - NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD, 114
  - NV\_ENC\_STAT, 116
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 121
- reserved3
  - NV\_ENC\_PIC\_PARAMS, 95
  - NV\_ENC\_PIC\_PARAMS\_H264, 97
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 102
- reserved4
  - NV\_ENC\_PIC\_PARAMS, 96
- reservedBitFields
  - NV\_ENC\_CONFIG\_H264, 55
  - NV\_ENC\_INITIALIZE\_PARAMS, 78
  - NV\_ENC\_LOCK\_BITSTREAM, 81
  - NV\_ENC\_LOCK\_INPUT\_BUFFER, 84
  - NV\_ENC\_PIC\_PARAMS, 95
  - NV\_ENC\_PIC\_PARAMS\_H264, 98
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 101
  - NV\_ENC\_RC\_PARAMS, 107
- resetEncoder
  - NV\_ENC\_RECONFIGURE\_PARAMS, 109
- resourceToRegister
  - NV\_ENC\_REGISTER\_RESOURCE, 111
- resourceType
  - NV\_ENC\_REGISTER\_RESOURCE, 110
- right
  - NVENC\_RECT, 125
- seiPayloadArray
  - NV\_ENC\_PIC\_PARAMS\_H264, 98
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 102
- seiPayloadArrayCnt
  - NV\_ENC\_PIC\_PARAMS\_H264, 98
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 102
- separateColourPlaneFlag
  - NV\_ENC\_CONFIG\_H264, 56
- size
  - NV\_ENC\_CREATE\_BITSTREAM\_BUFFER, 68
- sliceMode
  - NV\_ENC\_CONFIG\_H264, 57
  - NV\_ENC\_CONFIG\_HEVC, 66
  - NV\_ENC\_PIC\_PARAMS\_H264, 98
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 101
- sliceModeData
  - NV\_ENC\_CONFIG\_H264, 57
  - NV\_ENC\_CONFIG\_HEVC, 66
  - NV\_ENC\_PIC\_PARAMS\_H264, 99
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 101
- sliceModeDataUpdate
  - NV\_ENC\_PIC\_PARAMS\_H264, 98
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 101
- sliceOffsets
  - NV\_ENC\_LOCK\_BITSTREAM, 82
  - NV\_ENC\_STAT, 115
- sliceTypeArrayCnt
  - NV\_ENC\_PIC\_PARAMS\_H264, 98
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 101
- sliceTypeData
  - NV\_ENC\_PIC\_PARAMS\_H264, 98
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 101
- spsId

- NV\_ENC\_CONFIG\_H264, 56
- NV\_ENC\_CONFIG\_HEVC, 66
- NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD, 113
- spsppsBuffer
  - NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD, 113
- stereoMode
  - NV\_ENC\_CONFIG\_H264, 56
- strictGOPTarget
  - NV\_ENC\_RC\_PARAMS, 107
- subResourceIndex
  - NV\_ENC\_MAP\_INPUT\_RESOURCE, 86
  - NV\_ENC\_REGISTER\_RESOURCE, 110
- target
  - NV\_ENC\_INPUT\_RESOURCE\_OPENGL\_TEX, 80
- targetQuality
  - NV\_ENC\_RC\_PARAMS, 108
- targetQualityLSB
  - NV\_ENC\_RC\_PARAMS, 108
- temporalId
  - NV\_ENC\_PIC\_PARAMS\_HEVC, 100
- temporalLayerIdxMask
  - NV\_ENC\_RC\_PARAMS, 107
- temporalLayerQP
  - NV\_ENC\_RC\_PARAMS, 107
- texture
  - NV\_ENC\_INPUT\_RESOURCE\_OPENGL\_TEX, 80
- tier
  - NV\_ENC\_CONFIG\_HEVC, 63
- top
  - NVENC\_RECT, 125
- transferCharacteristics
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, 62
- useConstrainedIntraPred
  - NV\_ENC\_CONFIG\_H264, 55
  - NV\_ENC\_CONFIG\_HEVC, 64
- vbvBufferSize
  - NV\_ENC\_RC\_PARAMS, 106
- vbvInitialDelay
  - NV\_ENC\_RC\_PARAMS, 106
- version
  - NV\_ENC\_CAPS\_PARAM, 48
  - NV\_ENC\_CONFIG, 51
  - NV\_ENC\_CREATE\_BITSTREAM\_BUFFER, 68
  - NV\_ENC\_CREATE\_INPUT\_BUFFER, 70
  - NV\_ENC\_CREATE\_MV\_BUFFER, 72
  - NV\_ENC\_EVENT\_PARAMS, 73
  - NV\_ENC\_INITIALIZE\_PARAMS, 76
  - NV\_ENC\_LOCK\_BITSTREAM, 81
  - NV\_ENC\_LOCK\_INPUT\_BUFFER, 84
  - NV\_ENC\_MAP\_INPUT\_RESOURCE, 86
  - NV\_ENC\_MEONLY\_PARAMS, 88
  - NV\_ENC\_OPEN\_ENCODE\_SESSION\_EX\_PARAMS, 91
  - NV\_ENC\_PIC\_PARAMS, 93
  - NV\_ENC\_PRESET\_CONFIG, 103
  - NV\_ENC\_RECONFIGURE\_PARAMS, 109
  - NV\_ENC\_REGISTER\_RESOURCE, 110
  - NV\_ENC\_SEQUENCE\_PARAM\_PAYLOAD, 113
  - NV\_ENC\_STAT, 115
  - NV\_ENCODE\_API\_FUNCTION\_LIST, 118
- videoFormat
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, 61
- videoFullRangeFlag
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, 61
- videoSignalTypePresentFlag
  - NV\_ENC\_CONFIG\_H264\_VUI\_PARAMETERS, 61
- viewID
  - NV\_ENC\_MEONLY\_PARAMS, 89
- vpsId
  - NV\_ENC\_CONFIG\_HEVC, 65
- width
  - NV\_ENC\_CREATE\_INPUT\_BUFFER, 70
  - NV\_ENC\_REGISTER\_RESOURCE, 110
- zeroReorderDelay
  - NV\_ENC\_RC\_PARAMS, 107

### **Notice**

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

### **Trademarks**

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

### **Copyright**

© 2011-2017 NVIDIA Corporation. All rights reserved.