



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

November 11, 2016

Version 7.1



Contents

1	Legal Notice	1
2	Module Index	3
2.1	Modules	3
3	Data Structure Index	5
3.1	Data Structures	5
4	Module Documentation	7
4.1	NvEncodeAPI Data structures	7
4.1.1	Define Documentation	12
4.1.1.1	NV_ENC_CAPS_PARAM_VER	12
4.1.1.2	NV_ENC_CONFIG_VER	12
4.1.1.3	NV_ENC_CREATE_BITSTREAM_BUFFER_VER	12
4.1.1.4	NV_ENC_CREATE_INPUT_BUFFER_VER	12
4.1.1.5	NV_ENC_CREATE_MV_BUFFER_VER	12
4.1.1.6	NV_ENC_EVENT_PARAMS_VER	13
4.1.1.7	NV_ENC_INITIALIZE_PARAMS_VER	13
4.1.1.8	NV_ENC_LOCK_BITSTREAM_VER	13
4.1.1.9	NV_ENC_LOCK_INPUT_BUFFER_VER	13
4.1.1.10	NV_ENC_MAP_INPUT_RESOURCE_VER	13
4.1.1.11	NV_ENC_MEONLY_PARAMS_VER	13
4.1.1.12	NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS_VER	13
4.1.1.13	NV_ENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP	13
4.1.1.14	NV_ENC_PARAMS_RC_2_PASS_QUALITY	13
4.1.1.15	NV_ENC_PARAMS_RC_2_PASS_VBR	13
4.1.1.16	NV_ENC_PARAMS_RC_CBR2	13
4.1.1.17	NV_ENC_PARAMS_RC_VBR_MINQP	14
4.1.1.18	NV_ENC_PIC_PARAMS_VER	14

4.1.1.19	NV_ENC_PRESET_CONFIG_VER	14
4.1.1.20	NV_ENC_RC_PARAMS_VER	14
4.1.1.21	NV_ENC_RECONFIGURE_PARAMS_VER	14
4.1.1.22	NV_ENC_REGISTER_RESOURCE_VER	14
4.1.1.23	NV_ENC_SEQUENCE_PARAM_PAYLOAD_VER	14
4.1.1.24	NV_ENC_STAT_VER	14
4.1.2	Enumeration Type Documentation	14
4.1.2.1	NV_ENC_BUFFER_FORMAT	14
4.1.2.2	NV_ENC_CAPS	15
4.1.2.3	NV_ENC_DEVICE_TYPE	18
4.1.2.4	NV_ENC_H264_ADAPTIVE_TRANSFORM_MODE	18
4.1.2.5	NV_ENC_H264_BDIRECT_MODE	18
4.1.2.6	NV_ENC_H264_ENTROPY_CODING_MODE	18
4.1.2.7	NV_ENC_H264_FMO_MODE	18
4.1.2.8	NV_ENC_HEVC_CUSIZE	19
4.1.2.9	NV_ENC_INPUT_RESOURCE_TYPE	19
4.1.2.10	NV_ENC_LEVEL	19
4.1.2.11	NV_ENC_MEMORY_HEAP	19
4.1.2.12	NV_ENC_MV_PRECISION	19
4.1.2.13	NV_ENC_PARAMS_FRAME_FIELD_MODE	19
4.1.2.14	NV_ENC_PARAMS_RC_MODE	20
4.1.2.15	NV_ENC_PIC_FLAGS	20
4.1.2.16	NV_ENC_PIC_STRUCT	20
4.1.2.17	NV_ENC_PIC_TYPE	20
4.1.2.18	NV_ENC_STEREO_PACKING_MODE	21
4.1.2.19	NVENCSTATUS	21
4.2	NvEncodeAPI Functions	23
4.2.1	Function Documentation	26
4.2.1.1	NvEncCreateBitstreamBuffer	26
4.2.1.2	NvEncCreateInputBuffer	26
4.2.1.3	NvEncCreateMVBuffer	27
4.2.1.4	NvEncDestroyBitstreamBuffer	27
4.2.1.5	NvEncDestroyEncoder	27
4.2.1.6	NvEncDestroyInputBuffer	28
4.2.1.7	NvEncDestroyMVBuffer	28
4.2.1.8	NvEncEncodePicture	29
4.2.1.9	NvEncGetEncodeCaps	32

4.2.1.10	NvEncGetEncodeGUIDCount	32
4.2.1.11	NvEncGetEncodeGUIDs	32
4.2.1.12	NvEncGetEncodePresetConfig	33
4.2.1.13	NvEncGetEncodePresetCount	33
4.2.1.14	NvEncGetEncodePresetGUIDs	34
4.2.1.15	NvEncGetEncodeProfileGUIDCount	34
4.2.1.16	NvEncGetEncodeProfileGUIDs	35
4.2.1.17	NvEncGetEncodeStats	35
4.2.1.18	NvEncGetInputFormatCount	36
4.2.1.19	NvEncGetInputFormats	36
4.2.1.20	NvEncGetSequenceParams	37
4.2.1.21	NvEncInitializeEncoder	37
4.2.1.22	NvEncInvalidateRefFrames	39
4.2.1.23	NvEncLockBitstream	39
4.2.1.24	NvEncLockInputBuffer	40
4.2.1.25	NvEncMapInputResource	41
4.2.1.26	NvEncodeAPICreateInstance	41
4.2.1.27	NvEncodeAPIGetMaxSupportedVersion	41
4.2.1.28	NvEncOpenEncodeSession	42
4.2.1.29	NvEncOpenEncodeSessionEx	42
4.2.1.30	NvEncReconfigureEncoder	42
4.2.1.31	NvEncRegisterAsyncEvent	43
4.2.1.32	NvEncRegisterResource	43
4.2.1.33	NvEncRunMotionEstimationOnly	44
4.2.1.34	NvEncUnlockBitstream	44
4.2.1.35	NvEncUnlockInputBuffer	45
4.2.1.36	NvEncUnmapInputResource	45
4.2.1.37	NvEncUnregisterAsyncEvent	46
4.2.1.38	NvEncUnregisterResource	46
5	Data Structure Documentation	49
5.1	GUID Struct Reference	49
5.1.1	Detailed Description	49
5.1.2	Field Documentation	49
5.1.2.1	Data1	49
5.1.2.2	Data2	49
5.1.2.3	Data3	49

5.1.2.4	Data4	49
5.2	NV_ENC_CAPS_PARAM Struct Reference	50
5.2.1	Detailed Description	50
5.2.2	Field Documentation	50
5.2.2.1	capsToQuery	50
5.2.2.2	reserved	50
5.2.2.3	version	50
5.3	NV_ENC_CODEC_CONFIG Union Reference	51
5.3.1	Detailed Description	51
5.3.2	Field Documentation	51
5.3.2.1	h264Config	51
5.3.2.2	h264MeOnlyConfig	51
5.3.2.3	hevcConfig	51
5.3.2.4	hevcMeOnlyConfig	51
5.3.2.5	reserved	51
5.4	NV_ENC_CODEC_PIC_PARAMS Union Reference	52
5.4.1	Detailed Description	52
5.4.2	Field Documentation	52
5.4.2.1	h264PicParams	52
5.4.2.2	hevcPicParams	52
5.4.2.3	reserved	52
5.5	NV_ENC_CONFIG Struct Reference	53
5.5.1	Detailed Description	53
5.5.2	Field Documentation	53
5.5.2.1	encodeCodecConfig	53
5.5.2.2	frameFieldMode	53
5.5.2.3	frameIntervalP	53
5.5.2.4	gopLength	53
5.5.2.5	monoChromeEncoding	53
5.5.2.6	mvPrecision	54
5.5.2.7	profileGUID	54
5.5.2.8	rcParams	54
5.5.2.9	reserved	54
5.5.2.10	reserved2	54
5.5.2.11	version	54
5.6	NV_ENC_CONFIG_H264 Struct Reference	55
5.6.1	Detailed Description	56

5.6.2	Field Documentation	56
5.6.2.1	adaptiveTransformMode	56
5.6.2.2	bdirectMode	56
5.6.2.3	chromaFormatIDC	56
5.6.2.4	disableDeblockingFilterIDC	56
5.6.2.5	disableSPSPPS	56
5.6.2.6	enableConstrainedEncoding	56
5.6.2.7	enableIntraRefresh	56
5.6.2.8	enableLTR	56
5.6.2.9	enableStereoMVC	57
5.6.2.10	enableTemporalSVC	57
5.6.2.11	enableVFR	57
5.6.2.12	entropyCodingMode	57
5.6.2.13	fmoMode	57
5.6.2.14	h264VUIParameters	57
5.6.2.15	hierarchicalBFrames	57
5.6.2.16	hierarchicalPFrames	57
5.6.2.17	idrPeriod	57
5.6.2.18	intraRefreshCnt	57
5.6.2.19	intraRefreshPeriod	58
5.6.2.20	level	58
5.6.2.21	ltrNumFrames	58
5.6.2.22	ltrTrustMode	58
5.6.2.23	maxNumRefFrames	58
5.6.2.24	maxTemporalLayers	58
5.6.2.25	numTemporalLayers	58
5.6.2.26	outputAUD	58
5.6.2.27	outputBufferingPeriodSEI	58
5.6.2.28	outputFramePackingSEI	58
5.6.2.29	outputPictureTimingSEI	59
5.6.2.30	outputRecoveryPointSEI	59
5.6.2.31	ppsId	59
5.6.2.32	qpPrimeYZeroTransformBypassFlag	59
5.6.2.33	repeatSPSPPS	59
5.6.2.34	reserved1	59
5.6.2.35	reserved2	59
5.6.2.36	reservedBitFields	59

5.6.2.37	separateColourPlaneFlag	59
5.6.2.38	sliceMode	59
5.6.2.39	sliceModeData	60
5.6.2.40	spsId	60
5.6.2.41	stereoMode	60
5.6.2.42	useConstrainedIntraPred	60
5.7	NV_ENC_CONFIG_H264_MEONLY Struct Reference	61
5.7.1	Detailed Description	61
5.7.2	Field Documentation	61
5.7.2.1	bStereoEnable	61
5.7.2.2	disableIntraSearch	61
5.7.2.3	disablePartition16x16	61
5.7.2.4	disablePartition16x8	61
5.7.2.5	disablePartition8x16	61
5.7.2.6	disablePartition8x8	61
5.7.2.7	reserved	62
5.7.2.8	reserved1	62
5.7.2.9	reserved2	62
5.8	NV_ENC_CONFIG_H264_VUI_PARAMETERS Struct Reference	63
5.8.1	Detailed Description	63
5.8.2	Field Documentation	63
5.8.2.1	bitstreamRestrictionFlag	63
5.8.2.2	chromaSampleLocationBot	63
5.8.2.3	chromaSampleLocationFlag	63
5.8.2.4	chromaSampleLocationTop	63
5.8.2.5	colourDescriptionPresentFlag	63
5.8.2.6	colourMatrix	64
5.8.2.7	colourPrimaries	64
5.8.2.8	overscanInfo	64
5.8.2.9	overscanInfoPresentFlag	64
5.8.2.10	transferCharacteristics	64
5.8.2.11	videoFormat	64
5.8.2.12	videoFullRangeFlag	64
5.8.2.13	videoSignalTypePresentFlag	64
5.9	NV_ENC_CONFIG_HEVC Struct Reference	65
5.9.1	Detailed Description	65
5.9.2	Field Documentation	65

5.9.2.1	chromaFormatIDC	65
5.9.2.2	disableDeblockAcrossSliceBoundary	66
5.9.2.3	disableSPSPPS	66
5.9.2.4	enableIntraRefresh	66
5.9.2.5	enableLTR	66
5.9.2.6	hevcVUIParameters	66
5.9.2.7	idrPeriod	66
5.9.2.8	intraRefreshCnt	66
5.9.2.9	intraRefreshPeriod	66
5.9.2.10	level	66
5.9.2.11	ltrNumFrames	66
5.9.2.12	ltrTrustMode	67
5.9.2.13	maxCUSize	67
5.9.2.14	maxNumRefFramesInDPB	67
5.9.2.15	maxTemporalLayersMinus1	67
5.9.2.16	minCUSize	67
5.9.2.17	outputAUD	67
5.9.2.18	outputBufferingPeriodSEI	67
5.9.2.19	outputPictureTimingSEI	67
5.9.2.20	pixelBitDepthMinus8	67
5.9.2.21	ppsId	67
5.9.2.22	repeatSPSPPS	67
5.9.2.23	reserved	68
5.9.2.24	reserved1	68
5.9.2.25	reserved2	68
5.9.2.26	sliceMode	68
5.9.2.27	sliceModeData	68
5.9.2.28	spsId	68
5.9.2.29	tier	68
5.9.2.30	useConstrainedIntraPred	68
5.9.2.31	vpsId	68
5.10	NV_ENC_CONFIG_HEVC_MEONLY Struct Reference	69
5.10.1	Detailed Description	69
5.10.2	Field Documentation	69
5.10.2.1	reserved	69
5.10.2.2	reserved1	69
5.11	NV_ENC_CREATE_BITSTREAM_BUFFER Struct Reference	70

5.11.1	Detailed Description	70
5.11.2	Field Documentation	70
5.11.2.1	bitstreamBuffer	70
5.11.2.2	bitstreamBufferPtr	70
5.11.2.3	memoryHeap	70
5.11.2.4	reserved	70
5.11.2.5	reserved1	70
5.11.2.6	reserved2	70
5.11.2.7	size	71
5.11.2.8	version	71
5.12	NV_ENC_CREATE_INPUT_BUFFER Struct Reference	72
5.12.1	Detailed Description	72
5.12.2	Field Documentation	72
5.12.2.1	bufferFmt	72
5.12.2.2	height	72
5.12.2.3	inputBuffer	72
5.12.2.4	memoryHeap	72
5.12.2.5	pSysMemBuffer	72
5.12.2.6	reserved	72
5.12.2.7	reserved1	73
5.12.2.8	reserved2	73
5.12.2.9	version	73
5.12.2.10	width	73
5.13	NV_ENC_CREATE_MV_BUFFER Struct Reference	74
5.13.1	Detailed Description	74
5.13.2	Field Documentation	74
5.13.2.1	mvBuffer	74
5.13.2.2	reserved1	74
5.13.2.3	reserved2	74
5.13.2.4	version	74
5.14	NV_ENC_EVENT_PARAMS Struct Reference	75
5.14.1	Detailed Description	75
5.14.2	Field Documentation	75
5.14.2.1	completionEvent	75
5.14.2.2	reserved	75
5.14.2.3	reserved1	75
5.14.2.4	reserved2	75

5.14.2.5	version	75
5.15	NV_ENC_H264_MV_DATA Struct Reference	76
5.15.1	Detailed Description	76
5.15.2	Field Documentation	76
5.15.2.1	mbType	76
5.15.2.2	mv	76
5.15.2.3	partitionType	76
5.15.2.4	reserved	76
5.16	NV_ENC_HEVC_MV_DATA Struct Reference	77
5.16.1	Detailed Description	77
5.16.2	Field Documentation	77
5.16.2.1	cuSize	77
5.16.2.2	cuType	77
5.16.2.3	lastCUInCTB	77
5.16.2.4	mv	77
5.16.2.5	partitionMode	77
5.17	NV_ENC_INITIALIZE_PARAMS Struct Reference	78
5.17.1	Detailed Description	78
5.17.2	Field Documentation	78
5.17.2.1	darHeight	78
5.17.2.2	darWidth	78
5.17.2.3	enableEncodeAsync	78
5.17.2.4	enableExternalMEHints	79
5.17.2.5	enableMEOnlyMode	79
5.17.2.6	enablePTD	79
5.17.2.7	enableSubFrameWrite	79
5.17.2.8	encodeConfig	79
5.17.2.9	encodeGUID	79
5.17.2.10	encodeHeight	79
5.17.2.11	encodeWidth	79
5.17.2.12	frameRateDen	79
5.17.2.13	frameRateNum	80
5.17.2.14	maxEncodeHeight	80
5.17.2.15	maxEncodeWidth	80
5.17.2.16	maxMEHintCountsPerBlock	80
5.17.2.17	presetGUID	80
5.17.2.18	privData	80

5.17.2.19	privDataSize	80
5.17.2.20	reportSliceOffsets	80
5.17.2.21	reserved	80
5.17.2.22	reserved2	81
5.17.2.23	reservedBitFields	81
5.17.2.24	version	81
5.18	NV_ENC_LOCK_BITSTREAM Struct Reference	82
5.18.1	Detailed Description	82
5.18.2	Field Documentation	82
5.18.2.1	bitstreamBufferPtr	82
5.18.2.2	bitstreamSizeInBytes	82
5.18.2.3	doNotWait	82
5.18.2.4	frameAvgQP	83
5.18.2.5	frameIdx	83
5.18.2.6	frameSatd	83
5.18.2.7	hwEncodeStatus	83
5.18.2.8	ltrFrame	83
5.18.2.9	ltrFrameBitmap	83
5.18.2.10	ltrFrameIdx	83
5.18.2.11	numSlices	83
5.18.2.12	outputBitstream	83
5.18.2.13	outputDuration	83
5.18.2.14	outputTimeStamp	83
5.18.2.15	pictureStruct	84
5.18.2.16	pictureType	84
5.18.2.17	reserved	84
5.18.2.18	reserved2	84
5.18.2.19	reservedBitFields	84
5.18.2.20	sliceOffsets	84
5.18.2.21	version	84
5.19	NV_ENC_LOCK_INPUT_BUFFER Struct Reference	85
5.19.1	Detailed Description	85
5.19.2	Field Documentation	85
5.19.2.1	bufferDataPtr	85
5.19.2.2	doNotWait	85
5.19.2.3	inputBuffer	85
5.19.2.4	pitch	85

5.19.2.5	reserved1	85
5.19.2.6	reserved2	85
5.19.2.7	reservedBitFields	86
5.19.2.8	version	86
5.20	NV_ENC_MAP_INPUT_RESOURCE Struct Reference	87
5.20.1	Detailed Description	87
5.20.2	Field Documentation	87
5.20.2.1	inputResource	87
5.20.2.2	mappedBufferFmt	87
5.20.2.3	mappedResource	87
5.20.2.4	registeredResource	87
5.20.2.5	reserved1	87
5.20.2.6	reserved2	87
5.20.2.7	subResourceIndex	88
5.20.2.8	version	88
5.21	NV_ENC_MEONLY_PARAMS Struct Reference	89
5.21.1	Detailed Description	89
5.21.2	Field Documentation	89
5.21.2.1	bufferFmt	89
5.21.2.2	completionEvent	89
5.21.2.3	inputBuffer	89
5.21.2.4	inputHeight	89
5.21.2.5	inputWidth	89
5.21.2.6	mvBuffer	90
5.21.2.7	referenceFrame	90
5.21.2.8	reserved1	90
5.21.2.9	reserved2	90
5.21.2.10	version	90
5.21.2.11	viewID	90
5.22	NV_ENC_MVECTOR Struct Reference	91
5.22.1	Detailed Description	91
5.22.2	Field Documentation	91
5.22.2.1	mvx	91
5.22.2.2	mvy	91
5.23	NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS Struct Reference	92
5.23.1	Detailed Description	92
5.23.2	Field Documentation	92

5.23.2.1	apiVersion	92
5.23.2.2	device	92
5.23.2.3	deviceType	92
5.23.2.4	reserved	92
5.23.2.5	reserved1	92
5.23.2.6	reserved2	92
5.23.2.7	version	93
5.24	NV_ENC_PIC_PARAMS Struct Reference	94
5.24.1	Detailed Description	94
5.24.2	Field Documentation	94
5.24.2.1	bufferFmt	94
5.24.2.2	codecPicParams	94
5.24.2.3	completionEvent	95
5.24.2.4	encodePicFlags	95
5.24.2.5	frameIdx	95
5.24.2.6	inputBuffer	95
5.24.2.7	inputDuration	95
5.24.2.8	inputHeight	95
5.24.2.9	inputPitch	95
5.24.2.10	inputTimeStamp	95
5.24.2.11	inputWidth	95
5.24.2.12	meExternalHints	95
5.24.2.13	meHintCountsPerBlock	96
5.24.2.14	meHintRefPicDist	96
5.24.2.15	outputBitstream	96
5.24.2.16	pictureStruct	96
5.24.2.17	pictureType	96
5.24.2.18	qpDeltaMap	96
5.24.2.19	qpDeltaMapSize	96
5.24.2.20	reserved1	96
5.24.2.21	reserved2	96
5.24.2.22	reserved3	97
5.24.2.23	reserved4	97
5.24.2.24	reservedBitFields	97
5.24.2.25	version	97
5.25	NV_ENC_PIC_PARAMS_H264 Struct Reference	98
5.25.1	Detailed Description	98

5.25.2	Field Documentation	98
5.25.2.1	colourPlaneId	98
5.25.2.2	constrainedFrame	98
5.25.2.3	displayPOCSyntax	98
5.25.2.4	forceIntraRefreshWithFrameCnt	99
5.25.2.5	ltrMarkFrame	99
5.25.2.6	ltrMarkFrameIdx	99
5.25.2.7	ltrUsageMode	99
5.25.2.8	ltrUseFrameBitmap	99
5.25.2.9	ltrUseFrames	99
5.25.2.10	refPicFlag	99
5.25.2.11	reserved	99
5.25.2.12	reserved2	99
5.25.2.13	reserved3	99
5.25.2.14	reservedBitFields	99
5.25.2.15	seiPayloadArray	100
5.25.2.16	seiPayloadArrayCnt	100
5.25.2.17	sliceMode	100
5.25.2.18	sliceModeData	100
5.25.2.19	sliceModeDataUpdate	100
5.25.2.20	sliceTypeArrayCnt	100
5.25.2.21	sliceTypeData	100
5.26	NV_ENC_PIC_PARAMS_HEVC Struct Reference	101
5.26.1	Detailed Description	101
5.26.2	Field Documentation	101
5.26.2.1	constrainedFrame	101
5.26.2.2	displayPOCSyntax	101
5.26.2.3	forceIntraRefreshWithFrameCnt	101
5.26.2.4	ltrMarkFrame	102
5.26.2.5	ltrMarkFrameIdx	102
5.26.2.6	ltrUsageMode	102
5.26.2.7	ltrUseFrameBitmap	102
5.26.2.8	ltrUseFrames	102
5.26.2.9	refPicFlag	102
5.26.2.10	reserved	102
5.26.2.11	reserved2	102
5.26.2.12	reserved3	102

