

SynView  
Reference Guide

Generated by Doxygen 1.8.7

Sun Jan 21 2018 22:23:41



# Contents

- 1 SynView Reference Guide** **1**
  
- 2 Module Index** **3**
  - 2.1 Modules 3
  
- 3 Hierarchical Index** **5**
  - 3.1 Class Hierarchy 5
  
- 4 Class Index** **7**
  - 4.1 Class List 7
  
- 5 Module Documentation** **9**
  - 5.1 SynView Plain C API functions 9
    - 5.1.1 Detailed Description 9
  - 5.2 SynView General purpose functions 10
    - 5.2.1 Detailed Description 10
    - 5.2.2 Function Documentation 10
      - 5.2.2.1 LvCloseLibrary 10
      - 5.2.2.2 LvGetErrorMessage 10
      - 5.2.2.3 LvGetLastErrorMessage 10
      - 5.2.2.4 LvGetLibInfo 12
      - 5.2.2.5 LvGetLibInfoStr 12
      - 5.2.2.6 LvGetLibInfoStrSize 12
      - 5.2.2.7 LvGetVersion 12
      - 5.2.2.8 LvLog 13
      - 5.2.2.9 LvOpenLibrary 13
  - 5.3 SynView System module functions 14
    - 5.3.1 Detailed Description 14
    - 5.3.2 Function Documentation 14
      - 5.3.2.1 LvGetNumberOfSystems 14
      - 5.3.2.2 LvGetSystemId 14
      - 5.3.2.3 LvGetSystemIdSize 14
      - 5.3.2.4 LvSystemClose 15

5.3.2.5	LvSystemFindInterface	15
5.3.2.6	LvSystemGetInterfaceId	15
5.3.2.7	LvSystemGetInterfaceIdSize	16
5.3.2.8	LvSystemGetNumberOfInterfaces	16
5.3.2.9	LvSystemOpen	16
5.3.2.10	LvSystemUpdateInterfaceList	17
5.3.2.11	LvUpdateSystemList	17
5.4	SynView Interface module functions	18
5.4.1	Detailed Description	18
5.4.2	Function Documentation	18
5.4.2.1	LvInterfaceClose	18
5.4.2.2	LvInterfaceFindDevice	18
5.4.2.3	LvInterfaceGetDeviceId	19
5.4.2.4	LvInterfaceGetDeviceIdSize	19
5.4.2.5	LvInterfaceGetNumberOfDevices	19
5.4.2.6	LvInterfaceOpen	20
5.4.2.7	LvInterfaceUpdateDeviceList	20
5.5	SynView Device module functions	21
5.5.1	Detailed Description	21
5.5.2	Function Documentation	21
5.5.2.1	LoadBatch	21
5.5.2.2	LvDeviceAcquisitionAbort	21
5.5.2.3	LvDeviceAcquisitionArm	22
5.5.2.4	LvDeviceAcquisitionStart	22
5.5.2.5	LvDeviceAcquisitionStop	22
5.5.2.6	LvDeviceClose	22
5.5.2.7	LvDeviceGetNumberOfStreams	23
5.5.2.8	LvDeviceGetStreamId	23
5.5.2.9	LvDeviceGetStreamIdSize	23
5.5.2.10	LvDeviceLoadBatch	24
5.5.2.11	LvDeviceLoadSettings	24
5.5.2.12	LvDeviceOpen	24
5.5.2.13	LvDeviceReOpen	25
5.5.2.14	LvDeviceSaveSettings	25
5.5.2.15	LvDeviceUniGetLut	25
5.5.2.16	LvDeviceUniSetLut	26
5.6	SynView Stream module functions	27
5.6.1	Detailed Description	27
5.6.2	Function Documentation	27
5.6.2.1	LvStreamClose	27

---

5.6.2.2	LvStreamFlushQueue	27
5.6.2.3	LvStreamGetBufferAt	27
5.6.2.4	LvStreamOpen	28
5.6.2.5	LvStreamStart	28
5.6.2.6	LvStreamStop	28
5.7	SynView Buffer module functions	29
5.7.1	Detailed Description	29
5.7.2	Function Documentation	29
5.7.2.1	LvBufferAttachProcessBuffer	29
5.7.2.2	LvBufferClose	29
5.7.2.3	LvBufferGetImgInfo	30
5.7.2.4	LvBufferGetLastPaintRect	30
5.7.2.5	LvBufferOpen	30
5.7.2.6	LvBufferParseChunkData	31
5.7.2.7	LvBufferQueue	32
5.7.2.8	LvBufferSaveImageToBmpFile	32
5.7.2.9	LvBufferSaveImageToJpgFile	32
5.7.2.10	LvBufferSaveImageToTifFile	33
5.7.2.11	LvBufferUniCalculateWhiteBalance	34
5.8	SynView Event module functions	35
5.8.1	Detailed Description	35
5.8.2	Function Documentation	35
5.8.2.1	LvEventCallbackMustExit	35
5.8.2.2	LvEventClose	35
5.8.2.3	LvEventFlush	35
5.8.2.4	LvEventGetDataInfo	36
5.8.2.5	LvEventKill	36
5.8.2.6	LvEventOpen	36
5.8.2.7	LvEventPutData	37
5.8.2.8	LvEventSetCallback	38
5.8.2.9	LvEventSetCallbackNewBuffer	38
5.8.2.10	LvEventStartThread	38
5.8.2.11	LvEventStopThread	39
5.8.2.12	LvEventWaitAndGetData	40
5.8.2.13	LvEventWaitAndGetNewBuffer	40
5.9	SynView Renderer module functions	41
5.9.1	Detailed Description	41
5.9.2	Function Documentation	41
5.9.2.1	LvRendererCanDisplayImage	41
5.9.2.2	LvRendererClose	41

5.9.2.3	LvRendererDisplayImage	41
5.9.2.4	LvRendererOpen	42
5.9.2.5	LvRendererRepaint	42
5.9.2.6	LvRendererSetWindow	42
5.10	SynView Feature control functions	44
5.10.1	Detailed Description	45
5.10.2	Function Documentation	45
5.10.2.1	LvCmdExecute	45
5.10.2.2	LvCmdIsDone	46
5.10.2.3	LvGetAccess	47
5.10.2.4	LvGetBool	47
5.10.2.5	LvGetBuffer	47
5.10.2.6	LvGetBufferSize	48
5.10.2.7	LvGetEnum	48
5.10.2.8	LvGetEnumStr	48
5.10.2.9	LvGetEnumStrByVal	49
5.10.2.10	LvGetEnumValByStr	49
5.10.2.11	LvGetFeatureAt	49
5.10.2.12	LvGetFeatureByName	50
5.10.2.13	LvGetFloat	50
5.10.2.14	LvGetFloatRange	50
5.10.2.15	LvGetInfo	51
5.10.2.16	LvGetInfoStr	51
5.10.2.17	LvGetInfoStrSize	51
5.10.2.18	LvGetInt	52
5.10.2.19	LvGetInt32	52
5.10.2.20	LvGetInt32Range	52
5.10.2.21	LvGetInt64	53
5.10.2.22	LvGetInt64Range	53
5.10.2.23	LvGetIntRange	54
5.10.2.24	LvGetNumFeatures	55
5.10.2.25	LvGetPtr	55
5.10.2.26	LvGetString	55
5.10.2.27	LvGetStringSize	56
5.10.2.28	LvGetType	56
5.10.2.29	LvGetVisibility	56
5.10.2.30	LvIsAvailable	57
5.10.2.31	LvIsAvailableByName	57
5.10.2.32	LvIsAvailableEnumEntry	57
5.10.2.33	LvIsImplemented	58

5.10.2.34	LvIsImplementedByName	58
5.10.2.35	LvIsImplementedEnumEntry	58
5.10.2.36	LvIsReadable	58
5.10.2.37	LvIsWritable	59
5.10.2.38	LvPoll	59
5.10.2.39	LvRegisterFeatureCallback	59
5.10.2.40	LvSetBool	60
5.10.2.41	LvSetBuffer	60
5.10.2.42	LvSetEnum	60
5.10.2.43	LvSetEnumStr	61
5.10.2.44	LvSetFloat	61
5.10.2.45	LvSetInt	61
5.10.2.46	LvSetInt32	62
5.10.2.47	LvSetInt64	63
5.10.2.48	LvSetPtr	63
5.10.2.49	LvSetString	63
5.10.2.50	LvStartPollingThread	64
5.10.2.51	LvStopPollingThread	64
5.11	SynView Firmware update functions	65
5.11.1	Detailed Description	65
5.11.2	Function Documentation	65
5.11.2.1	LvFwGetFilePattern	65
5.11.2.2	LvFwGetLoadStatus	65
5.11.2.3	LvFwLoad	65
5.12	SynView C++ API functions	67
5.12.1	Detailed Description	67
5.13	SynView LvLibrary methods	68
5.13.1	Detailed Description	68
5.13.2	Function Documentation	68
5.13.2.1	CloseLibrary	68
5.13.2.2	GetErrorMessage	68
5.13.2.3	GetErrorMessage	69
5.13.2.4	GetLastErrorMessage	69
5.13.2.5	GetLastErrorMessage	69
5.13.2.6	GetLibInfo	70
5.13.2.7	GetLibInfoStr	70
5.13.2.8	GetLibInfoStr	70
5.13.2.9	GetLibInfoStrSize	70
5.13.2.10	GetNumberOfSystems	71
5.13.2.11	GetSystemId	71

5.13.2.12	GetSystemId	71
5.13.2.13	GetSystemIdSize	71
5.13.2.14	GetVersion	72
5.13.2.15	Log	72
5.13.2.16	OpenLibrary	72
5.13.2.17	SetThrowErrorEnable	72
5.13.2.18	UpdateSystemList	73
5.14	SynView LvSystem methods	74
5.14.1	Detailed Description	74
5.14.2	Function Documentation	74
5.14.2.1	Close	74
5.14.2.2	CloseEvent	74
5.14.2.3	CloseInterface	75
5.14.2.4	FindInterface	75
5.14.2.5	FindInterface	75
5.14.2.6	GetHandle	76
5.14.2.7	GetInterfaceId	76
5.14.2.8	GetInterfaceId	76
5.14.2.9	GetInterfaceIdSize	76
5.14.2.10	GetNumberOfInterfaces	77
5.14.2.11	Open	77
5.14.2.12	OpenEvent	77
5.14.2.13	OpenInterface	78
5.14.2.14	UpdateInterfaceList	78
5.15	SynView LvInterface methods	79
5.15.1	Detailed Description	79
5.15.2	Function Documentation	79
5.15.2.1	Close	79
5.15.2.2	CloseDevice	79
5.15.2.3	FindDevice	80
5.15.2.4	FindDevice	80
5.15.2.5	GetDeviceId	80
5.15.2.6	GetDeviceId	81
5.15.2.7	GetDeviceIdSize	81
5.15.2.8	GetHandle	81
5.15.2.9	GetNumberOfDevices	81
5.15.2.10	Open	82
5.15.2.11	OpenDevice	82
5.15.2.12	UpdateDeviceList	82
5.16	SynView LvDevice methods	84



---

5.16.1	Detailed Description	84
5.16.2	Function Documentation	84
5.16.2.1	AcquisitionAbort	84
5.16.2.2	AcquisitionArm	84
5.16.2.3	AcquisitionStart	85
5.16.2.4	AcquisitionStop	85
5.16.2.5	Close	85
5.16.2.6	CloseEvent	85
5.16.2.7	CloseStream	86
5.16.2.8	GetHandle	86
5.16.2.9	GetNumberOfStreams	86
5.16.2.10	GetStreamId	86
5.16.2.11	GetStreamId	87
5.16.2.12	GetStreamIdSize	87
5.16.2.13	LoadSettings	87
5.16.2.14	Open	88
5.16.2.15	OpenEvent	88
5.16.2.16	OpenStream	88
5.16.2.17	SaveSettings	89
5.16.2.18	UniGetLut	89
5.16.2.19	UniSetLut	89
5.17	SynView LvDevice firmware update methods	91
5.17.1	Detailed Description	91
5.17.2	Function Documentation	91
5.17.2.1	FwGetFilePattern	91
5.17.2.2	FwGetLoadStatus	91
5.17.2.3	FwLoad	91
5.18	SynView LvStream methods	93
5.18.1	Detailed Description	93
5.18.2	Function Documentation	93
5.18.2.1	Close	93
5.18.2.2	CloseBuffer	93
5.18.2.3	CloseEvent	94
5.18.2.4	CloseRenderer	94
5.18.2.5	FlushQueue	94
5.18.2.6	GetBufferAt	94
5.18.2.7	GetHandle	95
5.18.2.8	Open	95
5.18.2.9	OpenBuffer	95
5.18.2.10	OpenEvent	96

5.18.2.11	OpenRenderer	96
5.18.2.12	Start	96
5.18.2.13	Stop	97
5.19	SynView LvBuffer methods	98
5.19.1	Detailed Description	98
5.19.2	Function Documentation	98
5.19.2.1	AttachProcessBuffer	98
5.19.2.2	Close	98
5.19.2.3	GetHandle	99
5.19.2.4	GetImgInfo	99
5.19.2.5	GetLastPaintRect	99
5.19.2.6	GetUserPtr	99
5.19.2.7	Open	99
5.19.2.8	ParseChunkData	100
5.19.2.9	Queue	100
5.19.2.10	SavelImageToBmpFile	100
5.19.2.11	SavelImageToJpgFile	101
5.19.2.12	SavelImageToTifFile	101
5.19.2.13	UniCalculateWhiteBalance	101
5.20	SynView LvEvent methods	102
5.20.1	Detailed Description	102
5.20.2	Function Documentation	102
5.20.2.1	CallbackMustExit	102
5.20.2.2	Close	102
5.20.2.3	Flush	103
5.20.2.4	GetDataInfo	103
5.20.2.5	GetHandle	103
5.20.2.6	Kill	103
5.20.2.7	Open	103
5.20.2.8	Open	104
5.20.2.9	Open	104
5.20.2.10	PutData	104
5.20.2.11	SetCallback	105
5.20.2.12	SetCallbackNewBuffer	105
5.20.2.13	StartThread	105
5.20.2.14	StopThread	105
5.20.2.15	WaitAndGetData	106
5.20.2.16	WaitAndGetNewBuffer	106
5.21	SynView LvRenderer methods	107
5.21.1	Detailed Description	107

---

5.21.2	Function Documentation	107
5.21.2.1	Close	107
5.21.2.2	DisplayImage	107
5.21.2.3	GetHandle	107
5.21.2.4	Open	108
5.21.2.5	Repaint	109
5.21.2.6	SetWindow	109
5.22	SynView LvModule methods	110
5.22.1	Detailed Description	111
5.22.2	Function Documentation	111
5.22.2.1	CmdExecute	111
5.22.2.2	CmdIsDone	111
5.22.2.3	GetAccess	112
5.22.2.4	GetBool	113
5.22.2.5	GetBuffer	113
5.22.2.6	GetBufferSize	113
5.22.2.7	GetEnum	113
5.22.2.8	GetEnumStr	114
5.22.2.9	GetEnumStr	114
5.22.2.10	GetEnumStrByVal	114
5.22.2.11	GetEnumStrByVal	115
5.22.2.12	GetEnumValByStr	115
5.22.2.13	GetFeatureAt	115
5.22.2.14	GetFeatureByName	116
5.22.2.15	GetFloat	116
5.22.2.16	GetFloatRange	116
5.22.2.17	GetInfo	117
5.22.2.18	GetInfoStr	117
5.22.2.19	GetInfoStr	117
5.22.2.20	GetInfoStrSize	118
5.22.2.21	GetInt	118
5.22.2.22	GetInt32	118
5.22.2.23	GetInt32Range	119
5.22.2.24	GetInt64	119
5.22.2.25	GetInt64Range	119
5.22.2.26	GetIntRange	120
5.22.2.27	GetNumFeatures	120
5.22.2.28	GetPtr	120
5.22.2.29	GetString	121
5.22.2.30	GetString	121

5.22.2.31	GetStringSize	121
5.22.2.32	GetType	121
5.22.2.33	GetVisibility	122
5.22.2.34	IsAvailable	122
5.22.2.35	IsAvailableByName	122
5.22.2.36	IsAvailableEnumEntry	123
5.22.2.37	IsImplemented	124
5.22.2.38	IsImplementedByName	124
5.22.2.39	IsImplementedEnumEntry	124
5.22.2.40	IsReadable	124
5.22.2.41	IsWritable	125
5.22.2.42	Poll	125
5.22.2.43	RegisterFeatureCallback	125
5.22.2.44	SetBool	125
5.22.2.45	SetBuffer	126
5.22.2.46	SetEnum	126
5.22.2.47	SetEnumStr	126
5.22.2.48	SetFloat	126
5.22.2.49	SetInt	127
5.22.2.50	SetInt32	127
5.22.2.51	SetInt64	127
5.22.2.52	SetPtr	128
5.22.2.53	SetString	128
5.22.2.54	StartPollingThread	128
5.22.2.55	StopPollingThread	129
5.22.3	Variable Documentation	129
5.22.3.1	m_hModule	129
5.23	SynView	130
5.23.1	Detailed Description	130
5.24	SynView defines and typedefs	131
5.24.1	Detailed Description	131
5.24.2	Macro Definition Documentation	131
5.24.2.1	LV_DLENTY	131
5.24.2.2	LVIP_DLENTY	131
5.24.3	Typedef Documentation	131
5.24.3.1	LvEventCallbackFunct	131
5.24.3.2	LvEventCallbackNewBufferFunct	132
5.24.3.3	LvFeatureCallbackFunct	132
5.24.3.4	LvHModule	132
5.25	SynView enumerations	133

5.25.1	Detailed Description	134
5.25.2	Enumeration Type Documentation	134
5.25.2.1	LvEventDataInfo	134
5.25.2.2	LvEventType	134
5.25.2.3	LvFindBy	135
5.25.2.4	LvFtrAccess	135
5.25.2.5	LvFtrGroup	135
5.25.2.6	LvFtrGui	136
5.25.2.7	LvFtrInfo	137
5.25.2.8	LvFtrType	139
5.25.2.9	LvFtrVisibility	139
5.25.2.10	LvInfoDataType	140
5.25.2.11	LvLibInfo	140
5.25.2.12	LvQueueOperation	141
5.25.2.13	LvRenderFlags	141
5.26	SynView Image Processing Library	142
5.26.1	Detailed Description	142
5.27	SynView Image Processing Library defines, typedefs and enums	143
5.27.1	Detailed Description	143
5.27.2	Macro Definition Documentation	143
5.27.2.1	LVIP_LUT_BAYER	143
5.27.2.2	LVIP_LUT_BAYER_16	143
5.27.3	Enumeration Type Documentation	144
5.27.3.1	LvipColor	144
5.27.3.2	LvipImgAttr	144
5.27.3.3	LvipLutType	144
5.27.3.4	LvipOption	145
5.27.3.5	LvipTextAttr	145
5.28	Definitions for Enumeration Entry Info	147
5.28.1	Detailed Description	147
5.28.2	Macro Definition Documentation	147
5.28.2.1	LV_ENUMENTRY_CURRENT	147
5.28.3	Typedef Documentation	147
5.28.3.1	LvEnum	147
5.28.3.2	LvFeature	147
5.28.3.3	LvHBuffer	147
5.28.3.4	LvHDevice	147
5.28.3.5	LvHEvent	147
5.28.3.6	LvHInterface	148
5.28.3.7	LvHOverlay	148

5.28.3.8	LvHRenderer	148
5.28.3.9	LvHStream	148
5.28.3.10	LvHSystem	148
5.29	Features	149
5.29.1	Detailed Description	153
5.29.2	Enumeration Type Documentation	153
5.29.2.1	LvBufferFtr	153
5.29.2.2	LvDeviceFtr	155
5.29.2.3	LvEventFtr	175
5.29.2.4	LvInterfaceFtr	176
5.29.2.5	LvRendererFtr	177
5.29.2.6	LvStreamFtr	178
5.29.2.7	LvSystemFtr	179
5.30	Enumeration entries	181
5.30.1	Detailed Description	186
5.30.2	Enumeration Type Documentation	186
5.30.2.1	LvAcquisitionFrameRateControlMode	186
5.30.2.2	LvAcquisitionMode	186
5.30.2.3	LvAOIMode	186
5.30.2.4	LvBalanceRatioSelector	187
5.30.2.5	LvBalanceWhiteAuto	187
5.30.2.6	LvBayerDecoderAlgorithm	187
5.30.2.7	LvBlackLevelAuto	187
5.30.2.8	LvBlackLevelSelector	188
5.30.2.9	LvBootSwitch	188
5.30.2.10	LvChunkGainSelector	188
5.30.2.11	LvChunkLvExternalADCSelector	188
5.30.2.12	LvChunkSelector	188
5.30.2.13	LvColorTransformationSelector	189
5.30.2.14	LvColorTransformationValueSelector	189
5.30.2.15	LvCounterEventSource	190
5.30.2.16	LvCounterMode	190
5.30.2.17	LvCounterSelector	191
5.30.2.18	LvDeviceAccess	191
5.30.2.19	LvDeviceAccessStatus	191
5.30.2.20	LvDeviceClockSelector	191
5.30.2.21	LvDeviceEndiannessMechanism	192
5.30.2.22	LvDeviceScanType	192
5.30.2.23	LvDeviceTemperatureSelector	192
5.30.2.24	LvDeviceType	192

5.30.2.25 LvEventNotification	192
5.30.2.26 LvEventSelector	193
5.30.2.27 LvExposureAuto	193
5.30.2.28 LvExposureMode	193
5.30.2.29 LvExternalADCSelector	193
5.30.2.30 LvExternalDeviceControlMode	194
5.30.2.31 LvGainAuto	194
5.30.2.32 LvGainSelector	194
5.30.2.33 LvGevCCP	194
5.30.2.34 LvGevDeviceClass	194
5.30.2.35 LvGevDeviceModeCharacterSet	194
5.30.2.36 LvGevDeviceStreamCaptureMode	195
5.30.2.37 LvGevIPConfigurationStatus	195
5.30.2.38 LvGevSCPDirection	195
5.30.2.39 LvGevSupportedOptionSelector	195
5.30.2.40 LvImageStampSelector	196
5.30.2.41 LvInterfaceType	197
5.30.2.42 LvLensControlCalibrationStatus	197
5.30.2.43 LvLensControlTargetApproach	197
5.30.2.44 LvLineDebounceMode	197
5.30.2.45 LvLineFormat	197
5.30.2.46 LvLineMode	198
5.30.2.47 LvLineSelector	198
5.30.2.48 LvLineSource	199
5.30.2.49 LvLUTMode	199
5.30.2.50 LvLUTSelector	199
5.30.2.51 LvPixelFormat	200
5.30.2.52 LvPowerSwitchBoundADC	202
5.30.2.53 LvPowerSwitchCurrentAction	202
5.30.2.54 LvPowerSwitchDrive	203
5.30.2.55 LvPowerSwitchDriveAll	203
5.30.2.56 LvPowerSwitchSelector	203
5.30.2.57 LvRegionSelector	203
5.30.2.58 LvRenderType	203
5.30.2.59 LvSerialPortBaudRate	204
5.30.2.60 LvSerialPortCommandStatus	204
5.30.2.61 LvSerialPortDataBits	204
5.30.2.62 LvSerialPortParity	204
5.30.2.63 LvSerialPortStopBits	205
5.30.2.64 LvSpecialPurposeTriggerActivation	205

5.30.2.65	LvSpecialPurposeTriggerSelector	205
5.30.2.66	LvSpecialPurposeTriggerSource	205
5.30.2.67	LvStreamAcquisitionModeSelector	206
5.30.2.68	LvStreamType	206
5.30.2.69	LvStrobeDropMode	206
5.30.2.70	LvStrobeDurationMode	207
5.30.2.71	LvStrobeEnable	207
5.30.2.72	LvTimerSelector	207
5.30.2.73	LvTimerTriggerSource	207
5.30.2.74	LvTLType	208
5.30.2.75	LvTriggerActivation	208
5.30.2.76	LvTriggerCaching	209
5.30.2.77	LvTriggerMode	209
5.30.2.78	LvTriggerSelector	209
5.30.2.79	LvTriggerSource	209
5.30.2.80	LvUniBalanceRatioSelector	210
5.30.2.81	LvUniBalanceWhiteAuto	210
5.30.2.82	LvUniColorTransformationMode	211
5.30.2.83	LvUniColorTransformationSelector	211
5.30.2.84	LvUniColorTransformationValueSelector	211
5.30.2.85	LvUniLUTMode	211
5.30.2.86	LvUniLUTSelector	212
5.30.2.87	LvUniProcessExecution	212
5.30.2.88	LvUniProcessMode	212
5.30.2.89	LvUserOutputSelector	212
5.30.2.90	LvUserSetDefault	213
5.30.2.91	LvUserSetDefaultSelector	213
5.30.2.92	LvUserSetSelector	213
5.31	LvStreamStart() flags definitions	214
5.31.1	Detailed Description	214
5.31.2	Macro Definition Documentation	214
5.31.2.1	LvStreamStartFlags_Default	214
5.32	LvStreamStop() flags definitions	215
5.32.1	Detailed Description	215
5.32.2	Macro Definition Documentation	215
5.32.2.1	LvStreamStopFlags_Default	215
5.32.2.2	LvStreamStopFlags_Kill	215
5.33	LvDeviceUniSetLut() and LvDeviceUniGetLut() flags definitions	216
5.33.1	Detailed Description	216
5.33.2	Macro Definition Documentation	216



5.33.2.1	LvUniLutFlags_HwLut	216
5.34	LvSaveFlag definitions	217
5.34.1	Detailed Description	217
5.34.2	Macro Definition Documentation	217
5.34.2.1	LvSaveFlag_All	217
5.34.2.2	LvSaveFlag_GenTIFtr	217
5.34.2.3	LvSaveFlag_IgnoreModel	217
5.34.2.4	LvSaveFlag_IgnoreVersion	217
5.34.2.5	LvSaveFlag_LocalFtr	217
5.34.2.6	LvSaveFlag_RemoteFtr	217
5.35	LvPixelFormat definitions	218
5.35.1	Detailed Description	218
5.35.2	Macro Definition Documentation	218
5.35.2.1	LV_PIX_COLOR	218
5.35.2.2	LV_PIX_COLOR_MASK	219
5.35.2.3	LV_PIX_CUSTOM	219
5.35.2.4	LV_PIX_EFFECTIVE_PIXEL_SIZE_MASK	219
5.35.2.5	LV_PIX_EFFECTIVE_PIXEL_SIZE_SHIFT	219
5.35.2.6	LV_PIX_MONO	219
5.35.2.7	LV_PIX_OCCUPY12BIT	219
5.35.2.8	LV_PIX_OCCUPY16BIT	219
5.35.2.9	LV_PIX_OCCUPY24BIT	219
5.35.2.10	LV_PIX_OCCUPY32BIT	219
5.35.2.11	LV_PIX_OCCUPY36BIT	219
5.35.2.12	LV_PIX_OCCUPY48BIT	219
5.35.2.13	LV_PIX_OCCUPY8BIT	219
5.35.2.14	LvPixelFormat_BGR10Packed	220
5.35.2.15	LvPixelFormat_BGR12Packed	220
5.35.2.16	LvPixelFormat_BGR555p	220
5.35.2.17	LvPixelFormat_BGR565p	220
5.35.2.18	LvPixelFormat_BGR565Packed	220
5.35.2.19	LvPixelFormat_BGR8Packed	220
5.35.2.20	LvPixelFormat_BGRa8	220
5.35.2.21	LvPixelFormat_BGRA8Packed	220
5.35.2.22	LvPixelFormat_Mono8s	220
5.35.2.23	LvPixelFormat_Mono8Signed	220
5.35.2.24	LvPixelFormat_RGB10p32	220
5.35.2.25	LvPixelFormat_RGB10Packed	220
5.35.2.26	LvPixelFormat_RGB10Planar	221
5.35.2.27	LvPixelFormat_RGB10V2Packed	221

5.35.2.28	LvPixelFormat_RGB12Packed	221
5.35.2.29	LvPixelFormat_RGB12Planar	221
5.35.2.30	LvPixelFormat_RGB16Packed	221
5.35.2.31	LvPixelFormat_RGB16Planar	221
5.35.2.32	LvPixelFormat_RGB565p	221
5.35.2.33	LvPixelFormat_RGB565Packed	221
5.35.2.34	LvPixelFormat_RGB8Packed	221
5.35.2.35	LvPixelFormat_RGB8Planar	221
5.35.2.36	LvPixelFormat_RGBa8	221
5.35.2.37	LvPixelFormat_RGBA8Packed	221
5.35.2.38	LvPixelFormat_YUV411_8_UYYVYY	222
5.35.2.39	LvPixelFormat_YUV411Packed	222
5.35.2.40	LvPixelFormat_YUV422Packed	222
5.35.2.41	LvPixelFormat_YUV422UYVPacked	222
5.35.2.42	LvPixelFormat_YUV444Packed	222
5.35.2.43	LvPixelFormat_YUV8_UYV	222
5.36	SynView Image Processing Library functions	223
5.36.1	Detailed Description	223
5.37	Common functions	224
5.37.1	Detailed Description	224
5.37.2	Function Documentation	224
5.37.2.1	LvipGetStatusMsg	224
5.38	Image initialization functions	225
5.38.1	Detailed Description	225
5.38.2	Function Documentation	225
5.38.2.1	LvipAllocatелImageData	225
5.38.2.2	LvipDeallocatелImageData	225
5.38.2.3	LvipFillWithColor	225
5.38.2.4	LvipGetImageDataSize	226
5.38.2.5	LvipInitImgInfo	227
5.39	Region of Interest (ROI) functions	228
5.39.1	Detailed Description	228
5.39.2	Function Documentation	228
5.39.2.1	LvipCopyArea	228
5.40	Lookup Table (LUT) functions	229
5.40.1	Detailed Description	229
5.40.2	Function Documentation	229
5.40.2.1	LvipAddBrightnessAndContrastToLut	229
5.40.2.2	LvipAddGammaToLut	230
5.40.2.3	LvipAddOffsetAndGainToLut	230

5.40.2.4	LvipAddWbToLut	231
5.40.2.5	LvipAllocateLut	231
5.40.2.6	LvipApplyLut	231
5.40.2.7	LvipCalcWbFactors	232
5.40.2.8	LvipFreeLut	232
5.40.2.9	LvipGet10BitLut	233
5.40.2.10	LvipGet10BitLutValue	234
5.40.2.11	LvipGet12BitLut	234
5.40.2.12	LvipGet12BitLutValue	234
5.40.2.13	LvipGet8BitLut	235
5.40.2.14	LvipGet8BitLutValue	235
5.40.2.15	LvipResetLut	235
5.40.2.16	LvipSet10BitLut	236
5.40.2.17	LvipSet10BitLutValue	236
5.40.2.18	LvipSet12BitLut	236
5.40.2.19	LvipSet12BitLutValue	236
5.40.2.20	LvipSet8BitLut	237
5.40.2.21	LvipSet8BitLutValue	237
5.41	Bayer decoding/encoding functions	238
5.41.1	Detailed Description	238
5.41.2	Function Documentation	238
5.41.2.1	LvipBdBilinearColorCorrection	238
5.41.2.2	LvipBdBilinearInterpolation	238
5.41.2.3	LvipBdEncodeToBayer	239
5.41.2.4	LvipBdGreenToGreyscale	239
5.41.2.5	LvipBdNearestNeighbour	240
5.41.2.6	LvipBdPixelGrouping	240
5.41.2.7	LvipBdShowMosaic	241
5.41.2.8	LvipBdVariableGradients	241
5.42	Rotation and line manipulation functions	242
5.42.1	Detailed Description	242
5.42.2	Function Documentation	242
5.42.2.1	LvipDeinterlace	242
5.42.2.2	LvipMirror	242
5.42.2.3	LvipReverseLines	243
5.42.2.4	LvipReverseLinesFast	243
5.42.2.5	LvipRotate90	244
5.42.2.6	LvipRotate90AndMirror	244
5.43	Pixel format conversion functions	246
5.43.1	Detailed Description	246

5.43.2	Function Documentation	246
5.43.2.1	LvipCanConvertToPixelFormat	246
5.43.2.2	LvipConvertToPixelFormat	246
5.44	Saving/loading functions	247
5.44.1	Detailed Description	247
5.44.2	Function Documentation	247
5.44.2.1	LvipLoadFromBmp	247
5.44.2.2	LvipLoadFromJpeg	247
5.44.2.3	LvipLoadFromTiff	248
5.44.2.4	LvipSaveToBmp	248
5.44.2.5	LvipSaveToJpeg	248
5.44.2.6	LvipSaveToTiff	249
5.45	Overlay functions	250
5.46	RGB color correction and convolution functions	251
5.46.1	Detailed Description	251
5.46.2	Function Documentation	251
5.46.2.1	LvipApply3x3Convolution	251
5.46.2.2	LvipApplyRgbColorCorrection	251
5.46.2.3	LvipSet3x3MatrixSharpening	252
5.46.2.4	LvipSetSaturationMatrix	252
5.47	Shading correction functions	253
5.47.1	Detailed Description	253
5.47.2	Function Documentation	253
5.47.2.1	LvipApplyShadingCorrection	253
5.48	SynView INI file API	254
5.48.1	Detailed Description	254
5.48.2	Function Documentation	254
5.48.2.1	LvIniClose	254
5.48.2.2	LvIniDeleteItem	255
5.48.2.3	LvIniDeleteSection	255
5.48.2.4	LvIniGetBool	255
5.48.2.5	LvIniGetFloat	255
5.48.2.6	LvIniGetInteger	256
5.48.2.7	LvIniGetSectionRawLine	256
5.48.2.8	LvIniGetSectionRawLineSize	256
5.48.2.9	LvIniGetString	257
5.48.2.10	LvIniGetStringSize	258
5.48.2.11	LvIniItemExists	258
5.48.2.12	LvIniLoad	258
5.48.2.13	LvIniModified	259

5.48.2.14	LvIniOpen	259
5.48.2.15	LvIniSave	259
5.48.2.16	LvIniSectionExists	259
5.48.2.17	LvIniSetBool	259
5.48.2.18	LvIniSetFloat	260
5.48.2.19	LvIniSetInteger	260
5.48.2.20	LvIniSetParent	260
5.48.2.21	LvIniSetSectionRawLine	261
5.48.2.22	LvIniSetString	261
5.49	LvStatus definitions	262
5.49.1	Detailed Description	263
5.49.2	Macro Definition Documentation	263
5.49.2.1	LVSTATUS_ACQUISITION_CANNOT_BE_STARTED	263
5.49.2.2	LVSTATUS_ACQUISITION_CANNOT_BE_STOPPED	264
5.49.2.3	LVSTATUS_AVISAVER_TOO_MANY_INSTANCES	264
5.49.2.4	LVSTATUS_BUFFER_IS_QUEUED	264
5.49.2.5	LVSTATUS_BUFFER_NOT_FILLED	264
5.49.2.6	LVSTATUS_CANNOT_LOAD_GENTL	264
5.49.2.7	LVSTATUS_CANNOT_LOAD_XML	264
5.49.2.8	LVSTATUS_CANNOT_REOPEN_LIBRARY	264
5.49.2.9	LVSTATUS_CHUNK_ADAPTER_NOT_AVAILABLE	264
5.49.2.10	LVSTATUS_DEVICE_NOT_ACCESSIBLE	264
5.49.2.11	LVSTATUS_DEVICE_NOT_READWRITE	264
5.49.2.12	LVSTATUS_DEVICE_TOO_MANY_INSTANCES	264
5.49.2.13	LVSTATUS_DISABLED_BY_CALLBACK	265
5.49.2.14	LVSTATUS_DISPLAY_CANNOT_DISPLAY	265
5.49.2.15	LVSTATUS_DISPLAY_LIBRARY_NOT_LOADED	265
5.49.2.16	LVSTATUS_DISPLAY_NOT_OPEN	265
5.49.2.17	LVSTATUS_ENUM_ENTRY_INVALID	265
5.49.2.18	LVSTATUS_ENUM_ENTRY_NOT_AVAILABLE	265
5.49.2.19	LVSTATUS_ERROR	265
5.49.2.20	LVSTATUS_EVENT_NOT_POSSIBLE	265
5.49.2.21	LVSTATUS_EVENT_TOO_MANY_INSTANCES	265
5.49.2.22	LVSTATUS_FILE_CANNOT_CREATE	265
5.49.2.23	LVSTATUS_FILE_CANNOT_OPEN	265
5.49.2.24	LVSTATUS_GC_ABORT	265
5.49.2.25	LVSTATUS_GC_ACCESS_DENIED	266
5.49.2.26	LVSTATUS_GC_BUFFER_TOO_SMALL	266
5.49.2.27	LVSTATUS_GC_BUSY	266
5.49.2.28	LVSTATUS_GC_CUSTOM_ID	266

5.49.2.29 LVSTATUS_GC_ERROR . . . . .	266
5.49.2.30 LVSTATUS_GC_INVALID_ADDRESS . . . . .	266
5.49.2.31 LVSTATUS_GC_INVALID_BUFFER . . . . .	266
5.49.2.32 LVSTATUS_GC_INVALID_HANDLE . . . . .	266
5.49.2.33 LVSTATUS_GC_INVALID_ID . . . . .	266
5.49.2.34 LVSTATUS_GC_INVALID_INDEX . . . . .	266
5.49.2.35 LVSTATUS_GC_INVALID_PARAMETER . . . . .	266
5.49.2.36 LVSTATUS_GC_INVALID_VALUE . . . . .	266
5.49.2.37 LVSTATUS_GC_IO . . . . .	267
5.49.2.38 LVSTATUS_GC_NO_DATA . . . . .	267
5.49.2.39 LVSTATUS_GC_NOT_AVAILABLE . . . . .	267
5.49.2.40 LVSTATUS_GC_NOT_IMPLEMENTED . . . . .	267
5.49.2.41 LVSTATUS_GC_NOT_INITIALIZED . . . . .	267
5.49.2.42 LVSTATUS_GC_OUT_OF_MEMORY . . . . .	267
5.49.2.43 LVSTATUS_GC_PARSING_CHUNK_DATA . . . . .	267
5.49.2.44 LVSTATUS_GC_RESOURCE_EXHAUSTED . . . . .	267
5.49.2.45 LVSTATUS_GC_RESOURCE_IN_USE . . . . .	267
5.49.2.46 LVSTATUS_GC_TIMEOUT . . . . .	267
5.49.2.47 LVSTATUS_GC_UNKNOWN . . . . .	267
5.49.2.48 LVSTATUS_GENICAM_EXCEPTION . . . . .	267
5.49.2.49 LVSTATUS_HANDLE_INVALID . . . . .	268
5.49.2.50 LVSTATUS_INDEX_OUT_OF_RANGE . . . . .	268
5.49.2.51 LVSTATUS_INSUFFICIENT_BUFFER_SIZE . . . . .	268
5.49.2.52 LVSTATUS_INSUFFICIENT_STRING_BUFFER_SIZE . . . . .	268
5.49.2.53 LVSTATUS_INTERFACE_TOO_MANY_INSTANCES . . . . .	268
5.49.2.54 LVSTATUS_INVALID_ENUMENTRY_ID . . . . .	268
5.49.2.55 LVSTATUS_INVALID_IN_THIS_MODULE . . . . .	268
5.49.2.56 LVSTATUS_INVALID_IP_OR_MAC_ADDRESS_FORMAT . . . . .	268
5.49.2.57 LVSTATUS_ITEM_GROUP_INVALID . . . . .	268
5.49.2.58 LVSTATUS_ITEM_INVALID . . . . .	268
5.49.2.59 LVSTATUS_ITEM_NOT_APPLICABLE . . . . .	268
5.49.2.60 LVSTATUS_ITEM_NOT_AVAILABLE . . . . .	269
5.49.2.61 LVSTATUS_ITEM_NOT_READABLE . . . . .	269
5.49.2.62 LVSTATUS_ITEM_NOT_WRITABLE . . . . .	269
5.49.2.63 LVSTATUS_LAST_ERROR_NOT_AVAILABLE . . . . .	269
5.49.2.64 LVSTATUS_LIBRARY_NOT_LOADED . . . . .	269
5.49.2.65 LVSTATUS_LIBRARY_NOT_OPEN . . . . .	269
5.49.2.66 LVSTATUS_LICENSE_NOT_AVAILABLE . . . . .	269
5.49.2.67 LVSTATUS_LUT_NOT_AVAILABLE . . . . .	269
5.49.2.68 LVSTATUS_LUT_UNSUPPORTED_SIZE . . . . .	269

5.49.2.69	LVSTATUS_NO_CONSTANT_FOR_THIS_ENUMENTRY	269
5.49.2.70	LVSTATUS_NODE_MAP_CANNOT_GET	269
5.49.2.71	LVSTATUS_NOT_ENOUGH_BUFFERS	269
5.49.2.72	LVSTATUS_NOT_FOUND	270
5.49.2.73	LVSTATUS_NOT_IMPLEMENTED	270
5.49.2.74	LVSTATUS_NOT_SUPPORTED_FOR_THIS_EVENT	270
5.49.2.75	LVSTATUS_OK	270
5.49.2.76	LVSTATUS_PARAM_NOT_APPLICABLE	270
5.49.2.77	LVSTATUS_PARAMETER_INVALID	270
5.49.2.78	LVSTATUS_RENDERER_TOO_MANY_INSTANCES	270
5.49.2.79	LVSTATUS_SETTINGS_INCOMPATIBLE_ID	270
5.49.2.80	LVSTATUS_SETTINGS_INCOMPATIBLE_MODEL	270
5.49.2.81	LVSTATUS_SETTINGS_INCOMPATIBLE_VERSION	270
5.49.2.82	LVSTATUS_SRCGEN_SYMBOLIC_NOT_AVAILABLE	270
5.49.2.83	LVSTATUS_SRCGEN_TEMPLATE_NOT_AVAILABLE	270
5.49.2.84	LVSTATUS_STREAM_ALREADY_STARTED	271
5.49.2.85	LVSTATUS_STREAM_ALREADY_STOPPED	271
5.49.2.86	LVSTATUS_STREAM_TOO_MANY_INSTANCES	271
5.49.2.87	LVSTATUS_SYSTEM_TOO_MANY_INSTANCES	271
5.49.2.88	LVSTATUS_TIMEOUT	271
5.49.2.89	LVSTATUS_XML_UNZIP_ERROR	271
5.49.3	Typedef Documentation	271
5.49.3.1	LvStatus	271
5.50	SynView Image Processing Library LvStatus definitions	272
5.50.1	Detailed Description	272
5.50.2	Macro Definition Documentation	272
5.50.2.1	LVSTATUS_LVIP_BMP_CONTENTS_INVALID	272
5.50.2.2	LVSTATUS_LVIP_BMP_INCOMPATIBLE_LINE_INCREMENT	272
5.50.2.3	LVSTATUS_LVIP_BMP_INCOMPATIBLE_PIXEL_FORMAT	273
5.50.2.4	LVSTATUS_LVIP_CANNOT_CREATE_WRITE_FILE	273
5.50.2.5	LVSTATUS_LVIP_CANNOT_OPEN_READ_FILE	273
5.50.2.6	LVSTATUS_LVIP_DST_IMAGEINFO_NO_DATA	273
5.50.2.7	LVSTATUS_LVIP_DST_IMG_INFO_INCOMPATIBLE	273
5.50.2.8	LVSTATUS_LVIP_DST_RECT_OUTSIDE_SRC	273
5.50.2.9	LVSTATUS_LVIP_IMAGEINFO_NOT_EQUAL	273
5.50.2.10	LVSTATUS_LVIP_IMAGEINFO_NOT_INITIALIZED	273
5.50.2.11	LVSTATUS_LVIP_INCOMPATIBLE_REF_FLAGS	274
5.50.2.12	LVSTATUS_LVIP_INCOMPATIBLE_REF_PIXEL_FORMAT	274
5.50.2.13	LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_FLAGS	274
5.50.2.14	LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_PIXEL_FORMAT	274

5.50.2.15	LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_SIZE	274
5.50.2.16	LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_SIZE_ROTATED	274
5.50.2.17	LVSTATUS_LVIP_INVALID_DST_POINTER	274
5.50.2.18	LVSTATUS_LVIP_INVALID_LUT_HANDLE	274
5.50.2.19	LVSTATUS_LVIP_INVALID_LUT_TYPE	274
5.50.2.20	LVSTATUS_LVIP_INVALID_PIXEL_FORMAT	275
5.50.2.21	LVSTATUS_LVIP_INVALID_POINTER	275
5.50.2.22	LVSTATUS_LVIP_INVALID_SRC_POINTER	275
5.50.2.23	LVSTATUS_LVIP_JPEG_LOAD_FAILED	275
5.50.2.24	LVSTATUS_LVIP_JPEG_SAVE_FAILED	275
5.50.2.25	LVSTATUS_LVIP_LINEINCREMENT_TOO_BIG	275
5.50.2.26	LVSTATUS_LVIP_MEMORY_ALLOC_FAILED	275
5.50.2.27	LVSTATUS_LVIP_NOT_BAYER_PIXEL_FORMAT	275
5.50.2.28	LVSTATUS_LVIP_NOT_DISPLAYABLE_FORMAT	275
5.50.2.29	LVSTATUS_LVIP_SRC_IMAGEINFO_NO_DATA	276
5.50.2.30	LVSTATUS_LVIP_TIFF_CONTENTS_INVALID	276
5.50.2.31	LVSTATUS_LVIP_UNSUPPORTED	276
5.50.2.32	LVSTATUS_LVIP_UNSUPPORTED_BMP_HEADER	276
5.50.2.33	LVSTATUS_LVIP_UNSUPPORTED_COLOR_PLANES	276
5.50.2.34	LVSTATUS_LVIP_UNSUPPORTED_DST_PIXEL_FORMAT	276
5.50.2.35	LVSTATUS_LVIP_UNSUPPORTED_REVERSION	276
5.50.2.36	LVSTATUS_LVIP_UNSUPPORTED_SRC_PIXEL_FORMAT	276
<b>6</b>	<b>Class Documentation</b>	<b>277</b>
6.1	LvBuffer Class Reference	277
6.1.1	Detailed Description	277
6.2	LvDevice Class Reference	278
6.2.1	Detailed Description	279
6.3	LvEvent Class Reference	279
6.3.1	Detailed Description	280
6.4	LvException Class Reference	280
6.4.1	Detailed Description	280
6.5	LvInterface Class Reference	280
6.5.1	Detailed Description	281
6.6	LvImInfo Struct Reference	281
6.6.1	Detailed Description	282
6.6.2	Member Data Documentation	282
6.6.2.1	Attributes	282
6.6.2.2	BytesPerPixel	282
6.6.2.3	Height	282