5.26.2.13 reservedBitFields	102
5.26.2.14 seiPayloadArray	102
5.26.2.15 seiPayloadArrayCnt	103
5.26.2.16 sliceMode	103
5.26.2.17 sliceModeData	103
5.26.2.18 sliceModeDataUpdate	103
5.26.2.19 sliceTypeArrayCnt	103
5.26.2.20 sliceTypeData	103
5.26.2.21 temporalId	103
5.27 NV_ENC_PRESET_CONFIG Struct Reference	104
5.27.1 Detailed Description	104
5.27.2 Field Documentation	104
5.27.2.1 presetCfg	104
5.27.2.2 reserved1	104
5.27.2.3 reserved2	104
5.27.2.4 version	104
5.28 NV_ENC_QP Struct Reference	105
5.28.1 Detailed Description	105
5.29 NV_ENC_RC_PARAMS Struct Reference	106
5.29.1 Detailed Description	106
5.29.2 Field Documentation	106
5.29.2.1 aqStrength	106
5.29.2.2 averageBitRate	106
5.29.2.3 constQP	107
5.29.2.4 disableBadapt	107
5.29.2.5 disableIadapt	107
5.29.2.6 enableAQ	107
5.29.2.7 enableExtQPDeltaMap	107
5.29.2.8 enableInitialRCQP	107
5.29.2.9 enableLookahead	107
5.29.2.10 enableMaxQP	107
5.29.2.11 enableMinQP	107
5.29.2.12 enableNonRefP	107
5.29.2.13 enableTemporalAQ	107
5.29.2.14 initialRCQP	108
5.29.2.15 lookaheadDepth	108
5.29.2.16 maxBitRate	108

5.29.2.17	maxQP	108
5.29.2.18	minQP	108
5.29.2.19	rateControlMode	108
5.29.2.20	reservedBitFields	108
5.29.2.21	strictGOPTarget	108
5.29.2.22	targetQuality	108
5.29.2.23	temporalLayerIdxMask	108
5.29.2.24	temporalLayerQP	108
5.29.2.25	vbvBufferSize	109
5.29.2.26	vbvInitialDelay	109
5.29.2.27	zeroReorderDelay	109
5.30	NV_ENC_RECONFIGURE_PARAMS Struct Reference	110
5.30.1	Detailed Description	110
5.30.2	Field Documentation	110
5.30.2.1	forceIDR	110
5.30.2.2	reInitEncodeParams	110
5.30.2.3	resetEncoder	110
5.30.2.4	version	110
5.31	NV_ENC_REGISTER_RESOURCE Struct Reference	111
5.31.1	Detailed Description	111
5.31.2	Field Documentation	111
5.31.2.1	bufferFormat	111
5.31.2.2	height	111
5.31.2.3	pitch	111
5.31.2.4	registeredResource	111
5.31.2.5	reserved1	111
5.31.2.6	reserved2	111
5.31.2.7	resourceToRegister	112
5.31.2.8	resourceType	112
5.31.2.9	subResourceIndex	112
5.31.2.10	version	112
5.31.2.11	width	112
5.32	NV_ENC_SEI_PAYLOAD Struct Reference	113
5.32.1	Detailed Description	113
5.32.2	Field Documentation	113
5.32.2.1	payload	113
5.32.2.2	payloadSize	113

5.32.2.3	payloadType	113
5.33	NV_ENC_SEQUENCE_PARAM_PAYLOAD Struct Reference	114
5.33.1	Detailed Description	114
5.33.2	Field Documentation	114
5.33.2.1	inBufferSize	114
5.33.2.2	outSPSPSPayloadSize	114
5.33.2.3	ppsId	114
5.33.2.4	reserved	114
5.33.2.5	reserved2	114
5.33.2.6	spsId	114
5.33.2.7	spsppsBuffer	115
5.33.2.8	version	115
5.34	NV_ENC_STAT Struct Reference	116
5.34.1	Detailed Description	116
5.34.2	Field Documentation	116
5.34.2.1	bitStreamSize	116
5.34.2.2	lastValidByteOffset	116
5.34.2.3	outputBitStream	116
5.34.2.4	picIdx	116
5.34.2.5	picType	116
5.34.2.6	reserved	116
5.34.2.7	reserved1	117
5.34.2.8	reserved2	117
5.34.2.9	sliceOffsets	117
5.34.2.10	version	117
5.35	NV_ENCODE_API_FUNCTION_LIST Struct Reference	118
5.35.1	Detailed Description	118
5.35.2	Field Documentation	119
5.35.2.1	nvEncCreateBitstreamBuffer	119
5.35.2.2	nvEncCreateInputBuffer	119
5.35.2.3	nvEncCreateMVBuffer	119
5.35.2.4	nvEncDestroyBitstreamBuffer	119
5.35.2.5	nvEncDestroyEncoder	119
5.35.2.6	nvEncDestroyInputBuffer	119
5.35.2.7	nvEncDestroyMVBuffer	119
5.35.2.8	nvEncEncodePicture	119
5.35.2.9	nvEncGetEncodeCaps	119

5.35.2.10	<code>nvEncGetEncodeGUIDCount</code>	119
5.35.2.11	<code>nvEncGetEncodeGUIDs</code>	120
5.35.2.12	<code>nvEncGetEncodePresetConfig</code>	120
5.35.2.13	<code>nvEncGetEncodePresetCount</code>	120
5.35.2.14	<code>nvEncGetEncodePresetGUIDs</code>	120
5.35.2.15	<code>nvEncGetEncodeProfileGUIDCount</code>	120
5.35.2.16	<code>nvEncGetEncodeProfileGUIDs</code>	120
5.35.2.17	<code>nvEncGetEncodeStats</code>	120
5.35.2.18	<code>nvEncGetInputFormatCount</code>	120
5.35.2.19	<code>nvEncGetInputFormats</code>	120
5.35.2.20	<code>nvEncGetSequenceParams</code>	120
5.35.2.21	<code>nvEncInitializeEncoder</code>	121
5.35.2.22	<code>nvEncInvalidateRefFrames</code>	121
5.35.2.23	<code>nvEncLockBitstream</code>	121
5.35.2.24	<code>nvEncLockInputBuffer</code>	121
5.35.2.25	<code>nvEncMapInputResource</code>	121
5.35.2.26	<code>nvEncOpenEncodeSession</code>	121
5.35.2.27	<code>nvEncOpenEncodeSessionEx</code>	121
5.35.2.28	<code>nvEncReconfigureEncoder</code>	121
5.35.2.29	<code>nvEncRegisterAsyncEvent</code>	121
5.35.2.30	<code>nvEncRegisterResource</code>	121
5.35.2.31	<code>nvEncRunMotionEstimationOnly</code>	122
5.35.2.32	<code>nvEncUnlockBitstream</code>	122
5.35.2.33	<code>nvEncUnlockInputBuffer</code>	122
5.35.2.34	<code>nvEncUnmapInputResource</code>	122
5.35.2.35	<code>nvEncUnregisterAsyncEvent</code>	122
5.35.2.36	<code>nvEncUnregisterResource</code>	122
5.35.2.37	<code>reserved</code>	122
5.35.2.38	<code>reserved2</code>	122
5.35.2.39	<code>version</code>	122
5.36	<code>NVENC_EXTERNAL_ME_HINT</code> Struct Reference	123
5.36.1	Detailed Description	123
5.36.2	Field Documentation	123
5.36.2.1	<code>dir</code>	123
5.36.2.2	<code>lastOfMB</code>	123
5.36.2.3	<code>lastofPart</code>	123
5.36.2.4	<code>mvx</code>	123

5.36.2.5	mvy	123
5.36.2.6	partType	123
5.36.2.7	refidx	124
5.37	NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE Struct Reference	125
5.37.1	Detailed Description	125
5.37.2	Field Documentation	125
5.37.2.1	numCandsPerBlk16x16	125
5.37.2.2	numCandsPerBlk16x8	125
5.37.2.3	numCandsPerBlk8x16	125
5.37.2.4	numCandsPerBlk8x8	125
5.37.2.5	reserved	125
5.37.2.6	reserved1	125
5.38	NVENC_RECT Struct Reference	126
5.38.1	Detailed Description	126
5.38.2	Field Documentation	126
5.38.2.1	bottom	126
5.38.2.2	left	126
5.38.2.3	right	126
5.38.2.4	top	126

Chapter 1

Legal Notice

Copyright (c) 2011-2016 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	7
NvEncodeAPI Functions	23

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

GUID	49
NV_ENC_CAPS_PARAM	50
NV_ENC_CODEC_CONFIG	51
NV_ENC_CODEC_PIC_PARAMS	52
NV_ENC_CONFIG	53
NV_ENC_CONFIG_H264	55
NV_ENC_CONFIG_H264_MEONLY	61
NV_ENC_CONFIG_H264_VUI_PARAMETERS	63
NV_ENC_CONFIG_HEVC	65
NV_ENC_CONFIG_HEVC_MEONLY	69
NV_ENC_CREATE_BITSTREAM_BUFFER	70
NV_ENC_CREATE_INPUT_BUFFER	72
NV_ENC_CREATE_MV_BUFFER	74
NV_ENC_EVENT_PARAMS	75
NV_ENC_H264_MV_DATA	76
NV_ENC_HEVC_MV_DATA	77
NV_ENC_INITIALIZE_PARAMS	78
NV_ENC_LOCK_BITSTREAM	82
NV_ENC_LOCK_INPUT_BUFFER	85
NV_ENC_MAP_INPUT_RESOURCE	87
NV_ENC_MEONLY_PARAMS	89
NV_ENC_MVECTOR	91
NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS	92
NV_ENC_PIC_PARAMS	94
NV_ENC_PIC_PARAMS_H264	98
NV_ENC_PIC_PARAMS_HEVC	101
NV_ENC_PRESET_CONFIG	104
NV_ENC_QP	105
NV_ENC_RC_PARAMS	106
NV_ENC_RECONFIGURE_PARAMS	110
NV_ENC_REGISTER_RESOURCE	111
NV_ENC_SEI_PAYLOAD	113
NV_ENC_SEQUENCE_PARAM_PAYLOAD	114

NV_ENC_STAT	116
NV_ENCODE_API_FUNCTION_LIST	118
NVENC_EXTERNAL_ME_HINT	123
NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE	125
NVENC_RECT	126