---

6.6.2.4	LinePitch	282
6.6.2.5	pData	282
6.6.2.6	pDataB	282
6.6.2.7	pDataG	283
6.6.2.8	pDataR	283
6.6.2.9	PixelFormat	283
6.6.2.10	StructSize	283
6.6.2.11	Width	283
6.7	LvLibrary Class Reference	283
6.7.1	Detailed Description	284
6.8	LvModule Class Reference	284
6.8.1	Detailed Description	285
6.9	LvRenderer Class Reference	286
6.9.1	Detailed Description	286
6.10	LvStream Class Reference	286
6.10.1	Detailed Description	287
6.11	LvSystem Class Reference	287
6.11.1	Detailed Description	288



## **Chapter 1**

# **SynView Reference Guide**



## Chapter 2

# Module Index

### 2.1 Modules

Here is a list of all modules:

SynView	130
SynView Plain C API functions	9
SynView General purpose functions	10
SynView System module functions	14
SynView Interface module functions	18
SynView Device module functions	21
SynView Firmware update functions	65
SynView Stream module functions	27
SynView Buffer module functions	29
SynView Event module functions	35
SynView Renderer module functions	41
SynView Feature control functions	44
SynView C++ API functions	67
SynView LvLibrary methods	68
SynView LvSystem methods	74
SynView LvInterface methods	79
SynView LvDevice methods	84
SynView LvDevice firmware update methods	91
SynView LvStream methods	93
SynView LvBuffer methods	98
SynView LvEvent methods	102
SynView LvRenderer methods	107
SynView LvModule methods	110
SynView defines and typedefs	131
LvStreamStart() flags definitions	214
LvStreamStop() flags definitions	215
LvDeviceUniSetLut() and LvDeviceUniGetLut() flags definitions	216
LvSaveFlag definitions	217
LvPixelFormat definitions	218
LvStatus definitions	262
SynView enumerations	133
Features	149
Enumeration entries	181
SynView Image Processing Library	142
SynView Image Processing Library defines, typedefs and enums	143
Definitions for Enumeration Entry Info	147
SynView Image Processing Library functions	223

---

Common functions . . . . .	224
Image initialization functions . . . . .	225
Region of Interest (ROI) functions . . . . .	228
Lookup Table (LUT) functions . . . . .	229
Bayer decoding/encoding functions . . . . .	238
Rotation and line manipulation functions . . . . .	242
Pixel format conversion functions . . . . .	246
Saving/loading functions . . . . .	247
Overlay functions . . . . .	250
RGB color correction and convolution functions . . . . .	251
Shading correction functions . . . . .	253
SynView Image Processing Library LvStatus definitions . . . . .	272
SynView INI file API . . . . .	254

# Chapter 3

## Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- LvException . . . . . 280
- LvImgInfo . . . . . 281
- LvLibrary . . . . . 283
- LvModule . . . . . 284
  - LvBuffer . . . . . 277
  - LvDevice . . . . . 278
  - LvEvent . . . . . 279
  - LvInterface . . . . . 280
  - LvRenderer . . . . . 286
  - LvStream . . . . . 286
  - LvSystem . . . . . 287





# Chapter 4

## Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">LvBuffer</a>	<a href="#">277</a>
<a href="#">LvDevice</a>	<a href="#">278</a>
<a href="#">LvEvent</a>	<a href="#">279</a>
<a href="#">LvException</a>	<a href="#">280</a>
<a href="#">LvInterface</a>	<a href="#">280</a>
<a href="#">LvImgInfo</a>	<a href="#">281</a>
<a href="#">LvLibrary</a>	<a href="#">283</a>
<a href="#">LvModule</a>	<a href="#">284</a>
<a href="#">LvRenderer</a>	<a href="#">286</a>
<a href="#">LvStream</a>	<a href="#">286</a>
<a href="#">LvSystem</a>	<a href="#">287</a>



# Chapter 5

## Module Documentation

### 5.1 SynView Plain C API functions

#### Modules

- [SynView General purpose functions](#)
- [SynView System module functions](#)
- [SynView Interface module functions](#)
- [SynView Device module functions](#)
- [SynView Stream module functions](#)
- [SynView Buffer module functions](#)
- [SynView Event module functions](#)
- [SynView Renderer module functions](#)
- [SynView Feature control functions](#)

#### 5.1.1 Detailed Description

## 5.2 SynView General purpose functions

### Functions

- LV\_EXTC LV\_DLLIMPORT uint32\_t [LvGetVersion](#) ()
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvOpenLibrary](#) ()
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvCloseLibrary](#) ()
- LV\_EXTC LV\_DLLIMPORT void [LvGetErrorMessage](#) ([LvStatus](#) Error, char \*pMessage, size\_t Size)
- LV\_EXTC LV\_DLLIMPORT void [LvGetLastErrorMessage](#) (char \*pMessage, size\_t Size)
- LV\_EXTC LV\_DLLIMPORT void [LvLog](#) (const char \*pLogMessage)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetLibInfo](#) ([LvEnum](#) Info, int32\_t \*pInfo, int32\_t Param)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetLibInfoStr](#) ([LvEnum](#) Info, char \*pInfoStr, size\_t Size, int32\_t Param)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetLibInfoStrSize](#) ([LvEnum](#) Info, size\_t \*pSize, int32\_t Param)

### 5.2.1 Detailed Description

### 5.2.2 Function Documentation

#### 5.2.2.1 LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvCloseLibrary](#) ( )

Closes the SynView library. This must be done before you exit your application. Be sure to close first all dependent modules (System). If you are using SynView in a Windows DLL, avoid calling this in Windows DllMain() function - for proper functionality this function must be called when the application or DLL is still fully functional, which is not the case of PROCESS\_DETACH in the DllMain(). If you have called [LvOpenLibrary\(\)](#) multiple times, you must balance it by the same number of calls of this function. Only the last call actually does the uninitialization. IMPORTANT: The library must not be opened again once it was already uninitialized.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.2.2.2 LV\_EXTC LV\_DLLIMPORT void [LvGetErrorMessage](#) ( [LvStatus](#) Error, char \* pMessage, size\_t Size )

Returns a short description of the error. Note that only some of the errors are suitable for direct display to the user, many error values indicate states which are understandable to the programmer, but may not be understandable to the end user.

#### Parameters

<i>Error</i>	The error code (the return value of most SynView functions).
<i>pMessage</i>	Pointer to the text buffer.
<i>Size</i>	Size of the buffer.

#### See also

[LvStatus definitions](#).

#### 5.2.2.3 LV\_EXTC LV\_DLLIMPORT void [LvGetLastErrorMessage](#) ( char \* pMessage, size\_t Size )

Returns more detailed description of the last error, which happened in the thread from which this function was called. As the info is recorded inside SynView for each error, the description provides more detailed info, including the name of the function, in which the error happened, and possibly more diagnostic info. The difference to [LvGetErrorMessage\(\)](#) is that [LvGetLastErrorMessage\(\)](#) returns a static string from a numbered table of errors while this

function returns additionally info recorded at the time the error happened. If a function returns LVSTATUS\_OK, it does not reset this error message (for speed reasons) so the correct approach is to get the error number as the function return value and if this return value is not LVSTATUS\_OK, then you can get more info about the error using this function. be sure to call it from the same thread.

## Parameters

<i>pMessage</i>	Pointer to the text buffer.
<i>Size</i>	Size of the buffer.

## See also

[LvStatus definitions.](#)

5.2.2.4 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetLibInfo ( LvEnum *Info*, int32\_t \* *pInfo*, int32\_t *Param* )

Gets a general info in form of a 32-bit integer value.

## Parameters

<i>Info</i>	One of the <a href="#">LvLibInfo</a> values.
<i>pInfo</i>	The value is returned in this parameter.
<i>Param</i>	Additional parameter, required by some types of info.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.2.2.5 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetLibInfoStr ( LvEnum *Info*, char \* *pInfoStr*, size\_t *Size*, int32\_t *Param* )

Gets a general info in form of a string value.

## Parameters

<i>Info</i>	One of the <a href="#">LvLibInfo</a> values.
<i>pInfoStr</i>	The string value is returned in this parameter.
<i>Size</i>	Size of the buffer (to which pInfoStr points).
<i>Param</i>	Additional parameter, required by some types of info.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.2.2.6 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetLibInfoStrSize ( LvEnum *Info*, size\_t \* *pSize*, int32\_t *Param* )

Gets a buffer size needed for a general info in form of a string value.

## Parameters

<i>Info</i>	One of the <a href="#">LvLibInfo</a> values.
<i>pSize</i>	Size of the buffer is returned in this parameter.
<i>Param</i>	Additional parameter, required by some types of info.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.2.2.7 LV\_EXTC LV\_DLLIMPORT uint32\_t LvGetVersion ( )

Returns SynView version.

**Returns**

The returned doubleword contains the build version in the low word and the high word is the major version in the upper byte and subversion in the lower byte. For example:

```
uint32_t Version = LvGetVersion();
printf("SynView %d.%02d.%03d",
      ((Version >> 24) & 0xFF),
      ((Version >> 16) & 0xFF),
      (Version & 0xFFFF));
```

**5.2.2.8 LV\_EXTC LV\_DLLIMPORT void LvLog ( const char \* *pLogMessage* )**

Adds a line to the sv.synview.log. The SynView log is a tool for New Electronic Technology technical support, in some cases may be useful to put to the log additional info from your code.

**Parameters**

<i>pLogMessage</i>	Pointer to the null terminated string with the message.
--------------------	---

**5.2.2.9 LV\_EXTC LV\_DLLIMPORT LvStatus LvOpenLibrary ( )**

Opens the SynView library. This must be done before you can use any other SynView function (with the exception of [LvGetVersion\(\)](#) and [LvGetErrorMessage\(\)](#)). If you are using SynView in Windows DLL, avoid calling this in Windows DIIMain() function - for proper functionality this function must be called when the application or DLL is already fully initialized and there are no restrictions about synchronization (DIIMain has such restrictions). If you call this function multiple times, you must balance it by the same number of the [LvCloseLibrary\(\)](#) calls. Only the first call will actually do the initialization. IMPORTANT: The library must not be opened again once it was already uninitialized.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.3 SynView System module functions

### Functions

- LV\_EXTC LV\_DLLIMPORT [LvStatus LvUpdateSystemList](#) ()
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvGetNumberOfSystems](#) (uint32\_t \*pNumberOfSystems)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvGetSystemId](#) (uint32\_t Index, char \*pSystemId, size\_t Size)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvGetSystemIdSize](#) (uint32\_t Index, size\_t \*pSize)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvSystemOpen](#) (const char \*pSystemId, [LvHSystem](#) \*phSystem)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvSystemClose](#) ([LvHSystem](#) \*phSystem)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvSystemUpdateInterfaceList](#) ([LvHSystem](#) hSystem, uint32\_t Timeout)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvSystemGetNumberOfInterfaces](#) ([LvHSystem](#) hSystem, uint32\_t \*pNumberOfInterfaces)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvSystemGetInterfaceId](#) ([LvHSystem](#) hSystem, uint32\_t Index, char \*pInterfaceId, size\_t Size)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvSystemGetInterfaceIdSize](#) ([LvHSystem](#) hSystem, uint32\_t Index, size\_t \*pSize)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvSystemFindInterface](#) ([LvHSystem](#) hSystem, [LvEnum](#) FindBy, const char \*pFindStr, char \*pInterfaceId, size\_t Size)

### 5.3.1 Detailed Description

### 5.3.2 Function Documentation

#### 5.3.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetNumberOfSystems ( uint32\_t \* pNumberOfSystems )

Returns the number of systems found after the [LvUpdateSystemList\(\)](#) call. Typical use of this function is in iterating systems using the [LvGetSystemId\(\)](#) function.

#### Parameters

<i>pNumberOfSystems</i>	The number of systems found.
-------------------------	------------------------------

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.3.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetSystemId ( uint32\_t Index, char \* pSystemId, size\_t Size )

Returns the string ID of the system at given index. This ID is used in the [LvSystemOpen\(\)](#) function for opening the system.

#### Parameters

<i>Index</i>	Zero-based index of the system, a value $\geq 0$ and $<$ number of systems, returned by the <a href="#">LvGetNumberOfSystems()</a> function.
<i>pSystemId</i>	Pointer to a string buffer, where the system ID will be placed.
<i>Size</i>	Size of the buffer.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.3.2.3 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetSystemIdSize ( uint32\_t Index, size\_t \* pSize )

Returns the size of the string buffer needed to hold the system ID string, including the terminating zero character.



## Parameters

<i>Index</i>	Zero-based index of the system, a value $\geq 0$ and $<$ number of systems, returned by the <a href="#">LvGetNumberOfSystems()</a> function.
<i>pSize</i>	Size of the buffer is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.3.2.4 LV\_EXTC LV\_DLLIMPORT LvStatus LvSystemClose ( LvHSystem \* *phSystem* )

Closes the opened system. Actually it means freeing the corresponding GenTL library. Be sure you first close all dependent modules (Interface, Event etc.). If the System was opened multiple times, it only decreases the reference counter (see the note by the [LvSystemOpen\(\)](#)).

## Parameters

<i>phSystem</i>	Pointer to a handle to the System module, obtained from the <a href="#">LvSystemOpen()</a> function. The handle is assigned 0 after the operation.
-----------------	--

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.3.2.5 LV\_EXTC LV\_DLLIMPORT LvStatus LvSystemFindInterface ( LvHSystem *hSystem*, LvEnum *FindBy*, const char \* *pFindStr*, char \* *pInterfaceld*, size\_t *Size* )

Finds the interface according specified criteria and returns a string ID of the interface, which is used by the [LvInterfaceOpen\(\)](#) function. This function does not update the interface list - if you need to do so, call the [LvSystemUpdateInterfaceList\(\)](#) function before calling this function.

## Parameters

<i>hSystem</i>	A handle to the System module, obtained from the <a href="#">LvSystemOpen()</a> function.
<i>FindBy</i>	Specifies by which criteria to find the interface. Use one of the <a href="#">LvFindBy</a> constants.
<i>pFindStr</i>	Specifies the find string, the meaning of which is determined by the FindBy parameter, for example when using the <a href="#">LvFindBy_IPAddress</a> , this string should contain the IP address searched for. The searched string is not case sensitive and need not be complete (is searched as a substring).
<i>pInterfaceld</i>	Pointer to a string buffer, where the interface ID will be placed.
<i>Size</i>	Size of the string buffer.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#). If the Interface is found, the returned status is LVSTATUS\_OK.

5.3.2.6 LV\_EXTC LV\_DLLIMPORT LvStatus LvSystemGetInterfaceld ( LvHSystem *hSystem*, uint32\_t *Index*, char \* *pInterfaceld*, size\_t *Size* )

Returns a string ID of the interface, which is used by the [LvInterfaceOpen\(\)](#) function.

## Parameters

<i>hSystem</i>	A handle to the System module, obtained from the <a href="#">LvSystemOpen()</a> function.
<i>Index</i>	Zero-based index of the interface, a value $\geq 0$ and $<$ number of interfaces, returned by the <a href="#">LvSystemGetNumberOfInterfaces()</a> function.
<i>pInterfaceId</i>	Pointer to a string buffer, where the interface ID will be placed.
<i>Size</i>	Size of the string buffer.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.3.2.7 `LV_EXTC LV_DLLIMPORT LvStatus LvSystemGetInterfaceIdSize ( LvHSystem hSystem, uint32_t Index, size_t * pSize )`

Returns the size of the string buffer needed to hold the Interface ID string, including the terminating zero character.

## Parameters

<i>hSystem</i>	A handle to the System module, obtained from the <a href="#">LvSystemOpen()</a> function.
<i>Index</i>	Zero-based index of the interface, a value $\geq 0$ and $<$ number of interfaces, returned by the <a href="#">LvSystemGetNumberOfInterfaces()</a> function.
<i>pSize</i>	Size of the buffer is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.3.2.8 `LV_EXTC LV_DLLIMPORT LvStatus LvSystemGetNumberOfInterfaces ( LvHSystem hSystem, uint32_t * pNumberOfInterfaces )`

Returns the number of found interfaces, after the [LvSystemUpdateInterfaceList\(\)](#) call.

## Parameters

<i>hSystem</i>	A handle to the System module, obtained from the <a href="#">LvSystemOpen()</a> function.
<i>pNumberOfInterfaces</i>	Number of interfaces found.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.3.2.9 `LV_EXTC LV_DLLIMPORT LvStatus LvSystemOpen ( const char * pSystemId, LvHSystem * phSystem )`

Opens the System module. Opening the system actually means loading the corresponding GenTL library. Note that before you can open the System, the [LvOpenLibrary\(\)](#) must be called. The same system can be open multiple times (there is a reference counter inside); in such case there must be also the same number of [LvSystemClose\(\)](#) calls used (every open increase the reference count and every close decreases it).

## Parameters

<i>pSystemId</i>	A string ID of the system. This can be either an empty string - then the default system is opened, or it can be a string obtained from the <a href="#">LvGetSystemId()</a> function.
<i>phSystem</i>	Pointer to a handle to the opened System module.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.3.2.10 LV\_EXTC LV\_DLLIMPORT LvStatus LvSystemUpdateInterfaceList ( LvHSystem hSystem, uint32\_t Timeout )

Updates the internal list of available interfaces. You can then iterate through them by [LvSystemGetNumberOfInterfaces\(\)](#) and [LvSystemGetInterfaceId\(\)](#).

**Parameters**

<i>hSystem</i>	A handle to the System module, obtained from the <a href="#">LvSystemOpen()</a> function.
<i>Timeout</i>	Specifies a timeout in ms for searching the interfaces. This applies only to special cases of interfaces, where some delay can happen; common interfaces are detected without any significant delays.

**Returns**

If the timeout has expired while waiting for the completion, the function returns [LVSTATUS\\_TIMEOUT](#), otherwise it returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.3.2.11 LV\_EXTC LV\_DLLIMPORT LvStatus LvUpdateSystemList ( )

Updates the list of systems available. This function must be called before iterating through the systems by the [LvGetNumberOfSystems\(\)](#) and [LvGetSystemId\(\)](#) functions. The systems are physically represented by GenTL libraries available in the operating systems, this call searches for them in standard locations. See also the description of the sv.synview.ini file in the SynView User's Guide. Note that this function is seldom needed, most applications will work with the default system (see [LvSystemOpen\(\)](#) for details).

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.4 SynView Interface module functions

### Functions

- LV\_EXTC LV\_DLLIMPORT [LvStatus LvInterfaceOpen](#) ([LvHSystem](#) hSystem, const char \*pInterfaceld, [LvHInterface](#) \*phInterface)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvInterfaceClose](#) ([LvHInterface](#) \*phInterface)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvInterfaceUpdateDeviceList](#) ([LvHInterface](#) hInterface, uint32\_t Timeout)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvInterfaceGetNumberOfDevices](#) ([LvHInterface](#) hInterface, uint32\_t \*pDevices)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvInterfaceGetDeviceld](#) ([LvHInterface](#) hInterface, uint32\_t Index, char \*pDeviceld, size\_t Size)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvInterfaceGetDeviceldSize](#) ([LvHInterface](#) hInterface, uint32\_t Index, size\_t \*pSize)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvInterfaceFindDevice](#) ([LvHInterface](#) hInterface, [LvEnum](#) FindBy, const char \*pFindStr, char \*pDeviceld, size\_t Size)

### 5.4.1 Detailed Description

### 5.4.2 Function Documentation

#### 5.4.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvInterfaceClose ( LvHInterface \* phInterface )

Closes the interface. If the Interface was opened multiple times, it only decreases the reference counter (see a note by the [LvInterfaceOpen\(\)](#)). Be sure you first close all dependent modules (Device, Event etc.).

#### Parameters

<i>phInterface</i>	Pointer to a handle to the Interface module, obtained from the <a href="#">LvInterfaceOpen()</a> function. The handle is assigned 0 after the operation.
--------------------	--

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.4.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvInterfaceFindDevice ( LvHInterface hInterface, LvEnum FindBy, const char \* pFindStr, char \* pDeviceld, size\_t Size )

Finds the device according specified criteria and returns a string ID of the device, which can be used by the [LvDeviceOpen\(\)](#) function. This function does not update the device list - if you need to do so, call the [LvInterfaceUpdateDeviceList\(\)](#) function before calling this function.

#### Parameters

<i>hInterface</i>	A handle to the Interface module, obtained from the <a href="#">LvInterfaceOpen()</a> function.
<i>FindBy</i>	Specifies by which criteria to find the interface. Use one of the <a href="#">LvFindBy</a> constants.
<i>pFindStr</i>	Specifies the find string, the meaning of which is determined by the FindBy parameter, for example when using the <a href="#">LvFindBy_IPAddress</a> , this string should contain the IP address searched for. The searched string is not case sensitive and need not be complete (is searched as a substring).

<i>pDeviceId</i>	Pointer to a string buffer, where the device ID will be placed.
<i>Size</i>	Size of the buffer.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#). If the Device is found, the returned status is LVSTATUS\_OK.

#### 5.4.2.3 LV\_EXTC LV\_DLLIMPORT LvStatus LvInterfaceGetDeviceId ( LvHInterface hInterface, uint32\_t Index, char \* pDeviceId, size\_t Size )

Returns a string ID of the device at specified position in the list. Note that this device ID is stable (the same physical device has always the same ID) and it is unique (no other physical device can have the same ID). To hardcode directly the device ID in your application is not recommended, as the application would not be usable, when a defective device needs to be replaced. The SynView User's Guide discuss the ways, how to solve such maintainability demands.

**Parameters**

<i>hInterface</i>	A handle to the Interface module, obtained from the <a href="#">LvInterfaceOpen()</a> function.
<i>Index</i>	Zero-based index of the device, a value $\geq 0$ and $<$ number of devices, returned by the <a href="#">LvInterfaceGetNumberOfDevices()</a> function.
<i>pDeviceId</i>	Pointer to a string buffer, where the device ID will be placed.
<i>Size</i>	Size of the buffer.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.4.2.4 LV\_EXTC LV\_DLLIMPORT LvStatus LvInterfaceGetDeviceIdSize ( LvHInterface hInterface, uint32\_t Index, size\_t \* pSize )

Returns the size of the string buffer needed to hold the Device ID string, including the terminating zero character.

**Parameters**

<i>hInterface</i>	A handle to the Interface module, obtained from the <a href="#">LvInterfaceOpen()</a> function.
<i>Index</i>	Zero-based index of the device, a value $\geq 0$ and $<$ number of devices, returned by the <a href="#">LvInterfaceGetNumberOfDevices()</a> function.
<i>pSize</i>	Size of the buffer is returned in this parameter.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.4.2.5 LV\_EXTC LV\_DLLIMPORT LvStatus LvInterfaceGetNumberOfDevices ( LvHInterface hInterface, uint32\_t \* pDevices )

Returns the number of devices found by the [LvInterfaceUpdateDeviceList\(\)](#) function.

**Parameters**

<i>hInterface</i>	A handle to the Interface module, obtained from the <a href="#">LvInterfaceOpen()</a> function.
<i>pDevices</i>	Number of devices found.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.4.2.6 LV\_EXTC LV\_DLLIMPORT LvStatus LvInterfaceOpen ( LvHSystem hSystem, const char \* pInterfaceId, LvHInterface \* phInterface )

Opens the Interface module. The same Interface can be open multiple times (there is a reference counter inside); in such case there must be also the same number of [LvInterfaceClose\(\)](#) calls used (every open increase the reference count and every close decreases it) .