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_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_SIDE BYSIDE = 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 }
- enum NV_ENC_DEVICE_TYPE {
NV_ENC_DEVICE_TYPE_DIRECTX = 0x0,
NV_ENC_DEVICE_TYPE_CUDA = 0x1 }
- 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_EXPOSED_COUNT }

```

- enum NV_ENC_HEVC_CUSIZE

4.1.1 Define Documentation

4.1.1.1 #define NV_ENC_CAPS_PARAM_VER NVENC_API_STRUCT_VERSION(1)

NV_ENC_CAPS_PARAM struct version.

4.1.1.2 #define NV_ENC_CONFIG_VER (NVENC_API_STRUCT_VERSION(6) | (1 << 31))

macro for constructing the version field of NV_ENC_CONFIG

4.1.1.3 #define NV_ENC_CREATE_BITSTREAM_BUFFER_VER NVENC_API_STRUCT_VERSION(1)

NV_ENC_CREATE_BITSTREAM_BUFFER struct version.

4.1.1.4 #define NV_ENC_CREATE_INPUT_BUFFER_VER NVENC_API_STRUCT_VERSION(1)

NV_ENC_CREATE_INPUT_BUFFER struct version.

4.1.1.5 #define NV_ENC_CREATE_MV_BUFFER_VER NVENC_API_STRUCT_VERSION(1)

NV_ENC_CREATE_MV_BUFFER struct version

4.1.1.6 #define NV_ENC_EVENT_PARAMS_VER NVENCAPI_STRUCT_VERSION(1)

Macro for constructing the version field of `_NV_ENC_EVENT_PARAMS`

4.1.1.7 #define NV_ENC_INITIALIZE_PARAMS_VER (NVENCAPI_STRUCT_VERSION(5) | (1 << 31))

macro for constructing the version field of `_NV_ENC_INITIALIZE_PARAMS`

4.1.1.8 #define NV_ENC_LOCK_BITSTREAM_VER NVENCAPI_STRUCT_VERSION(1)

Macro for constructing the version field of `_NV_ENC_LOCK_BITSTREAM`

4.1.1.9 #define NV_ENC_LOCK_INPUT_BUFFER_VER NVENCAPI_STRUCT_VERSION(1)

Macro for constructing the version field of `_NV_ENC_LOCK_INPUT_BUFFER`

4.1.1.10 #define NV_ENC_MAP_INPUT_RESOURCE_VER NVENCAPI_STRUCT_VERSION(4)

Macro for constructing the version field of `_NV_ENC_MAP_INPUT_RESOURCE`

4.1.1.11 #define NV_ENC_MEONLY_PARAMS_VER NVENCAPI_STRUCT_VERSION(3)

[NV_ENC_MEONLY_PARAMS](#) struct version

4.1.1.12 #define NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS_VER NVENCAPI_STRUCT_VERSION(1)

Macro for constructing the version field of `_NV_ENC_OPEN_ENCODE_SESSIONEX_PARAMS`

4.1.1.13 #define NV_ENC_PARAMS_RC_2_PASS_FRAME_SIZE_CAP NV_ENC_PARAMS_RC_CBR_HQ

Deprecated

4.1.1.14 #define NV_ENC_PARAMS_RC_2_PASS_QUALITY NV_ENC_PARAMS_RC_CBR_LOWDELAY_HQ

Deprecated

4.1.1.15 #define NV_ENC_PARAMS_RC_2_PASS_VBR NV_ENC_PARAMS_RC_VBR_HQ

Deprecated

4.1.1.16 #define NV_ENC_PARAMS_RC_CBR2 NV_ENC_PARAMS_RC_CBR

Deprecated

4.1.1.17 #define NV_ENC_PARAMS_RC_VBR_MINQP (NV_ENC_PARAMS_RC_MODE)0x4

Deprecated

4.1.1.18 #define NV_ENC_PIC_PARAMS_VER (NVENCAPI_STRUCT_VERSION(4) | (1 << 31))

Macro for constructing the version field of `_NV_ENC_PIC_PARAMS`

4.1.1.19 #define NV_ENC_PRESET_CONFIG_VER (NVENCAPI_STRUCT_VERSION(4) | (1 << 31))

macro for constructing the version field of `_NV_ENC_PRESET_CONFIG`

4.1.1.20 #define NV_ENC_RC_PARAMS_VER NVENCAPI_STRUCT_VERSION(1)

macro for constructing the version field of `_NV_ENC_RC_PARAMS`

4.1.1.21 #define NV_ENC_RECONFIGURE_PARAMS_VER (NVENCAPI_STRUCT_VERSION(1) | (1 << 31))

macro for constructing the version field of `_NV_ENC_RECONFIGURE_PARAMS`

4.1.1.22 #define NV_ENC_REGISTER_RESOURCE_VER NVENCAPI_STRUCT_VERSION(3)

Macro for constructing the version field of `_NV_ENC_REGISTER_RESOURCE`

4.1.1.23 #define NV_ENC_SEQUENCE_PARAM_PAYLOAD_VER NVENCAPI_STRUCT_VERSION(1)

Macro for constructing the version field of `_NV_ENC_SEQUENCE_PARAM_PAYLOAD`

4.1.1.24 #define NV_ENC_STAT_VER NVENCAPI_STRUCT_VERSION(1)

Macro for constructing the version field of `_NV_ENC_STAT`

4.1.2 Enumeration Type Documentation**4.1.2.1 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
- NV_ENC_BUFFER_FORMAT_ARGB10*** 10 bit Packed A2R10G10B10. Each pixel of size 2 bytes. Most Significant 10 bits contain pixel data.
- NV_ENC_BUFFER_FORMAT_AYUV*** 8 bit Packed A8Y8U8V8
- NV_ENC_BUFFER_FORMAT_ABGR*** 8 bit Packed A8B8G8R8
- NV_ENC_BUFFER_FORMAT_ABGR10*** 10 bit Packed A2B10G10R10. Each pixel of size 2 bytes. Most Significant 10 bits contain pixel data.

4.1.2.2 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 Bufer 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.
- 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_EXPOSED_COUNT*** Reserved - Not to be used by clients.

4.1.2.3 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

4.1.2.4 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

4.1.2.5 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

4.1.2.6 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

4.1.2.7 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

4.1.2.8 enum NV_ENC_HEVC_CUSIZE

HEVC CU SIZE

4.1.2.9 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

4.1.2.10 enum NV_ENC_LEVEL

Encoding levels

4.1.2.11 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

4.1.2.12 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

4.1.2.13 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

4.1.2.14 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)

4.1.2.15 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

4.1.2.16 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

4.1.2.17 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

4.1.2.18 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

4.1.2.19 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_REGISTERED 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.

4.2 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](#) input-Buffer)
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](#) bit-streamBuffer)
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](#) bitstream-Buffer)
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](#) input-Buffer)
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 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](#)

4.2.1.2 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](#)

4.2.1.3 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](#)

4.2.1.4 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](#)

4.2.1.5 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](#)

4.2.1.6 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](#)

4.2.1.7 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

4.2.1.8 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`. In case 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`

4.2.1.9 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](#)

4.2.1.10 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](#)

4.2.1.11 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 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](#)

4.2.1.12 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 guid. 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 guid.

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](#)

4.2.1.13 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](#)

4.2.1.14 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](#)

4.2.1.15 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 guids 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](#)

4.2.1.16 NVENCSTATUS NVENCAPI NvEncGetEncodeProfileGUIDs (void * *encoder*, GUID *encodeGUID*, GUID * *profileGUIDs*, uint32_t *guidArraySize*, uint32_t * *GUIDCount*)

The function returns an array of supported profile guids for a particular codec guid. The client must allocate an array where the NvEncodeAPI interface can populate the profile guids. 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 guids 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](#)

4.2.1.17 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`

4.2.1.18 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`

4.2.1.19 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 [NvEnc-CreateInputBuffer\(\)](#) 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

4.2.1.20 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

4.2.1.21 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](#)

4.2.1.22 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 encoder 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](#)

4.2.1.23 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`

4.2.1.24 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`

4.2.1.25 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 direct3d or cuda 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

4.2.1.26 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

4.2.1.27 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](#)

4.2.1.28 NVENCSTATUS NVENCAPI NvEncOpenEncodeSession (void * *device*, uint32_t *deviceType*, void ** *encoder*)

Deprecated.

Returns:

[NV_ENC_ERR_INVALID_CALL](#)

4.2.1.29 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/CUDA interface in the **device* parameter. 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](#)

4.2.1.30 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](#)

**4.2.1.31 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](#)

**4.2.1.32 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](#)

4.2.1.33 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](#)

4.2.1.34 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](#)

4.2.1.35 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](#)

4.2.1.36 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](#)

4.2.1.37 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](#)

4.2.1.38 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

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

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 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.

5.2.2.2 [uint32_t](#) NV_ENC_CAPS_PARAM::reserved[62]

[in]: Reserved and must be set to 0

5.2.2.3 [uint32_t](#) NV_ENC_CAPS_PARAM::version

[in]: Struct version. Must be set to [NV_ENC_CAPS_PARAM_VER](#)

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

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_H264_MEONLY NV_ENC_CODEC_CONFIG::h264MeOnlyConfig

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

5.3.2.3 NV_ENC_CONFIG_HEVC NV_ENC_CODEC_CONFIG::hevcConfig

[in]: Specifies the HEVC-specific 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. Currently unsupported and must not to be used.

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

Encoder configuration parameters to be set during initialization.

5.5.2 Field Documentation

5.5.2.1 NV_ENC_CODEC_CONFIG NV_ENC_CONFIG::encodeCodecConfig

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

5.5.2.2 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.

5.5.2.3 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.

5.5.2.4 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.

5.5.2.5 uint32_t NV_ENC_CONFIG::monoChromeEncoding

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

5.5.2.6 NV_ENC_MV_PRECISION NV_ENC_CONFIG::mvPrecision

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

5.5.2.7 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.

5.5.2.8 NV_ENC_RC_PARAMS NV_ENC_CONFIG::rcParams

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

5.5.2.9 uint32_t NV_ENC_CONFIG::reserved[278]

[in]: Reserved and must be set to 0

5.5.2.10 void* NV_ENC_CONFIG::reserved2[64]

[in]: Reserved and must be set to NULL

5.5.2.11 uint32_t NV_ENC_CONFIG::version

[in]: Struct version. Must be set to [NV_ENC_CONFIG_VER](#).

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

H264 encoder configuration parameters

5.6.2 Field Documentation

5.6.2.1 `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.

5.6.2.2 `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.

5.6.2.3 `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.

5.6.2.4 `uint32_t NV_ENC_CONFIG_H264::disableDeblockingFilterIDC`

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

5.6.2.5 `uint32_t NV_ENC_CONFIG_H264::disableSPSPPS`

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

5.6.2.6 `uint32_t NV_ENC_CONFIG_H264::enableConstrainedEncoding`

[in]: Set this to 1 to enable constrainedFrame encoding where each slice in the constarined picture is independent of other slices Check support for constrained encoding using [NV_ENC_CAPS_SUPPORT_CONSTRAINED_ENCODING](#) caps.

5.6.2.7 `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

5.6.2.8 `uint32_t NV_ENC_CONFIG_H264::enableLTR`

[in]: Currently this feature is not available and must be set to 0. Set to 1 to enable LTR support and auto-mark the first

5.6.2.9 uint32_t NV_ENC_CONFIG_H264::enableStereoMVC

[in]: Set to 1 to enable stereo MVC

5.6.2.10 uint32_t NV_ENC_CONFIG_H264::enableTemporalSVC

[in]: Set to 1 to enable SVC temporal

5.6.2.11 uint32_t NV_ENC_CONFIG_H264::enableVFR

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

5.6.2.12 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.

5.6.2.13 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.

5.6.2.14 NV_ENC_CONFIG_H264_VUI_PARAMETERS NV_ENC_CONFIG_H264::h264VUIParameters

[in]: Specifies the H264 video usability info pamameters

5.6.2.15 uint32_t NV_ENC_CONFIG_H264::hierarchicalBFrames

[in]: Set to 1 to enable hierarchical BFrames

5.6.2.16 uint32_t NV_ENC_CONFIG_H264::hierarchicalPFrames

[in]: Set to 1 to enable hierarchical PFrames

5.6.2.17 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.

5.6.2.18 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

5.6.2.19 `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`.

5.6.2.20 `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.

5.6.2.21 `uint32_t NV_ENC_CONFIG_H264::ltrNumFrames`

[in]: Specifies the number of LTR frames used. If `ltrTrustMode=1`, encoder will mark first `numLTRFrames` base layer reference frames within each IDR interval as LTR. If `ltrMarkFrame=1`, `ltrNumFrames` specifies maximum number of ltr frames in DPB. If `ltrNumFrames` value is more than DPB size(`maxNumRefFrames`) encoder will take decision on its own.

5.6.2.22 `uint32_t NV_ENC_CONFIG_H264::ltrTrustMode`

[in]: Specifies the LTR operating mode. Set to 0 to disallow encoding using LTR frames until later specified. Set to 1 to allow encoding using LTR frames unless later invalidated.

5.6.2.23 `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.

5.6.2.24 `uint32_t NV_ENC_CONFIG_H264::maxTemporalLayers`

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

5.6.2.25 `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]`

5.6.2.26 `uint32_t NV_ENC_CONFIG_H264::outputAUD`

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

5.6.2.27 `uint32_t NV_ENC_CONFIG_H264::outputBufferingPeriodSEI`

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

5.6.2.28 `uint32_t NV_ENC_CONFIG_H264::outputFramePackingSEI`

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

5.6.2.29 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

5.6.2.30 uint32_t NV_ENC_CONFIG_H264::outputRecoveryPointSEI

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

5.6.2.31 uint32_t NV_ENC_CONFIG_H264::ppsId

[in]: Specifies the PPS id of the picture header. Currently reserved and must be set to 0.

5.6.2.32 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.

5.6.2.33 uint32_t NV_ENC_CONFIG_H264::repeatSPSPS

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

5.6.2.34 uint32_t NV_ENC_CONFIG_H264::reserved1[270]

[in]: Reserved and must be set to 0

5.6.2.35 void* NV_ENC_CONFIG_H264::reserved2[64]

[in]: Reserved and must be set to NULL

5.6.2.36 uint32_t NV_ENC_CONFIG_H264::reservedBitFields

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

5.6.2.37 uint32_t NV_ENC_CONFIG_H264::separateColourPlaneFlag

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

5.6.2.38 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

5.6.2.39 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

5.6.2.40 uint32_t NV_ENC_CONFIG_H264::spsId

[in]: Specifies the SPS id of the sequence header. Currently reserved and must be set to 0.

5.6.2.41 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

5.6.2.42 uint32_t NV_ENC_CONFIG_H264::useConstrainedIntraPred

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

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

H264 encoder configuration parameters for ME only Mode

5.7.2 Field Documentation

5.7.2.1 uint32_t NV_ENC_CONFIG_H264_MEONLY::bStereoEnable

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

5.7.2.2 uint32_t NV_ENC_CONFIG_H264_MEONLY::disableIntraSearch

[in]: Disable Intra search during MotionEstimation

5.7.2.3 uint32_t NV_ENC_CONFIG_H264_MEONLY::disablePartition16x16

[in]: Disable MotionEstimation on 16x16 blocks

5.7.2.4 uint32_t NV_ENC_CONFIG_H264_MEONLY::disablePartition16x8

[in]: Disable MotionEstimation on 16x8 blocks

5.7.2.5 uint32_t NV_ENC_CONFIG_H264_MEONLY::disablePartition8x16

[in]: Disable MotionEstimation on 8x16 blocks

5.7.2.6 uint32_t NV_ENC_CONFIG_H264_MEONLY::disablePartition8x8

[in]: Disable MotionEstimation on 8x8 blocks

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

H264 Video Usability Info parameters

5.8.2 Field Documentation

5.8.2.1 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.

5.8.2.2 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)

5.8.2.3 uint32_t NV_ENC_CONFIG_H264_VUI_PARAMETERS::chromaSampleLocationFlag

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

5.8.2.4 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)

5.8.2.5 uint32_t NV_ENC_CONFIG_H264_VUI_PARAMETERS::colourDescriptionPresentFlag

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

5.8.2.6 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).

5.8.2.7 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::overscanInfo

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

5.8.2.9 uint32_t NV_ENC_CONFIG_H264_VUI_PARAMETERS::overscanInfoPresentFlag

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

5.8.2.10 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)

5.8.2.11 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).

5.8.2.12 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).

5.8.2.13 uint32_t NV_ENC_CONFIG_H264_VUI_PARAMETERS::videoSignalTypePresentFlag

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

5.9 NV_ENC_CONFIG_HEVC Struct Reference

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

Data Fields

- uint32_t [level](#)
- uint32_t [tier](#)
- [NV_ENC_HEVC_CUSIZE minCUSize](#)
- [NV_ENC_HEVC_CUSIZE maxCUSize](#)
- 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

HEVC encoder configuration parameters to be set during initialization.

5.9.2 Field Documentation

5.9.2.1 uint32_t NV_ENC_CONFIG_HEVC::chromaFormatIDC

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

5.9.2.2 `uint32_t NV_ENC_CONFIG_HEVC::disableDeblockAcrossSliceBoundary`

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

5.9.2.3 `uint32_t NV_ENC_CONFIG_HEVC::disableSPSPPS`

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

5.9.2.4 `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

5.9.2.5 `uint32_t NV_ENC_CONFIG_HEVC::enableLTR`

[in]: Set 1 to enable use of long term reference pictures for inter prediction.

5.9.2.6 `NV_ENC_CONFIG_HEVC_VUI_PARAMETERS NV_ENC_CONFIG_HEVC::hevcVUIParameters`

[in]: Specifies the HEVC video usability info parameters

5.9.2.7 `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.

5.9.2.8 `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`

5.9.2.9 `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`.

5.9.2.10 `uint32_t NV_ENC_CONFIG_HEVC::level`

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

5.9.2.11 `uint32_t NV_ENC_CONFIG_HEVC::ltrNumFrames`

[in]: Specifies the number of LTR frames used. If `ltrTrustMode=1`, encoder will mark first `numLTRFrames` base layer reference frames within each IDR interval as LTR. If `ltrMarkFrame=1`, `ltrNumFrames` specifies maximum number of ltr frames in DPB. If `ltrNumFrames` value is more than DPB size(`maxNumRefFramesInDPB`) encoder will take decision on its own.

5.9.2.12 uint32_t NV_ENC_CONFIG_HEVC::ltrTrustMode

[in]: Specifies the LTR operating mode. Set to 0 to disallow encoding using LTR frames until later specified. Set to 1 to allow encoding using LTR frames unless later invalidated.

5.9.2.13 NV_ENC_HEVC_CUSIZE NV_ENC_CONFIG_HEVC::maxCUSize

[in]: Specifies the maximum size of luma coding unit. Currently NVENC SDK only supports maxCUSize equal to NV_ENC_HEVC_CUSIZE_32x32.

5.9.2.14 uint32_t NV_ENC_CONFIG_HEVC::maxNumRefFramesInDPB

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

5.9.2.15 uint32_t NV_ENC_CONFIG_HEVC::maxTemporalLayersMinus1

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

5.9.2.16 NV_ENC_HEVC_CUSIZE NV_ENC_CONFIG_HEVC::minCUSize

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

5.9.2.17 uint32_t NV_ENC_CONFIG_HEVC::outputAUD

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

5.9.2.18 uint32_t NV_ENC_CONFIG_HEVC::outputBufferingPeriodSEI

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

5.9.2.19 uint32_t NV_ENC_CONFIG_HEVC::outputPictureTimingSEI

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

5.9.2.20 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.

5.9.2.21 uint32_t NV_ENC_CONFIG_HEVC::ppsId

[in]: Specifies the PPS id of the picture header. Currently reserved and must be set to 0.

5.9.2.22 uint32_t NV_ENC_CONFIG_HEVC::repeatSPSPPS

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

5.9.2.23 uint32_t NV_ENC_CONFIG_HEVC::reserved

[in]: Reserved bitfields.

5.9.2.24 uint32_t NV_ENC_CONFIG_HEVC::reserved1[217]

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

5.9.2.25 void* NV_ENC_CONFIG_HEVC::reserved2[64]

[in]: Reserved and must be set to NULL

5.9.2.26 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

5.9.2.27 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

5.9.2.28 uint32_t NV_ENC_CONFIG_HEVC::spsId

[in]: Specifies the SPS id of the sequence header. Currently reserved and must be set to 0.

5.9.2.29 uint32_t NV_ENC_CONFIG_HEVC::tier

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

5.9.2.30 uint32_t NV_ENC_CONFIG_HEVC::useConstrainedIntraPred

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

5.9.2.31 uint32_t NV_ENC_CONFIG_HEVC::vpsId

[in]: Specifies the VPS id of the video parameter set. Currently reserved and must be set to 0.

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

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 NV_ENC_OUTPUT_PTR NV_ENC_CREATE_BITSTREAM_BUFFER::bitstreamBuffer

[out]: Pointer to the output bitstream buffer

5.11.2.2 void* NV_ENC_CREATE_BITSTREAM_BUFFER::bitstreamBufferPtr

[out]: Reserved and should not be used

5.11.2.3 NV_ENC_MEMORY_HEAP NV_ENC_CREATE_BITSTREAM_BUFFER::memoryHeap

[in]: Deprecated. Will be removed in sdk 8.0

5.11.2.4 uint32_t NV_ENC_CREATE_BITSTREAM_BUFFER::reserved

[in]: Reserved and must be set to 0

5.11.2.5 uint32_t NV_ENC_CREATE_BITSTREAM_BUFFER::reserved1[58]

[in]: Reserved and should be set to 0

5.11.2.6 void* NV_ENC_CREATE_BITSTREAM_BUFFER::reserved2[64]

[in]: Reserved and should be set to NULL

5.11.2.7 uint32_t NV_ENC_CREATE_BITSTREAM_BUFFER::size

[in]: Size of the bitstream buffer to be created

5.11.2.8 uint32_t NV_ENC_CREATE_BITSTREAM_BUFFER::version

[in]: Struct version. Must be set to [NV_ENC_CREATE_BITSTREAM_BUFFER_VER](#)

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 NV_ENC_BUFFER_FORMAT NV_ENC_CREATE_INPUT_BUFFER::bufferFmt

[in]: Input buffer format

5.12.2.2 uint32_t NV_ENC_CREATE_INPUT_BUFFER::height

[in]: Input buffer width

5.12.2.3 NV_ENC_INPUT_PTR NV_ENC_CREATE_INPUT_BUFFER::inputBuffer

[out]: Pointer to input buffer

5.12.2.4 NV_ENC_MEMORY_HEAP NV_ENC_CREATE_INPUT_BUFFER::memoryHeap

[in]: Deprecated. Will be removed in sdk 8.0

5.12.2.5 void* NV_ENC_CREATE_INPUT_BUFFER::pSysMemBuffer

[in]: Pointer to existing system buffer

5.12.2.6 uint32_t NV_ENC_CREATE_INPUT_BUFFER::reserved

[in]: Reserved and must be set to 0

5.12.2.7 uint32_t NV_ENC_CREATE_INPUT_BUFFER::reserved1[57]

[in]: Reserved and must be set to 0

5.12.2.8 void* NV_ENC_CREATE_INPUT_BUFFER::reserved2[63]

[in]: Reserved and must be set to NULL

5.12.2.9 uint32_t NV_ENC_CREATE_INPUT_BUFFER::version

[in]: Struct version. Must be set to [NV_ENC_CREATE_INPUT_BUFFER_VER](#)

5.12.2.10 uint32_t NV_ENC_CREATE_INPUT_BUFFER::width

[in]: Input buffer width

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 NV_ENC_OUTPUT_PTR NV_ENC_CREATE_MV_BUFFER::mvBuffer

[out]: Pointer to the output motion vector buffer

5.13.2.2 uint32_t NV_ENC_CREATE_MV_BUFFER::reserved1[255]

[in]: Reserved and should be set to 0

5.13.2.3 void* NV_ENC_CREATE_MV_BUFFER::reserved2[63]

[in]: Reserved and should be set to NULL

5.13.2.4 uint32_t NV_ENC_CREATE_MV_BUFFER::version

[in]: Struct version. Must be set to NV_ENC_CREATE_MV_BUFFER_VER

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 void* NV_ENC_EVENT_PARAMS::completionEvent

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

5.14.2.2 uint32_t NV_ENC_EVENT_PARAMS::reserved

[in]: Reserved and must be set to 0

5.14.2.3 uint32_t NV_ENC_EVENT_PARAMS::reserved1[253]

[in]: Reserved and must be set to 0

5.14.2.4 void* NV_ENC_EVENT_PARAMS::reserved2[64]

[in]: Reserved and must be set to NULL

5.14.2.5 uint32_t NV_ENC_EVENT_PARAMS::version

[in]: Struct version. Must be set to [NV_ENC_EVENT_PARAMS_VER](#).

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 `uint8_t NV_ENC_H264_MV_DATA::mbType`

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

5.15.2.2 `NV_ENC_MVECTOR NV_ENC_H264_MV_DATA::mv[4]`

up to 4 vectors for 8x8 partition

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 `uint8_t NV_ENC_HEVC_MV_DATA::cuSize`

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

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::lastCUInCTB`

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

5.16.2.4 `NV_ENC_MVECTOR NV_ENC_HEVC_MV_DATA::mv[4]`

up to 4 vectors within a CU

5.16.2.5 `uint8_t NV_ENC_HEVC_MV_DATA::partitionMode`

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

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 reservedBitFields:28](#)
- [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

Encode Session Initialization parameters.

5.17.2 Field Documentation

5.17.2.1 [uint32_t NV_ENC_INITIALIZE_PARAMS::darHeight](#)

[in]: Specifies the display aspect ratio height.

5.17.2.2 [uint32_t NV_ENC_INITIALIZE_PARAMS::darWidth](#)

[in]: Specifies the display aspect ratio Width.

5.17.2.3 [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.

5.17.2.4 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](#).

5.17.2.5 uint32_t NV_ENC_INITIALIZE_PARAMS::enableMEOnlyMode

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

5.17.2.6 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.

5.17.2.7 uint32_t NV_ENC_INITIALIZE_PARAMS::enableSubFrameWrite

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

5.17.2.8 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.

5.17.2.9 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.

5.17.2.10 uint32_t NV_ENC_INITIALIZE_PARAMS::encodeHeight

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

5.17.2.11 uint32_t NV_ENC_INITIALIZE_PARAMS::encodeWidth

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

5.17.2.12 uint32_t NV_ENC_INITIALIZE_PARAMS::frameRateDen

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

5.17.2.13 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).

5.17.2.14 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.

5.17.2.15 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.

5.17.2.16 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_PIC_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.

5.17.2.17 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.

5.17.2.18 void* NV_ENC_INITIALIZE_PARAMS::privData

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

5.17.2.19 uint32_t NV_ENC_INITIALIZE_PARAMS::privDataSize

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

5.17.2.20 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

5.17.2.21 uint32_t NV_ENC_INITIALIZE_PARAMS::reserved[289]

[in]: Reserved and must be set to 0

5.17.2.22 void* NV_ENC_INITIALIZE_PARAMS::reserved2[64]

[in]: Reserved and must be set to NULL

5.17.2.23 uint32_t NV_ENC_INITIALIZE_PARAMS::reservedBitFields

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

5.17.2.24 uint32_t NV_ENC_INITIALIZE_PARAMS::version

[in]: Struct version. Must be set to [NV_ENC_INITIALIZE_PARAMS_VER](#).

5.18 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]

5.18.1 Detailed Description

Bitstream buffer lock parameters.

5.18.2 Field Documentation

5.18.2.1 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

5.18.2.2 uint32_t NV_ENC_LOCK_BITSTREAM::bitstreamSizeInBytes

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

5.18.2.3 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.

5.18.2.4 uint32_t NV_ENC_LOCK_BITSTREAM::frameAvgQP

[out]: Average QP of the frame.

5.18.2.5 uint32_t NV_ENC_LOCK_BITSTREAM::frameIdx

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

5.18.2.6 uint32_t NV_ENC_LOCK_BITSTREAM::frameSatd

[out]: Total SATD cost for whole frame.

5.18.2.7 uint32_t NV_ENC_LOCK_BITSTREAM::hwEncodeStatus

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

5.18.2.8 uint32_t NV_ENC_LOCK_BITSTREAM::ltrFrame

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

5.18.2.9 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.

5.18.2.10 uint32_t NV_ENC_LOCK_BITSTREAM::ltrFrameIdx

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

5.18.2.11 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.

5.18.2.12 void* NV_ENC_LOCK_BITSTREAM::outputBitstream

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

5.18.2.13 uint64_t NV_ENC_LOCK_BITSTREAM::outputDuration

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

5.18.2.14 uint64_t NV_ENC_LOCK_BITSTREAM::outputTimeStamp

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

5.18.2.15 NV_ENC_PIC_STRUCT NV_ENC_LOCK_BITSTREAM::pictureStruct

[out]: Structure of the generated output picture.

5.18.2.16 NV_ENC_PIC_TYPE NV_ENC_LOCK_BITSTREAM::pictureType

[out]: Picture type of the encoded picture.

5.18.2.17 uint32_t NV_ENC_LOCK_BITSTREAM::reserved[236]

[in]: Reserved and must be set to 0

5.18.2.18 void* NV_ENC_LOCK_BITSTREAM::reserved2[64]

[in]: Reserved and must be set to NULL

5.18.2.19 uint32_t NV_ENC_LOCK_BITSTREAM::reservedBitFields

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

5.18.2.20 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.

5.18.2.21 uint32_t NV_ENC_LOCK_BITSTREAM::version

[in]: Struct version. Must be set to [NV_ENC_LOCK_BITSTREAM_VER](#).

5.19 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]

5.19.1 Detailed Description

Uncompressed Input Buffer lock parameters.

5.19.2 Field Documentation

5.19.2.1 void* NV_ENC_LOCK_INPUT_BUFFER::bufferDataPtr

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

5.19.2.2 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.

5.19.2.3 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.

5.19.2.4 uint32_t NV_ENC_LOCK_INPUT_BUFFER::pitch

[out]: Pitch of the locked input buffer.

5.19.2.5 uint32_t NV_ENC_LOCK_INPUT_BUFFER::reserved1[251]

[in]: Reserved and must be set to 0

5.19.2.6 void* NV_ENC_LOCK_INPUT_BUFFER::reserved2[64]

[in]: Reserved and must be set to NULL

5.19.2.7 uint32_t NV_ENC_LOCK_INPUT_BUFFER::reservedBitFields

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

5.19.2.8 uint32_t NV_ENC_LOCK_INPUT_BUFFER::version

[in]: Struct version. Must be set to [NV_ENC_LOCK_INPUT_BUFFER_VER](#).

5.20 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]

5.20.1 Detailed Description

Map an input resource to a Nvidia Encoder Input Buffer

5.20.2 Field Documentation

5.20.2.1 void* NV_ENC_MAP_INPUT_RESOURCE::inputResource

[in]: Deprecated. Do not use.

5.20.2.2 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.

5.20.2.3 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.

5.20.2.4 NV_ENC_REGISTERED_PTR NV_ENC_MAP_INPUT_RESOURCE::registeredResource

[in]: The Registered resource handle obtained by calling [NvEncRegisterInputResource](#).

5.20.2.5 uint32_t NV_ENC_MAP_INPUT_RESOURCE::reserved1[251]

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

5.20.2.6 void* NV_ENC_MAP_INPUT_RESOURCE::reserved2[63]

[in]: Reserved and must be set to NULL

5.20.2.7 uint32_t NV_ENC_MAP_INPUT_RESOURCE::subResourceIndex

[in]: Deprecated. Do not use.

5.20.2.8 uint32_t NV_ENC_MAP_INPUT_RESOURCE::version

[in]: Struct version. Must be set to [NV_ENC_MAP_INPUT_RESOURCE_VER](#).

5.21 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](#)
- uint32_t [reserved1](#) [251]
- void * [reserved2](#) [60]

5.21.1 Detailed Description

MEOnly parameters that need to be sent on a per motion estimation basis.

5.21.2 Field Documentation

5.21.2.1 NV_ENC_BUFFER_FORMAT NV_ENC_MEONLY_PARAMS::bufferFmt

[in]: Specifies the input buffer format.

5.21.2.2 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.

5.21.2.3 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.

5.21.2.4 uint32_t NV_ENC_MEONLY_PARAMS::inputHeight

[in]: Specifies the input buffer height

5.21.2.5 uint32_t NV_ENC_MEONLY_PARAMS::inputWidth

[in]: Specifies the input buffer width

5.21.2.6 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.

5.21.2.7 NV_ENC_INPUT_PTR NV_ENC_MEONLY_PARAMS::referenceFrame

[in]: Specifies the reference frame pointer

5.21.2.8 uint32_t NV_ENC_MEONLY_PARAMS::reserved1[251]

[in]: Reserved and must be set to 0

5.21.2.9 void* NV_ENC_MEONLY_PARAMS::reserved2[60]

[in]: Reserved and must be set to NULL

5.21.2.10 uint32_t NV_ENC_MEONLY_PARAMS::version

[in]: Struct version. Must be set to NV_ENC_MEONLY_PARAMS_VER.

5.21.2.11 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 NV_ENC_MVECTOR Struct Reference

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

Data Fields

- [int16_t mvx](#)
- [int16_t mvy](#)

5.22.1 Detailed Description

Structs needed for ME only mode.

5.22.2 Field Documentation

5.22.2.1 `int16_t NV_ENC_MVECTOR::mvx`

the x component of MV in qpel units

5.22.2.2 `int16_t NV_ENC_MVECTOR::mvy`

the y component of MV in qpel units

5.23 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 \[64\]](#)

5.23.1 Detailed Description

Encoder Session Creation parameters

5.23.2 Field Documentation

5.23.2.1 `uint32_t NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS::apiVersion`

[in]: API version. Should be set to NVENCAPI_VERSION.

5.23.2.2 `void* NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS::device`

[in]: Pointer to client device.

5.23.2.3 `NV_ENC_DEVICE_TYPE NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS::deviceType`

[in]: Specified the device Type

5.23.2.4 `void* NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS::reserved`

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

5.23.2.5 `uint32_t NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS::reserved1[253]`

[in]: Reserved and must be set to 0

5.23.2.6 `void* NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS::reserved2[64]`

[in]: Reserved and must be set to NULL

5.23.2.7 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 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]

5.24.1 Detailed Description

Encoding parameters that need to be sent on a per frame basis.

5.24.2 Field Documentation

5.24.2.1 NV_ENC_BUFFER_FORMAT NV_ENC_PIC_PARAMS::bufferFmt

[in]: Specifies the input buffer format.

5.24.2.2 NV_ENC_CODEC_PIC_PARAMS NV_ENC_PIC_PARAMS::codecPicParams

[in]: Specifies the codec specific per-picture encoding parameters.

5.24.2.3 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.

5.24.2.4 uint32_t NV_ENC_PIC_PARAMS::encodePicFlags

[in]: Specifies bit-wise OR'ed encode pic flags. See [NV_ENC_PIC_FLAGS](#) enum.

5.24.2.5 uint32_t NV_ENC_PIC_PARAMS::frameIdx

[in]: Specifies the frame index associated with the input frame [optional].

5.24.2.6 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.

5.24.2.7 uint64_t NV_ENC_PIC_PARAMS::inputDuration

[in]: Specifies duration of the input picture

5.24.2.8 uint32_t NV_ENC_PIC_PARAMS::inputHeight

[in]: Specifies the input buffer height

5.24.2.9 uint32_t NV_ENC_PIC_PARAMS::inputPitch

[in]: Specifies the input buffer pitch. If pitch value is not known, set this to inputWidth.

5.24.2.10 uint64_t NV_ENC_PIC_PARAMS::inputTimeStamp

[in]: Specifies presentation timestamp associated with the input picture.

5.24.2.11 uint32_t NV_ENC_PIC_PARAMS::inputWidth

[in]: Specifies the input buffer width

5.24.2.12 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 multiplied by 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)

5.24.2.13 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[1x] must never exceed NV_ENC_INITIALIZE_PARAMS::maxMEHintCountsPerBlock[1x] provided during encoder initialization.

5.24.2.14 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

5.24.2.15 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.

5.24.2.16 NV_ENC_PIC_STRUCT NV_ENC_PIC_PARAMS::pictureStruct

[in]: Specifies structure of the input picture.