**Parameters**

<i>hSystem</i>	A handle to the System module, obtained from the <a href="#">LvSystemOpen()</a> function.
<i>pInterfaceId</i>	A string interface ID, obtained by the <a href="#">LvSystemGetInterfaceId()</a> .
<i>phInterface</i>	In this parameter the handle to the interface is returned.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.4.2.7 LV\_EXTC LV\_DLLIMPORT LvStatus LvInterfaceUpdateDeviceList ( LvHInterface hInterface, uint32\_t Timeout )

Updates the Device list. The available devices are searched.

**Parameters**

<i>hInterface</i>	A handle to the Interface module, obtained from the <a href="#">LvInterfaceOpen()</a> function.
<i>Timeout</i>	Specifies a timeout in ms for searching the devices.

**Returns**

If the timeout has expired while waiting for the completion, the function returns [LVSTATUS\\_TIMEOUT](#), otherwise it returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.5 SynView Device module functions

### Modules

- [SynView Firmware update functions](#)

### Functions

- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceOpen](#) ([LvHInterface](#) hInterface, const char \*pDeviceId, [LvHDevice](#) \*phDevice, [LvEnum](#) Access)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceReOpen](#) ([LvHInterface](#) hInterface, const char \*pDeviceId, [LvHDevice](#) hDevice, [LvEnum](#) Access=[LvDeviceAccess\\_Exclusive](#))
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceClose](#) ([LvHDevice](#) \*phDevice)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceGetNumberOfStreams](#) ([LvHDevice](#) hDevice, uint32\_t \*pNumberOfStreams)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceGetStreamId](#) ([LvHDevice](#) hDevice, uint32\_t Index, char \*pStreamId, size\_t Size)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceGetStreamIdSize](#) ([LvHDevice](#) hDevice, uint32\_t Index, size\_t \*pSize)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceAcquisitionStart](#) ([LvHDevice](#) hDevice, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceAcquisitionStop](#) ([LvHDevice](#) hDevice, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceAcquisitionAbort](#) ([LvHDevice](#) hDevice, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceAcquisitionArm](#) ([LvHDevice](#) hDevice, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceSaveSettings](#) ([LvHDevice](#) hDevice, const char \*pId, const char \*pFileName, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceLoadSettings](#) ([LvHDevice](#) hDevice, const char \*pId, const char \*pFileName, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceLoadBatch](#) ([LvHDevice](#) hDevice, const char \*pFileName)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceUniSetLut](#) ([LvHDevice](#) hDevice, [LvEnum](#) Selector, void \*pLUT, size\_t Size, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceUniGetLut](#) ([LvHDevice](#) hDevice, [LvEnum](#) Selector, void \*pLUT, size\_t Size, uint32\_t Options)
- [LvStatus LvDevice::LoadBatch](#) (const char \*pFileName)

### 5.5.1 Detailed Description

### 5.5.2 Function Documentation

#### 5.5.2.1 [LvStatus LvDevice::LoadBatch](#) ( const char \* *pFileName* )

Loads device batch commands from a file.

#### Parameters

<i>pFileName</i>	The file specification, where the configuration is stored. It is a text file.
------------------	---

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.5.2.2 LV\_EXTC LV\_DLLIMPORT [LvStatus LvDeviceAcquisitionAbort](#) ( [LvHDevice](#) *hDevice*, uint32\_t *Options* )

Aborts the acquisition immediately, without completing the current Frame or waiting on a trigger.

## Parameters

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>Options</i>	Reserved for future use, must be 0 or omitted.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

### 5.5.2.3 LV\_EXTC LV\_DLLIMPORT LvStatus LvDeviceAcquisitionArm ( LvHDevice hDevice, uint32\_t Options )

Prepares the device for acquisition, so that the acquisition using the [LvDeviceAcquisitionStart\(\)](#) function then can start fast. If it is not called before [LvDeviceAcquisitionStart\(\)](#), it is called automatically inside the [LvDeviceAcquisitionStart\(\)](#).

## Parameters

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>Options</i>	Reserved for future use, must be 0 or omitted.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

### 5.5.2.4 LV\_EXTC LV\_DLLIMPORT LvStatus LvDeviceAcquisitionStart ( LvHDevice hDevice, uint32\_t Options )

Starts the acquisition. This function includes more than just calling the AcquisitionStart remote command on the device - it checks the size of the buffers, prepares the streams for the start, locks GenTL params and then starts the acquisition on the device itself. Always check the success of this function call.

## Parameters

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>Options</i>	Reserved for future use, must be 0 or omitted.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

### 5.5.2.5 LV\_EXTC LV\_DLLIMPORT LvStatus LvDeviceAcquisitionStop ( LvHDevice hDevice, uint32\_t Options )

Stops the acquisition.

## Parameters

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>Options</i>	Reserved for future use, must be 0 or omitted.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

### 5.5.2.6 LV\_EXTC LV\_DLLIMPORT LvStatus LvDeviceClose ( LvHDevice \* phDevice )

Closes the Device. Be sure you first close all dependent modules (Stream, Event etc.).



## Parameters

<i>phDevice</i>	Pointer to a handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function. This handle is assigned 0 after the operation.
-----------------	---

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.5.2.7 LV\_EXTC LV\_DLLIMPORT LvStatus LvDeviceGetNumberOfStreams ( LvHDevice hDevice, uint32\_t \* pNumberOfStreams )

Returns the number of available stream types for this device. You can then iterate the streams by the [LvDeviceGetStreamId\(\)](#) function.

## Parameters

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>pNumberOfStreams</i>	The number of streams is returned here.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.5.2.8 LV\_EXTC LV\_DLLIMPORT LvStatus LvDeviceGetStreamId ( LvHDevice hDevice, uint32\_t Index, char \* pStreamId, size\_t Size )

Returns a string Stream ID, needed for opening the stream.

## Parameters

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>Index</i>	Zero-based index of the stream type, a value $\geq 0$ and $<$ number of streams, returned by the <a href="#">LvDeviceGetNumberOfStreams()</a> function.
<i>pStreamId</i>	Pointer to a string buffer, where the stream ID will be placed.
<i>Size</i>	Size of the buffer.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.5.2.9 LV\_EXTC LV\_DLLIMPORT LvStatus LvDeviceGetStreamIdSize ( LvHDevice hDevice, uint32\_t Index, size\_t \* pSize )

Returns the size of the string buffer needed to hold the stream ID at given index, including the space for the terminating zero character.

## Parameters

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>Index</i>	Zero-based index of the stream type, a value $\geq 0$ and $<$ number of streams, returned by the <a href="#">LvDeviceGetNumberOfStreams()</a> function.

<i>pSize</i>	Size of the buffer is returned in this parameter.
--------------	---

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.5.2.10 LV\_EXTC LV\_DLLIMPORT LvStatus LvDeviceLoadBatch ( LvHDevice *hDevice*, const char \* *pFileName* )

Loads device batch commands from a file.

**Parameters**

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>pFileName</i>	The file specification, where the configuration is stored. It is a text file.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.5.2.11 LV\_EXTC LV\_DLLIMPORT LvStatus LvDeviceLoadSettings ( LvHDevice *hDevice*, const char \* *pId*, const char \* *pFileName*, uint32\_t *Options* )

Loads the device settings from a file. In the Options can be specified which parts of the device configuration are to be loaded - the Remote, Local and/or GenTL part. Note that there are several factors, which can break the compatibility of the settings file with the current device:

- When the current device is of different vendor/model, the settings file is most probably not compatible.
- When the current device is of the same vendor/model, but uses a different firmware version - this could mean that some remote device features are not present or behave differently.
- When the XML version of the Local and GenTL features changes - again this could mean that some features are not present or behave differently. For this reason this function checks the versions and if not the same, it returns either the [LVSTATUS\\_SETTINGS\\_INCOMPATIBLE\\_MODEL](#) or [LVSTATUS\\_SETTINGS\\_INCOMPATIBLE\\_VERSION](#) error states. As the difference in versions might not necessarily mean a real incompatibility, you can use the [LvSaveFlag\\_IgnoreVersion](#) and [LvSaveFlag\\_IgnoreModel](#) flags in the Options parameter in order to force this function to try to load the settings even if the possible incompatibility is found.

**Parameters**

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>pId</i>	A string ID enabling to protect the file. If you specify a non-empty ID in <a href="#">LvDeviceSaveSettings()</a> , you must use the same ID in <a href="#">LvDeviceLoadSettings()</a> , otherwise the settings are not loaded.
<i>pFileName</i>	The file specification, where the configuration is stored. It is a text file.
<i>Options</i>	One or or-ed combination of <a href="#">LvSaveFlag definitions</a> .

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.5.2.12 LV\_EXTC LV\_DLLIMPORT LvStatus LvDeviceOpen ( LvHInterface *hInterface*, const char \* *pDeviceId*, LvHDevice \* *phDevice*, LvEnum *Access* )

Opens the Device module. This physically means opening a connection with the device and retrieving a list of device remote features. Always check the success of this function call; the opening may fail for example when you request an exclusive access and the device is already open by some other application.

## Parameters

<i>hInterface</i>	A handle to the Interface module, obtained from the <a href="#">LvInterfaceOpen()</a> function.
<i>pDeviceId</i>	A string ID of the device, obtained by <a href="#">LvInterfaceGetDeviceId()</a> function.
<i>phDevice</i>	In this parameter the handle to the Device is returned.
<i>Access</i>	Desired device access, one of the <a href="#">LvDeviceAccess</a> constants.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.5.2.13** `LV_EXTC LV_DLLIMPORT LvStatus LvDeviceReOpen ( LvHInterface hInterface, const char * pDeviceId, LvHDevice hDevice, LvEnum Access = LvDeviceAccess_Exclusive )`

Re-Opens the Device. does not create a new class, just tries to reopen the connection

## Parameters

<i>hInterface</i>	A handle to the Interface module, obtained from the <a href="#">LvInterfaceOpen()</a> function.
<i>pDeviceId</i>	A string ID of the device, obtained by <a href="#">LvInterfaceGetDeviceId()</a> function.
<i>hDevice</i>	The handle to the Device
<i>Access</i>	Desired device access, one of the <a href="#">LvDeviceAccess</a> constants.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.5.2.14** `LV_EXTC LV_DLLIMPORT LvStatus LvDeviceSaveSettings ( LvHDevice hDevice, const char * pId, const char * pFileName, uint32_t Options )`

Saves the device settings to a file. In the Options can be specified which parts of the device configuration are to be saved - the Remote, Local and/or GenTL part. See also notes by [LvDeviceLoadSettings\(\)](#).

## Parameters

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>pId</i>	A string ID enabling to protect the file. If you specify a non-empty ID in <a href="#">LvDeviceSaveSettings()</a> , you must use the same ID in <a href="#">LvDeviceLoadSettings()</a> , otherwise the settings are not loaded.
<i>pFileName</i>	The file specification, to which the configuration is stored. It is a text file.
<i>Options</i>	One or or-ed combination of <a href="#">LvSaveFlag definitions</a> .

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.5.2.15** `LV_EXTC LV_DLLIMPORT LvStatus LvDeviceUniGetLut ( LvHDevice hDevice, LvEnum Selector, void * pLUT, size_t Size, uint32_t Options )`

Gets the lookup table. See [LvDeviceUniSetLut\(\)](#) for details. The LUT is automatically recalculated to appropriate type, if you use different LUT bit depth than is the actually used for the current pixel format. So you can for example read the 12-bit LUT to 8-bit LUT array.

## Parameters

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>Selector</i>	Lookup table selector, see <a href="#">LvLUTSelector</a> .
<i>pLUT</i>	Pointer to the lookup table.
<i>Size</i>	Size of the lookup table. The only valid values are <ul style="list-style-type: none"> <li>• 256 for 8-bit LUT</li> <li>• 2048 for 10-bit LUT</li> <li>• 8192 for 12-bit LUT</li> </ul>
<i>Options</i>	The <code>LvUniLutFlags_HwLut</code> option can be used to apply to function directly on HW LUT.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.5.2.16 `LV_EXTC LV_DLLIMPORT LvStatus LvDeviceUniSetLut ( LvHDevice hDevice, LvEnum Selector, void * pLUT, size_t Size, uint32_t Options )`

Sets the lookup table. If the hardware lookup table is available, it is used, otherwise a software lookup table is set. This function belongs to a set of functions, which unify the functionality of devices with real-time processing embedded in hardware (RTF) and devices without real-time processing, for which the processing is made by software. The LUT is automatically recalculated to appropriate type, if you use different LUT bit depth than is the actually used for the current pixel format.

## Parameters

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>Selector</i>	Lookup table selector, see <a href="#">LvLUTSelector</a> .
<i>pLUT</i>	Pointer to the lookup table.
<i>Size</i>	Size of the lookup table. The only valid values are <ul style="list-style-type: none"> <li>• 256 for 8-bit LUT</li> <li>• 2048 for 10-bit LUT</li> <li>• 8192 for 12-bit LUT</li> </ul>
<i>Options</i>	The <code>LvUniLutFlags_HwLut</code> option can be used to apply to function directly on HW LUT.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.6 SynView Stream module functions

### Functions

- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvStreamOpen](#) ([LvHDevice](#) hDevice, const char \*pStreamId, [LvHStream](#) \*phStream)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvStreamClose](#) ([LvHStream](#) \*phStream)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvStreamGetBufferAt](#) ([LvHStream](#) hStream, uint32\_t BufferIndex, [LvHBuffer](#) \*phBuffer)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvStreamFlushQueue](#) ([LvHStream](#) hStream, [LvEnum](#) Operation)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvStreamStart](#) ([LvHStream](#) hStream, uint32\_t StartFlags, uint32\_t ImagesToAcquire)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvStreamStop](#) ([LvHStream](#) hStream, uint32\_t StopFlags)

### 5.6.1 Detailed Description

### 5.6.2 Function Documentation

#### 5.6.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvStreamClose ( LvHStream \* phStream )

Closes the Stream. Be sure you first close all dependent modules (Buffers, Event, Renderer etc.).

#### Parameters

<i>phStream</i>	Pointer to a handle to the Stream module, obtained from the <a href="#">LvStreamOpen()</a> function. This handle is assigned 0 after the operation.
-----------------	---

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.6.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvStreamFlushQueue ( LvHStream hStream, LvEnum Operation )

Moves the buffers according to the [LvQueueOperation](#) specified.

#### Parameters

<i>hStream</i>	A handle to the Stream module, obtained from the <a href="#">LvStreamOpen()</a> function.
<i>Operation</i>	One of the <a href="#">LvQueueOperation</a> .

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.6.2.3 LV\_EXTC LV\_DLLIMPORT LvStatus LvStreamGetBufferAt ( LvHStream hStream, uint32\_t BufferIndex, LvHBuffer \* phBuffer )

Returns the buffer handle at given index.

#### Parameters

<i>hStream</i>	A handle to the Stream module, obtained from the <a href="#">LvStreamOpen()</a> function.
<i>BufferIndex</i>	Zero-based index.
<i>phBuffer</i>	In this parameter the buffer handle is returned.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.6.2.4 `LV_EXTC LV_DLLIMPORT LvStatus LvStreamOpen ( LvHDevice hDevice, const char * pStreamId, LvHStream * phStream )`

Opens the stream module, associated with the device.

**Parameters**

<i>hDevice</i>	A handle to the Device module, obtained from the <a href="#">LvDeviceOpen()</a> function.
<i>pStreamId</i>	A string ID of the stream, obtained from <a href="#">LvDeviceGetStreamId()</a> . If an empty string is used, the first found stream is opened. This is usually the image data stream.
<i>phStream</i>	In this parameter the handle to the Stream is returned.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.6.2.5 `LV_EXTC LV_DLLIMPORT LvStatus LvStreamStart ( LvHStream hStream, uint32_t StartFlags, uint32_t ImagesToAcquire )`

Starts the stream. This function need not be used on the image stream, where it is called automatically in the [LvDeviceAcquisitionStart\(\)](#) function.

**Parameters**

<i>hStream</i>	A handle to the Stream module, obtained from the <a href="#">LvStreamOpen()</a> function.
<i>StartFlags</i>	One of the <code>GroupSynView_StreamStartFlags</code> .
<i>ImagesToAcquire</i>	Number of images to acquire.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.6.2.6 `LV_EXTC LV_DLLIMPORT LvStatus LvStreamStop ( LvHStream hStream, uint32_t StopFlags )`

Stops the stream. This function need not be used on the image stream, where it is called automatically in the [LvDeviceAcquisitionStop\(\)](#) function.

**Parameters**

<i>hStream</i>	A handle to the Stream module, obtained from the <a href="#">LvStreamOpen()</a> function.
<i>StopFlags</i>	One of the <code>GroupSynView_StreamStopFlags</code> .

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.7 SynView Buffer module functions

### Functions

- LV\_EXTC LV\_DLLIMPORT [LvStatus LvBufferOpen](#) ([LvHStream](#) hStream, void \*pDataPointer, size\_t DataSize, void \*pUserPointer, uint32\_t Options, [LvHBuffer](#) \*phBuffer)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvBufferClose](#) ([LvHBuffer](#) \*phBuffer)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvBufferAttachProcessBuffer](#) ([LvHBuffer](#) hBuffer, void \*pDataPointer, size\_t DataSize)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvBufferQueue](#) ([LvHBuffer](#) hBuffer)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvBufferParseChunkData](#) ([LvHBuffer](#) hBuffer, uint32\_t UpdateLayout)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvBufferSaveImageToBmpFile](#) ([LvHBuffer](#) hBuffer, const char \*pFileName)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvBufferSaveImageToJpgFile](#) ([LvHBuffer](#) hBuffer, const char \*pFileName, uint32\_t Quality)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvBufferSaveImageToTifFile](#) ([LvHBuffer](#) hBuffer, const char \*pFileName, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvBufferGetImgInfo](#) ([LvHBuffer](#) hBuffer, [LvImgInfo](#) \*pImgInfo, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvBufferGetLastPaintRect](#) ([LvHBuffer](#) hBuffer, int32\_t \*pX, int32\_t \*pY, int32\_t \*pWidth, int32\_t \*pHeight)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvBufferUniCalculateWhiteBalance](#) ([LvHBuffer](#) hBuffer)

#### 5.7.1 Detailed Description

#### 5.7.2 Function Documentation

##### 5.7.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvBufferAttachProcessBuffer ( LvHBuffer hBuffer, void \* pDataPointer, size\_t DataSize )

Attaches a process buffer to a buffer. The process buffer may be needed for software processing, for example Bayer decoding, if the device hardware is not capable of it. The process buffer can be either supplied by the application by this function, or allocated automatically by SynView upon need.

#### Parameters

<i>hBuffer</i>	A handle to the Buffer module, obtained from the <a href="#">LvBufferOpen()</a> function.
<i>pDataPointer</i>	Pointer to the supplied buffer.
<i>DataSize</i>	Size of the buffer.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

##### 5.7.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvBufferClose ( LvHBuffer \* phBuffer )

Closes the buffer. On the GenTL level it corresponds to the DSRevokeBuffer() function.

#### Parameters

<i>phBuffer</i>	A handle to the Buffer module, obtained from the <a href="#">LvBufferOpen()</a> function. This handle is assigned 0 after the operation.
-----------------	--

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

### 5.7.2.3 LV\_EXTC LV\_DLLIMPORT LvStatus LvBufferGetImgInfo ( LvHBuffer hBuffer, LvplmgInfo \* plmgInfo, uint32\_t Options )

Fills the [LvplmgInfo](#) structure for the image in the buffer. This simplifies a direct use of the [SynView Image Processing Library](#). If the image is processed, the image info points to the processed image, otherwise it points to the original image.

#### Parameters

<i>hBuffer</i>	A handle to the Buffer module, obtained from the <a href="#">LvBufferOpen()</a> function.
<i>plmgInfo</i>	Pointer to the <a href="#">ImgInfo</a> structure, to which are the image parameters stored.
<i>Options</i>	Currently unused, must be 0.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

### 5.7.2.4 LV\_EXTC LV\_DLLIMPORT LvStatus LvBufferGetLastPaintRect ( LvHBuffer hBuffer, int32\_t \* pX, int32\_t \* pY, int32\_t \* pWidth, int32\_t \* pHeight )

Returns the rectangle to which the buffer was last painted. This is useful namely in case you have a tile mode and want to identify the buffer according a mouse click location. If the buffer was not yet painted by the renderer, the returned values are 0.

#### Parameters

<i>hBuffer</i>	A handle to the Buffer module, obtained from the <a href="#">LvBufferOpen()</a> function.
<i>pX</i>	Pointer to X offset in pixels.
<i>pY</i>	Pointer to Y offset in pixels.
<i>pWidth</i>	Pointer to Width in pixels.
<i>pHeight</i>	Pointer to Height in pixels.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

### 5.7.2.5 LV\_EXTC LV\_DLLIMPORT LvStatus LvBufferOpen ( LvHStream hStream, void \* pDataPointer, size\_t DataSize, void \* pUserPointer, uint32\_t Options, LvHBuffer \* phBuffer )

Opens a buffer. On the GenTL level it corresponds to [DSAnnounceBuffer\(\)](#) or [DSAllocAndAnnounceBuffer\(\)](#).

#### Parameters

<i>hStream</i>	A handle to the Stream module, obtained from the <a href="#">LvStreamOpen()</a> function.
<i>pDataPointer</i>	Pointer to image data buffer. This can be supplied by the application (in such case the <a href="#">DataSize</a> must be set to the actual size of the buffer), or can be left NULL - in such case the buffer is allocated by SynView.
<i>DataSize</i>	Size of the buffer supplied, or 0 if the <i>pDataPointer</i> is NULL.
<i>pUserPointer</i>	A user pointer, which is then passed back in the <a href="#">LvEventCallbackNewBufferFunc()</a> . It enables the application to reference some own data structure associated with the buffer.
<i>Options</i>	Currently unused, must be 0.
<i>phBuffer</i>	To this parameter the handle to the buffer is returned.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).



5.7.2.6 LV\_EXTC LV\_DLLIMPORT LvStatus LvBufferParseChunkData ( LvHBuffer *hBuffer*, uint32\_t *UpdateLayout* )

Parses the chunk data of the image. The chunk data are then accessible as device remote features (for example [LvDevice\\_ChunkTimestamp](#)).

## Parameters

<i>hBuffer</i>	A handle to the Buffer module, obtained from the <a href="#">LvBufferOpen()</a> function.
<i>UpdateLayout</i>	If set to 1, the layout of chunk data is decoded. If set to 0, the data are only read from already decoded layout - this is faster. Usually, the layout of the chunk data is constant, so it needs to be decoded only at first call of this function.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.7.2.7 LV\_EXTC LV\_DLLIMPORT LvStatus LvBufferQueue ( LvHBuffer hBuffer )

Puts the buffer to the input buffer pool. This is an important part of the image handling loop: after the buffer with the acquired image is passed to the application, the application must return it to the input buffer pool by this function after processing.

## Parameters

<i>hBuffer</i>	A handle to the Buffer module, obtained from the <a href="#">LvBufferOpen()</a> function.
----------------	---

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.7.2.8 LV\_EXTC LV\_DLLIMPORT LvStatus LvBufferSaveImageToBmpFile ( LvHBuffer hBuffer, const char \* pFileName )

Saves the image to a file in Windows BMP format. If the image is in the pixel format not compatible with the BMP format, it is automatically converted.

## Parameters

<i>hBuffer</i>	A handle to the Buffer module, obtained from the <a href="#">LvBufferOpen()</a> function.
<i>pFileName</i>	The file name. Be sure to specify it with the full path.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.7.2.9 LV\_EXTC LV\_DLLIMPORT LvStatus LvBufferSaveImageToJpgFile ( LvHBuffer hBuffer, const char \* pFileName, uint32\_t Quality )

Saves the image to a file in JPEG format. If the image is in the pixel format not compatible with the JPEG format, it is automatically converted.

## Parameters

<i>hBuffer</i>	A handle to the Buffer module, obtained from the <a href="#">LvBufferOpen()</a> function.
<i>pFileName</i>	The file name. Be sure to specify it with the full path.
<i>Quality</i>	The quality factor in range from from 1 to 100. The higher is the factor, the higher is the quality and lower the compression.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.7.2.10 LV\_EXTC LV\_DLLIMPORT LvStatus LvBufferSaveImageToTifFile ( LvHBuffer *hBuffer*, const char \* *pFileName*, uint32\_t *Options* )

Saves the image to a file in the TIFF format. If the image is in the pixel format not compatible with the TIF format, it is automatically converted.

## Parameters

<i>hBuffer</i>	A handle to the Buffer module, obtained from the <a href="#">LvBufferOpen()</a> function.
<i>pFileName</i>	The file name. Be sure to specify it with the full path.
<i>Options</i>	Options for saved pixel format. The <a href="#">LvipOption_TiffConvertTo16Bit</a> flag can be used there.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.7.2.11 LV\_EXTC LV\_DLLIMPORT LvStatus LvBufferUniCalculateWhiteBalance ( LvHBuffer hBuffer )

Calculates white balance factors from the current image.

## Parameters

<i>hBuffer</i>	A handle to the Buffer module, obtained from the <a href="#">LvBufferOpen()</a> function.
----------------	---

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.8 SynView Event module functions

### Functions

- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvEventOpen](#) ([LvHModule](#) hModule, [LvEnum](#) EventType, [LvHEvent](#) \*phEvent)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvEventClose](#) ([LvHEvent](#) \*phEvent)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvEventKill](#) ([LvHEvent](#) hEvent)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvEventFlush](#) ([LvHEvent](#) hEvent)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvEventWaitAndGetData](#) ([LvHEvent](#) hEvent, void \*pBuffer, size\_t \*pSize, uint32\_t Timeout)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvEventWaitAndGetNewBuffer](#) ([LvHEvent](#) hEvent, [LvHBuffer](#) \*phBuffer, uint32\_t Timeout)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvEventGetDataInfo](#) ([LvHEvent](#) hEvent, void \*pInBuffer, size\_t InSize, [LvEnum](#) Info, void \*pBuffer, size\_t \*pSize, [LvEnum](#) \*pType, int32\_t Param)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvEventPutData](#) ([LvHEvent](#) hEvent, void \*pBuffer, size\_t Size)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvEventSetCallback](#) ([LvHEvent](#) hEvent, [LvEventCallbackFunc](#) pFunction, void \*pUserParam)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvEventSetCallbackNewBuffer](#) ([LvHEvent](#) hEvent, [LvEventCallbackNewBufferFunc](#) pFunction, void \*pUserParam)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvEventStartThread](#) ([LvHEvent](#) hEvent)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvEventStopThread](#) ([LvHEvent](#) hEvent)
- LV\_EXTC LV\_DLLIMPORT uint32\_t [LvEventCallbackMustExit](#) ([LvHEvent](#) hEvent)

### 5.8.1 Detailed Description

### 5.8.2 Function Documentation

#### 5.8.2.1 LV\_EXTC LV\_DLLIMPORT uint32\_t LvEventCallbackMustExit ( LvHEvent hEvent )

Indicates that the event callback function for [LvEventSetCallback](#) or [LvEventSetCallbackNewBuffer](#) needs to exit as soon as possible

#### Parameters

<i>hEvent</i>	A handle to the Event module, obtained from the <a href="#">LvEventOpen()</a> function.
---------------	---

#### Returns

Returns the must exit status

#### 5.8.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvEventClose ( LvHEvent \* phEvent )

Closes the Event module.

#### Parameters

<i>phEvent</i>	A handle to the Event module, obtained from the <a href="#">LvEventOpen()</a> function. This handle is assigned 0 after the operation.
----------------	--

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.8.2.3 LV\_EXTC LV\_DLLIMPORT LvStatus LvEventFlush ( LvHEvent hEvent )

Discards all buffers in the output buffer queue (waiting to be delivered to the application).

## Parameters

<i>hEvent</i>	A handle to the Event module, obtained from the <a href="#">LvEventOpen()</a> function.
---------------	---

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.8.2.4** LV\_EXTC LV\_DLLIMPORT LvStatus LvEventGetDataInfo ( LvHEvent hEvent, void \* pInBuffer, size\_t InSize, LvEnum Info, void \* pBuffer, size\_t \* pSize, LvEnum \* pType, int32\_t Param )

Enables to parse the buffer from LvEventWaitAndGetData.

## Parameters

<i>hEvent</i>	A handle to the Event module, obtained from the <a href="#">LvEventOpen()</a> function.
<i>pInBuffer</i>	Pointer to a buffer containing event data. This value must not be NULL.
<i>InSize</i>	Size of the provided pInBuffer in bytes.
<i>Info</i>	One of the <a href="#">LvEventDataInfo</a> .
<i>pBuffer</i>	Pointer to a user allocated buffer to receive the requested information. If this parameter is NULL, pSize will contain the minimal size of pBuffer in bytes. If the pType is a string, the size includes the terminating 0.
<i>pSize</i>	Size of the buffer is returned in this parameter.
<i>pType</i>	One of the <a href="#">LvInfoDataType</a> .
<i>Param</i>	Additional parameter, if used, its role is explained by the <a href="#">LvEventDataInfo</a> .

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.8.2.5** LV\_EXTC LV\_DLLIMPORT LvStatus LvEventKill ( LvHEvent hEvent )

Terminates a single wait in the [LvEventWaitAndGetData\(\)](#) function.

## Parameters

<i>hEvent</i>	A handle to the Event module, obtained from the <a href="#">LvEventOpen()</a> function.
---------------	---

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.8.2.6** LV\_EXTC LV\_DLLIMPORT LvStatus LvEventOpen ( LvHModule hModule, LvEnum EventType, LvHEvent \* phEvent )

Opens the Event module for specified owner module.

## Parameters

<i>hModule</i>	A handle to the System, Device or Stream module.
<i>EventType</i>	One of the <a href="#">LvEventType</a> .
<i>phEvent</i>	To this parameter the Event handle is stored.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.8.2.7 LV\_EXTC LV\_DLLIMPORT LvStatus LvEventPutData ( LvHEvent *hEvent*, void \* *pBuffer*, size\_t *Size* )

Puts a new event to Event output queue. This function can be used only for user-defined events.

## Parameters

<i>hEvent</i>	A handle to the Event module, obtained from the <a href="#">LvEventOpen()</a> function.
<i>pBuffer</i>	Pointer to event data.
<i>Size</i>	Size of the event data.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.8.2.8** LV\_EXTC LV\_DLLIMPORT LvStatus LvEventSetCallback ( LvHEvent *hEvent*, LvEventCallbackFunct *pFunction*, void \* *pUserParam* )

Specifies a callback function for the event thread. Note that the callback function cannot be a method of a class.

## Parameters

<i>hEvent</i>	A handle to the Event module, obtained from the <a href="#">LvEventOpen()</a> function.
<i>pFunction</i>	The callback function in the forms of <a href="#">LvEventCallbackFunct</a> .
<i>pUserParam</i>	User parameter, which will be passed to each callback call.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.8.2.9** LV\_EXTC LV\_DLLIMPORT LvStatus LvEventSetCallbackNewBuffer ( LvHEvent *hEvent*, LvEventCallbackNewBufferFunct *pFunction*, void \* *pUserParam* )

Specifies a callback function for the thread of the Event of the [LvEventType\\_NewBuffer](#). Once the application specifies this callback, it becomes responsible for returning the image buffers to the input buffer pool. Note that the callback function cannot be a method of a class.

## Parameters

<i>hEvent</i>	A handle to the Event module, obtained from the <a href="#">LvEventOpen()</a> function.
<i>pFunction</i>	The callback function in the forms of <a href="#">LvEventCallbackNewBufferFunct</a> .
<i>pUserParam</i>	User parameter, which will be passed to each callback call.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.8.2.10** LV\_EXTC LV\_DLLIMPORT LvStatus LvEventStartThread ( LvHEvent *hEvent* )

Starts an internal thread, which waits for events and passes them to specified callback function. When the thread is started, the application must no longer call the [LvEventWaitAndGetData\(\)](#) or [LvEventWaitAndGetNewBufer\(\)](#) functions - this is called internally in the thread and upon return from this function a callback function is called.

## Parameters

<i>hEvent</i>	A handle to the Event module, obtained from the <a href="#">LvEventOpen()</a> function.
---------------	---

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).



5.8.2.11 LV\_EXTC LV\_DLLIMPORT LvStatus LvEventStopThread ( LvHEvent *hEvent* )

Stops the event internal thread.

## Parameters

<i>hEvent</i>	A handle to the Event module, obtained from the <a href="#">LvEventOpen()</a> function.
---------------	---

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.8.2.12** `LV_EXTC LV_DLLIMPORT LvStatus LvEventWaitAndGetData ( LvHEvent hEvent, void * pBuffer, size_t * pSize, uint32_t Timeout )`

Waits for the event and gets its data in one atomic operation. Use this function only for events other than [LvEventType\\_NewBuffer](#), for the the [LvEventType\\_NewBuffer](#) event type use the [LvEventWaitAndGetNewBuffer\(\)](#) function instead. Do not use this function if you use the callback - see [LvEventSetCallback\(\)](#) or [LvEventSetCallbackNewBuffer\(\)](#).

## Parameters

<i>hEvent</i>	A handle to the Event module, obtained from the <a href="#">LvEventOpen()</a> function.
<i>pBuffer</i>	Pointer to a user allocated buffer to receive the event data. The buffer can be parsed by the <a href="#">LvEventGetDataInfo()</a> function.
<i>pSize</i>	Size of the buffer must be specified in this parameter and after the function returns, the actual size is returned in this parameter.
<i>Timeout</i>	The wait timeout in milliseconds. The value 0xFFFFFFFF is considered as infinite. Note that you can also kill waiting from another thread using the <a href="#">LvEventKill()</a> function.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.8.2.13** `LV_EXTC LV_DLLIMPORT LvStatus LvEventWaitAndGetNewBuffer ( LvHEvent hEvent, LvHBuffer * pBuffer, uint32_t Timeout )`

Waits for the event and gets its data in one atomic operation. Use this function only for events of the [LvEventType\\_NewBuffer](#) type. Do not use this function if you use the callback - see [LvEventSetCallback\(\)](#) or [LvEventSetCallbackNewBuffer\(\)](#).

## Parameters

<i>hEvent</i>	A handle to the Event module, obtained from the <a href="#">LvEventOpen()</a> function.
<i>pBuffer</i>	The handle to the received buffer is returned in this parameter.
<i>Timeout</i>	The wait timeout in milliseconds. The value 0xFFFFFFFF is considered as infinite. Note that you can also kill waiting from another thread using the <a href="#">LvEventKill()</a> function.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.9 SynView Renderer module functions

### Functions

- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvRendererOpen](#) ([LvHStream](#) hStream, [LvHRenderer](#) \*phRenderer)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvRendererClose](#) ([LvHRenderer](#) \*phRenderer)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvRendererSetWindow](#) ([LvHRenderer](#) hRenderer, void \*pDisplay, int64\_t hWindow)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvRendererCanDisplayImage](#) ([LvHRenderer](#) hRenderer, [LvHBuffer](#) hBuffer, uint32\_t RenderFlags)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvRendererDisplayImage](#) ([LvHRenderer](#) hRenderer, [LvHBuffer](#) hBuffer, uint32\_t RenderFlags)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvRendererRepaint](#) ([LvHRenderer](#) hRenderer, uint32\_t RenderFlags)

### 5.9.1 Detailed Description

### 5.9.2 Function Documentation

#### 5.9.2.1 LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvRendererCanDisplayImage](#) ( [LvHRenderer](#) hRenderer, [LvHBuffer](#) hBuffer, uint32\_t RenderFlags )

Checks, if the image can be displayed. Namely the possibility to convert the image to desired display pixel format is checked.

#### Parameters

<i>hRenderer</i>	A handle to the Renderer module, obtained from the <a href="#">LvRendererOpen()</a> function.
<i>hBuffer</i>	The buffer to be displayed.
<i>RenderFlags</i>	Zero or a combination of <a href="#">LvRenderFlags</a> .

#### Returns

Returns the [LvStatus](#) value; the value LVSTATUS\_OK indicates the display is possible, the value LVSTATUS\_CANNOT\_DISPLAY indicates impossibility of pixel format conversion or a misconfiguration of the renderer. See [LvStatus definitions](#).

#### 5.9.2.2 LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvRendererClose](#) ( [LvHRenderer](#) \* phRenderer )

Closes the Renderer module.

#### Parameters

<i>phRenderer</i>	A handle to the Renderer module, obtained from the <a href="#">LvRendererOpen()</a> function.
-------------------	---

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.9.2.3 LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvRendererDisplayImage](#) ( [LvHRenderer](#) hRenderer, [LvHBuffer](#) hBuffer, uint32\_t RenderFlags )

Displays the image. The image display mode is set by Renderer features, see [LvRendererFtr](#).

## Parameters

<i>hRenderer</i>	A handle to the Renderer module, obtained from the <a href="#">LvRendererOpen()</a> function.
<i>hBuffer</i>	The buffer to be displayed.
<i>RenderFlags</i>	Zero or a combination of <a href="#">LvRenderFlags</a> .

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.9.2.4 LV\_EXTC LV\_DLLIMPORT LvStatus LvRendererOpen ( LvHStream hStream, LvHRenderer \* phRenderer )

Opens the Renderer module for image display. The renderer attempts to load the sv.synview.display library. In case of SynView installation in an operating system without possibility to graphically display (for example Linux without XWindows), the load of this library fails and the calls to Renderer functions will return errors.

## Parameters

<i>hStream</i>	A handle to the Stream module, obtained from the <a href="#">LvStreamOpen()</a> function.
<i>phRenderer</i>	In this parameter the handle to the renderer is returned.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.9.2.5 LV\_EXTC LV\_DLLIMPORT LvStatus LvRendererRepaint ( LvHRenderer hRenderer, uint32\_t RenderFlags )

Repaints the contents of the display window. In order to be able to repaint, all images to be displayed must be still held by the application, i.e. must not be returned to the input buffer pool. See also [LvStream\\_LvPostponeQueue←Buffers](#) feature. A typical usage of this function is in the WM\_PAINT handler in a Windows application.

## Parameters

<i>hRenderer</i>	A handle to the Renderer module, obtained from the <a href="#">LvRendererOpen()</a> function.
<i>RenderFlags</i>	Zero or a combination of <a href="#">LvRenderFlags</a> .

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.9.2.6 LV\_EXTC LV\_DLLIMPORT LvStatus LvRendererSetWindow ( LvHRenderer hRenderer, void \* pDisplay, int64\_t hWindow )

Sets the target window, in which the renderer has to display. Note that the application itself assure any repainting (when the window need to be repainted due to a movement of overlapping) - use [LvRendererRepaint\(\)](#) in such case.

## Parameters

<i>hRenderer</i>	A handle to the Renderer module, obtained from the <a href="#">LvRendererOpen()</a> function.
<i>pDisplay</i>	Pointer to the display. It is defined as void* in order to make SynView header files independent on XWindows.

---

<i>hWindow</i>	Handle to the window.
----------------	-----------------------

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.10 SynView Feature control functions

### Functions

- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetNumFeatures](#) ([LvHModule](#) hModule, [LvEnum](#) FeatureGroup, [uint32\\_t](#) \*pNumItems)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetFeatureAt](#) ([LvHModule](#) hModule, [LvEnum](#) FeatureGroup, [uint32\\_t](#) Index, [LvFeature](#) \*pItem, [uint32\\_t](#) \*pLevel)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetFeatureByName](#) ([LvHModule](#) hModule, [LvEnum](#) FeatureGroup, [const char](#) \*pName, [LvFeature](#) \*pItem)
- LV\_EXTC LV\_DLLIMPORT [uint32\\_t](#) [LvIsImplemented](#) ([LvHModule](#) hModule, [LvFeature](#) Feature)
- LV\_EXTC LV\_DLLIMPORT [uint32\\_t](#) [LvIsImplementedByName](#) ([LvHModule](#) hModule, [LvEnum](#) FeatureGroup, [const char](#) \*pName)
- LV\_EXTC LV\_DLLIMPORT [uint32\\_t](#) [LvIsImplementedEnumEntry](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [LvEnum](#) EnumEntry)
- LV\_EXTC LV\_DLLIMPORT [uint32\\_t](#) [LvIsAvailable](#) ([LvHModule](#) hModule, [LvFeature](#) Feature)
- LV\_EXTC LV\_DLLIMPORT [uint32\\_t](#) [LvIsAvailableByName](#) ([LvHModule](#) hModule, [LvEnum](#) FeatureGroup, [const char](#) \*pName)
- LV\_EXTC LV\_DLLIMPORT [uint32\\_t](#) [LvIsAvailableEnumEntry](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [LvEnum](#) EnumEntry)
- LV\_EXTC LV\_DLLIMPORT [uint32\\_t](#) [LvIsReadable](#) ([LvHModule](#) hModule, [LvFeature](#) Feature)
- LV\_EXTC LV\_DLLIMPORT [uint32\\_t](#) [LvIsWritable](#) ([LvHModule](#) hModule, [LvFeature](#) Feature)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetType](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [LvEnum](#) \*pFtrType, [LvEnum](#) \*pFtrGui, [LvEnum](#) \*pFtrGroup)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetBool](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [int32\\_t](#) \*pValue)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvSetBool](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [int32\\_t](#) Value)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetInt32](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [int32\\_t](#) \*pValue)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvSetInt32](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [int32\\_t](#) Value)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetInt32Range](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [int32\\_t](#) \*pMinValue, [int32\\_t](#) \*pMaxValue, [int32\\_t](#) \*pIncrement)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetInt64](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [int64\\_t](#) \*pValue)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvSetInt64](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [int64\\_t](#) Value)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetInt64Range](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [int64\\_t](#) \*pMinValue, [int64\\_t](#) \*pMaxValue, [int64\\_t](#) \*pIncrement)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetInt](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [int64\\_t](#) \*pValue)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvSetInt](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [int64\\_t](#) Value)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetIntRange](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [int64\\_t](#) \*pMinValue, [int64\\_t](#) \*pMaxValue, [int64\\_t](#) \*pIncrement)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetFloat](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [double](#) \*pValue)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvSetFloat](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [double](#) Value)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetFloatRange](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [double](#) \*pMinValue, [double](#) \*pMaxValue, [double](#) \*pIncrement)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetString](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [char](#) \*pValue, [size\\_t](#) Size)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetStringSize](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [size\\_t](#) \*pSize)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvSetString](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [const char](#) \*pValue)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetBuffer](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [void](#) \*pBuffer, [size\\_t](#) Size)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetBufferSize](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [size\\_t](#) \*pSize)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvSetBuffer](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [void](#) \*pBuffer, [size\\_t](#) Size)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetPtr](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [void](#) \*\*ppValue)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvSetPtr](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [void](#) \*pValue)

- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetEnum](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [LvEnum](#) \*p↔ Value)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvSetEnum](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [LvEnum](#) Value)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetEnumStr](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, char \*p↔ SymbolicName, size\_t Size)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvSetEnumStr](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, const char \*pSymbolicName)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetEnumValByStr](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, const char \*pSymbolicName, [LvEnum](#) \*pValue, [LvEnum](#) \*pFtrAccess)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetEnumStrByVal](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [LvEnum](#) Value, char \*pSymbolicName, size\_t SymbolicNameSize, [LvEnum](#) \*pFtrAccess)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvCmdExecute](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, uint32\_t↔ Timeout)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvCmdIsDone](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, uint32\_t \*p↔ IsDone)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetAccess](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [LvEnum](#) \*p↔ FtrAccess)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetVisibility](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [LvEnum](#) \*p↔ FtrVisibility)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetInfo](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [LvEnum](#) FtrInfo, int32\_t \*pInfo, int32\_t Param)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetInfoStr](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [LvEnum](#) FtrInfo, char \*pInfoStr, size\_t Size, int32\_t Param)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvGetInfoStrSize](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [LvEnum](#) FtrInfo, size\_t \*pSize, int32\_t Param)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvRegisterFeatureCallback](#) ([LvHModule](#) hModule, [LvFeature](#) Feature, [LvFeatureCallbackFunc](#) pFunction, void \*pUserParam, void \*pFeatureParam)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvStartPollingThread](#) ([LvHModule](#) hModule, uint32\_t PollingTime, int32\_t PollChildren)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvStopPollingThread](#) ([LvHModule](#) hModule)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvPoll](#) ([LvHModule](#) hModule)

### 5.10.1 Detailed Description

### 5.10.2 Function Documentation

#### 5.10.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvCmdExecute ( LvHModule hModule, LvFeature Feature, uint32\_t Timeout )

Executes a command.

#### Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>Timeout</i>	If greater than 0, the <a href="#">LvCmdIsDone()</a> is called in a loop to wait for the command completion, until the <a href="#">LvCmdIsDone()</a> returns true or the Timeout (in milliseconds) expires. If set to 0, no wait is done.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvCmdlsDone ( LvHModule *hModule*, LvFeature *Feature*, uint32\_t \*  
*plsDone* )

Checks if the command execution has completed.



## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>plsDone</i>	In this parameter is returned 1, if the command is completed, otherwise 0.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.3 `LV_EXTC LV_DLLIMPORT LvStatus LvGetAccess ( LvHModule hModule, LvFeature Feature, LvEnum * pFtrAccess )`

Gets the access mode of the feature.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pFtrAccess</i>	The access is returned in this parameter. One of the <a href="#">LvFtrAccess</a> .

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.4 `LV_EXTC LV_DLLIMPORT LvStatus LvGetBool ( LvHModule hModule, LvFeature Feature, int32_t * pValue )`

Gets a Boolean value (as 32-bit integer).

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pValue</i>	The bool value (as 32-bit integer) is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.5 `LV_EXTC LV_DLLIMPORT LvStatus LvGetBuffer ( LvHModule hModule, LvFeature Feature, void * pBuffer, size_t Size )`

Gets a block of data.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.

<i>pBuffer</i>	Pointer to a buffer, to which the data will be stored.
<i>Size</i>	Size of the buffer.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.10.2.6 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetBufferSize ( LvHModule hModule, LvFeature Feature, size\_t \* pSize )

Gets the block data size.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pSize</i>	The needed size of the buffer is returned in this parameter.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.10.2.7 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetEnum ( LvHModule hModule, LvFeature Feature, LvEnum \* pValue )

Gets the SynView constant for the enumeration entry, if exists. If does not exist, you must work with the string enumeration entry value.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pValue</i>	SynView constant for the enum entry is returned in this parameter.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.10.2.8 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetEnumStr ( LvHModule hModule, LvFeature Feature, char \* pSymbolicName, size\_t Size )

Gets the enumeration entry as string (symbolic name). It is not possible to get the needed size for this single feature, instead, it is possible to get the maximum size of the all enum values of this feature, by the [LvGetInfo\(LvFtrInfo\\_EnumEntryNameMaxSize\)](#) function.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.