5.24.2.17 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.

5.24.2.18 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.

5.24.2.19 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

5.24.2.20 uint32_t NV_ENC_PIC_PARAMS::reserved1[6]

[in]: Reserved and must be set to 0

5.24.2.21 void* NV_ENC_PIC_PARAMS::reserved2[2]

[in]: Reserved and must be set to NULL

5.24.2.22 uint32_t NV_ENC_PIC_PARAMS::reserved3[286]

[in]: Reserved and must be set to 0

5.24.2.23 void* NV_ENC_PIC_PARAMS::reserved4[60]

[in]: Reserved and must be set to NULL

5.24.2.24 uint32_t NV_ENC_PIC_PARAMS::reservedBitFields

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

5.24.2.25 uint32_t NV_ENC_PIC_PARAMS::version

[in]: Struct version. Must be set to [NV_ENC_PIC_PARAMS_VER](#).

5.25 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]

5.25.1 Detailed Description

H264 specific enc pic params. sent on a per frame basis.

5.25.2 Field Documentation

5.25.2.1 uint32_t NV_ENC_PIC_PARAMS_H264::colourPlaneId

[in]: Specifies the colour plane ID associated with the current input.

5.25.2.2 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

5.25.2.3 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.

5.25.2.4 uint32_t NV_ENC_PIC_PARAMS_H264::forceIntraRefreshWithFrameCnt

[in]: Forces an intra refresh with duration equal to `intraRefreshFrameCnt`. When `outputRecoveryPointSEI` is set this 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

5.25.2.5 uint32_t NV_ENC_PIC_PARAMS_H264::ltrMarkFrame

[in]: Set to 1 if client wants to mark this frame as LTR

5.25.2.6 uint32_t NV_ENC_PIC_PARAMS_H264::ltrMarkFrameIdx

[in]: Specifies the long term referenceframe index to use for marking this frame as LTR.

5.25.2.7 uint32_t NV_ENC_PIC_PARAMS_H264::ltrUsageMode

[in]: Specifies additional usage constraints for encoding using LTR frames from this point further. 0: no constraints, 1: no short term refs older than current, no previous LTR frames.

5.25.2.8 uint32_t NV_ENC_PIC_PARAMS_H264::ltrUseFrameBitmap

[in]: Specifies the the associated bitmap of LTR frame indices when encoding this frame.

5.25.2.9 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`

5.25.2.10 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.

5.25.2.11 uint32_t NV_ENC_PIC_PARAMS_H264::reserved[243]

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

5.25.2.12 void* NV_ENC_PIC_PARAMS_H264::reserved2[62]

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

5.25.2.13 uint32_t NV_ENC_PIC_PARAMS_H264::reserved3

[in]: Reserved and must be set to 0

5.25.2.14 uint32_t NV_ENC_PIC_PARAMS_H264::reservedBitFields

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

5.25.2.15 NV_ENC_SEI_PAYLOAD* NV_ENC_PIC_PARAMS_H264::seiPayloadArray

[in]: Array of SEI payloads which will be inserted for this frame.

5.25.2.16 uint32_t NV_ENC_PIC_PARAMS_H264::seiPayloadArrayCnt

[in]: Specifies the number of elements allocated in seiPayloadArray array.

5.25.2.17 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

5.25.2.18 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

5.25.2.19 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

5.25.2.20 uint32_t NV_ENC_PIC_PARAMS_H264::sliceTypeArrayCnt

[in]: Deprecated.

5.25.2.21 uint8_t* NV_ENC_PIC_PARAMS_H264::sliceTypeData

[in]: Deprecated.

5.26 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]

5.26.1 Detailed Description

HEVC 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_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

5.26.2.2 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.

5.26.2.3 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

5.26.2.4 uint32_t NV_ENC_PIC_PARAMS_HEVC::ltrMarkFrame

[in]: Set to 1 if client wants to mark this frame as LTR

5.26.2.5 uint32_t NV_ENC_PIC_PARAMS_HEVC::ltrMarkFrameIdx

[in]: Specifies the long term reference frame index to use for marking this frame as LTR.

5.26.2.6 uint32_t NV_ENC_PIC_PARAMS_HEVC::ltrUsageMode

[in]: Specifies additional usage constraints for encoding using LTR frames from this point further. 0: no constraints, 1: no short term refs older than current, no previous LTR frames.

5.26.2.7 uint32_t NV_ENC_PIC_PARAMS_HEVC::ltrUseFrameBitmap

[in]: Specifies the associated bitmap of LTR frame indices when encoding this frame.

5.26.2.8 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

5.26.2.9 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.

5.26.2.10 uint32_t NV_ENC_PIC_PARAMS_HEVC::reserved

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

5.26.2.11 uint32_t NV_ENC_PIC_PARAMS_HEVC::reserved2[244]

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

5.26.2.12 void* NV_ENC_PIC_PARAMS_HEVC::reserved3[61]

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

5.26.2.13 uint32_t NV_ENC_PIC_PARAMS_HEVC::reservedBitFields

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

5.26.2.14 NV_ENC_SEI_PAYLOAD* NV_ENC_PIC_PARAMS_HEVC::seiPayloadArray

[in]: Array of SEI payloads which will be inserted for this frame.

5.26.2.15 uint32_t NV_ENC_PIC_PARAMS_HEVC::seiPayloadArrayCnt

[in]: Specifies the number of elements allocated in seiPayloadArray array.

5.26.2.16 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

5.26.2.17 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

5.26.2.18 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

5.26.2.19 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

5.26.2.20 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](#)

5.26.2.21 uint32_t NV_ENC_PIC_PARAMS_HEVC::temporalId

[in]: Specifies the temporal id of the picture

5.27 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]

5.27.1 Detailed Description

Encoder preset config

5.27.2 Field Documentation

5.27.2.1 NV_ENC_CONFIG NV_ENC_PRESET_CONFIG::presetCfg

[out]: preset config returned by the Nvidia Video Encoder interface.

5.27.2.2 [uint32_t](#) NV_ENC_PRESET_CONFIG::reserved1[255]

[in]: Reserved and must be set to 0

5.27.2.3 [void*](#) NV_ENC_PRESET_CONFIG::reserved2[64]

[in]: Reserved and must be set to NULL

5.27.2.4 [uint32_t](#) NV_ENC_PRESET_CONFIG::version

[in]: Struct version. Must be set to [NV_ENC_PRESET_CONFIG_VER](#).

5.28 NV_ENC_QP Struct Reference

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

5.28.1 Detailed Description

QP value for frames

5.29 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]
- [uint16_t](#) targetQuality
- [uint16_t](#) lookaheadDepth

5.29.1 Detailed Description

Rate Control Configuration Paramters

5.29.2 Field Documentation

5.29.2.1 [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. Currently supported only with h264

5.29.2.2 [uint32_t NV_ENC_RC_PARAMS::averageBitRate](#)

[in]: Specifies the average bitrate(in bits/sec) used for encoding.

5.29.2.3 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.

5.29.2.4 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)

5.29.2.5 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)

5.29.2.6 uint32_t NV_ENC_RC_PARAMS::enableAQ

[in]: Set this to 1 to enable adaptive quantization (Spatial).

5.29.2.7 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)

5.29.2.8 uint32_t NV_ENC_RC_PARAMS::enableInitialRCQP

[in]: Set this to 1 if user supplied initial QP is used for rate control.

5.29.2.9 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)

5.29.2.10 uint32_t NV_ENC_RC_PARAMS::enableMaxQP

[in]: Set this to 1 if maximum QP used for rate control.

5.29.2.11 uint32_t NV_ENC_RC_PARAMS::enableMinQP

[in]: Set this to 1 if minimum QP used for rate control.

5.29.2.12 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)

5.29.2.13 uint32_t NV_ENC_RC_PARAMS::enableTemporalAQ

[in]: Set this to 1 to enable temporal AQ for H.264

5.29.2.14 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.

5.29.2.15 uint16_t NV_ENC_RC_PARAMS::lookaheadDepth

[in]: Maximum depth of lookahead with range 0-32 (only used if enableLookahead=1)

5.29.2.16 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.

5.29.2.17 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.

5.29.2.18 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.

5.29.2.19 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.

5.29.2.20 uint32_t NV_ENC_RC_PARAMS::reservedBitFields

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

5.29.2.21 uint32_t NV_ENC_RC_PARAMS::strictGOPTarget

[in]: Set this to 1 to minimize GOP-to-GOP rate fluctuations

5.29.2.22 uint16_t NV_ENC_RC_PARAMS::targetQuality

[in]: Target CQ (Constant Quality) level for VBR mode (range 0-51 with 0-automatic)

5.29.2.23 uint32_t NV_ENC_RC_PARAMS::temporallayerIdxMask

[in]: Specifies the temporal layers (as a bitmask) whose QPs have changed. Valid max bitmask is $[2^{NV_ENC_CAPS_NUM_MAX_TEMPORAL_LAYERS} - 1]$

5.29.2.24 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

5.29.2.25 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.

5.29.2.26 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 .

5.29.2.27 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 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](#)

5.30.1 Detailed Description

Encode Session Reconfigured parameters.

5.30.2 Field Documentation

5.30.2.1 [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](#)].

5.30.2.2 [NV_ENC_INITIALIZE_PARAMS NV_ENC_RECONFIGURE_PARAMS::reInitEncodeParams](#)

[in]: Encoder session re-initialization parameters.

5.30.2.3 [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

5.30.2.4 [uint32_t NV_ENC_RECONFIGURE_PARAMS::version](#)

[in]: Struct version. Must be set to [NV_ENC_RECONFIGURE_PARAMS_VER](#).

5.31 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 \[62\]](#)

5.31.1 Detailed Description

Register a resource for future use with the Nvidia Video Encoder Interface.

5.31.2 Field Documentation

5.31.2.1 NV_ENC_BUFFER_FORMAT NV_ENC_REGISTER_RESOURCE::bufferFormat

[in]: Buffer format of resource to be registered.

5.31.2.2 uint32_t NV_ENC_REGISTER_RESOURCE::height

[in]: Input buffer Height.

5.31.2.3 uint32_t NV_ENC_REGISTER_RESOURCE::pitch

[in]: Input buffer Pitch.

5.31.2.4 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.

5.31.2.5 uint32_t NV_ENC_REGISTER_RESOURCE::reserved1[248]

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

5.31.2.6 void* NV_ENC_REGISTER_RESOURCE::reserved2[62]

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

5.31.2.7 void* NV_ENC_REGISTER_RESOURCE::resourceToRegister

[in]: Handle to the resource that is being registered.

5.31.2.8 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](#).

5.31.2.9 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.

5.31.2.10 uint32_t NV_ENC_REGISTER_RESOURCE::version

[in]: Struct version. Must be set to [NV_ENC_REGISTER_RESOURCE_VER](#).

5.31.2.11 uint32_t NV_ENC_REGISTER_RESOURCE::width

[in]: Input buffer Width.

5.32 NV_ENC_SEI_PAYLOAD Struct Reference

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

Data Fields

- [uint32_t payloadSize](#)
- [uint32_t payloadType](#)
- [uint8_t * payload](#)

5.32.1 Detailed Description

User SEI message

5.32.2 Field Documentation

5.32.2.1 `uint8_t* NV_ENC_SEI_PAYLOAD::payload`

[in] pointer to user data

5.32.2.2 `uint32_t NV_ENC_SEI_PAYLOAD::payloadSize`

[in] SEI payload size in bytes. SEI payload must be byte aligned, as described in Annex D

5.32.2.3 `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 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]`

5.33.1 Detailed Description

Sequence and picture parameters payload.

5.33.2 Field Documentation

5.33.2.1 `uint32_t NV_ENC_SEQUENCE_PARAM_PAYLOAD::inBufferSize`

[in]: Specifies the size of the `spsppsBuffer` provided by the client

5.33.2.2 `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 `SPSPPSBuffer`.

5.33.2.3 `uint32_t NV_ENC_SEQUENCE_PARAM_PAYLOAD::ppsId`

[in]: Specifies the PPS id to be used in picture header. Default value is 0.

5.33.2.4 `uint32_t NV_ENC_SEQUENCE_PARAM_PAYLOAD::reserved[250]`

[in]: Reserved and must be set to 0

5.33.2.5 `void* NV_ENC_SEQUENCE_PARAM_PAYLOAD::reserved2[64]`

[in]: Reserved and must be set to NULL

5.33.2.6 `uint32_t NV_ENC_SEQUENCE_PARAM_PAYLOAD::spsId`

[in]: Specifies the SPS id to be used in sequence header. Default value is 0.

5.33.2.7 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.

5.33.2.8 uint32_t NV_ENC_SEQUENCE_PARAM_PAYLOAD::version

[in]: Struct version. Must be set to [NV_ENC_INITIALIZE_PARAMS_VER](#).

5.34 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]

5.34.1 Detailed Description

Encode Stats structure.

5.34.2 Field Documentation

5.34.2.1 `uint32_t NV_ENC_STAT::bitStreamSize`

[out]: Size of generated bitstream in bytes.

5.34.2.2 `uint32_t NV_ENC_STAT::lastValidByteOffset`

[out]: Offset of last valid bytes of completed bitstream

5.34.2.3 `NV_ENC_OUTPUT_PTR NV_ENC_STAT::outputBitStream`

[out]: Specifies the pointer to output bitstream.

5.34.2.4 `uint32_t NV_ENC_STAT::picIdx`

[out]: Picture number

5.34.2.5 `uint32_t NV_ENC_STAT::picType`

[out]: Picture type of encoded picture. See [NV_ENC_PIC_TYPE](#).

5.34.2.6 `uint32_t NV_ENC_STAT::reserved`

[in]: Reserved and must be set to 0

5.34.2.7 uint32_t NV_ENC_STAT::reserved1[233]

[in]: Reserved and must be set to 0

5.34.2.8 void* NV_ENC_STAT::reserved2[64]

[in]: Reserved and must be set to NULL

5.34.2.9 uint32_t NV_ENC_STAT::sliceOffsets[16]

[out]: Offsets of each slice

5.34.2.10 uint32_t NV_ENC_STAT::version

[in]: Struct version. Must be set to [NV_ENC_STAT_VER](#).

5.35 NV_ENCODE_API_FUNCTION_LIST Struct Reference

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

Data Fields

- uint32_t [version](#)
- uint32_t [reserved](#)
- PNVENCOOPENENCODESESSION [nvEncOpenEncodeSession](#)
- PNVENCGETENCODEGUIDCOUNT [nvEncGetEncodeGUIDCount](#)
- PNVENCGETENCODEPRESETCOUNT [nvEncGetEncodeProfileGUIDCount](#)
- PNVENCGETENCODEPRESETGUIDS [nvEncGetEncodeProfileGUIDs](#)
- PNVENCGETENCODEGUIDS [nvEncGetEncodeGUIDs](#)
- PNVENCGETINPUTFORMATCOUNT [nvEncGetInputFormatCount](#)
- PNVENCGETINPUTFORMATS [nvEncGetInputFormats](#)
- PNVENCGETENCODECAPS [nvEncGetEncodeCaps](#)
- PNVENCGETENCODEPRESETCOUNT [nvEncGetEncodePresetCount](#)
- PNVENCGETENCODEPRESETGUIDS [nvEncGetEncodePresetGUIDs](#)
- PNVENCGETENCODEPRESETCONFIG [nvEncGetEncodePresetConfig](#)
- PNVENCCREATEENCODER [nvEncInitializeEncoder](#)
- PNVENCCREATEINPUTBUFFER [nvEncCreateInputBuffer](#)
- PNVENCCDESTROYINPUTBUFFER [nvEncDestroyInputBuffer](#)
- PNVENCCREATEBITSTREAMBUFFER [nvEncCreateBitstreamBuffer](#)
- PNVENCCDESTROYBITSTREAMBUFFER [nvEncDestroyBitstreamBuffer](#)
- PNVENCCENCODEPICTURE [nvEncEncodePicture](#)
- PNVENCCLOCKBITSTREAM [nvEncLockBitstream](#)
- PNVENCCUNLOCKBITSTREAM [nvEncUnlockBitstream](#)
- PNVENCCLOCKINPUTBUFFER [nvEncLockInputBuffer](#)
- PNVENCCUNLOCKINPUTBUFFER [nvEncUnlockInputBuffer](#)
- PNVENCGETENCODESTATS [nvEncGetEncodeStats](#)
- PNVENCGETSEQUENCEPARAMS [nvEncGetSequenceParams](#)
- PNVENCCREGISTERASYNCEVENT [nvEncRegisterAsyncEvent](#)
- PNVENCCUNREGISTERASYNCEVENT [nvEncUnregisterAsyncEvent](#)
- PNVENCCMAPINPUTRESOURCE [nvEncMapInputResource](#)
- PNVENCCUNMAPINPUTRESOURCE [nvEncUnmapInputResource](#)
- PNVENCCDESTROYENCODER [nvEncDestroyEncoder](#)
- PNVENCCINVALIDATEREFFRAMES [nvEncInvalidateRefFrames](#)
- PNVENCCOPENENCODESESSIONEX [nvEncOpenEncodeSessionEx](#)
- PNVENCCREGISTERRESOURCE [nvEncRegisterResource](#)
- PNVENCCUNREGISTERRESOURCE [nvEncUnregisterResource](#)
- PNVENCCRECONFIGUREENCODER [nvEncReconfigureEncoder](#)
- PNVENCCCREATEMVBUFFER [nvEncCreateMVBuffer](#)
- PNVENCCDESTROYMVBUFFER [nvEncDestroyMVBuffer](#)
- PNVENCCRUNMOTIONESTIMATIONONLY [nvEncRunMotionEstimationOnly](#)
- void * [reserved2](#) [281]

5.35.1 Detailed Description

[NV_ENCODE_API_FUNCTION_LIST](#)

5.35.2 Field Documentation

5.35.2.1 PNVENCCREATEBITSTREAMBUFFER NV_ENCODE_API_FUNCTION_LIST::nvEncCreateBitstreamBuffer

[out]: Client should access [NvEncCreateBitstreamBuffer\(\)](#) API through this pointer.

5.35.2.2 PNVENCCREATEINPUTBUFFER NV_ENCODE_API_FUNCTION_LIST::nvEncCreateInputBuffer

[out]: Client should access [NvEncCreateInputBuffer\(\)](#) API through this pointer.

5.35.2.3 PNVENCCREATEMVBUFFER NV_ENCODE_API_FUNCTION_LIST::nvEncCreateMVBuffer

[out]: Client should access [NvEncCreateMVBuffer](#) API through this pointer.

5.35.2.4 PNVENCDESTROYBITSTREAMBUFFER NV_ENCODE_API_FUNCTION_LIST::nvEncDestroyBitstreamBuffer

[out]: Client should access [NvEncDestroyBitstreamBuffer\(\)](#) API through this pointer.

5.35.2.5 PNVENCDESTROYENCODER NV_ENCODE_API_FUNCTION_LIST::nvEncDestroyEncoder

[out]: Client should access [NvEncDestroyEncoder\(\)](#) API through this pointer.

5.35.2.6 PNVENCDESTROYINPUTBUFFER NV_ENCODE_API_FUNCTION_LIST::nvEncDestroyInputBuffer

[out]: Client should access [NvEncDestroyInputBuffer\(\)](#) API through this pointer.

5.35.2.7 PNVENCDESTROYMVBUFFER NV_ENCODE_API_FUNCTION_LIST::nvEncDestroyMVBuffer

[out]: Client should access [NvEncDestroyMVBuffer](#) API through this pointer.

5.35.2.8 PNVENCENCODEPICTURE NV_ENCODE_API_FUNCTION_LIST::nvEncEncodePicture

[out]: Client should access [NvEncEncodePicture\(\)](#) API through this pointer.

5.35.2.9 PNVENCGETENCODECAPS NV_ENCODE_API_FUNCTION_LIST::nvEncGetEncodeCaps

[out]: Client should access [NvEncGetEncodeCaps\(\)](#) API through this pointer.

5.35.2.10 PNVENCGETENCODEGUIDCOUNT NV_ENCODE_API_FUNCTION_LIST::nvEncGetEncodeGUIDCount

[out]: Client should access [NvEncGetEncodeGUIDCount\(\)](#) API through this pointer.

5.35.2.11 PNVENCGETENCODEGUIDS NV_ENCODE_API_FUNCTION_LIST::nvEncGetEncodeGUIDs

[out]: Client should access [NvEncGetEncodeGUIDs\(\)](#) API through this pointer.

5.35.2.12 PNVENCGETENCODEPRESETCONFIG NV_ENCODE_API_FUNCTION_LIST::nvEncGetEncodePresetConfig

[out]: Client should access [NvEncGetEncodePresetConfig\(\)](#) API through this pointer.

5.35.2.13 PNVENCGETENCODEPRESETCOUNT NV_ENCODE_API_FUNCTION_LIST::nvEncGetEncodePresetCount

[out]: Client should access [NvEncGetEncodePresetCount\(\)](#) API through this pointer.

5.35.2.14 PNVENCGETENCODEPRESETGUIDS NV_ENCODE_API_FUNCTION_LIST::nvEncGetEncodePresetGUIDs

[out]: Client should access [NvEncGetEncodePresetGUIDs\(\)](#) API through this pointer.

5.35.2.15 PNVENCGETENCODEPRESETCOUNT NV_ENCODE_API_FUNCTION_LIST::nvEncGetEncodeProfileGUIDCount

[out]: Client should access [NvEncGetEncodeProfileGUIDCount\(\)](#) API through this pointer.

5.35.2.16 PNVENCGETENCODEPRESETGUIDS NV_ENCODE_API_FUNCTION_LIST::nvEncGetEncodeProfileGUIDs