<i>pSymbolicName</i>	A pointer to a string buffer, where the symbolic name will be returned.
<i>Size</i>	Size of the buffer.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.10.2.9 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetEnumStrByVal ( LvHModule hModule, LvFeature Feature, LvEnum Value, char \* pSymbolicName, size\_t SymbolicNameSize, LvEnum \* pFtrAccess )

Returns a string symbolic name of the enum entry for the SynView constant.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>Value</i>	The SynView constant for the enum entry.
<i>pSymbolicName</i>	Pointer to string buffer, where the symbolic name is returned. Can be NULL.
<i>SymbolicNameSize</i>	Size of pSymbolicName buffer.
<i>pFtrAccess</i>	The access mode of the enum entry is returned in this parameter - one of <a href="#">LvFtrAccess</a> . Can be NULL.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.10.2.10 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetEnumValByStr ( LvHModule hModule, LvFeature Feature, const char \* pSymbolicName, LvEnum \* pValue, LvEnum \* pFtrAccess )

Gets the SynView constant for the enumeration entry, if exists.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pSymbolicName</i>	A string with symbolic name of the enum entry.
<i>pValue</i>	The SynView constant for the enum entry is returned in this parameter. If the SynView constant does not exist for this enumeration entry, 0 is returned (no error is indicated).
<i>pFtrAccess</i>	The feature access is returned in this parameter - one of <a href="#">GroupSynView_LvFtrAccess</a> . Can be NULL.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.10.2.11 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetFeatureAt ( LvHModule hModule, LvEnum FeatureGroup, uint32\_t Index, LvFeature \* pItem, uint32\_t \* pLevel )

Returns the feature ID at specified position. Can be used to iterate all the features in a list.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>FeatureGroup</i>	One of the <a href="#">LvFtrGroup</a> .
<i>Index</i>	Zero based index of the feature in the list.
<i>pItem</i>	Feature ID is returned in this parameter.
<i>pLevel</i>	Feature Level expressing its position in the tree is returned in this parameter. The base level has value 1.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.12 `LV_EXTC LV_DLLIMPORT LvStatus LvGetFeatureByName ( LvHModule hModule, LvEnum FeatureGroup, const char * pName, LvFeature * pItem )`

Returns a feature ID based on the feature name. This function is a substantial function for the generic approach to the feature - by this function you can get the ID of any existing feature, that means also for those, for which a SynView constant is not defined. Be sure to check the success of this function - if the feature is not mandatory, it may not exist.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>FeatureGroup</i>	One of the <a href="#">LvFtrGroup</a> .
<i>pName</i>	Name of the feature.
<i>pItem</i>	Feature ID is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.13 `LV_EXTC LV_DLLIMPORT LvStatus LvGetFloat ( LvHModule hModule, LvFeature Feature, double * pValue )`

Gets a float value.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pValue</i>	The float value is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.14 `LV_EXTC LV_DLLIMPORT LvStatus LvGetFloatRange ( LvHModule hModule, LvFeature Feature, double * pMinValue, double * pMaxValue, double * pIncrement )`

Returns a range of a float feature.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pMinValue</i>	The minimum value is returned in this parameter. Can be NULL.
<i>pMaxValue</i>	The maximum value is returned in this parameter. Can be NULL.
<i>pIncrement</i>	The increment value is returned in this parameter. If the increment is not defined, 0 is returned. Can be NULL.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.15 `LV_EXTC LV_DLLIMPORT LvStatus LvGetInfo ( LvHModule hModule, LvFeature Feature, LvEnum FtrInfo, int32_t * pInfo, int32_t Param )`

Gets an info in form of a 32-bit integer value.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>FtrInfo</i>	One of the <a href="#">LvFtrInfo</a> .
<i>pInfo</i>	The value is returned in this parameter.
<i>Param</i>	Additional parameter, required by some types of info.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.16 `LV_EXTC LV_DLLIMPORT LvStatus LvGetInfoStr ( LvHModule hModule, LvFeature Feature, LvEnum FtrInfo, char * pInfoStr, size_t Size, int32_t Param )`

Gets an info in form of a string value.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>FtrInfo</i>	One of the <a href="#">LvFtrInfo</a> .
<i>pInfoStr</i>	The string value is returned in this parameter.
<i>Size</i>	Size of the buffer (to which pInfoStr points).
<i>Param</i>	Additional parameter, required by some types of info.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.17 `LV_EXTC LV_DLLIMPORT LvStatus LvGetInfoStrSize ( LvHModule hModule, LvFeature Feature, LvEnum FtrInfo, size_t * pSize, int32_t Param )`

Gets a buffer size needed for an info in form of a string value.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>FtrInfo</i>	One of the <a href="#">LvFtrInfo</a> .
<i>pSize</i>	Size of the buffer is returned in this parameter.
<i>Param</i>	Additional parameter, required by some types of info.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.10.2.18 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetInt ( LvHModule hModule, LvFeature Feature, int64\_t \* pValue )

Gets a 64-bit integer value.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pValue</i>	The integer value is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

This function is equal to the [LvGetInt64\(\)](#) function.

## 5.10.2.19 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetInt32 ( LvHModule hModule, LvFeature Feature, int32\_t \* pValue )

Gets a 32-bit integer value.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pValue</i>	The integer value is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

The value is internally kept always as a 64-bit value; the functions for setting and getting a 32-bit value are provided just for convenience.

## 5.10.2.20 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetInt32Range ( LvHModule hModule, LvFeature Feature, int32\_t \* pMinValue, int32\_t \* pMaxValue, int32\_t \* pIncrement )

Returns a range and increment of an 32-bit integer feature.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pMinValue</i>	The minimum value is returned in this parameter. Can be NULL.
<i>pMaxValue</i>	The maximum value is returned in this parameter. Can be NULL.
<i>pIncrement</i>	The increment value is returned in this parameter. Can be NULL.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

The value is internally kept always as a 64-bit value; the functions for setting and getting a 32-bit value are provided just for convenience.

## 5.10.2.21 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetInt64 ( LvHModule hModule, LvFeature Feature, int64\_t \* pValue )

Gets a 64-bit integer value.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pValue</i>	The integer value is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

This function is equal to the [LvGetInt\(\)](#) function.

## 5.10.2.22 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetInt64Range ( LvHModule hModule, LvFeature Feature, int64\_t \* pMinValue, int64\_t \* pMaxValue, int64\_t \* pIncrement )

Returns a range and increment of an 64-bit integer feature.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pMinValue</i>	The minimum value is returned in this parameter. Can be NULL.
<i>pMaxValue</i>	The maximum value is returned in this parameter. Can be NULL.
<i>pIncrement</i>	The increment value is returned in this parameter. Can be NULL.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

This function is equal to the [LvGetIntRange\(\)](#) function.

5.10.2.23 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetIntRange ( LvHModule *hModule*, LvFeature *Feature*, int64\_t \* *pMinValue*, int64\_t \* *pMaxValue*, int64\_t \* *pIncrement* )

Returns a range and increment of an 64-bit integer feature.



## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pMinValue</i>	The minimum value is returned in this parameter. Can be NULL.
<i>pMaxValue</i>	The maximum value is returned in this parameter. Can be NULL.
<i>pIncrement</i>	The increment value is returned in this parameter. Can be NULL.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

This function is equal to the [LvGetInt64Range\(\)](#) function.

#### 5.10.2.24 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetNumFeatures ( LvHModule hModule, LvEnum FeatureGroup, uint32\_t \* pNumItems )

Returns a number of features for specified group. This is useful for building a list of all available features (like the tree in lvexplorer).

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>FeatureGroup</i>	One of the <a href="#">LvFtrGroup</a> .
<i>pNumItems</i>	The number of features is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.10.2.25 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetPtr ( LvHModule hModule, LvFeature Feature, void \*\* ppValue )

Gets a pointer.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>ppValue</i>	The pointer is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.10.2.26 LV\_EXTC LV\_DLLIMPORT LvStatus LvGetString ( LvHModule hModule, LvFeature Feature, char \* pValue, size\_t Size )

Gets a string value. If you need first to get the string size, use the [LvGetStringSize\(\)](#) function.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pValue</i>	Pointer to a null-terminated string buffer.
<i>Size</i>	Size of the buffer.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.27 `LV_EXTC LV_DLLIMPORT LvStatus LvGetStringSize ( LvHModule hModule, LvFeature Feature, size_t * pSize )`

Gets a buffer size needed for a string.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pSize</i>	Size of the buffer (including space for terminating zero) is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.28 `LV_EXTC LV_DLLIMPORT LvStatus LvGetType ( LvHModule hModule, LvFeature Feature, LvEnum * pFtrType, LvEnum * pFtrGui, LvEnum * pFtrGroup )`

Returns the feature type, GUI representation and group.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pFtrType</i>	The feature type is returned in this parameter. The returned value is one of the <a href="#">LvFtrType</a> . Can be NULL.
<i>pFtrGui</i>	The feature GUI representation is returned in this parameter. The returned value is one of the <a href="#">LvFtrGui</a> . Can be NULL.
<i>pFtrGroup</i>	The feature group, to which the feature belongs. The returned value is one of the <a href="#">LvFtrGroup</a> . Can be NULL.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.29 `LV_EXTC LV_DLLIMPORT LvStatus LvGetVisibility ( LvHModule hModule, LvFeature Feature, LvEnum * pFtrVisibility )`

Gets the feature visibility (beginner-expert-guru).

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pFtrVisibility</i>	The visibility is returned in this parameter. One of the <a href="#">LvFtrVisibility</a> .

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.10.2.30 LV\_EXTC LV\_DLLIMPORT uint32\_t LvsAvailable ( LvHModule hModule, LvFeature Feature )

A helper function, allowing simply to determine, if a feature is available. It is a wrapper around the [LvGetAccess\(\)](#) function.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.

## Returns

If the feature is available, returns 1, otherwise 0.

## 5.10.2.31 LV\_EXTC LV\_DLLIMPORT uint32\_t LvsAvailableByName ( LvHModule hModule, LvEnum FeatureGroup, const char \* pName )

A helper function, allowing simply to determine, if a feature is available. It is a wrapper around the [LvGetAccess\(\)](#) and [LvGetFeatureByName\(\)](#) functions.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>FeatureGroup</i>	One of the <a href="#">LvFtrGroup</a> .
<i>pName</i>	Name of the feature.

## Returns

If the feature is available, returns 1, otherwise 0.

## 5.10.2.32 LV\_EXTC LV\_DLLIMPORT uint32\_t LvsAvailableEnumEntry ( LvHModule hModule, LvFeature Feature, LvEnum EnumEntry )

A helper function, allowing simply to determine, if an enum entry of an enum feature is available.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.

<i>EnumEntry</i>	The SynView constant for the enum entry.
------------------	--

**Returns**

If the enum entry is available, returns 1, otherwise 0.

#### 5.10.2.33 LV\_EXTC LV\_DLLIMPORT uint32\_t LvlIsImplemented ( LvHModule *hModule*, LvFeature *Feature* )

A helper function, allowing simply to determine, if a feature is implemented. It is a wrapper around the [LvGetAccess\(\)](#) function.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.

**Returns**

If the feature is implemented, returns 1, otherwise 0.

#### 5.10.2.34 LV\_EXTC LV\_DLLIMPORT uint32\_t LvlIsImplementedByName ( LvHModule *hModule*, LvEnum *FeatureGroup*, const char \* *pName* )

A helper function, allowing simply to determine, if a feature is implemented. It is a wrapper around the [LvGetAccess\(\)](#) and [LvGetFeatureByName\(\)](#) functions.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>FeatureGroup</i>	One of the <a href="#">LvFtrGroup</a> .
<i>pName</i>	Name of the feature.

**Returns**

If the feature is implemented, returns 1, otherwise 0.

#### 5.10.2.35 LV\_EXTC LV\_DLLIMPORT uint32\_t LvlIsImplementedEnumEntry ( LvHModule *hModule*, LvFeature *Feature*, LvEnum *EnumEntry* )

A helper function, allowing simply to determine, if an enum entry of an enum feature is implemented.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>EnumEntry</i>	The SynView constant for the enum entry.

**Returns**

If the enum entry is implemented, returns 1, otherwise 0.

#### 5.10.2.36 LV\_EXTC LV\_DLLIMPORT uint32\_t LvlIsReadable ( LvHModule *hModule*, LvFeature *Feature* )

A helper function, allowing simply to determine, if a feature is readable. It is a wrapper around the [LvGetAccess\(\)](#) function.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.

## Returns

If the feature is readable, returns 1, otherwise 0.

## 5.10.2.37 LV\_EXTC LV\_DLLIMPORT uint32\_t LvsWritable ( LvHModule hModule, LvFeature Feature )

A helper function, allowing simply to determine, if a feature is writable. It is a wrapper around the [LvGetAccess\(\)](#) function.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.

## Returns

If the feature is writable, returns 1, otherwise 0.

## 5.10.2.38 LV\_EXTC LV\_DLLIMPORT LvStatus LvPoll ( LvHModule hModule )

Polls all the non-cached features of the module. If the feature polling interval expires, the value is read and the feature callback is called.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream or Renderer module.
----------------	---

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.10.2.39 LV\_EXTC LV\_DLLIMPORT LvStatus LvRegisterFeatureCallback ( LvHModule hModule, LvFeature Feature, LvFeatureCallbackFunc pFunction, void \* pUserParam, void \* pFeatureParam )

Registers or unregisters a callback function for the feature. This callback is produced by GenApi when a feature changes its value or status. The application should process this callback fast. Note that the callback can be called also from another thread - see [LvEventType\\_FeatureDevEvent](#). Important note: The feature callback function should never set any other feature. Doing so can lead to recursions, which would be probably hard to diagnose and could cause unexpected behavior.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.

<i>pFunction</i>	The callback function in the form of <a href="#">LvFeatureCallbackFunc</a> . If you want to unregister the function, use NULL at this parameter.
<i>pUserParam</i>	User parameter, which will be passed to each callback call.
<i>pFeatureParam</i>	Second user parameter, which will be passed to each callback call.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.10.2.40 LV\_EXTC LV\_DLLIMPORT LvStatus LvSetBool ( LvHModule *hModule*, LvFeature *Feature*, int32\_t *Value* )

Sets a Boolean value.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>Value</i>	Value to be set (in form of 32_bit integer value).

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.10.2.41 LV\_EXTC LV\_DLLIMPORT LvStatus LvSetBuffer ( LvHModule *hModule*, LvFeature *Feature*, void \* *pBuffer*, size\_t *Size* )

Sets a block of data.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pBuffer</i>	Pointer to the data.
<i>Size</i>	Size of the data.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.10.2.42 LV\_EXTC LV\_DLLIMPORT LvStatus LvSetEnum ( LvHModule *hModule*, LvFeature *Feature*, LvEnum *Value* )

Sets the enumeration entry by the SynView constant. If the SynView constant is not defined for the feature, then use [LvSetEnumStr\(\)](#) to set the enum entry by a string.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.

<i>Value</i>	SynView constant for the requested enumeration entry.
--------------	---

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.43 `LV_EXTC LV_DLLIMPORT LvStatus LvSetEnumStr ( LvHModule hModule, LvFeature Feature, const char * pSymbolicName )`

Sets enumeration entry by its string symbolic name.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pSymbolicName</i>	A pointer to a string with the symbolic name of the enumeration entry.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.44 `LV_EXTC LV_DLLIMPORT LvStatus LvSetFloat ( LvHModule hModule, LvFeature Feature, double Value )`

Sets a float value.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>Value</i>	The value to be set.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.45 `LV_EXTC LV_DLLIMPORT LvStatus LvSetInt ( LvHModule hModule, LvFeature Feature, int64_t Value )`

Sets a 64-bit integer value.

**Parameters**

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>Value</i>	Value to be set.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**Note**

This function is equal to the [LvSetInt64\(\)](#) function.

5.10.2.46 LV\_EXTC LV\_DLLIMPORT LvStatus LvSetInt32 ( LvHModule *hModule*, LvFeature *Feature*, int32\_t *Value* )

Sets a 32-bit value.



## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>Value</i>	Value to be set.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

The value is internally kept always as a 64-bit value; the functions for setting and getting a 32-bit value are provided just for convenience.

5.10.2.47 LV\_EXTC LV\_DLLIMPORT LvStatus LvSetInt64 ( LvHModule *hModule*, LvFeature *Feature*, int64\_t *Value* )

Sets a 64-bit integer value.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>Value</i>	Value to be set.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

This function is equal to the [LvSetInt\(\)](#) function.

5.10.2.48 LV\_EXTC LV\_DLLIMPORT LvStatus LvSetPtr ( LvHModule *hModule*, LvFeature *Feature*, void \* *pValue* )

Sets a pointer.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pValue</i>	The pointer to be set.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.49 LV\_EXTC LV\_DLLIMPORT LvStatus LvSetString ( LvHModule *hModule*, LvFeature *Feature*, const char \* *pValue* )

Sets a string value.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvGetFeatureByName()</a> function.
<i>pValue</i>	The string value (null-terminated).

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.50 LV\_EXTC LV\_DLLIMPORT LvStatus LvStartPollingThread ( LvHModule *hModule*, uint32\_t *PollingTime*, int32\_t *PollChildren* )

Starts a thread, which in a loop polls the non-cached features. If the feature polling interval expires, the value is read and the feature callback is called.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream or Renderer module.
<i>PollingTime</i>	A time in milliseconds between 2 calls to poll the features.
<i>PollChildren</i>	If set to true, also the features in all children modules are polled. For example, if your application uses only one System module, then it is a parent of all other modules, so the polling will be propagated to all modules from a single thread. If a module has started own polling thread, then it is excluded from the propagating.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.10.2.51 LV\_EXTC LV\_DLLIMPORT LvStatus LvStopPollingThread ( LvHModule *hModule* )

Stops the polling thread. See [LvStartPollingThread\(\)](#) for details.

## Parameters

<i>hModule</i>	A handle to the System, Interface, Device, Stream or Renderer module.
----------------	---

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.11 SynView Firmware update functions

### Functions

- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvFwGetFilePattern](#) ([LvHModule](#) hModule, uint32\_t Which, char \*pFilePattern, size\_t Size)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvFwLoad](#) ([LvHModule](#) hModule, uint32\_t Which, const char \*pFilePath)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvFwGetLoadStatus](#) ([LvHModule](#) hModule, uint32\_t Which, uint32\_t \*pCurrentByteCount, uint32\_t \*pIsLoading)

#### 5.11.1 Detailed Description

#### 5.11.2 Function Documentation

##### 5.11.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvFwGetFilePattern ( LvHModule hModule, uint32\_t Which, char \* pFilePattern, size\_t Size )

Returns the file name mask (with wildcard characters), for searching the file with the appropriate firmware update. The files with the FW update have in their names coded the hardware IDs, so using this mask (for example in a filter in a file open dialog box) assures the file appropriate for this device is used.

##### Parameters

<i>hModule</i>	A handle to the Device module.
<i>Which</i>	An ID specific for a hardware. Discussed in the SynView User's Guide.
<i>pFilePattern</i>	In this parameter the file pattern is returned.
<i>Size</i>	Size of the buffer (to which the pFilePattern points).

##### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

##### 5.11.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvFwGetLoadStatus ( LvHModule hModule, uint32\_t Which, uint32\_t \* pCurrentByteCount, uint32\_t \* pIsLoading )

Returns the byte count and whether the loading is still in progress.

##### Parameters

<i>hModule</i>	A handle to the Device module.
<i>Which</i>	An ID specific for a hardware. Discussed in the SynView User's Guide.
<i>pCurrentByteCount</i>	Returns number of bytes transferred so far.
<i>pIsLoading</i>	Returns 1 if the loading is still in progress.

##### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

##### 5.11.2.3 LV\_EXTC LV\_DLLIMPORT LvStatus LvFwLoad ( LvHModule hModule, uint32\_t Which, const char \* pFilePath )

Loads the firmware from a file to the hardware. It can be very long process (taking minutes) and this functions blocks the thread during this process. It is recommended to check the load status from another thread using the [LvFwGetLoadStatus\(\)](#) function.

**Parameters**

<i>hModule</i>	A handle to the Device module.
<i>Which</i>	An ID specific for a hardware. Discussed in the SynView User's Guide.
<i>pFilePath</i>	File specification, with full path.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.12 SynView C++ API functions

### Modules

- [SynView LvLibrary methods](#)
- [SynView LvSystem methods](#)
- [SynView LvInterface methods](#)
- [SynView LvDevice methods](#)
- [SynView LvStream methods](#)
- [SynView LvBuffer methods](#)
- [SynView LvEvent methods](#)
- [SynView LvRenderer methods](#)
- [SynView LvModule methods](#)

### 5.12.1 Detailed Description

## 5.13 SynView LvLibrary methods

### Functions

- static uint32\_t LvLibrary::GetVersion ()
- static LvStatus LvLibrary::OpenLibrary ()
- static LvStatus LvLibrary::CloseLibrary ()
- static void LvLibrary::GetErrorMessage (LvStatus Error, char \*pMessage, size\_t Size)
- static std::string LvLibrary::GetErrorMessage (LvStatus Error)
- static void LvLibrary::GetLastErrorMessage (char \*pMessage, size\_t Size)
- static std::string LvLibrary::GetLastErrorMessage ()
- static void LvLibrary::Log (const char \*pLogMessage)
- static LvStatus LvLibrary::GetLibInfo (LvEnum Info, int32\_t \*pInfo, int32\_t Param=0)
- static LvStatus LvLibrary::GetLibInfoStr (LvEnum Info, char \*pInfoStr, size\_t Size, int32\_t Param=0)
- static LvStatus LvLibrary::GetLibInfoStrSize (LvEnum Info, size\_t \*pSize, int32\_t Param=0)
- static LvStatus LvLibrary::GetLibInfoStr (LvEnum Info, std::string &sInfo, int32\_t Param=0)
- static LvStatus LvLibrary::UpdateSystemList ()
- static LvStatus LvLibrary::GetNumberOfSystems (uint32\_t \*pNumberOfSystems)
- static LvStatus LvLibrary::GetSystemId (uint32\_t Index, char \*pSystemId, size\_t Size)
- static LvStatus LvLibrary::GetSystemIdSize (uint32\_t Index, size\_t \*pSize)
- static LvStatus LvLibrary::GetSystemId (uint32\_t Index, std::string &sSystemId)
- static void LvLibrary::SetThrowErrorEnable (bool bEnable)

#### 5.13.1 Detailed Description

#### 5.13.2 Function Documentation

##### 5.13.2.1 static LvStatus LvLibrary::CloseLibrary ( ) [static]

Closes the SynView library. This must be performed before you exit your application. Be sure to close first all dependent modules (System). If you are using SynView in a Windows DLL, avoid calling this in Windows DllMain() function - for proper functionality this function must be called when the application or DLL is still fully functional, which is not the case of PROCESS\_DETACH in the DllMain(). If you have called LvLibrary::OpenLibrary() multiple times, you must balance it by the same number of calls of this function. Only the last call actually does the uninitialization. IMPORTANT: The library must not be opened again once it was already uninitialized.

#### Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

##### 5.13.2.2 static void LvLibrary::GetErrorMessage ( LvStatus Error, char \* pMessage, size\_t Size ) [static]

Returns a short description of the error. Note that only some of the errors are suitable for direct display to the user, many error values indicate states which are understandable to the programmer, but may not be understandable to the end user.

#### Parameters

<i>Error</i>	The error code (the return value of most SynView functions).
<i>pMessage</i>	Pointer to the text buffer.

<i>Size</i>	Size of the buffer.
-------------	---------------------

See also

[LvStatus definitions.](#)

#### 5.13.2.3 static std::string LvLibrary::GetErrorMessage ( LvStatus *Error* ) [static]

Returns a short description of the error. Note that only some of the errors are suitable for direct display to the user, many error values indicate states which are understandable to the programmer, but may not be understandable to the end user.

Parameters

<i>Error</i>	The error code (the return value of most SynView functions).
--------------	--

Returns

Error message in std::string.

See also

[LvStatus definitions.](#)

#### 5.13.2.4 static void LvLibrary::GetLastErrorMessage ( char \* *pMessage*, size\_t *Size* ) [static]

Returns more detailed description of the last error, which happened in the thread from which this function was called. As the info is recorded inside SynView for each error, the description provides more detailed info, including the name of the function, in which the error happened, and possibly more diagnostic info. The difference to [LvLibrary::GetErrorMessage\(\)](#) is that [LvLibrary::GetErrorMessage\(\)](#) returns a static string from a numbered table of errors while this function returns additionally info recorded at the time the error happened. If a function returns LVSTATUS\_OK, it does not reset this error message (for speed reasons) so the correct approach is to get the error number as the function return value and if this return value is not LVSTATUS\_OK, then you can get more info about the error using this function. be sure to call it from the same thread.

Parameters

<i>pMessage</i>	Pointer to the text buffer.
<i>Size</i>	Size of the buffer.

See also

[LvStatus definitions.](#)

#### 5.13.2.5 static std::string LvLibrary::GetLastErrorMessage ( ) [static]

Returns more detailed description of the last error, which happened in the thread from which this function was called. As the info is recorded inside SynView for each error, the description provides more detailed info, including the name of the function, in which the error happened, and possibly more diagnostic info. The difference to [LvLibrary::GetErrorMessage\(\)](#) is that [LvLibrary::GetErrorMessage\(\)](#) returns a static string from a numbered table of errors while this function returns additionally info recorded at the time the error happened. If a function returns LVSTATUS\_OK, it does not reset this error message (for speed reasons) so the correct approach is to get the error number as the function return value and if this return value is not LVSTATUS\_OK, then you can get more info about the error using this function. be sure to call it from the same thread.

**Returns**

Error message in `std::string`.

**See also**

[LvStatus definitions](#).

**5.13.2.6** `static LvStatus LvLibrary::GetLibInfo ( LvEnum Info, int32_t * pInfo, int32_t Param = 0 )` [static]

Gets a general info in form of a 32-bit integer value.

**Parameters**

<i>Info</i>	One of the <a href="#">LvLibInfo</a> values.
<i>pInfo</i>	The value is returned in this parameter.
<i>Param</i>	Additional parameter, required by some types of info.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.13.2.7** `static LvStatus LvLibrary::GetLibInfoStr ( LvEnum Info, char * pInfoStr, size_t Size, int32_t Param = 0 )`  
[static]

Gets a general info in form of a string value.

**Parameters**

<i>Info</i>	One of the <a href="#">LvLibInfo</a> values.
<i>pInfoStr</i>	The string value is returned in this parameter.
<i>Size</i>	Size of the buffer (to which pInfoStr points).
<i>Param</i>	Additional parameter, required by some types of info.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.13.2.8** `static LvStatus LvLibrary::GetLibInfoStr ( LvEnum Info, std::string & sInfo, int32_t Param = 0 )` [static]

Gets a general info in form of a `std::string` value.

**Parameters**

<i>Info</i>	One of the <a href="#">LvLibInfo</a> values.
<i>sInfo</i>	The string value is returned in this parameter.
<i>Param</i>	Additional parameter, required by some types of info.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.13.2.9** `static LvStatus LvLibrary::GetLibInfoStrSize ( LvEnum Info, size_t * pSize, int32_t Param = 0 )` [static]

Gets a buffer size needed for a general info in form of a string value.



## Parameters

<i>Info</i>	One of the <a href="#">LvLibInfo</a> values.
<i>pSize</i>	Size of the buffer is returned in this parameter.
<i>Param</i>	Additional parameter, required by some types of info.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.13.2.10 static LvStatus LvLibrary::GetNumberOfSystems ( uint32\_t \* pNumberOfSystems ) [static]

Returns the number of systems found after the [LvLibrary::UpdateSystemList\(\)](#) call. Typical use of this function is in iterating systems using the [LvLibrary::GetSystemId\(\)](#) function.

## Parameters

<i>pNumberOfSystems</i>	The number of systems found.
-------------------------	------------------------------

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.13.2.11 static LvStatus LvLibrary::GetSystemId ( uint32\_t Index, char \* pSystemId, size\_t Size ) [static]

Returns the string ID of the system at given index. This ID is used in the [LvSystem::Open\(\)](#) function for opening the system.

## Parameters

<i>Index</i>	Zero-based index of the system, a value $\geq 0$ and $<$ number of systems, returned by the <a href="#">LvLibrary::GetNumberOfSystems()</a> function.
<i>pSystemId</i>	Pointer to a string buffer, where the system ID will be placed.
<i>Size</i>	Size of the buffer.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.13.2.12 static LvStatus LvLibrary::GetSystemId ( uint32\_t Index, std::string & sSystemId ) [static]

Returns the string ID of the system at given index. This ID is used in the [LvSystem::Open\(\)](#) function for opening the system.

## Parameters

<i>Index</i>	Zero-based index of the system, a value $\geq 0$ and $<$ number of systems, returned by the <a href="#">LvLibrary::GetNumberOfSystems()</a> function.
<i>sSystemId</i>	String, where the system ID will be placed.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.13.2.13 static LvStatus LvLibrary::GetSystemIdSize ( uint32\_t Index, size\_t \* pSize ) [static]

Returns the size of the string buffer needed to hold the system ID string, including the terminating zero character.

## Parameters

<i>Index</i>	Zero-based index of the system, a value $\geq 0$ and $<$ number of systems, returned by the <a href="#">LvLibrary::GetNumberOfSystems()</a> function.
<i>pSize</i>	Size of the buffer is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.13.2.14 `static uint32_t LvLibrary::GetVersion ( ) [static]`

Returns SynView version.

## Returns

The returned doubleword contains the build version in the low word and the high word is the major version in the upper byte and subversion in the lower byte. For example:

```
uint32_t Version = LvLibrary::GetVersion();
printf("SynView %d.%02d.%03d",
      ((Version >> 24) & 0xFF),
      ((Version >> 16) & 0xFF),
      (Version & 0xFFFF));
```

5.13.2.15 `static void LvLibrary::Log ( const char * pLogMessage ) [static]`

Adds a line to the sv.synview.log. The SynView log is a tool for New Electronic Technology technical support, but in some cases may be useful to put to the log additional info from your code.

## Parameters

<i>pLogMessage</i>	Pointer to the null terminated string with the message.
--------------------	---

5.13.2.16 `static LvStatus LvLibrary::OpenLibrary ( ) [static]`

Opens the SynView library. This must be done before you can use any other SynView function (with the exception of [LvLibrary::GetVersion\(\)](#) and [LvLibrary::GetErrorMessage\(\)](#)). If you are using SynView in Windows DLL, avoid calling this in Windows DllMain() function - for proper functionality this function must be called when the application or DLL is already fully initialized and there are no restrictions about synchronization (DllMain has such restrictions). If you call this function multiple times, you must balance it by the same number of the [LvLibrary::CloseLibrary\(\)](#) calls. Only the first call will actually do the initialization. IMPORTANT: The library must not be opened again once it was already uninitialized.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.13.2.17 `static void LvLibrary::SetThrowErrorEnable ( bool bEnable ) [static]`

Enables/disables the conversion of LvStatus return values (not equal to LVSTATUS\_OK) to C++ exceptions of the [LvException](#) type.

## Parameters

<i>bEnable</i>	Enable/disable the exception throwing.
----------------	--

5.13.2.18 `static LvStatus LvLibrary::UpdateSystemList( ) [static]`

Updates the list of systems available. This function must be called before iterating through the systems by the [LvLibrary::GetNumberOfSystems\(\)](#) and [LvLibrary::GetSystemId\(\)](#) functions. The systems are physically represented by GenTL libraries available in the operating systems, this call searches for them in standard locations. See also the description of the `sv.synview.ini` file in the SynView User's Guide. Note that this function is seldom needed, most applications will work with the default system (see [LvSystem::Open\(\)](#) for details).

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.14 SynView LvSystem methods

### Functions

- static [LvStatus LvSystem::Open](#) (const char \*pSystemId, [LvSystem](#) \*&pSystem)
- static [LvStatus LvSystem::Close](#) ([LvSystem](#) \*&pSystem)
- [LvStatus LvSystem::UpdateInterfaceList](#) (uint32\_t Timeout=0xFFFFFFFF)
- [LvStatus LvSystem::GetNumberOfInterfaces](#) (uint32\_t \*pNumberOfInterfaces)
- [LvStatus LvSystem::GetInterfaceId](#) (uint32\_t Index, char \*pInterfaceId, size\_t Size)
- [LvStatus LvSystem::GetInterfaceIdSize](#) (uint32\_t Index, size\_t \*pSize)
- [LvStatus LvSystem::GetInterfaceId](#) (uint32\_t Index, std::string &sInterfaceId)
- [LvStatus LvSystem::FindInterface](#) ([LvFindBy](#) FindBy, const char \*pFindStr, char \*pInterfaceId, size\_t Size)
- [LvStatus LvSystem::FindInterface](#) ([LvFindBy](#) FindBy, const char \*pFindStr, std::string &sInterfaceId)
- [LvHSystem LvSystem::GetHandle](#) ()
- [LvStatus LvSystem::OpenInterface](#) (const char \*pInterfaceId, [LvInterface](#) \*&pInterface)
- [LvStatus LvSystem::CloseInterface](#) ([LvInterface](#) \*&pInterface)
- [LvStatus LvSystem::OpenEvent](#) ([LvEventType](#) EventType, [LvEvent](#) \*&pEvent)
- [LvStatus LvSystem::CloseEvent](#) ([LvEvent](#) \*&pEvent)

### 5.14.1 Detailed Description

### 5.14.2 Function Documentation

#### 5.14.2.1 static [LvStatus LvSystem::Close](#) ( [LvSystem](#) \*& pSystem ) [static]

Deletes the [LvSystem](#) class instance. Actually it means freeing the corresponding GenTL library. Be sure you first close all dependent modules ([LvInterface](#), [LvEvent](#) etc.). If the System was opened multiple times, it only decreases the reference counter (see the note by the [LvSystem::Open\(\)](#)).

#### Parameters

<i>pSystem</i>	Pointer to <a href="#">LvSystem</a> instance, obtained from the <a href="#">LvSystem::Open()</a> function. The pointer is assigned NULL after the operation.
----------------	--

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.14.2.2 [LvStatus LvSystem::CloseEvent](#) ( [LvEvent](#) \*& pEvent )

Deletes the [LvEvent](#) class instance. This method is provided just for convenience, it has the same functionality as the [LvEvent::Close\(\)](#) static method.

#### Parameters

<i>pEvent</i>	Pointer the Event class instance, is assigned NULL after the closing is done.
---------------	---

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### See also

[LvEvent::Close\(\)](#).

## 5.14.2.3 LvStatus LvSystem::CloseInterface ( LvInterface \*&amp; pInterface )

Deletes the [LvInterface](#) class instance. This method is provided just for convenience, it has the same functionality as the [LvInterface::Close\(\)](#) static method. If the Interface was opened multiple times, it only decreases the reference counter (see a note by the [LvInterface::Open\(\)](#)). Be sure you first close all dependent modules ([LvDevice](#), [LvEvent](#) etc.).

## Parameters

<i>pInterface</i>	Pointer to the <a href="#">LvInterface</a> instance, obtained from the <a href="#">LvInterface::Open()</a> function. The pointer is assigned NULL after the operation.
-------------------	--

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## See also

[LvInterface::Close\(\)](#).

## 5.14.2.4 LvStatus LvSystem::FindInterface ( LvFindBy FindBy, const char \* pFindStr, char \* pInterfaceld, size\_t Size )

Finds the interface according specified criteria and returns a string ID of the interface, which is used by the [LvInterface::Open\(\)](#) function. This function does not update the interface list - if you need to do so, call the [LvSystem::UpdateInterfaceList\(\)](#) function before calling this function.

## Parameters

<i>FindBy</i>	Specifies by which criteria to find the interface. Use one of the <a href="#">LvFindBy</a> constants.
<i>pFindStr</i>	Specifies the find string, the meaning of which is determined by the FindBy parameter, for example when using the <a href="#">LvFindBy_IPAddress</a> , this string should contain the IP address searched for. The searched string is not case sensitive and need not be complete (is searched as a substring).
<i>pInterfaceld</i>	Pointer to a string buffer, where the interface ID will be placed.
<i>Size</i>	Size of the string buffer.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#). If the Interface is found, the returned status is LVSTATUS\_OK.

## 5.14.2.5 LvStatus LvSystem::FindInterface ( LvFindBy FindBy, const char \* pFindStr, std::string &amp; sInterfaceld )

Finds the interface according specified criteria and returns a string ID of the interface, which is used by the [LvInterface::Open\(\)](#) function. This function does not update the interface list - if you need to do so, call the [LvSystem::UpdateInterfaceList\(\)](#) function before calling this function.

## Parameters

<i>FindBy</i>	Specifies by which criteria to find the interface. Use one of the <a href="#">LvFindBy</a> constants.
<i>pFindStr</i>	Specifies the find string, the meaning of which is determined by the FindBy parameter, for example when using the <a href="#">LvFindBy_IPAddress</a> , this string should contain the IP address searched for. The searched string is not case sensitive and need not be complete (is searched as a substring).

<i>sInterfaceld</i>	String to which the interface ID will be placed.
---------------------	--

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#). If the Interface is found, the returned status is LVSTATUS\_OK.

**5.14.2.6 LvHSystem LvSystem::GetHandle ( )**

Returns a handle of the System (used in the Plain C API), associated with this class.

**Returns**

The Plain C API handle.

**5.14.2.7 LvStatus LvSystem::GetInterfaceld ( uint32\_t Index, char \* pInterfaceld, size\_t Size )**

Returns a string ID of the interface, which is used by the [LvInterface::Open\(\)](#) function.

**Parameters**

<i>Index</i>	Zero-based index of the interface, a value $\geq 0$ and $<$ number of interfaces, returned by the <a href="#">LvSystem::GetNumberOfInterfaces()</a> function.
<i>pInterfaceld</i>	Pointer to a string buffer, where the interface ID will be placed.
<i>Size</i>	Size of the string buffer.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.14.2.8 LvStatus LvSystem::GetInterfaceld ( uint32\_t Index, std::string & sInterfaceld )**

Returns a string ID of the interface, which is used by the [LvInterface::Open\(\)](#) function.

**Parameters**

<i>Index</i>	Zero-based index of the interface, a value $\geq 0$ and $<$ number of interfaces, returned by the <a href="#">LvSystem::GetNumberOfInterfaces()</a> function.
<i>sInterfaceld</i>	String, where the interface ID will be placed.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.14.2.9 LvStatus LvSystem::GetInterfaceldSize ( uint32\_t Index, size\_t \* pSize )**

Returns the size of the string buffer needed to hold the Interface ID string, including the terminating zero character.

**Parameters**

<i>Index</i>	Zero-based index of the interface, a value $\geq 0$ and $<$ number of interfaces, returned by the <a href="#">LvSystem::GetNumberOfInterfaces()</a> function.
<i>pSize</i>	Size of the buffer is returned in this parameter.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.14.2.10 LvStatus LvSystem::GetNumberOfInterfaces ( uint32\_t\* pNumberOfInterfaces )

Returns the number of found interfaces, after the [LvSystem::UpdateInterfaceList\(\)](#) call.

**Parameters**

<i>pNumberOfInterfaces</i>	Number of interfaces found.
----------------------------	-----------------------------

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.14.2.11 static LvStatus LvSystem::Open ( const char\* pSystemId, LvSystem\*& pSystem ) [static]

Creates the [LvSystem](#) class instance. Opening the system actually means loading the corresponding GenT↔L library. Note that before you can open the System, the [LvOpenLibrary\(\)](#) must be called. The same system can be open multiple times (there is a reference counter inside); in such case there must be also the same number of [LvSystem::Close\(\)](#) calls used (every open increase the reference count and every close decreases it).

**Parameters**

<i>pSystemId</i>	A string ID of the system. This can be either an empty string - then the default system is opened, or it can be a string obtained from the <a href="#">LvGetSystemId()</a> function.
<i>pSystem</i>	Pointer to the opened <a href="#">LvSystem</a> instance is returned here in case the opening succeeds, NULL if fails

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.14.2.12 LvStatus LvSystem::OpenEvent ( LvEventType EventType, LvEvent\*& pEvent )

Creates the [LvEvent](#) class instance, owned by the System. This method is provided just for convenience, it has the same functionality as the [LvEvent::Open\(\)](#) static method.

**Parameters**

<i>EventType</i>	One of the <a href="#">LvEventType</a> .
<i>pEvent</i>	To this parameter the Event class instance is stored.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**See also**

[LvEvent::Open\(\)](#).

#### 5.14.2.13 `LvStatus LvSystem::OpenInterface ( const char * pInterfaceId, LvInterface *& pInterface )`

Creates the `LvInterface` class instance. This method is provided just for convenience, it has the same functionality as the `LvInterface::Open()` static method. The same Interface can be open multiple times (there is a reference counter inside); in such case there must be also the same number of `LvInterface::Close()` or `LvSystem::InterfaceClose()` calls used (every open increase the reference count and every close decreases it).

##### Parameters

<i>pInterfaceId</i>	A string interface ID, obtained by the <code>LvSystem::GetInterfaceId()</code> .
<i>pInterface</i>	In this parameter the pointer to the <code>LvInterface</code> instance is returned.

##### Returns

Returns the `LvStatus` value indicating the success of the requested operation. See [LvStatus definitions](#).

##### See also

[LvInterface::Open\(\)](#).

#### 5.14.2.14 `LvStatus LvSystem::UpdateInterfaceList ( uint32_t Timeout = 0xFFFFFFFF )`

Updates the internal list of available interfaces. You can then iterate through them by `LvSystem::GetNumberOfInterfaces()` and `LvSystem::GetInterfaceId()`.

##### Parameters

<i>Timeout</i>	Specifies a timeout in ms for searching the interfaces. This applies only to special cases of interfaces, where some delay can happen; common interfaces are detected without any significant delays.
----------------	---

##### Returns

If the timeout has expired while waiting for the completion, the function returns `LVSTATUS_TIMEOUT`, otherwise it returns the `LvStatus` value indicating the success of the requested operation. See [LvStatus definitions](#).



## 5.15 SynView LvInterface methods

### Functions

- static [LvStatus LvInterface::Open](#) ([LvSystem](#) \*pSystem, const char \*pInterfaceld, [LvInterface](#) \*&pInterface)
- static [LvStatus LvInterface::Close](#) ([LvInterface](#) \*&pInterface)
- [LvStatus LvInterface::UpdateDeviceList](#) (uint32\_t Timeout=0xFFFFFFFF)
- [LvStatus LvInterface::GetNumberOfDevices](#) (uint32\_t \*pDevices)
- [LvStatus LvInterface::GetDeviceld](#) (uint32\_t Index, char \*pDeviceld, size\_t Size)
- [LvStatus LvInterface::GetDeviceldSize](#) (uint32\_t Index, size\_t \*pSize)
- [LvStatus LvInterface::GetDeviceld](#) (uint32\_t Index, std::string &sDeviceld)
- [LvStatus LvInterface::FindDevice](#) ([LvFindBy](#) FindBy, const char \*pFindStr, char \*pDeviceld, size\_t Size)
- [LvStatus LvInterface::FindDevice](#) ([LvFindBy](#) FindBy, const char \*pFindStr, std::string &sDeviceld)
- [LvHInterface LvInterface::GetHandle](#) ()
- [LvStatus LvInterface::OpenDevice](#) (const char \*pDeviceld, [LvDevice](#) \*&pDevice, [LvDeviceAccess](#) Access=[LvDeviceAccess\\_Exclusive](#))
- [LvStatus LvInterface::CloseDevice](#) ([LvDevice](#) \*&pDevice)

### 5.15.1 Detailed Description

### 5.15.2 Function Documentation

#### 5.15.2.1 static [LvStatus LvInterface::Close](#) ( [LvInterface](#) \*& *pInterface* ) [static]

Deletes the [LvInterface](#) class instance. If the Interface was opened multiple times, it only decreases the reference counter (see a note by the [LvInterface::Open\(\)](#)). Be sure you first close all dependent modules ([LvDevice](#), [LvEvent](#) etc.).

#### Parameters

<i>pInterface</i>	Pointer to the <a href="#">LvInterface</a> instance, obtained from the <a href="#">LvInterface::Open()</a> function. The pointer is assigned NULL after the operation.
-------------------	--

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### See also

[LvSystem::CloseInterface\(\)](#).

#### 5.15.2.2 [LvStatus LvInterface::CloseDevice](#) ( [LvDevice](#) \*& *pDevice* )

Deletes the [LvDevice](#) class instance. This method is provided just for convenience, it has the same functionality as the [LvDevice::Close\(\)](#) static method. Be sure you first close all dependent modules ([LvStream](#), [LvEvent](#) etc.).

#### Parameters

<i>pDevice</i>	Pointer to the <a href="#">LvDevice</a> instance, obtained from the <a href="#">LvDevice::Open()</a> function. This pointer is assigned NULL after the operation.
----------------	---

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### See also

[LvDevice::Close\(\)](#).

### 5.15.2.3 `LvStatus LvInterface::FindDevice ( LvFindBy FindBy, const char * pFindStr, char * pDeviceld, size_t Size )`

Finds the device according specified criteria and returns a string ID of the device, which can be used by the [LvDevice::Open\(\)](#) function. This function does not update the device list - if you need to do so, call the [LvInterface::UpdateDeviceList\(\)](#) function before calling this function.

#### Parameters

<i>FindBy</i>	Specifies by which criteria to find the interface. Use one of the <a href="#">LvFindBy</a> constants.
<i>pFindStr</i>	Specifies the find string, the meaning of which is determined by the FindBy parameter, for example when using the <a href="#">LvFindBy_IPAddress</a> , this string should contain the IP address searched for. The searched string is not case sensitive and need not be complete (is searched as a substring).
<i>pDeviceld</i>	Pointer to a string buffer, where the device ID will be placed.
<i>Size</i>	Size of the buffer.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#). If the Device is found, the returned status is `LVSTATUS_OK`.

### 5.15.2.4 `LvStatus LvInterface::FindDevice ( LvFindBy FindBy, const char * pFindStr, std::string & sDeviceld )`

Finds the device according specified criteria and returns a string ID of the device, which can be used by the [LvDevice::Open\(\)](#) function. This function does not update the device list - if you need to do so, call the [LvInterface::UpdateDeviceList\(\)](#) function before calling this function.