[out]: Client should access [NvEncGetEncodeProfileGUIDs\(\)](#) API through this pointer.

5.35.2.17 PNVENCGETENCODESTATS NV_ENCODE_API_FUNCTION_LIST::nvEncGetEncodeStats

[out]: Client should access [NvEncGetEncodeStats\(\)](#) API through this pointer.

5.35.2.18 PNVENCGETINPUTFORMATCOUNT NV_ENCODE_API_FUNCTION_LIST::nvEncGetInputFormatCount

[out]: Client should access [NvEncGetInputFormatCount\(\)](#) API through this pointer.

5.35.2.19 PNVENCGETINPUTFORMATS NV_ENCODE_API_FUNCTION_LIST::nvEncGetInputFormats

[out]: Client should access [NvEncGetInputFormats\(\)](#) API through this pointer.

5.35.2.20 PNVENCGETSEQUENCEPARAMS NV_ENCODE_API_FUNCTION_LIST::nvEncGetSequenceParams

[out]: Client should access [NvEncGetSequenceParams\(\)](#) API through this pointer.

5.35.2.21 PNVENCINITIALIZEENCODER NV_ENCODE_API_FUNCTION_LIST::nvEncInitializeEncoder

[out]: Client should access [NvEncInitializeEncoder\(\)](#) API through this pointer.

5.35.2.22 PNVENCINVALIDATEREFFRAMES NV_ENCODE_API_FUNCTION_LIST::nvEncInvalidateRefFrames

[out]: Client should access [NvEncInvalidateRefFrames\(\)](#) API through this pointer.

5.35.2.23 PNVENCLOCKBITSTREAM NV_ENCODE_API_FUNCTION_LIST::nvEncLockBitstream

[out]: Client should access [NvEncLockBitstream\(\)](#) API through this pointer.

5.35.2.24 PNVENCLOCKINPUTBUFFER NV_ENCODE_API_FUNCTION_LIST::nvEncLockInputBuffer

[out]: Client should access [NvEncLockInputBuffer\(\)](#) API through this pointer.

5.35.2.25 PNVENCMAPINPUTRESOURCE NV_ENCODE_API_FUNCTION_LIST::nvEncMapInputResource

[out]: Client should access [NvEncMapInputResource\(\)](#) API through this pointer.

5.35.2.26 PNVENCOPENENCODESESSION NV_ENCODE_API_FUNCTION_LIST::nvEncOpenEncodeSession

[out]: Client should access [NvEncOpenEncodeSession\(\)](#) API through this pointer.

5.35.2.27 PNVENCOPENENCODESESSIONEX NV_ENCODE_API_FUNCTION_LIST::nvEncOpenEncodeSessionEx

[out]: Client should access [NvEncOpenEncodeSession\(\)](#) API through this pointer.

5.35.2.28 PNVENCRECONFIGUREENCODER NV_ENCODE_API_FUNCTION_LIST::nvEncReconfigureEncoder

[out]: Client should access [NvEncReconfigureEncoder\(\)](#) API through this pointer.

5.35.2.29 PNVENCREGISTERASYNC EVENT NV_ENCODE_API_FUNCTION_LIST::nvEncRegisterAsyncEvent

[out]: Client should access [NvEncRegisterAsyncEvent\(\)](#) API through this pointer.

5.35.2.30 PNVENCREGISTERRESOURCE NV_ENCODE_API_FUNCTION_LIST::nvEncRegisterResource

[out]: Client should access [NvEncRegisterResource\(\)](#) API through this pointer.

5.35.2.31 PNVENCRUNMOTIONESTIMATIONONLY NV_ENCODE_API_FUNCTION_LIST::nvEncRunMotionEstimationOnly

[out]: Client should access [NvEncRunMotionEstimationOnly](#) API through this pointer.

5.35.2.32 PNVENCUNLOCKBITSTREAM NV_ENCODE_API_FUNCTION_LIST::nvEncUnlockBitstream

[out]: Client should access [NvEncUnlockBitstream\(\)](#) API through this pointer.

5.35.2.33 PNVENCUNLOCKINPUTBUFFER NV_ENCODE_API_FUNCTION_LIST::nvEncUnlockInputBuffer

[out]: Client should access [NvEncUnlockInputBuffer\(\)](#) API through this pointer.

5.35.2.34 PNVENCUNMAPINPUTRESOURCE NV_ENCODE_API_FUNCTION_LIST::nvEncUnmapInputResource

[out]: Client should access [NvEncUnmapInputResource\(\)](#) API through this pointer.

5.35.2.35 PNVENCUNREGISTERASYNCEVENT NV_ENCODE_API_FUNCTION_LIST::nvEncUnregisterAsyncEvent

[out]: Client should access [NvEncUnregisterAsyncEvent\(\)](#) API through this pointer.

5.35.2.36 PNVENCUNREGISTERRESOURCE NV_ENCODE_API_FUNCTION_LIST::nvEncUnregisterResource

[out]: Client should access [NvEncUnregisterResource\(\)](#) API through this pointer.

5.35.2.37 uint32_t NV_ENCODE_API_FUNCTION_LIST::reserved

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

5.35.2.38 void* NV_ENCODE_API_FUNCTION_LIST::reserved2[281]

[in]: Reserved and must be set to NULL

5.35.2.39 uint32_t NV_ENCODE_API_FUNCTION_LIST::version

[in]: Client should pass NV_ENCODE_API_FUNCTION_LIST_VER.

5.36 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

5.36.1 Detailed Description

External Motion Vector hint structure.

5.36.2 Field Documentation

5.36.2.1 `int32_t NVENC_EXTERNAL_ME_HINT::dir`

[in]: Specifies the direction of motion estimation . 0=L0 1=L1.

5.36.2.2 `int32_t NVENC_EXTERNAL_ME_HINT::lastOfMB`

[in]: Set to 1 for the last MV of macroblock.

5.36.2.3 `int32_t NVENC_EXTERNAL_ME_HINT::lastofPart`

[in]: Set to 1 for the last MV of (sub) partition

5.36.2.4 `int32_t NVENC_EXTERNAL_ME_HINT::mvx`

[in]: Specifies the x component of integer pixel MV (relative to current MB) S12.0.

5.36.2.5 `int32_t NVENC_EXTERNAL_ME_HINT::mvy`

[in]: Specifies the y component of integer pixel MV (relative to current MB) S10.0 .

5.36.2.6 `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).

5.36.2.7 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 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]

5.37.1 Detailed Description

External motion vector hint counts per block type.

5.37.2 Field Documentation

5.37.2.1 `uint32_t NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE::numCandsPerBlk16x16`

[in]: Specifies the number of candidates per 16x16 block.

5.37.2.2 `uint32_t NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE::numCandsPerBlk16x8`

[in]: Specifies the number of candidates per 16x8 block.

5.37.2.3 `uint32_t NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE::numCandsPerBlk8x16`

[in]: Specifies the number of candidates per 8x16 block.

5.37.2.4 `uint32_t NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE::numCandsPerBlk8x8`

[in]: Specifies the number of candidates per 8x8 block.

5.37.2.5 `uint32_t NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE::reserved`

[in]: Reserved for padding.

5.37.2.6 `uint32_t NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE::reserved1[3]`

[in]: Reserved for future use.

5.38 NVENC_RECT Struct Reference

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

Data Fields

- [uint32_t left](#)
- [uint32_t top](#)
- [uint32_t right](#)
- [uint32_t bottom](#)

5.38.1 Detailed Description

Defines a Rectangle. Used in NV_ENC_PREPROCESS_FRAME.

5.38.2 Field Documentation

5.38.2.1 `uint32_t NVENC_RECT::bottom`

[in]: Y coordinate of the bottom right corner of the rectangular area to be specified.

5.38.2.2 `uint32_t NVENC_RECT::left`

[in]: X coordinate of the upper left corner of rectangular area to be specified.

5.38.2.3 `uint32_t NVENC_RECT::right`

[in]: X coordinate of the bottom right corner of the rectangular area to be specified.

5.38.2.4 `uint32_t NVENC_RECT::top`

[in]: Y coordinate of the upper left corner of the rectangular area to be specified.

Index

- adaptiveTransformMode
 - NV_ENC_CONFIG_H264, 56
- apiVersion
 - NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS, 92
- aqStrength
 - NV_ENC_RC_PARAMS, 106
- averageBitRate
 - NV_ENC_RC_PARAMS, 106
- bdirectMode
 - NV_ENC_CONFIG_H264, 56
- bitstreamBuffer
 - NV_ENC_CREATE_BITSTREAM_BUFFER, 70
- bitstreamBufferPtr
 - NV_ENC_CREATE_BITSTREAM_BUFFER, 70
 - NV_ENC_LOCK_BITSTREAM, 82
- bitstreamRestrictionFlag
 - NV_ENC_CONFIG_H264_VUI_PARAMETERS, 63
- bitStreamSize
 - NV_ENC_STAT, 116
- bitstreamSizeInBytes
 - NV_ENC_LOCK_BITSTREAM, 82
- bottom
 - NVENC_RECT, 126
- bStereoEnable
 - NV_ENC_CONFIG_H264_MEONLY, 61
- bufferDataPtr
 - NV_ENC_LOCK_INPUT_BUFFER, 85
- bufferFmt
 - NV_ENC_CREATE_INPUT_BUFFER, 72
 - NV_ENC_MEONLY_PARAMS, 89
 - NV_ENC_PIC_PARAMS, 94
- bufferFormat
 - NV_ENC_REGISTER_RESOURCE, 111
- capsToQuery
 - NV_ENC_CAPS_PARAM, 50
- chromaFormatIDC
 - NV_ENC_CONFIG_H264, 56
 - NV_ENC_CONFIG_HEVC, 65
- chromaSampleLocationBot
 - NV_ENC_CONFIG_H264_VUI_PARAMETERS, 63
- chromaSampleLocationFlag
 - NV_ENC_CONFIG_H264_VUI_PARAMETERS, 63
- chromaSampleLocationTop
 - NV_ENC_CONFIG_H264_VUI_PARAMETERS, 63
- codecPicParams
 - NV_ENC_PIC_PARAMS, 94
- colourDescriptionPresentFlag
 - NV_ENC_CONFIG_H264_VUI_PARAMETERS, 63
- colourMatrix
 - NV_ENC_CONFIG_H264_VUI_PARAMETERS, 63
- colourPlaneId
 - NV_ENC_PIC_PARAMS_H264, 98
- colourPrimaries
 - NV_ENC_CONFIG_H264_VUI_PARAMETERS, 64
- completionEvent
 - NV_ENC_EVENT_PARAMS, 75
 - NV_ENC_MEONLY_PARAMS, 89
 - NV_ENC_PIC_PARAMS, 94
- constQP
 - NV_ENC_RC_PARAMS, 106
- constrainedFrame
 - NV_ENC_PIC_PARAMS_H264, 98
 - NV_ENC_PIC_PARAMS_HEVC, 101
- cuSize
 - NV_ENC_HEVC_MV_DATA, 77
- cuType
 - NV_ENC_HEVC_MV_DATA, 77
- darHeight
 - NV_ENC_INITIALIZE_PARAMS, 78
- darWidth
 - NV_ENC_INITIALIZE_PARAMS, 78
- Data1
 - GUID, 49
- Data2
 - GUID, 49
- Data3
 - GUID, 49
- Data4
 - GUID, 49

- device
 - NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS, 92
- deviceType
 - NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS, 92
- dir
 - NVENC_EXTERNAL_ME_HINT, 123
- disableBadapt
 - NV_ENC_RC_PARAMS, 107
- disableDeblockAcrossSliceBoundary
 - NV_ENC_CONFIG_HEVC, 65
- disableDeblockingFilterIDC
 - NV_ENC_CONFIG_H264, 56
- disableIadapt
 - NV_ENC_RC_PARAMS, 107
- disableIntraSearch
 - NV_ENC_CONFIG_H264_MEONLY, 61
- disablePartition16x16
 - NV_ENC_CONFIG_H264_MEONLY, 61
- disablePartition16x8
 - NV_ENC_CONFIG_H264_MEONLY, 61
- disablePartition8x16
 - NV_ENC_CONFIG_H264_MEONLY, 61
- disablePartition8x8
 - NV_ENC_CONFIG_H264_MEONLY, 61
- disableSPSPPS
 - NV_ENC_CONFIG_H264, 56
 - NV_ENC_CONFIG_HEVC, 66
- displayPOCSyntax
 - NV_ENC_PIC_PARAMS_H264, 98
 - NV_ENC_PIC_PARAMS_HEVC, 101
- doNotWait
 - NV_ENC_LOCK_BITSTREAM, 82
 - NV_ENC_LOCK_INPUT_BUFFER, 85
- enableAQ
 - NV_ENC_RC_PARAMS, 107
- enableConstrainedEncoding
 - NV_ENC_CONFIG_H264, 56
- enableEncodeAsync
 - NV_ENC_INITIALIZE_PARAMS, 78
- enableExternalMEHints
 - NV_ENC_INITIALIZE_PARAMS, 78
- enableExtQPDeltaMap
 - NV_ENC_RC_PARAMS, 107
- enableInitialRCQP
 - NV_ENC_RC_PARAMS, 107
- enableIntraRefresh
 - NV_ENC_CONFIG_H264, 56
 - NV_ENC_CONFIG_HEVC, 66
- enableLookahead
 - NV_ENC_RC_PARAMS, 107
- enableLTR
 - NV_ENC_CONFIG_H264, 56
 - NV_ENC_CONFIG_HEVC, 66
- enableMaxQP
 - NV_ENC_RC_PARAMS, 107
- enableMEOnlyMode
 - NV_ENC_INITIALIZE_PARAMS, 79
- enableMinQP
 - NV_ENC_RC_PARAMS, 107
- enableNonRefP
 - NV_ENC_RC_PARAMS, 107
- enablePTD
 - NV_ENC_INITIALIZE_PARAMS, 79
- enableStereoMVC
 - NV_ENC_CONFIG_H264, 56
- enableSubFrameWrite
 - NV_ENC_INITIALIZE_PARAMS, 79
- enableTemporalAQ
 - NV_ENC_RC_PARAMS, 107
- enableTemporalSVC
 - NV_ENC_CONFIG_H264, 57
- enableVFR
 - NV_ENC_CONFIG_H264, 57
- ENCODE_FUNC
 - NvEncCreateBitstreamBuffer, 26
 - NvEncCreateInputBuffer, 26
 - NvEncCreateMVBuffer, 26
 - NvEncDestroyBitstreamBuffer, 27
 - NvEncDestroyEncoder, 27
 - NvEncDestroyInputBuffer, 28
 - NvEncDestroyMVBuffer, 28
 - NvEncEncodePicture, 29
 - NvEncGetEncodeCaps, 31
 - NvEncGetEncodeGUIDCount, 32
 - NvEncGetEncodeGUIDs, 32
 - NvEncGetEncodePresetConfig, 33
 - NvEncGetEncodePresetCount, 33
 - NvEncGetEncodePresetGUIDs, 34
 - NvEncGetEncodeProfileGUIDCount, 34
 - NvEncGetEncodeProfileGUIDs, 35
 - NvEncGetEncodeStats, 35
 - NvEncGetInputFormatCount, 36
 - NvEncGetInputFormats, 36
 - NvEncGetSequenceParams, 37
 - NvEncInitializeEncoder, 37
 - NvEncInvalidateRefFrames, 39
 - NvEncLockBitstream, 39
 - NvEncLockInputBuffer, 40
 - NvEncMapInputResource, 40
 - NvEncodeAPICreateInstance, 41
 - NvEncodeAPIGetMaxSupportedVersion, 41
 - NvEncOpenEncodeSession, 42
 - NvEncOpenEncodeSessionEx, 42
 - NvEncReconfigureEncoder, 42
 - NvEncRegisterAsyncEvent, 43

- NvEncRegisterResource, 43
- NvEncRunMotionEstimationOnly, 44
- NvEncUnlockBitstream, 44
- NvEncUnlockInputBuffer, 45
- NvEncUnmapInputResource, 45
- NvEncUnregisterAsyncEvent, 46
- NvEncUnregisterResource, 46
- encodeCodecConfig
 - NV_ENC_CONFIG, 53
- encodeConfig
 - NV_ENC_INITIALIZE_PARAMS, 79
- encodeGUID
 - NV_ENC_INITIALIZE_PARAMS, 79
- encodeHeight
 - NV_ENC_INITIALIZE_PARAMS, 79
- encodePicFlags
 - NV_ENC_PIC_PARAMS, 95
- ENCODER_STRUCTURE
 - NV_ENC_BUFFER_FORMAT_ABGR, 15
 - NV_ENC_BUFFER_FORMAT_ABGR10, 15
 - NV_ENC_BUFFER_FORMAT_ARGB, 15
 - NV_ENC_BUFFER_FORMAT_ARGB10, 15
 - NV_ENC_BUFFER_FORMAT_AYUV, 15
 - 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, 15
 - NV_ENC_BUFFER_FORMAT_YV12, 14
 - NV_ENC_CAPS_ASYNC_ENCODE_SUPPORT, 17
 - NV_ENC_CAPS_EXPOSED_COUNT, 17
 - NV_ENC_CAPS_HEIGHT_MAX, 16
 - NV_ENC_CAPS_LEVEL_MAX, 16
 - NV_ENC_CAPS_LEVEL_MIN, 16
 - NV_ENC_CAPS_MB_NUM_MAX, 17
 - NV_ENC_CAPS_MB_PER_SEC_MAX, 17
 - NV_ENC_CAPS_NUM_MAX_BFRAMES, 15
 - NV_ENC_CAPS_NUM_MAX_TEMPORAL_LAYERS, 15
 - NV_ENC_CAPS_PREPROC_SUPPORT, 17
 - NV_ENC_CAPS_SEPARATE_COLOUR_PLANE, 16
 - NV_ENC_CAPS_SUPPORT_10BIT_ENCODE, 17
 - NV_ENC_CAPS_SUPPORT_ADAPTIVE_TRANSFORM, 15
 - NV_ENC_CAPS_SUPPORT_BDIRECT_MODE, 15
 - NV_ENC_CAPS_SUPPORT_CABAC, 15
 - NV_ENC_CAPS_SUPPORT_CONSTRAINED_ENCODING, 16
 - NV_ENC_CAPS_SUPPORT_CUSTOM_VBV_BUF_SIZE, 17
 - NV_ENC_CAPS_SUPPORT_DYN_BITRATE_CHANGE, 16
 - NV_ENC_CAPS_SUPPORT_DYN_FORCE_CONSTQP, 16
 - NV_ENC_CAPS_SUPPORT_DYN_RC_MODE_CHANGE, 16
 - NV_ENC_CAPS_SUPPORT_DYN_RES_CHANGE, 16
 - NV_ENC_CAPS_SUPPORT_DYNAMIC_SLICE_MODE, 17
 - NV_ENC_CAPS_SUPPORT_FIELD_ENCODING, 15
 - NV_ENC_CAPS_SUPPORT_FMO, 15
 - NV_ENC_CAPS_SUPPORT_HIERARCHICAL_BFRAMES, 16
 - NV_ENC_CAPS_SUPPORT_HIERARCHICAL_PFRAMES, 15
 - NV_ENC_CAPS_SUPPORT_INTRA_REFRESH, 16
 - NV_ENC_CAPS_SUPPORT_LOOKAHEAD, 17
 - NV_ENC_CAPS_SUPPORT_LOSSLESS_ENCODE, 17
 - NV_ENC_CAPS_SUPPORT_MEONLY_MODE, 17
 - NV_ENC_CAPS_SUPPORT_MONOCHROME, 15
 - NV_ENC_CAPS_SUPPORT_QPELMV, 15
 - NV_ENC_CAPS_SUPPORT_REF_PIC_INVALIDATION, 17
 - NV_ENC_CAPS_SUPPORT_RESERVED, 15
 - NV_ENC_CAPS_SUPPORT_SAO, 17
 - NV_ENC_CAPS_SUPPORT_SUBFRAME_READBACK, 16
 - NV_ENC_CAPS_SUPPORT_TEMPORAL_AQ, 17
 - NV_ENC_CAPS_SUPPORT_TEMPORAL_SVC, 16
 - NV_ENC_CAPS_SUPPORT_YUV444_ENCODE, 17
 - NV_ENC_CAPS_SUPPORTED_RATECONTROL_MODES, 15
 - NV_ENC_CAPS_WIDTH_MAX, 16
 - NV_ENC_DEVICE_TYPE_CUDA, 18
 - NV_ENC_DEVICE_TYPE_DIRECTX, 18
 - NV_ENC_ERR_DEVICE_NOT_EXIST, 21
 - NV_ENC_ERR_ENCODER_BUSY, 22
 - NV_ENC_ERR_ENCODER_NOT_INITIALIZED, 21
 - NV_ENC_ERR_EVENT_NOT_REGISTERD, 22
 - NV_ENC_ERR_GENERIC, 22
 - NV_ENC_ERR_INCOMPATIBLE_CLIENT_KEY, 22

- NV_ENC_ERR_INVALID_CALL, 21
- NV_ENC_ERR_INVALID_DEVICE, 21
- NV_ENC_ERR_INVALID_ENCODERDEVICE, 21
- NV_ENC_ERR_INVALID_EVENT, 21
- NV_ENC_ERR_INVALID_PARAM, 21
- NV_ENC_ERR_INVALID_PTR, 21
- NV_ENC_ERR_INVALID_VERSION, 22
- NV_ENC_ERR_LOCK_BUSY, 21
- NV_ENC_ERR_MAP_FAILED, 22
- NV_ENC_ERR_NEED_MORE_INPUT, 22
- NV_ENC_ERR_NO_ENCODE_DEVICE, 21
- NV_ENC_ERR_NOT_ENOUGH_BUFFER, 21
- NV_ENC_ERR_OUT_OF_MEMORY, 21
- NV_ENC_ERR_RESOURCE_NOT_MAPPED, 22
- NV_ENC_ERR_RESOURCE_NOT_REGISTERED, 22
- NV_ENC_ERR_RESOURCE_REGISTER_FAILED, 22
- NV_ENC_ERR_UNIMPLEMENTED, 22
- NV_ENC_ERR_UNSUPPORTED_DEVICE, 21
- NV_ENC_ERR_UNSUPPORTED_PARAM, 21
- NV_ENC_H264_ADAPTIVE_TRANSFORM_AUTOSELECT, 18
- NV_ENC_H264_ADAPTIVE_TRANSFORM_DISABLE, 18
- NV_ENC_H264_ADAPTIVE_TRANSFORM_ENABLE, 18
- NV_ENC_H264_BDIRECT_MODE_AUTOSELECT, 18
- NV_ENC_H264_BDIRECT_MODE_DISABLE, 18
- NV_ENC_H264_BDIRECT_MODE_SPATIAL, 18
- NV_ENC_H264_BDIRECT_MODE_TEMPORAL, 18
- NV_ENC_H264_ENTROPY_CODING_MODE_AUTOSELECT, 18
- NV_ENC_H264_ENTROPY_CODING_MODE_CABAC, 18
- NV_ENC_H264_ENTROPY_CODING_MODE_CAVLC, 18
- NV_ENC_H264_FMO_AUTOSELECT, 18
- NV_ENC_H264_FMO_DISABLE, 18
- NV_ENC_H264_FMO_ENABLE, 18
- NV_ENC_INPUT_RESOURCE_TYPE_CUDAARRAY, 19
- NV_ENC_INPUT_RESOURCE_TYPE_CUDEVICEPTR, 19
- NV_ENC_INPUT_RESOURCE_TYPE_DIRECTX, 19
- NV_ENC_MEMORY_HEAP_AUTOSELECT, 19
- NV_ENC_MEMORY_HEAP_SYSMEM_CACHED, 19
- NV_ENC_MEMORY_HEAP_SYSMEM_UNCACHED, 19
- NV_ENC_MEMORY_HEAP_VID, 19
- NV_ENC_MV_PRECISION_DEFAULT, 19
- NV_ENC_MV_PRECISION_FULL_PEL, 19
- NV_ENC_MV_PRECISION_HALF_PEL, 19
- NV_ENC_MV_PRECISION_QUARTER_PEL, 19
- NV_ENC_PARAMS_FRAME_FIELD_MODE_FIELD, 19
- NV_ENC_PARAMS_FRAME_FIELD_MODE_FRAME, 19
- NV_ENC_PARAMS_FRAME_FIELD_MODE_MBAFF, 19
- NV_ENC_PARAMS_RC_CBR, 20
- NV_ENC_PARAMS_RC_CBR_HQ, 20
- NV_ENC_PARAMS_RC_CBR_LOWDELAY_HQ, 20
- NV_ENC_PARAMS_RC_CONSTQP, 20
- NV_ENC_PARAMS_RC_VBR, 20
- NV_ENC_PARAMS_RC_VBR_HQ, 20
- NV_ENC_PIC_FLAG_EOS, 20
- NV_ENC_PIC_FLAG_FORCEIDR, 20
- NV_ENC_PIC_FLAG_FORCEINTRA, 20
- NV_ENC_PIC_FLAG_OUTPUT_SPSPPS, 20
- NV_ENC_PIC_STRUCT_FIELD_BOTTOM_TOP, 20
- NV_ENC_PIC_STRUCT_FIELD_TOP_BOTTOM, 20
- NV_ENC_PIC_STRUCT_FRAME, 20
- NV_ENC_PIC_TYPE_B, 20
- NV_ENC_PIC_TYPE_BI, 20
- NV_ENC_PIC_TYPE_I, 20
- NV_ENC_PIC_TYPE_IDR, 20
- NV_ENC_PIC_TYPE_INTRA_REFRESH, 20
- NV_ENC_PIC_TYPE_P, 20
- NV_ENC_PIC_TYPE_SKIPPED, 20
- NV_ENC_PIC_TYPE_UNKNOWN, 20
- NV_ENC_STEREO_PACKING_MODE_CHECKERBOARD, 21
- NV_ENC_STEREO_PACKING_MODE_COLINTERLEAVE, 21
- NV_ENC_STEREO_PACKING_MODE_FRAMESEQ, 21
- NV_ENC_STEREO_PACKING_MODE_NONE, 21
- NV_ENC_STEREO_PACKING_MODE_ROWINTERLEAVE, 21
- NV_ENC_STEREO_PACKING_MODE_SIDE_BY_SIDE, 21
- NV_ENC_STEREO_PACKING_MODE_TOPBOTTOM, 21
- NV_ENC_SUCCESS, 21
- ENCODER_STRUCTURE
- NV_ENC_BUFFER_FORMAT, 14

- NV_ENC_CAPS, 15
- NV_ENC_CAPS_PARAM_VER, 12
- NV_ENC_CONFIG_VER, 12
- NV_ENC_CREATE_BITSTREAM_BUFFER_-
VER, 12
- NV_ENC_CREATE_INPUT_BUFFER_VER, 12
- NV_ENC_CREATE_MV_BUFFER_VER, 12
- NV_ENC_DEVICE_TYPE, 17
- NV_ENC_EVENT_PARAMS_VER, 12
- NV_ENC_H264_ADAPTIVE_TRANSFORM_-
MODE, 18
- NV_ENC_H264_BDIRECT_MODE, 18
- NV_ENC_H264_ENTROPY_CODING_MODE,
18
- NV_ENC_H264_FMO_MODE, 18
- NV_ENC_HEVC_CUSIZE, 18
- NV_ENC_INITIALIZE_PARAMS_VER, 13
- NV_ENC_INPUT_RESOURCE_TYPE, 19
- NV_ENC_LEVEL, 19
- NV_ENC_LOCK_BITSTREAM_VER, 13
- NV_ENC_LOCK_INPUT_BUFFER_VER, 13
- NV_ENC_MAP_INPUT_RESOURCE_VER, 13
- NV_ENC_MEMORY_HEAP, 19
- NV_ENC_MEONLY_PARAMS_VER, 13
- NV_ENC_MV_PRECISION, 19
- NV_ENC_OPEN_ENCODE_SESSION_EX_-
PARAMS_VER, 13
- NV_ENC_PARAMS_FRAME_FIELD_MODE, 19
- NV_ENC_PARAMS_RC_2_PASS_FRAMESIZE_-
CAP, 13
- NV_ENC_PARAMS_RC_2_PASS_QUALITY, 13
- NV_ENC_PARAMS_RC_2_PASS_VBR, 13
- NV_ENC_PARAMS_RC_CBR2, 13
- NV_ENC_PARAMS_RC_MODE, 19
- NV_ENC_PARAMS_RC_VBR_MINQP, 13
- NV_ENC_PIC_FLAGS, 20
- NV_ENC_PIC_PARAMS_VER, 14
- NV_ENC_PIC_STRUCT, 20
- NV_ENC_PIC_TYPE, 20
- NV_ENC_PRESET_CONFIG_VER, 14
- NV_ENC_RC_PARAMS_VER, 14
- NV_ENC_RECONFIGURE_PARAMS_VER, 14
- NV_ENC_REGISTER_RESOURCE_VER, 14
- NV_ENC_SEQUENCE_PARAM_PAYLOAD_-
VER, 14
- NV_ENC_STAT_VER, 14
- NV_ENC_STEREO_PACKING_MODE, 20
- NVENCSTATUS, 21
- encodeWidth
 - NV_ENC_INITIALIZE_PARAMS, 79
- entropyCodingMode
 - NV_ENC_CONFIG_H264, 57
- fmoMode
 - NV_ENC_CONFIG_H264, 57
- forceIDR
 - NV_ENC_RECONFIGURE_PARAMS, 110
- forceIntraRefreshWithFrameCnt
 - NV_ENC_PIC_PARAMS_H264, 98
 - NV_ENC_PIC_PARAMS_HEVC, 101
- frameAvgQP
 - NV_ENC_LOCK_BITSTREAM, 82
- frameFieldMode
 - NV_ENC_CONFIG, 53
- frameIdx
 - NV_ENC_LOCK_BITSTREAM, 83
 - NV_ENC_PIC_PARAMS, 95
- frameIntervalP
 - NV_ENC_CONFIG, 53
- frameRateDen
 - NV_ENC_INITIALIZE_PARAMS, 79
- frameRateNum
 - NV_ENC_INITIALIZE_PARAMS, 79
- frameSatd
 - NV_ENC_LOCK_BITSTREAM, 83
- gopLength
 - NV_ENC_CONFIG, 53
- GUID, 49
 - Data1, 49
 - Data2, 49
 - Data3, 49
 - Data4, 49
- h264Config
 - NV_ENC_CODEC_CONFIG, 51
- h264MeOnlyConfig
 - NV_ENC_CODEC_CONFIG, 51
- h264PicParams
 - NV_ENC_CODEC_PIC_PARAMS, 52
- h264VUIParameters
 - NV_ENC_CONFIG_H264, 57
- height
 - NV_ENC_CREATE_INPUT_BUFFER, 72
 - NV_ENC_REGISTER_RESOURCE, 111
- hevcConfig
 - NV_ENC_CODEC_CONFIG, 51
- hevcMeOnlyConfig
 - NV_ENC_CODEC_CONFIG, 51
- hevcPicParams
 - NV_ENC_CODEC_PIC_PARAMS, 52
- hevcVUIParameters
 - NV_ENC_CONFIG_HEVC, 66
- hierarchicalBFrames
 - NV_ENC_CONFIG_H264, 57
- hierarchicalPFrames
 - NV_ENC_CONFIG_H264, 57
- hwEncodeStatus

- NV_ENC_LOCK_BITSTREAM, 83
- idrPeriod
 - NV_ENC_CONFIG_H264, 57
 - NV_ENC_CONFIG_HEVC, 66
- inBufferSize
 - NV_ENC_SEQUENCE_PARAM_PAYLOAD, 114
- initialRCQP
 - NV_ENC_RC_PARAMS, 107
- inputBuffer
 - NV_ENC_CREATE_INPUT_BUFFER, 72
 - NV_ENC_LOCK_INPUT_BUFFER, 85
 - NV_ENC_MEONLY_PARAMS, 89
 - NV_ENC_PIC_PARAMS, 95
- inputDuration
 - NV_ENC_PIC_PARAMS, 95
- inputHeight
 - NV_ENC_MEONLY_PARAMS, 89
 - NV_ENC_PIC_PARAMS, 95
- inputPitch
 - NV_ENC_PIC_PARAMS, 95
- inputResource
 - NV_ENC_MAP_INPUT_RESOURCE, 87
- inputTimeStamp
 - NV_ENC_PIC_PARAMS, 95
- inputWidth
 - NV_ENC_MEONLY_PARAMS, 89
 - NV_ENC_PIC_PARAMS, 95
- intraRefreshCnt
 - NV_ENC_CONFIG_H264, 57
 - NV_ENC_CONFIG_HEVC, 66
- intraRefreshPeriod
 - NV_ENC_CONFIG_H264, 57
 - NV_ENC_CONFIG_HEVC, 66
- lastCUInCTB
 - NV_ENC_HEVC_MV_DATA, 77
- lastOfMB
 - NVENC_EXTERNAL_ME_HINT, 123
- lastofPart
 - NVENC_EXTERNAL_ME_HINT, 123
- lastValidByteOffset
 - NV_ENC_STAT, 116
- left
 - NVENC_RECT, 126
- level
 - NV_ENC_CONFIG_H264, 58
 - NV_ENC_CONFIG_HEVC, 66
- lookaheadDepth
 - NV_ENC_RC_PARAMS, 108
- ltrFrame
 - NV_ENC_LOCK_BITSTREAM, 83
- ltrFrameBitmap
 - NV_ENC_LOCK_BITSTREAM, 83
- ltrFrameIdx
 - NV_ENC_LOCK_BITSTREAM, 83
- ltrMarkFrame
 - NV_ENC_PIC_PARAMS_H264, 99
 - NV_ENC_PIC_PARAMS_HEVC, 101
- ltrMarkFrameIdx
 - NV_ENC_PIC_PARAMS_H264, 99
 - NV_ENC_PIC_PARAMS_HEVC, 102
- ltrNumFrames
 - NV_ENC_CONFIG_H264, 58
 - NV_ENC_CONFIG_HEVC, 66
- ltrTrustMode
 - NV_ENC_CONFIG_H264, 58
 - 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, 99
 - NV_ENC_PIC_PARAMS_HEVC, 102
- mappedBufferFmt
 - NV_ENC_MAP_INPUT_RESOURCE, 87
- mappedResource
 - NV_ENC_MAP_INPUT_RESOURCE, 87
- maxBitRate
 - NV_ENC_RC_PARAMS, 108
- maxCUSize
 - NV_ENC_CONFIG_HEVC, 67
- maxEncodeHeight
 - NV_ENC_INITIALIZE_PARAMS, 80
- maxEncodeWidth
 - NV_ENC_INITIALIZE_PARAMS, 80
- maxMEHintCountsPerBlock
 - NV_ENC_INITIALIZE_PARAMS, 80
- maxNumRefFrames
 - NV_ENC_CONFIG_H264, 58
- maxNumRefFramesInDPB
 - NV_ENC_CONFIG_HEVC, 67
- maxQP
 - NV_ENC_RC_PARAMS, 108
- maxTemporalLayers
 - NV_ENC_CONFIG_H264, 58
- maxTemporalLayersMinus1
 - NV_ENC_CONFIG_HEVC, 67
- mbType
 - NV_ENC_H264_MV_DATA, 76
- meExternalHints
 - NV_ENC_PIC_PARAMS, 95
- meHintCountsPerBlock
 - NV_ENC_PIC_PARAMS, 95

- meHintRefPicDist
 - NV_ENC_PIC_PARAMS, 96
- memoryHeap
 - NV_ENC_CREATE_BITSTREAM_BUFFER, 70
 - NV_ENC_CREATE_INPUT_BUFFER, 72
- minCUSize
 - NV_ENC_CONFIG_HEVC, 67
- minQP
 - NV_ENC_RC_PARAMS, 108
- monoChromeEncoding
 - NV_ENC_CONFIG, 53
- mv
 - NV_ENC_H264_MV_DATA, 76
 - NV_ENC_HEVC_MV_DATA, 77
- mvBuffer
 - NV_ENC_CREATE_MV_BUFFER, 74
 - NV_ENC_MEONLY_PARAMS, 89
- mvPrecision
 - NV_ENC_CONFIG, 53
- mvx
 - NV_ENC_MVECTOR, 91
 - NVENC_EXTERNAL_ME_HINT, 123
- mvy
 - NV_ENC_MVECTOR, 91
 - NVENC_EXTERNAL_ME_HINT, 123
- numCandsPerBlk16x16
 - NVENC_EXTERNAL_ME_HINT_COUNTS_-
PER_BLOCKTYPE, 125
- numCandsPerBlk16x8
 - NVENC_EXTERNAL_ME_HINT_COUNTS_-
PER_BLOCKTYPE, 125
- numCandsPerBlk8x16
 - NVENC_EXTERNAL_ME_HINT_COUNTS_-
PER_BLOCKTYPE, 125
- numCandsPerBlk8x8
 - NVENC_EXTERNAL_ME_HINT_COUNTS_-
PER_BLOCKTYPE, 125
- numSlices
 - NV_ENC_LOCK_BITSTREAM, 83
- numTemporalLayers
 - NV_ENC_CONFIG_H264, 58
- NV_ENC_BUFFER_FORMAT_ABGR
 - ENCODER_STRUCTURE, 15
- NV_ENC_BUFFER_FORMAT_ABGR10
 - ENCODER_STRUCTURE, 15
- NV_ENC_BUFFER_FORMAT_ARGB
 - ENCODER_STRUCTURE, 15
- NV_ENC_BUFFER_FORMAT_ARGB10
 - ENCODER_STRUCTURE, 15
- NV_ENC_BUFFER_FORMAT_AYUV
 - ENCODER_STRUCTURE, 15
- 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, 15
- NV_ENC_BUFFER_FORMAT_YV12
 - ENCODER_STRUCTURE, 14
- NV_ENC_CAPS_ASYNC_ENCODE_SUPPORT
 - ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_EXPOSED_COUNT
 - ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_HEIGHT_MAX
 - ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_LEVEL_MAX
 - ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_LEVEL_MIN
 - ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_MB_NUM_MAX
 - ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_MB_PER_SEC_MAX
 - ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_NUM_MAX_BFRAMES
 - ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_NUM_MAX_TEMPORAL_LAYERS
 - ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_PREPROC_SUPPORT
 - ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_SEPARATE_COLOUR_PLANE
 - ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_SUPPORT_10BIT_ENCODE
 - ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_SUPPORT_ADAPTIVE_-
TRANSFORM
 - ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_SUPPORT_BDIRECT_MODE
 - ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_SUPPORT_CABAC
 - ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_SUPPORT_CONSTRAINED_-
ENCODING
 - ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_SUPPORT_CUSTOM_VBV_BUF_-
SIZE
 - ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_SUPPORT_DYN_BITRATE_-
CHANGE
 - ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_SUPPORT_DYN_FORCE_-
CONSTQP

- ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_SUPPORT_DYN_RCMODE_-
CHANGE
ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_SUPPORT_DYN_RES_CHANGE
ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_SUPPORT_DYNAMIC_SLICE_-
MODE
ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_SUPPORT_FIELD_ENCODING
ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_SUPPORT_FMO
ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_SUPPORT_HIERARCHICAL_-
BFRAMES
ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_SUPPORT_HIERARCHICAL_-
PFRAMES
ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_SUPPORT_INTRA_REFRESH
ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_SUPPORT_LOOKAHEAD
ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_SUPPORT_LOSSLESS_ENCODE
ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_SUPPORT_MEONLY_MODE
ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_SUPPORT_MONOCHROME
ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_SUPPORT_QPELMV
ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_SUPPORT_REF_PIC_-
INVALIDATION
ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_SUPPORT_RESERVED
ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_SUPPORT_SAO
ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_SUPPORT_SUBFRAME_-
READBACK
ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_SUPPORT_TEMPORAL_AQ
ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_SUPPORT_TEMPORAL_SVC
ENCODER_STRUCTURE, 16
- NV_ENC_CAPS_SUPPORT_YUV444_ENCODE
ENCODER_STRUCTURE, 17
- NV_ENC_CAPS_SUPPORTED_RATECONTROL_-
MODES
ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_WIDTH_MAX
ENCODER_STRUCTURE, 16
- NV_ENC_DEVICE_TYPE_CUDA
ENCODER_STRUCTURE, 18
- NV_ENC_DEVICE_TYPE_DIRECTX
ENCODER_STRUCTURE, 18
- NV_ENC_ERR_DEVICE_NOT_EXIST
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_ENCODER_BUSY
ENCODER_STRUCTURE, 22
- NV_ENC_ERR_ENCODER_NOT_INITIALIZED
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_EVENT_NOT_REGISTERD
ENCODER_STRUCTURE, 22
- NV_ENC_ERR_GENERIC
ENCODER_STRUCTURE, 22
- NV_ENC_ERR_INCOMPATIBLE_CLIENT_KEY
ENCODER_STRUCTURE, 22
- NV_ENC_ERR_INVALID_CALL
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_INVALID_DEVICE
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_INVALID_ENCODERDEVICE
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_INVALID_EVENT
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_INVALID_PARAM
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_INVALID_PTR
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_INVALID_VERSION
ENCODER_STRUCTURE, 22
- NV_ENC_ERR_LOCK_BUSY
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_MAP_FAILED
ENCODER_STRUCTURE, 22
- NV_ENC_ERR_NEED_MORE_INPUT
ENCODER_STRUCTURE, 22
- NV_ENC_ERR_NO_ENCODE_DEVICE
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_NOT_ENOUGH_BUFFER
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_OUT_OF_MEMORY
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_RESOURCE_NOT_MAPPED
ENCODER_STRUCTURE, 22
- NV_ENC_ERR_RESOURCE_NOT_REGISTERED
ENCODER_STRUCTURE, 22
- NV_ENC_ERR_RESOURCE_REGISTER_FAILED
ENCODER_STRUCTURE, 22
- NV_ENC_ERR_UNIMPLEMENTED
ENCODER_STRUCTURE, 22
- NV_ENC_ERR_UNSUPPORTED_DEVICE
ENCODER_STRUCTURE, 21
- NV_ENC_ERR_UNSUPPORTED_PARAM
ENCODER_STRUCTURE, 21
- NV_ENC_H264_ADAPTIVE_TRANSFORM_-
AUTOSELECT

- ENCODER_STRUCTURE, 18
- NV_ENC_H264_ADAPTIVE_TRANSFORM_-
DISABLE
ENCODER_STRUCTURE, 18
- NV_ENC_H264_ADAPTIVE_TRANSFORM_-
ENABLE
ENCODER_STRUCTURE, 18
- NV_ENC_H264_BDIRECT_MODE_AUTOSELECT
ENCODER_STRUCTURE, 18
- NV_ENC_H264_BDIRECT_MODE_DISABLE
ENCODER_STRUCTURE, 18
- NV_ENC_H264_BDIRECT_MODE_SPATIAL
ENCODER_STRUCTURE, 18
- NV_ENC_H264_BDIRECT_MODE_TEMPORAL
ENCODER_STRUCTURE, 18
- NV_ENC_H264_ENTROPY_CODING_MODE_-
AUTOSELECT
ENCODER_STRUCTURE, 18
- NV_ENC_H264_ENTROPY_CODING_MODE_-
CABAC
ENCODER_STRUCTURE, 18
- NV_ENC_H264_ENTROPY_CODING_MODE_-
CAVLC
ENCODER_STRUCTURE, 18
- NV_ENC_H264_FMO_AUTOSELECT
ENCODER_STRUCTURE, 18
- NV_ENC_H264_FMO_DISABLE
ENCODER_STRUCTURE, 18
- NV_ENC_H264_FMO_ENABLE
ENCODER_STRUCTURE, 18
- NV_ENC_INPUT_RESOURCE_TYPE_CUDAARRAY
ENCODER_STRUCTURE, 19
- NV_ENC_INPUT_RESOURCE_TYPE_-
CUDADEVICEPTR
ENCODER_STRUCTURE, 19
- NV_ENC_INPUT_RESOURCE_TYPE_DIRECTX
ENCODER_STRUCTURE, 19
- NV_ENC_MEMORY_HEAP_AUTOSELECT
ENCODER_STRUCTURE, 19
- NV_ENC_MEMORY_HEAP_SYSTEMEM_CACHED
ENCODER_STRUCTURE, 19
- NV_ENC_MEMORY_HEAP_SYSTEMEM_UNCACHED
ENCODER_STRUCTURE, 19
- NV_ENC_MEMORY_HEAP_VID
ENCODER_STRUCTURE, 19
- NV_ENC_MV_PRECISION_DEFAULT
ENCODER_STRUCTURE, 19
- NV_ENC_MV_PRECISION_FULL_PEL
ENCODER_STRUCTURE, 19
- NV_ENC_MV_PRECISION_HALF_PEL
ENCODER_STRUCTURE, 19
- NV_ENC_MV_PRECISION_QUARTER_PEL
ENCODER_STRUCTURE, 19
- NV_ENC_PARAMS_FRAME_FIELD_MODE_FIELD
ENCODER_STRUCTURE, 19
- NV_ENC_PARAMS_FRAME_FIELD_MODE_-
FRAME
ENCODER_STRUCTURE, 19
- NV_ENC_PARAMS_FRAME_FIELD_MODE_-
MBAFF
ENCODER_STRUCTURE, 19
- NV_ENC_PARAMS_RC_CBR
ENCODER_STRUCTURE, 20
- NV_ENC_PARAMS_RC_CBR_HQ
ENCODER_STRUCTURE, 20
- NV_ENC_PARAMS_RC_CBR_LOWDELAY_HQ
ENCODER_STRUCTURE, 20
- NV_ENC_PARAMS_RC_CONSTQP
ENCODER_STRUCTURE, 20
- NV_ENC_PARAMS_RC_VBR
ENCODER_STRUCTURE, 20
- NV_ENC_PARAMS_RC_VBR_HQ
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_FLAG_EOS
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_FLAG_FORCEIDR
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_FLAG_FORCEINTRA
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_FLAG_OUTPUT_SPSPPS
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_STRUCT_FIELD_BOTTOM_TOP
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_STRUCT_FIELD_TOP_BOTTOM
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_STRUCT_FRAME
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_TYPE_B
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_TYPE_BI
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_TYPE_I
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_TYPE_IDR
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_TYPE_INTRA_REFRESH
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_TYPE_P
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_TYPE_SKIPPED
ENCODER_STRUCTURE, 20
- NV_ENC_PIC_TYPE_UNKNOWN
ENCODER_STRUCTURE, 20
- NV_ENC_STEREO_PACKING_MODE_-
CHECKERBOARD
ENCODER_STRUCTURE, 21
- NV_ENC_STEREO_PACKING_MODE_-
COLINTERLEAVE

- ENCODER_STRUCTURE, 21
- NV_ENC_STEREO_PACKING_MODE_FRAMESEQ
 - ENCODER_STRUCTURE, 21
- NV_ENC_STEREO_PACKING_MODE_NONE
 - ENCODER_STRUCTURE, 21
- NV_ENC_STEREO_PACKING_MODE_-
 - ROWINTERLEAVE
 - ENCODER_STRUCTURE, 21
- NV_ENC_STEREO_PACKING_MODE_SIDE BYSIDE
 - ENCODER_STRUCTURE, 21
- NV_ENC_STEREO_PACKING_MODE_-
 - TOPBOTTOM
 - ENCODER_STRUCTURE, 21
- NV_ENC_SUCCESS
 - ENCODER_STRUCTURE, 21
- NV_ENC_BUFFER_FORMAT
 - ENCODER_STRUCTURE, 14
- NV_ENC_CAPS
 - ENCODER_STRUCTURE, 15
- NV_ENC_CAPS_PARAM, 50
 - capsToQuery, 50
 - reserved, 50
 - version, 50
- NV_ENC_CAPS_PARAM_VER
 - ENCODER_STRUCTURE, 12
- NV_ENC_CODEC_CONFIG, 51
 - h264Config, 51
 - h264MeOnlyConfig, 51
 - hevcConfig, 51
 - hevcMeOnlyConfig, 51
 - reserved, 51
- NV_ENC_CODEC_PIC_PARAMS, 52
 - h264PicParams, 52
 - hevcPicParams, 52
 - reserved, 52
- NV_ENC_CONFIG, 53
 - encodeCodecConfig, 53
 - frameFieldMode, 53
 - frameIntervalP, 53
 - gopLength, 53
 - monoChromeEncoding, 53
 - mvPrecision, 53
 - profileGUID, 54
 - rcParams, 54
 - reserved, 54
 - reserved2, 54
 - version, 54
- NV_ENC_CONFIG_H264, 55
 - adaptiveTransformMode, 56
 - bdirectMode, 56
 - chromaFormatIDC, 56
 - disableDeblockingFilterIDC, 56
 - disableSPSPPS, 56
 - enableConstrainedEncoding, 56
 - enableIntraRefresh, 56
 - enableLTR, 56
 - enableStereoMVC, 56
 - enableTemporalSVC, 57
 - enableVFR, 57
 - entropyCodingMode, 57
 - fmoMode, 57
 - h264VUIParameters, 57
 - hierarchicalBFrames, 57
 - hierarchicalPFrames, 57
 - idrPeriod, 57
 - intraRefreshCnt, 57
 - intraRefreshPeriod, 57
 - level, 58
 - ltrNumFrames, 58
 - ltrTrustMode, 58
 - maxNumRefFrames, 58
 - maxTemporalLayers, 58
 - numTemporalLayers, 58
 - outputAUD, 58
 - outputBufferingPeriodSEI, 58
 - outputFramePackingSEI, 58
 - outputPictureTimingSEI, 58
 - outputRecoveryPointSEI, 59
 - ppsId, 59
 - qpPrimeYZeroTransformBypassFlag, 59
 - repeatSPSPPS, 59
 - reserved1, 59
 - reserved2, 59
 - reservedBitFields, 59
 - separateColourPlaneFlag, 59
 - sliceMode, 59
 - sliceModeData, 59
 - spsId, 60
 - stereoMode, 60
 - useConstrainedIntraPred, 60
- NV_ENC_CONFIG_H264_MEONLY, 61
 - bStereoEnable, 61
 - disableIntraSearch, 61
 - disablePartition16x16, 61
 - disablePartition16x8, 61
 - disablePartition8x16, 61
 - disablePartition8x8, 61
 - reserved, 61
 - reserved1, 62
 - reserved2, 62
- NV_ENC_CONFIG_H264_VUI_PARAMETERS, 63
 - bitstreamRestrictionFlag, 63
 - chromaSampleLocationBot, 63
 - chromaSampleLocationFlag, 63
 - chromaSampleLocationTop, 63
 - colourDescriptionPresentFlag, 63
 - colourMatrix, 63
 - colourPrimaries, 64

- overscanInfo, 64
- overscanInfoPresentFlag, 64
- transferCharacteristics, 64
- videoFormat, 64
- videoFullRangeFlag, 64
- videoSignalTypePresentFlag, 64
- NV_ENC_CONFIG_HEVC, 65
 - chromaFormatIDC, 65
 - disableDeblockAcrossSliceBoundary, 65
 - disableSPSPPS, 66
 - enableIntraRefresh, 66
 - enableLTR, 66
 - hevcVUIParameters, 66
 - idrPeriod, 66
 - intraRefreshCnt, 66
 - intraRefreshPeriod, 66
 - level, 66
 - ltrNumFrames, 66
 - ltrTrustMode, 66
 - maxCUSize, 67
 - maxNumRefFramesInDPB, 67
 - maxTemporalLayersMinus1, 67
 - minCUSize, 67
 - outputAUD, 67
 - outputBufferingPeriodSEI, 67
 - outputPictureTimingSEI, 67
 - pixelBitDepthMinus8, 67
 - ppsId, 67
 - repeatSPSPPS, 67
 - reserved, 67
 - reserved1, 68
 - reserved2, 68
 - sliceMode, 68
 - sliceModeData, 68
 - spsId, 68
 - tier, 68
 - useConstrainedIntraPred, 68
 - vpsId, 68
- NV_ENC_CONFIG_HEVC_MEONLY, 69
 - reserved, 69
 - reserved1, 69
- NV_ENC_CONFIG_VER
 - ENCODER_STRUCTURE, 12
- NV_ENC_CREATE_BITSTREAM_BUFFER, 70
 - bitstreamBuffer, 70
 - bitstreamBufferPtr, 70
 - memoryHeap, 70
 - reserved, 70
 - reserved1, 70
 - reserved2, 70
 - size, 70
 - version, 71
- NV_ENC_CREATE_BITSTREAM_BUFFER_VER
 - ENCODER_STRUCTURE, 12
- NV_ENC_CREATE_INPUT_BUFFER, 72
 - bufferFmt, 72
 - height, 72
 - inputBuffer, 72
 - memoryHeap, 72
 - pSysMemBuffer, 72
 - reserved, 72
 - reserved1, 72
 - reserved2, 73
 - version, 73
 - width, 73
- NV_ENC_CREATE_INPUT_BUFFER_VER
 - ENCODER_STRUCTURE, 12
- NV_ENC_CREATE_MV_BUFFER, 74
 - mvBuffer, 74
 - reserved1, 74
 - reserved2, 74
 - version, 74
- NV_ENC_CREATE_MV_BUFFER_VER
 - ENCODER_STRUCTURE, 12
- NV_ENC_DEVICE_TYPE
 - ENCODER_STRUCTURE, 17
- NV_ENC_EVENT_PARAMS, 75
 - completionEvent, 75
 - reserved, 75
 - reserved1, 75
 - reserved2, 75
 - version, 75
- NV_ENC_EVENT_PARAMS_VER
 - ENCODER_STRUCTURE, 12
- NV_ENC_H264_ADAPTIVE_TRANSFORM_MODE
 - ENCODER_STRUCTURE, 18
- NV_ENC_H264_BDIRECT_MODE
 - ENCODER_STRUCTURE, 18
- NV_ENC_H264_ENTROPY_CODING_MODE
 - ENCODER_STRUCTURE, 18
- NV_ENC_H264_FMO_MODE
 - ENCODER_STRUCTURE, 18
- NV_ENC_H264_MV_DATA, 76
 - mbType, 76
 - mv, 76
 - partitionType, 76
 - reserved, 76
- NV_ENC_HEVC_CUSIZE
 - ENCODER_STRUCTURE, 18
- NV_ENC_HEVC_MV_DATA, 77
 - cuSize, 77
 - cuType, 77
 - lastCUInCTB, 77
 - mv, 77
 - partitionMode, 77
- NV_ENC_INITIALIZE_PARAMS, 78
 - darHeight, 78
 - darWidth, 78

- enableEncodeAsync, 78
- enableExternalMEHints, 78
- enableMEOnlyMode, 79
- enablePTD, 79
- enableSubFrameWrite, 79
- encodeConfig, 79
- encodeGUID, 79
- encodeHeight, 79
- encodeWidth, 79
- frameRateDen, 79
- frameRateNum, 79
- maxEncodeHeight, 80
- maxEncodeWidth, 80
- maxMEHintCountsPerBlock, 80
- presetGUID, 80
- privData, 80
- privDataSize, 80
- reportSliceOffsets, 80
- reserved, 80
- reserved2, 80
- reservedBitFields, 81
- version, 81
- NV_ENC_INITIALIZE_PARAMS_VER
ENCODER_STRUCTURE, 13
- NV_ENC_INPUT_RESOURCE_TYPE
ENCODER_STRUCTURE, 19
- NV_ENC_LEVEL
ENCODER_STRUCTURE, 19
- NV_ENC_LOCK_BITSTREAM, 82
 - bitstreamBufferPtr, 82
 - bitstreamSizeInBytes, 82
 - doNotWait, 82
 - frameAvgQP, 82
 - frameIdx, 83
 - frameSatd, 83
 - hwEncodeStatus, 83
 - ltrFrame, 83
 - ltrFrameBitmap, 83
 - ltrFrameIdx, 83
 - numSlices, 83
 - outputBitstream, 83
 - outputDuration, 83
 - outputTimeStamp, 83
 - pictureStruct, 83
 - pictureType, 84
 - reserved, 84
 - reserved2, 84
 - reservedBitFields, 84
 - sliceOffsets, 84
 - version, 84
- NV_ENC_LOCK_BITSTREAM_VER
ENCODER_STRUCTURE, 13
- NV_ENC_LOCK_INPUT_BUFFER, 85
 - bufferDataPtr, 85
 - doNotWait, 85
 - inputBuffer, 85
 - pitch, 85
 - reserved1, 85
 - reserved2, 85
 - reservedBitFields, 85
 - version, 86
- NV_ENC_LOCK_INPUT_BUFFER_VER
ENCODER_STRUCTURE, 13
- NV_ENC_MAP_INPUT_RESOURCE, 87
 - inputResource, 87
 - mappedBufferFmt, 87
 - mappedResource, 87
 - registeredResource, 87
 - reserved1, 87
 - reserved2, 87
 - subResourceIndex, 87
 - version, 88
- NV_ENC_MAP_INPUT_RESOURCE_VER
ENCODER_STRUCTURE, 13
- NV_ENC_MEMORY_HEAP
ENCODER_STRUCTURE, 19
- NV_ENC_MEONLY_PARAMS, 89
 - bufferFmt, 89
 - completionEvent, 89
 - inputBuffer, 89
 - inputHeight, 89
 - inputWidth, 89
 - mvBuffer, 89
 - referenceFrame, 90
 - reserved1, 90
 - reserved2, 90
 - version, 90
 - viewID, 90
- NV_ENC_MEONLY_PARAMS_VER
ENCODER_STRUCTURE, 13
- NV_ENC_MV_PRECISION
ENCODER_STRUCTURE, 19
- NV_ENC_MVECTOR, 91
 - mvx, 91
 - mvy, 91
- NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS, 92
 - apiVersion, 92
 - device, 92
 - deviceType, 92
 - reserved, 92
 - reserved1, 92
 - reserved2, 92
 - version, 92
- NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS_VER
ENCODER_STRUCTURE, 13
- NV_ENC_PARAMS_FRAME_FIELD_MODE

- ENCODER_STRUCTURE, 19
- NV_ENC_PARAMS_RC_2_PASS_FRAMESIZE_CAP
 - ENCODER_STRUCTURE, 13
- NV_ENC_PARAMS_RC_2_PASS_QUALITY
 - ENCODER_STRUCTURE, 13
- NV_ENC_PARAMS_RC_2_PASS_VBR
 - ENCODER_STRUCTURE, 13
- NV_ENC_PARAMS_RC_CBR2
 - ENCODER_STRUCTURE, 13
- NV_ENC_PARAMS_RC_MODE
 - ENCODER_STRUCTURE, 19
- NV_ENC_PARAMS_RC_VBR_MINQP
 - ENCODER_STRUCTURE, 13
- NV_ENC_PIC_FLAGS
 - ENCODER_STRUCTURE, 20
- NV_ENC_PIC_PARAMS, 94
 - bufferFmt, 94
 - codecPicParams, 94
 - completionEvent, 94
 - encodePicFlags, 95
 - frameIdx, 95
 - inputBuffer, 95
 - inputDuration, 95
 - inputHeight, 95
 - inputPitch, 95
 - inputTimeStamp, 95
 - inputWidth, 95
 - meExternalHints, 95
 - meHintCountsPerBlock, 95
 - meHintRefPicDist, 96
 - outputBitstream, 96
 - pictureStruct, 96
 - pictureType, 96
 - qpDeltaMap, 96
 - qpDeltaMapSize, 96
 - reserved1, 96
 - reserved2, 96
 - reserved3, 96
 - reserved4, 97
 - reservedBitFields, 97
 - version, 97
- NV_ENC_PIC_PARAMS_H264, 98
 - colourPlaneId, 98
 - constrainedFrame, 98
 - displayPOCSyntax, 98
 - forceIntraRefreshWithFrameCnt, 98
 - ltrMarkFrame, 99
 - ltrMarkFrameIdx, 99
 - ltrUsageMode, 99
 - ltrUseFrameBitmap, 99
 - ltrUseFrames, 99
 - refPicFlag, 99
 - reserved, 99
 - reserved2, 99
 - reserved3, 99
 - reservedBitFields, 99
 - seiPayloadArray, 99
 - seiPayloadArrayCnt, 100
 - sliceMode, 100
 - sliceModeData, 100
 - sliceModeDataUpdate, 100
 - sliceTypeArrayCnt, 100
 - sliceTypeData, 100
- NV_ENC_PIC_PARAMS_HEVC, 101
 - constrainedFrame, 101
 - displayPOCSyntax, 101
 - forceIntraRefreshWithFrameCnt, 101
 - ltrMarkFrame, 101
 - ltrMarkFrameIdx, 102
 - ltrUsageMode, 102
 - ltrUseFrameBitmap, 102
 - ltrUseFrames, 102
 - refPicFlag, 102
 - reserved, 102
 - reserved2, 102
 - reserved3, 102
 - reservedBitFields, 102
 - seiPayloadArray, 102
 - seiPayloadArrayCnt, 102
 - sliceMode, 103
 - sliceModeData, 103
 - sliceModeDataUpdate, 103
 - sliceTypeArrayCnt, 103
 - sliceTypeData, 103
 - temporalId, 103
- NV_ENC_PIC_PARAMS_VER
 - ENCODER_STRUCTURE, 14
- NV_ENC_PIC_STRUCT
 - ENCODER_STRUCTURE, 20
- NV_ENC_PIC_TYPE
 - ENCODER_STRUCTURE, 20
- NV_ENC_PRESET_CONFIG, 104
 - presetCfg, 104
 - reserved1, 104
 - reserved2, 104
 - version, 104
- NV_ENC_PRESET_CONFIG_VER
 - ENCODER_STRUCTURE, 14
- NV_ENC_QP, 105
- NV_ENC_RC_PARAMS, 106
 - aqStrength, 106
 - averageBitRate, 106
 - constQP, 106
 - disableBadapt, 107
 - disableIadapt, 107
 - enableAQ, 107
 - enableExtQPDeltaMap, 107
 - enableInitialRCQP, 107

- enableLookahead, [107](#)
- enableMaxQP, [107](#)
- enableMinQP, [107](#)
- enableNonRefP, [107](#)
- enableTemporalAQ, [107](#)
- initialRCQP, [107](#)
- lookaheadDepth, [108](#)
- maxBitRate, [108](#)
- maxQP, [108](#)
- minQP, [108](#)
- rateControlMode, [108](#)
- reservedBitFields, [108](#)
- strictGOPTarget, [108](#)
- targetQuality, [108](#)
- temporalLayerIdxMask, [108](#)
- temporalLayerQP, [108](#)
- vbvBufferSize, [108](#)
- vbvInitialDelay, [109](#)
- zeroReorderDelay, [109](#)
- NV_ENC_RC_PARAMS_VER
ENCODER_STRUCTURE, [14](#)
- NV_ENC_RECONFIGURE_PARAMS, [110](#)
- forceIDR, [110](#)
- reInitEncodeParams, [110](#)
- resetEncoder, [110](#)
- version, [110](#)
- NV_ENC_RECONFIGURE_PARAMS_VER
ENCODER_STRUCTURE, [14](#)
- NV_ENC_REGISTER_RESOURCE, [111](#)
- bufferFormat, [111](#)
- height, [111](#)
- pitch, [111](#)
- registeredResource, [111](#)
- reserved1, [111](#)
- reserved2, [111](#)
- resourceToRegister, [111](#)
- resourceType, [112](#)
- subResourceIndex, [112](#)
- version, [112](#)
- width, [112](#)
- NV_ENC_REGISTER_RESOURCE_VER
ENCODER_STRUCTURE, [14](#)
- NV_ENC_SEI_PAYLOAD, [113](#)
- payload, [113](#)
- payloadSize, [113](#)
- payloadType, [113](#)
- NV_ENC_SEQUENCE_PARAM_PAYLOAD, [114](#)
- inBufferSize, [114](#)
- outSPSPSPayloadSize, [114](#)
- ppsId, [114](#)
- reserved, [114](#)
- reserved2, [114](#)
- spsId, [114](#)
- spsppsBuffer, [114](#)
- version, [115](#)
- NV_ENC_SEQUENCE_PARAM_PAYLOAD_VER
ENCODER_STRUCTURE, [14](#)
- NV_ENC_STAT, [116](#)
- bitStreamSize, [116](#)
- lastValidByteOffset, [116](#)
- outputBitStream, [116](#)
- picIdx, [116](#)
- picType, [116](#)
- reserved, [116](#)
- reserved1, [116](#)
- reserved2, [117](#)
- sliceOffsets, [117](#)
- version, [117](#)
- NV_ENC_STAT_VER
ENCODER_STRUCTURE, [14](#)
- NV_ENC_STEREO_PACKING_MODE
ENCODER_STRUCTURE, [20](#)
- NV_ENCODE_API_FUNCTION_LIST, [118](#)
- nvEncCreateBitstreamBuffer, [119](#)
- nvEncCreateInputBuffer, [119](#)
- nvEncCreateMVBuffer, [119](#)
- nvEncDestroyBitstreamBuffer, [119](#)
- nvEncDestroyEncoder, [119](#)
- nvEncDestroyInputBuffer, [119](#)
- nvEncDestroyMVBuffer, [119](#)
- nvEncEncodePicture, [119](#)
- nvEncGetEncodeCaps, [119](#)
- nvEncGetEncodeGUIDCount, [119](#)
- nvEncGetEncodeGUIDs, [119](#)
- nvEncGetEncodePresetConfig, [120](#)
- nvEncGetEncodePresetCount, [120](#)
- nvEncGetEncodePresetGUIDs, [120](#)
- nvEncGetEncodeProfileGUIDCount, [120](#)
- nvEncGetEncodeProfileGUIDs, [120](#)
- nvEncGetEncodeStats, [120](#)
- nvEncGetInputFormatCount, [120](#)
- nvEncGetInputFormats, [120](#)
- nvEncGetSequenceParams, [120](#)
- nvEncInitializeEncoder, [120](#)
- nvEncInvalidateRefFrames, [121](#)
- nvEncLockBitstream, [121](#)
- nvEncLockInputBuffer, [121](#)
- nvEncMapInputResource, [121](#)
- nvEncOpenEncodeSession, [121](#)
- nvEncOpenEncodeSessionEx, [121](#)
- nvEncReconfigureEncoder, [121](#)
- nvEncRegisterAsyncEvent, [121](#)
- nvEncRegisterResource, [121](#)
- nvEncRunMotionEstimationOnly, [121](#)
- nvEncUnlockBitstream, [122](#)
- nvEncUnlockInputBuffer, [122](#)
- nvEncUnmapInputResource, [122](#)
- nvEncUnregisterAsyncEvent, [122](#)

- nvEncUnregisterResource, [122](#)
- reserved, [122](#)
- reserved2, [122](#)
- version, [122](#)
- NVENC_EXTERNAL_ME_HINT, [123](#)
 - dir, [123](#)
 - lastOfMB, [123](#)
 - lastofPart, [123](#)
 - mx, [123](#)
 - my, [123](#)
 - partType, [123](#)
 - refidx, [123](#)
- NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE, [125](#)
 - numCandsPerBlk16x16, [125](#)
 - numCandsPerBlk16x8, [125](#)
 - numCandsPerBlk8x16, [125](#)
 - numCandsPerBlk8x8, [125](#)
 - reserved, [125](#)
 - reserved1, [125](#)
- NVENC_RECT, [126](#)
 - bottom, [126](#)
 - left, [126](#)
 - right, [126](#)
 - top, [126](#)
- NvEncCreateBitstreamBuffer
 - ENCODE_FUNC, [26](#)
- nvEncCreateBitstreamBuffer
 - NV_ENCODE_API_FUNCTION_LIST, [119](#)
- NvEncCreateInputBuffer
 - ENCODE_FUNC, [26](#)
- nvEncCreateInputBuffer
 - NV_ENCODE_API_FUNCTION_LIST, [119](#)
- NvEncCreateMVBuffer
 - ENCODE_FUNC, [26](#)
- nvEncCreateMVBuffer
 - NV_ENCODE_API_FUNCTION_LIST, [119](#)
- NvEncDestroyBitstreamBuffer
 - ENCODE_FUNC, [27](#)
- nvEncDestroyBitstreamBuffer
 - NV_ENCODE_API_FUNCTION_LIST, [119](#)
- NvEncDestroyEncoder
 - ENCODE_FUNC, [27](#)
- nvEncDestroyEncoder
 - NV_ENCODE_API_FUNCTION_LIST, [119](#)
- NvEncDestroyInputBuffer
 - ENCODE_FUNC, [28](#)
- nvEncDestroyInputBuffer
 - NV_ENCODE_API_FUNCTION_LIST, [119](#)
- NvEncDestroyMVBuffer
 - ENCODE_FUNC, [28](#)
- nvEncDestroyMVBuffer
 - NV_ENCODE_API_FUNCTION_LIST, [119](#)
- NvEncEncodePicture
 - ENCODE_FUNC, [29](#)
- nvEncEncodePicture
 - NV_ENCODE_API_FUNCTION_LIST, [119](#)
- NvEncGetEncodeCaps
 - ENCODE_FUNC, [31](#)
- nvEncGetEncodeCaps
 - NV_ENCODE_API_FUNCTION_LIST, [119](#)
- NvEncGetEncodeGUIDCount
 - ENCODE_FUNC, [32](#)
- nvEncGetEncodeGUIDCount
 - NV_ENCODE_API_FUNCTION_LIST, [119](#)
- NvEncGetEncodeGUIDs
 - ENCODE_FUNC, [32](#)
- nvEncGetEncodeGUIDs
 - NV_ENCODE_API_FUNCTION_LIST, [119](#)
- NvEncGetEncodePresetConfig
 - ENCODE_FUNC, [33](#)
- nvEncGetEncodePresetConfig
 - NV_ENCODE_API_FUNCTION_LIST, [120](#)
- NvEncGetEncodePresetCount
 - ENCODE_FUNC, [33](#)
- nvEncGetEncodePresetCount
 - NV_ENCODE_API_FUNCTION_LIST, [120](#)
- NvEncGetEncodePresetGUIDs
 - ENCODE_FUNC, [34](#)
- nvEncGetEncodePresetGUIDs
 - NV_ENCODE_API_FUNCTION_LIST, [120](#)
- NvEncGetEncodeProfileGUIDCount
 - ENCODE_FUNC, [34](#)
- nvEncGetEncodeProfileGUIDCount
 - NV_ENCODE_API_FUNCTION_LIST, [120](#)
- NvEncGetEncodeProfileGUIDs
 - ENCODE_FUNC, [35](#)
- nvEncGetEncodeProfileGUIDs
 - NV_ENCODE_API_FUNCTION_LIST, [120](#)
- NvEncGetEncodeStats
 - ENCODE_FUNC, [35](#)
- nvEncGetEncodeStats
 - NV_ENCODE_API_FUNCTION_LIST, [120](#)
- NvEncGetInputFormatCount
 - ENCODE_FUNC, [36](#)
- nvEncGetInputFormatCount
 - NV_ENCODE_API_FUNCTION_LIST, [120](#)
- NvEncGetInputFormats
 - ENCODE_FUNC, [36](#)
- nvEncGetInputFormats
 - NV_ENCODE_API_FUNCTION_LIST, [120](#)
- NvEncGetSequenceParams
 - ENCODE_FUNC, [37](#)
- nvEncGetSequenceParams
 - NV_ENCODE_API_FUNCTION_LIST, [120](#)
- NvEncInitializeEncoder
 - ENCODE_FUNC, [37](#)
- nvEncInitializeEncoder

- NV_ENCODE_API_FUNCTION_LIST, 120
- NvEncInvalidateRefFrames
 - ENCODE_FUNC, 39
- nvEncInvalidateRefFrames
 - NV_ENCODE_API_FUNCTION_LIST, 121
- NvEncLockBitstream
 - ENCODE_FUNC, 39
- nvEncLockBitstream
 - NV_ENCODE_API_FUNCTION_LIST, 121
- NvEncLockInputBuffer
 - ENCODE_FUNC, 40
- nvEncLockInputBuffer
 - NV_ENCODE_API_FUNCTION_LIST, 121
- NvEncMapInputResource
 - ENCODE_FUNC, 40
- nvEncMapInputResource
 - NV_ENCODE_API_FUNCTION_LIST, 121
- NvEncodeAPI Data structures, 7
- NvEncodeAPI Functions, 23
- NvEncodeAPICreateInstance
 - ENCODE_FUNC, 41
- NvEncodeAPIGetMaxSupportedVersion
 - ENCODE_FUNC, 41
- NvEncOpenEncodeSession
 - ENCODE_FUNC, 42
- nvEncOpenEncodeSession
 - NV_ENCODE_API_FUNCTION_LIST, 121
- NvEncOpenEncodeSessionEx
 - ENCODE_FUNC, 42
- nvEncOpenEncodeSessionEx
 - NV_ENCODE_API_FUNCTION_LIST, 121
- NvEncReconfigureEncoder
 - ENCODE_FUNC, 42
- nvEncReconfigureEncoder
 - NV_ENCODE_API_FUNCTION_LIST, 121
- NvEncRegisterAsyncEvent
 - ENCODE_FUNC, 43
- nvEncRegisterAsyncEvent
 - NV_ENCODE_API_FUNCTION_LIST, 121
- NvEncRegisterResource
 - ENCODE_FUNC, 43
- nvEncRegisterResource
 - NV_ENCODE_API_FUNCTION_LIST, 121
- NvEncRunMotionEstimationOnly
 - ENCODE_FUNC, 44
- nvEncRunMotionEstimationOnly
 - NV_ENCODE_API_FUNCTION_LIST, 121
- NVENCSTATUS
 - ENCODER_STRUCTURE, 21
- NvEncUnlockBitstream
 - ENCODE_FUNC, 44
- nvEncUnlockBitstream
 - NV_ENCODE_API_FUNCTION_LIST, 122
- NvEncUnlockInputBuffer
 - ENCODE_FUNC, 45
- nvEncUnlockInputBuffer
 - NV_ENCODE_API_FUNCTION_LIST, 122
- NvEncUnmapInputResource
 - ENCODE_FUNC, 45
- nvEncUnmapInputResource
 - NV_ENCODE_API_FUNCTION_LIST, 122
- NvEncUnregisterAsyncEvent
 - ENCODE_FUNC, 46
- nvEncUnregisterAsyncEvent
 - NV_ENCODE_API_FUNCTION_LIST, 122
- NvEncUnregisterResource
 - ENCODE_FUNC, 46
- nvEncUnregisterResource
 - NV_ENCODE_API_FUNCTION_LIST, 122
- outputAUD
 - NV_ENC_CONFIG_H264, 58
 - NV_ENC_CONFIG_HEVC, 67
- outputBitStream
 - NV_ENC_STAT, 116
- outputBitstream
 - NV_ENC_LOCK_BITSTREAM, 83
 - NV_ENC_PIC_PARAMS, 96
- outputBufferingPeriodSEI
 - NV_ENC_CONFIG_H264, 58
 - NV_ENC_CONFIG_HEVC, 67
- outputDuration
 - NV_ENC_LOCK_BITSTREAM, 83
- outputFramePackingSEI
 - NV_ENC_CONFIG_H264, 58
- outputPictureTimingSEI
 - NV_ENC_CONFIG_H264, 58
 - NV_ENC_CONFIG_HEVC, 67
- outputRecoveryPointSEI
 - NV_ENC_CONFIG_H264, 59
- outputTimeStamp
 - NV_ENC_LOCK_BITSTREAM, 83
- outSPSPSPayloadSize
 - NV_ENC_SEQUENCE_PARAM_PAYLOAD, 114
- overscanInfo
 - NV_ENC_CONFIG_H264_VUI_PARAMETERS, 64
- overscanInfoPresentFlag
 - NV_ENC_CONFIG_H264_VUI_PARAMETERS, 64
- partitionMode
 - NV_ENC_HEVC_MV_DATA, 77
- partitionType
 - NV_ENC_H264_MV_DATA, 76
- partType
 - NVENC_EXTERNAL_ME_HINT, 123
- payload

- NV_ENC_SEI_PAYLOAD, 113
- payloadSize
 - NV_ENC_SEI_PAYLOAD, 113
- payloadType
 - NV_ENC_SEI_PAYLOAD, 113
- picIdx
 - NV_ENC_STAT, 116
- pictureStruct
 - NV_ENC_LOCK_BITSTREAM, 83
 - NV_ENC_PIC_PARAMS, 96
- pictureType
 - NV_ENC_LOCK_BITSTREAM, 84
 - NV_ENC_PIC_PARAMS, 96
- picType
 - NV_ENC_STAT, 116
- pitch
 - NV_ENC_LOCK_INPUT_BUFFER, 85
 - NV_ENC_REGISTER_RESOURCE, 111
- pixelBitDepthMinus8
 - NV_ENC_CONFIG_HEVC, 67
- ppsId
 - NV_ENC_CONFIG_H264, 59
 - NV_ENC_CONFIG_HEVC, 67
 - NV_ENC_SEQUENCE_PARAM_PAYLOAD, 114
- presetCfg
 - NV_ENC_PRESET_CONFIG, 104
- presetGUID
 - NV_ENC_INITIALIZE_PARAMS, 80
- privData
 - NV_ENC_INITIALIZE_PARAMS, 80
- privDataSize
 - NV_ENC_INITIALIZE_PARAMS, 80
- profileGUID
 - NV_ENC_CONFIG, 54
- pSysMemBuffer
 - NV_ENC_CREATE_INPUT_BUFFER, 72

- qpDeltaMap
 - NV_ENC_PIC_PARAMS, 96
- qpDeltaMapSize
 - NV_ENC_PIC_PARAMS, 96
- qpPrimeYZeroTransformBypassFlag
 - NV_ENC_CONFIG_H264, 59

- rateControlMode
 - NV_ENC_RC_PARAMS, 108
- rcParams
 - NV_ENC_CONFIG, 54
- referenceFrame
 - NV_ENC_MEONLY_PARAMS, 90
- refIdx
 - NVENC_EXTERNAL_ME_HINT, 123
- refPicFlag
 - NV_ENC_PIC_PARAMS_H264, 99

- NV_ENC_PIC_PARAMS_HEVC, 102
- registeredResource
 - NV_ENC_MAP_INPUT_RESOURCE, 87
 - NV_ENC_REGISTER_RESOURCE, 111
- reInitEncodeParams
 - NV_ENC_RECONFIGURE_PARAMS, 110
- repeatSPSPPS
 - NV_ENC_CONFIG_H264, 59
 - NV_ENC_CONFIG_HEVC, 67
- reportSliceOffsets
 - NV_ENC_INITIALIZE_PARAMS, 80
- reserved
 - NV_ENC_CAPS_PARAM, 50
 - NV_ENC_CODEC_CONFIG, 51
 - NV_ENC_CODEC_PIC_PARAMS, 52
 - NV_ENC_CONFIG, 54
 - NV_ENC_CONFIG_H264_MEONLY, 61
 - NV_ENC_CONFIG_HEVC, 67
 - NV_ENC_CONFIG_HEVC_MEONLY, 69
 - NV_ENC_CREATE_BITSTREAM_BUFFER, 70
 - NV_ENC_CREATE_INPUT_BUFFER, 72
 - NV_ENC_EVENT_PARAMS, 75
 - NV_ENC_H264_MV_DATA, 76
 - NV_ENC_INITIALIZE_PARAMS, 80
 - NV_ENC_LOCK_BITSTREAM, 84
 - NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS, 92
 - NV_ENC_PIC_PARAMS_H264, 99
 - NV_ENC_PIC_PARAMS_HEVC, 102
 - NV_ENC_SEQUENCE_PARAM_PAYLOAD, 114
 - NV_ENC_STAT, 116
 - NV_ENCODE_API_FUNCTION_LIST, 122
 - NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE, 125
- reserved1
 - NV_ENC_CONFIG_H264, 59
 - NV_ENC_CONFIG_H264_MEONLY, 62
 - NV_ENC_CONFIG_HEVC, 68
 - NV_ENC_CONFIG_HEVC_MEONLY, 69
 - NV_ENC_CREATE_BITSTREAM_BUFFER, 70
 - NV_ENC_CREATE_INPUT_BUFFER, 72
 - NV_ENC_CREATE_MV_BUFFER, 74
 - NV_ENC_EVENT_PARAMS, 75
 - NV_ENC_LOCK_INPUT_BUFFER, 85
 - NV_ENC_MAP_INPUT_RESOURCE, 87
 - NV_ENC_MEONLY_PARAMS, 90
 - NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS, 92
 - NV_ENC_PIC_PARAMS, 96
 - NV_ENC_PRESET_CONFIG, 104
 - NV_ENC_REGISTER_RESOURCE, 111
 - NV_ENC_STAT, 116
 - NVENC_EXTERNAL_ME_HINT_COUNTS_PER_BLOCKTYPE, 125

- reserved2
 - NV_ENC_CONFIG, 54
 - NV_ENC_CONFIG_H264, 59
 - NV_ENC_CONFIG_H264_MEONLY, 62
 - NV_ENC_CONFIG_HEVC, 68
 - NV_ENC_CREATE_BITSTREAM_BUFFER, 70
 - NV_ENC_CREATE_INPUT_BUFFER, 73
 - NV_ENC_CREATE_MV_BUFFER, 74
 - NV_ENC_EVENT_PARAMS, 75
 - NV_ENC_INITIALIZE_PARAMS, 80
 - NV_ENC_LOCK_BITSTREAM, 84
 - NV_ENC_LOCK_INPUT_BUFFER, 85
 - NV_ENC_MAP_INPUT_RESOURCE, 87
 - NV_ENC_MEONLY_PARAMS, 90
 - NV_ENC_OPEN_ENCODE_SESSION_EX_PARAMS, 92
 - NV_ENC_PIC_PARAMS, 96
 - NV_ENC_PIC_PARAMS_H264, 99
 - NV_ENC_PIC_PARAMS_HEVC, 102
 - NV_ENC_PRESET_CONFIG, 104
 - NV_ENC_REGISTER_RESOURCE, 111
 - NV_ENC_SEQUENCE_PARAM_PAYLOAD, 114
 - NV_ENC_STAT, 117
 - NV_ENCODE_API_FUNCTION_LIST, 122
- reserved3
 - NV_ENC_PIC_PARAMS, 96
 - NV_ENC_PIC_PARAMS_H264, 99
 - NV_ENC_PIC_PARAMS_HEVC, 102
- reserved4
 - NV_ENC_PIC_PARAMS, 97
- reservedBitFields
 - NV_ENC_CONFIG_H264, 59
 - NV_ENC_INITIALIZE_PARAMS, 81
 - NV_ENC_LOCK_BITSTREAM, 84
 - NV_ENC_LOCK_INPUT_BUFFER, 85
 - NV_ENC_PIC_PARAMS, 97
 - NV_ENC_PIC_PARAMS_H264, 99
 - NV_ENC_PIC_PARAMS_HEVC, 102
 - NV_ENC_RC_PARAMS, 108
- resetEncoder
 - NV_ENC_RECONFIGURE_PARAMS, 110
- resourceToRegister
 - NV_ENC_REGISTER_RESOURCE, 111
- resourceType
 - NV_ENC_REGISTER_RESOURCE, 112
- right
 - NVENC_RECT, 126
- seiPayloadArray
 - NV_ENC_PIC_PARAMS_H264, 99
 - NV_ENC_PIC_PARAMS_HEVC, 102
- seiPayloadArrayCnt
 - NV_ENC_PIC_PARAMS_H264, 100
 - NV_ENC_PIC_PARAMS_HEVC, 102
- separateColourPlaneFlag
 - NV_ENC_CONFIG_H264, 59
- size
 - NV_ENC_CREATE_BITSTREAM_BUFFER, 70
- sliceMode
 - NV_ENC_CONFIG_H264, 59
 - NV_ENC_CONFIG_HEVC, 68
 - NV_ENC_PIC_PARAMS_H264, 100
 - NV_ENC_PIC_PARAMS_HEVC, 103
- sliceModeData
 - NV_ENC_CONFIG_H264, 59
 - NV_ENC_CONFIG_HEVC, 68
 - NV_ENC_PIC_PARAMS_H264, 100
 - NV_ENC_PIC_PARAMS_HEVC, 103
- sliceModeDataUpdate
 - NV_ENC_PIC_PARAMS_H264, 100
 - NV_ENC_PIC_PARAMS_HEVC, 103
- sliceOffsets
 - NV_ENC_LOCK_BITSTREAM, 84
 - NV_ENC_STAT, 117
- sliceTypeArrayCnt
 - NV_ENC_PIC_PARAMS_H264, 100
 - NV_ENC_PIC_PARAMS_HEVC, 103
- sliceTypeData
 - NV_ENC_PIC_PARAMS_H264, 100
 - NV_ENC_PIC_PARAMS_HEVC, 103
- spsId
 - NV_ENC_CONFIG_H264, 60
 - NV_ENC_CONFIG_HEVC, 68
 - NV_ENC_SEQUENCE_PARAM_PAYLOAD, 114
- spsppsBuffer
 - NV_ENC_SEQUENCE_PARAM_PAYLOAD, 114
- stereoMode
 - NV_ENC_CONFIG_H264, 60
- strictGOPTarget
 - NV_ENC_RC_PARAMS, 108
- subResourceIndex
 - NV_ENC_MAP_INPUT_RESOURCE, 87
 - NV_ENC_REGISTER_RESOURCE, 112
- targetQuality
 - NV_ENC_RC_PARAMS, 108
- temporalId
 - NV_ENC_PIC_PARAMS_HEVC, 103
- temporalLayerIdxMask
 - NV_ENC_RC_PARAMS, 108
- temporalLayerQP
 - NV_ENC_RC_PARAMS, 108
- tier
 - NV_ENC_CONFIG_HEVC, 68
- top
 - NVENC_RECT, 126
- transferCharacteristics

- NV_ENC_CONFIG_H264_VUI_PARAMETERS,
64
- useConstrainedIntraPred
 - NV_ENC_CONFIG_H264, 60
 - NV_ENC_CONFIG_HEVC, 68
- vbvBufferSize
 - NV_ENC_RC_PARAMS, 108
- vbvInitialDelay
 - NV_ENC_RC_PARAMS, 109
- version
 - NV_ENC_CAPS_PARAM, 50
 - NV_ENC_CONFIG, 54
 - NV_ENC_CREATE_BITSTREAM_BUFFER, 71
 - NV_ENC_CREATE_INPUT_BUFFER, 73
 - NV_ENC_CREATE_MV_BUFFER, 74
 - NV_ENC_EVENT_PARAMS, 75
 - NV_ENC_INITIALIZE_PARAMS, 81
 - NV_ENC_LOCK_BITSTREAM, 84
 - NV_ENC_LOCK_INPUT_BUFFER, 86
 - NV_ENC_MAP_INPUT_RESOURCE, 88
 - NV_ENC_MEONLY_PARAMS, 90
 - NV_ENC_OPEN_ENCODE_SESSION_EX_-
PARAMS, 92
 - NV_ENC_PIC_PARAMS, 97
 - NV_ENC_PRESET_CONFIG, 104
 - NV_ENC_RECONFIGURE_PARAMS, 110
 - NV_ENC_REGISTER_RESOURCE, 112
 - NV_ENC_SEQUENCE_PARAM_PAYLOAD, 115
 - NV_ENC_STAT, 117
 - NV_ENCODE_API_FUNCTION_LIST, 122
- videoFormat
 - NV_ENC_CONFIG_H264_VUI_PARAMETERS,
64
- videoFullRangeFlag
 - NV_ENC_CONFIG_H264_VUI_PARAMETERS,
64
- videoSignalTypePresentFlag
 - NV_ENC_CONFIG_H264_VUI_PARAMETERS,
64
- viewID
 - NV_ENC_MEONLY_PARAMS, 90
- vpsId
 - NV_ENC_CONFIG_HEVC, 68
- width
 - NV_ENC_CREATE_INPUT_BUFFER, 73
 - NV_ENC_REGISTER_RESOURCE, 112
- zeroReorderDelay
 - NV_ENC_RC_PARAMS, 109

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

© 2011-2016 NVIDIA Corporation. All rights reserved.