#### Parameters

<i>FindBy</i>	Specifies by which criteria to find the interface. Use one of the <a href="#">LvFindBy</a> constants.
<i>pFindStr</i>	Specifies the find string, the meaning of which is determined by the FindBy parameter, for example when using the <a href="#">LvFindBy_IPAddress</a> , this string should contain the IP address searched for. The searched string is not case sensitive and need not be complete (is searched as a substring).
<i>sDeviceld</i>	To this string the found device ID is placed.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#). If the Device is found, the returned status is `LVSTATUS_OK`.

### 5.15.2.5 `LvStatus LvInterface::GetDeviceld ( uint32_t Index, char * pDeviceld, size_t Size )`

Returns a string ID of the device at specified position in the list. Note that this device ID is stable (the same physical device has always the same ID) and it is unique (no other physical device can have the same ID). To hardcode directly the device ID in your application is not recommended, as the application would not be usable, when a defective device needs to be replaced. The SynView User's Guide discuss the ways, how to solve such maintainability demands.

#### Parameters

<i>Index</i>	Zero-based index of the device, a value $\geq 0$ and $<$ number of devices, returned by the <a href="#">LvInterface::GetNumberOfDevices()</a> function.
--------------	---

<i>pDeviceId</i>	Pointer to a string buffer, where the device ID will be placed.
<i>Size</i>	Size of the buffer.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.15.2.6 LvStatus LvInterface::GetDeviceId ( uint32\_t Index, std::string & sDeviceId )**

Returns a string ID of the device at specified position in the list. Note that this device ID is stable (the same physical device has always the same ID) and it is unique (no other physical device can have the same ID). To hardcode directly the device ID in your application is not recommended, as the application would not be usable, when a defective device needs to be replaced. The SynView User's Guide discuss the ways, how to solve such maintainability demands.

**Parameters**

<i>Index</i>	Zero-based index of the device, a value $\geq 0$ and $<$ number of devices, returned by the <a href="#">LvInterface::GetNumberOfDevices()</a> function.
<i>sDeviceId</i>	String to which the device ID will be placed.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.15.2.7 LvStatus LvInterface::GetDeviceIdSize ( uint32\_t Index, size\_t \* pSize )**

Returns the size of the string buffer needed to hold the Device ID string, including the terminating zero character.

**Parameters**

<i>Index</i>	Zero-based index of the device, a value $\geq 0$ and $<$ number of devices, returned by the <a href="#">LvInterface::GetNumberOfDevices()</a> function.
<i>pSize</i>	Size of the buffer is returned in this parameter.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.15.2.8 LvHInterface LvInterface::GetHandle ( )**

Returns a handle of the Interface (used in the Plain C API), associated with this class.

**Returns**

The Plain C API handle.

**5.15.2.9 LvStatus LvInterface::GetNumberOfDevices ( uint32\_t \* pDevices )**

Returns the number of devices found by the [LvInterface::UpdateDeviceList\(\)](#) function.

## Parameters

<i>pDevices</i>	Number of devices found.
-----------------	--------------------------

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.15.2.10** `static LvStatus LvInterface::Open ( LvSystem * pSystem, const char * pInterfaceId, LvInterface *& pInterface ) [static]`

Creates the [LvInterface](#) class instance. The same Interface can be open multiple times (there is a reference counter inside); in such case there must be also the same number of [LvInterface::Close\(\)](#) or [LvSystem::InterfaceClose\(\)](#) calls used (every open increase the reference count and every close decreases it) .

## Parameters

<i>pSystem</i>	A pointer to the <a href="#">LvSystem</a> instance, obtained from the <a href="#">LvSystem::Open()</a> function.
<i>pInterfaceId</i>	A string interface ID, obtained by the <a href="#">LvSystem::GetInterfaceId()</a> .
<i>pInterface</i>	In this parameter the pointer to the <a href="#">LvInterface</a> instance is returned.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## See also

[LvSystem::OpenInterface\(\)](#).

**5.15.2.11** `LvStatus LvInterface::OpenDevice ( const char * pDeviceId, LvDevice *& pDevice, LvDeviceAccess Access = LvDeviceAccess_Exclusive )`

Creates the [LvDevice](#) class instance. This method is provided just for convenience, it has the same functionality as the [LvDevice::Open\(\)](#) static method. This physically means opening a connection with the device and retrieving a list of device remote features. Always check the success of this function call; the opening may fail for example when you request an exclusive access and the device is already open by some other application.

## Parameters

<i>pDeviceId</i>	A string ID of the device, obtained by <a href="#">LvInterface::GetDeviceId()</a> function.
<i>pDevice</i>	In this parameter the pointer to the <a href="#">LvDevice</a> instance is returned.
<i>Access</i>	Desired device access, one of the <a href="#">LvDeviceAccess</a> constants.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## See also

[LvDevice::Open\(\)](#).

**5.15.2.12** `LvStatus LvInterface::UpdateDeviceList ( uint32_t Timeout = 0xFFFFFFFF )`

Updates the Device list. The available devices are searched.

**Parameters**

<i>Timeout</i>	Specifies a timeout in ms for searching the devices.
----------------	--

**Returns**

If the timeout has expired while waiting for the completion, the function returns [LVSTATUS\\_TIMEOUT](#), otherwise it returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.16 SynView LvDevice methods

### Modules

- [SynView LvDevice firmware update methods](#)

### Functions

- static `LvStatus LvDevice::Open` (`LvInterface *pInterface`, `const char *pDeviceId`, `LvDevice *&pDevice`, `LvDeviceAccess Access=LvDeviceAccess_Exclusive`)
- static `LvStatus LvDevice::Close` (`LvDevice *&pDevice`)
- `LvStatus LvDevice::GetNumberOfStreams` (`uint32_t *pNumberOfStreams`)
- `LvStatus LvDevice::GetStreamId` (`uint32_t Index`, `char *pStreamId`, `size_t Size`)
- `LvStatus LvDevice::GetStreamIdSize` (`uint32_t Index`, `size_t *pSize`)
- `LvStatus LvDevice::GetStreamId` (`uint32_t Index`, `std::string &sStreamId`)
- `LvStatus LvDevice::AcquisitionStart` (`uint32_t Options=0`)
- `LvStatus LvDevice::AcquisitionStop` (`uint32_t Options=0`)
- `LvStatus LvDevice::AcquisitionAbort` (`uint32_t Options=0`)
- `LvStatus LvDevice::AcquisitionArm` (`uint32_t Options=0`)
- `LvStatus LvDevice::SaveSettings` (`const char *pId`, `const char *pFileName`, `uint32_t Options`)
- `LvStatus LvDevice::LoadSettings` (`const char *pId`, `const char *pFileName`, `uint32_t Options`)
- `LvStatus LvDevice::UniSetLut` (`LvLUTSelector Selector`, `void *pLUT`, `size_t Size`, `uint32_t Options=0`)
- `LvStatus LvDevice::UniGetLut` (`LvLUTSelector Selector`, `void *pLUT`, `size_t Size`, `uint32_t Options=0`)
- `LvStatus LvDevice::OpenStream` (`const char *pStreamId`, `LvStream *&pStream`)
- `LvStatus LvDevice::CloseStream` (`LvStream *&pStream`)
- `LvStatus LvDevice::OpenEvent` (`LvEventType EventType`, `LvEvent *&pEvent`)
- `LvStatus LvDevice::CloseEvent` (`LvEvent *&pEvent`)
- `LvHDevice LvDevice::GetHandle` ()

### 5.16.1 Detailed Description

### 5.16.2 Function Documentation

#### 5.16.2.1 `LvStatus LvDevice::AcquisitionAbort ( uint32_t Options = 0 )`

Aborts the acquisition immediately, without completing the current Frame or waiting on a trigger.

#### Parameters

<i>Options</i>	Reserved for future use, must be 0 or omitted.
----------------	--

#### Returns

Returns the `LvStatus` value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.16.2.2 `LvStatus LvDevice::AcquisitionArm ( uint32_t Options = 0 )`

Prepares the device for acquisition, so that the acquisition using the `LvDevice::AcquisitionStart()` function then can start fast. If it is not called before `LvDevice::AcquisitionStart()`, it is called automatically inside the `LvDevice::AcquisitionStart()`.

## Parameters

<i>Options</i>	Reserved for future use, must be 0 or omitted.
----------------	--

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.16.2.3 **LvStatus** LvDevice::AcquisitionStart ( uint32\_t *Options* = 0 )

Starts the acquisition. This function includes more than just calling the remote AcquisitionStart command on the device - it checks the size of the buffers, prepares the streams for the start, locks GenTL params and then starts the acquisition on the device itself. Always check the success of this function call.

## Parameters

<i>Options</i>	Reserved for future use, must be 0 or omitted.
----------------	--

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.16.2.4 **LvStatus** LvDevice::AcquisitionStop ( uint32\_t *Options* = 0 )

Stops the acquisition.

## Parameters

<i>Options</i>	Reserved for future use, must be 0 or omitted.
----------------	--

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.16.2.5 **static LvStatus** LvDevice::Close ( LvDevice \*& *pDevice* ) [static]

Deletes the [LvDevice](#) class instance. Be sure you first close all dependent modules ([LvStream](#), [LvEvent](#) etc.).

## Parameters

<i>pDevice</i>	Pointer to the <a href="#">LvDevice</a> instance, obtained from the <a href="#">LvDevice::Open()</a> function. This pointer is assigned NULL after the operation.
----------------	---

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## See also

[LvInterface::CloseDevice\(\)](#).

5.16.2.6 **LvStatus** LvDevice::CloseEvent ( LvEvent \*& *pEvent* )

Deletes the [LvEvent](#) class instance. This method is provided just for convenience, it has the same functionality as the [LvEvent::Close\(\)](#) static method.

## Parameters

<i>pEvent</i>	A pointer to the <a href="#">LvEvent</a> class instance, obtained from the <a href="#">LvEvent::Open()</a> or <a href="#">LvDevice::OpenEvent()</a> function. This pointer is assigned NULL after the operation.
---------------	--

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## See also

[LvEvent::Close\(\)](#).

5.16.2.7 [LvStatus](#) [LvDevice::CloseStream \( LvStream \\*& pStream \)](#)

Deletes the [LvStream](#) class instance. This method is provided just for convenience, it has the same functionality as the [LvStream::Close\(\)](#) static method. Be sure you first close all dependent modules ([LvBuffers](#), [LvEvent](#), [LvRenderer](#) etc.).

## Parameters

<i>pStream</i>	Pointer to the <a href="#">LvStream</a> instance, obtained from the <a href="#">LvStream::Open()</a> or <a href="#">LvDevice::OpenStream()</a> function. This pointer is assigned NULL after the operation.
----------------	---

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## See also

[LvStream::Close\(\)](#).

5.16.2.8 [LvHDevice](#) [LvDevice::GetHandle \( \)](#)

Returns a handle of the Device (used in the Plain C API), associated with this class.

## Returns

The Plain C API handle.

5.16.2.9 [LvStatus](#) [LvDevice::GetNumberOfStreams \( uint32\\_t \\* pNumberOfStreams \)](#)

Returns the number of available stream types for this device. You can then iterate the streams by the [LvDevice::GetStreamId\(\)](#) function.

## Parameters

<i>pNumberOfStreams</i>	The number of streams is returned here.
-------------------------	---

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.16.2.10 [LvStatus](#) [LvDevice::GetStreamId \( uint32\\_t Index, char \\* pStreamId, size\\_t Size \)](#)

Returns a string Stream ID, needed for opening the stream.



## Parameters

<i>Index</i>	Zero-based index of the stream type, a value $\geq 0$ and $<$ number of streams, returned by the <a href="#">LvDevice::GetNumberOfStreams()</a> function.
<i>pStreamId</i>	Pointer to a string buffer, where the stream ID will be placed.
<i>Size</i>	Size of the buffer.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.16.2.11 [LvStatus](#) LvDevice::GetStreamId ( `uint32_t Index`, `std::string & sStreamId` )

Returns a string Stream ID, needed for opening the stream.

## Parameters

<i>Index</i>	Zero-based index of the stream type, a value $\geq 0$ and $<$ number of streams, returned by the <a href="#">LvDevice::GetNumberOfStreams()</a> function.
<i>sStreamId</i>	A string to which the stream ID is placed.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.16.2.12 [LvStatus](#) LvDevice::GetStreamIdSize ( `uint32_t Index`, `size_t * pSize` )

Returns the size of the string buffer needed to hold the stream ID at given index, including the space for the terminating zero character.

## Parameters

<i>Index</i>	Zero-based index of the stream type, a value $\geq 0$ and $<$ number of streams, returned by the <a href="#">LvDevice::GetNumberOfStreams()</a> function.
<i>pSize</i>	Size of the buffer is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.16.2.13 [LvStatus](#) LvDevice::LoadSettings ( `const char * pId`, `const char * pFileName`, `uint32_t Options` )

Loads the device settings from a file. In the Options can be specified which parts of the device configuration are to be loaded.

## Parameters

<i>pId</i>	A string ID enabling to distinguish more configurations in one file. If empty, the "Default" will be used.
<i>pFileName</i>	The file specification, where the configuration is stored. It is a text file.
<i>Options</i>	One or or-ed combination of <a href="#">LvSaveFlag definitions</a> .

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.16.2.14** `static LvStatus LvDevice::Open ( LvInterface * pInterface, const char * pDeviceId, LvDevice *& pDevice, LvDeviceAccess Access = LvDeviceAccess_Exclusive ) [static]`

Creates the [LvDevice](#) class instance.

This physically means opening a connection with the device and retrieving a list of device remote features. Always check the success of this function call; the opening may fail for example when you request an exclusive access and the device is already open by some other application.

#### Parameters

<i>pInterface</i>	A pointer to the <a href="#">LvInterface</a> instance, obtained from the <a href="#">LvInterface::Open()</a> function.
<i>pDeviceId</i>	A string ID of the device, obtained by <a href="#">LvInterface::GetDeviceId()</a> function.
<i>pDevice</i>	In this parameter the pointer to the <a href="#">LvDevice</a> instance is returned.
<i>Access</i>	Desired device access, one of the <a href="#">LvDeviceAccess</a> constants.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### See also

[LvInterface::OpenDevice\(\)](#).

**5.16.2.15** `LvStatus LvDevice::OpenEvent ( LvEventType EventType, LvEvent *& pEvent )`

Creates the [LvEvent](#) class instance for specified owner module. This method is provided just for convenience, it has the same functionality as the [LvEvent::Open\(\)](#) static method.

#### Parameters

<i>EventType</i>	One of the <a href="#">LvEventType</a> .
<i>pEvent</i>	To this parameter the pointer <a href="#">LvEvent</a> instance is stored.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### See also

[LvEvent::Open\(\)](#).

**5.16.2.16** `LvStatus LvDevice::OpenStream ( const char * pStreamId, LvStream *& pStream )`

Creates the [LvStream](#) class instance associated with the device. This method is provided just for convenience, it has the same functionality as the [LvStream::Open\(\)](#) static method.

#### Parameters

<i>pStreamId</i>	A string ID of the stream, obtained from <a href="#">LvDevice::GetStreamId()</a> . If an empty string is used, the first found stream is opened. This is usually the image data stream.
<i>pStream</i>	In this parameter the pointer to the <a href="#">LvStream</a> instance is returned.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### See also

[LvStream::Open\(\)](#).

#### 5.16.2.17 LvStatus LvDevice::SaveSettings ( const char \* *pld*, const char \* *pFileName*, uint32\_t *Options* )

Saves the device settings to a file. In the Options can be specified which parts of the device configuration are to be saved.

##### Parameters

<i>pld</i>	A string ID enabling to distinguish more configurations in one file. If empty, the "Default" will be used.
<i>pFileName</i>	The file specification, to which the configuration is stored. It is a text file.
<i>Options</i>	One or or-ed combination of <a href="#">LvSaveFlag definitions</a> .

##### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.16.2.18 LvStatus LvDevice::UniGetLut ( LvLUTSelector *Selector*, void \* *pLUT*, size\_t *Size*, uint32\_t *Options* = 0 )

Gets the lookup table. See [LvDevice::UniSetLut\(\)](#) for details. The LUT is automatically recalculated to appropriate type, if you use different LUT bit depth than is the actually used for the current pixel format. So you can for example read the 12-bit LUT to 8-bit LUT array.

##### Parameters

<i>Selector</i>	Lookup table selector, see <a href="#">LvLUTSelector</a> .
<i>pLUT</i>	Pointer to the lookup table.
<i>Size</i>	Size of the lookup table. The only valid values are <ul style="list-style-type: none"> <li>• 256 for 8-bit LUT</li> <li>• 2048 for 10-bit LUT</li> <li>• 8192 for 12-bit LUT</li> </ul>
<i>Options</i>	The <a href="#">LvUniLutFlags_HwLut</a> option can be used to apply to function directly on HW LUT.

##### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.16.2.19 LvStatus LvDevice::UniSetLut ( LvLUTSelector *Selector*, void \* *pLUT*, size\_t *Size*, uint32\_t *Options* = 0 )

Sets the lookup table. If the hardware lookup table is available, it is used, otherwise a software lookup table is set. This function belongs to a set of functions, which unify the functionality of devices with real-time processing embedded in hardware (RTF) and devices without real-time processing, for which the processing is made by software. The LUT is automatically recalculated to appropriate type, if you use different LUT bit depth than is the actually used for the current pixel format.

##### Parameters

<i>Selector</i>	Lookup table selector, see <a href="#">LvLUTSelector</a> .
<i>pLUT</i>	Pointer to the lookup table.
<i>Size</i>	Size of the lookup table. The only valid values are <ul style="list-style-type: none"> <li>• 256 for 8-bit LUT</li> <li>• 2048 for 10-bit LUT</li> <li>• 8192 for 12-bit LUT</li> </ul>
<i>Options</i>	The <a href="#">LvUniLutFlags_HwLut</a> option can be used to apply to function directly on HW LUT.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.17 SynView LvDevice firmware update methods

### Functions

- [LvStatus LvDevice::FwGetFilePattern](#) (uint32\_t Which, char \*pFilePattern, size\_t Size)
- [LvStatus LvDevice::FwLoad](#) (uint32\_t Which, const char \*pFilePath)
- [LvStatus LvDevice::FwGetLoadStatus](#) (uint32\_t Which, uint32\_t \*pCurrentByteCount, bool \*plsLoading)

#### 5.17.1 Detailed Description

#### 5.17.2 Function Documentation

##### 5.17.2.1 LvStatus LvDevice::FwGetFilePattern ( uint32\_t Which, char \* pFilePattern, size\_t Size )

Returns the file name mask (with wildcard characters), for searching the file with the appropriate firmware update. The files with the FW update have in their names coded the hardware IDs, so using this mask (for example in a filter in a file open dialog box) assures the file appropriate for this device is used.

##### Parameters

<i>Which</i>	An ID specific for a hardware. Discussed in the SynView User's Guide.
<i>pFilePattern</i>	In this parameter the file pattern is returned.
<i>Size</i>	Size of the buffer (to which the pFilePattern points).

##### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

##### 5.17.2.2 LvStatus LvDevice::FwGetLoadStatus ( uint32\_t Which, uint32\_t \* pCurrentByteCount, bool \* plsLoading )

Returns the byte count and whether the loading is still in progress.

##### Parameters

<i>Which</i>	An ID specific for a hardware. Discussed in the SynView User's Guide.
<i>pCurrentByteCount</i>	Returns number of bytes transferred so far.
<i>plsLoading</i>	Returns true if the loading is still in progress.

##### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

##### 5.17.2.3 LvStatus LvDevice::FwLoad ( uint32\_t Which, const char \* pFilePath )

Loads the firmware from a file to the hardware. It can be very long process (taking minutes) and this functions blocks the thread during this process. It is recommended to check the load status from another thread using the [LvFwGetLoadStatus\(\)](#) function.

##### Parameters

<i>Which</i>	An ID specific for a hardware. Discussed in the SynView User's Guide.
--------------	---

---

<i>pFilePath</i>	File specification, with full path.
------------------	-------------------------------------

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.18 SynView LvStream methods

### Functions

- static [LvStatus LvStream::Open](#) ([LvDevice](#) \*pDevice, const char \*pStreamId, [LvStream](#) \*&pStream)
- static [LvStatus LvStream::Close](#) ([LvStream](#) \*&pStream)
- [LvStatus LvStream::GetBufferAt](#) (uint32\_t BufferIndex, [LvBuffer](#) \*&pBuffer)
- [LvStatus LvStream::FlushQueue](#) ([LvQueueOperation](#) Operation)
- [LvStatus LvStream::Start](#) (uint32\_t StartFlags=0x00000000, uint32\_t ImagesToAcquire=0xFFFFFFFF)
- [LvStatus LvStream::Stop](#) (uint32\_t StopFlags=0x00000000)
- [LvHStream LvStream::GetHandle](#) ()
- [LvStatus LvStream::OpenBuffer](#) (void \*pDataPointer, size\_t DataSize, void \*pUserPointer, uint32\_t Options, [LvBuffer](#) \*&pBuffer)
- [LvStatus LvStream::CloseBuffer](#) ([LvBuffer](#) \*&pBuffer)
- [LvStatus LvStream::OpenEvent](#) ([LvEventType](#) EventType, [LvEvent](#) \*&pEvent)
- [LvStatus LvStream::CloseEvent](#) ([LvEvent](#) \*&pEvent)
- [LvStatus LvStream::OpenRenderer](#) ([LvRenderer](#) \*&pRenderer)
- [LvStatus LvStream::CloseRenderer](#) ([LvRenderer](#) \*&pRenderer)

### 5.18.1 Detailed Description

### 5.18.2 Function Documentation

#### 5.18.2.1 static [LvStatus LvStream::Close](#) ( [LvStream](#) \*& pStream ) [static]

Deletes the [LvStream](#) class instance. Be sure you first close all dependent modules ([LvBuffers](#), [LvEvent](#), [LvRenderer](#) etc.).

#### Parameters

<i>pStream</i>	Pointer to the <a href="#">LvStream</a> instance, obtained from the <a href="#">LvStream::Open()</a> function. This pointer is assigned NULL after the operation.
----------------	---

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### See also

[LvDevice::CloseStream\(\)](#).

#### 5.18.2.2 [LvStatus LvStream::CloseBuffer](#) ( [LvBuffer](#) \*& pBuffer )

Deletes the [LvBuffer](#) class instance. On the GenTL level it corresponds to the [DSRevokeBuffer\(\)](#) function. This method is provided just for convenience, it has the same functionality as the [LvBuffer::Close\(\)](#) static method.

#### Parameters

<i>pBuffer</i>	A pointer to the <a href="#">LvBuffer</a> instance, obtained from the <a href="#">LvBuffer::Open()</a> function. This pointer is assigned NULL after the operation.
----------------	---

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### See also

[LvBuffer::Close\(\)](#).

### 5.18.2.3 LvStatus LvStream::CloseEvent ( LvEvent \*& pEvent )

Deletes the [LvEvent](#) class instance. This method is provided just for convenience, it has the same functionality as the [LvEvent::Close\(\)](#) static method.

#### Parameters

<i>pEvent</i>	Pointer the Event class instance, is assigned NULL after the closing is done.
---------------	---

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### See also

[LvEvent::Close\(\)](#).

### 5.18.2.4 LvStatus LvStream::CloseRenderer ( LvRenderer \*& pRenderer )

Deletes the [LvRenderer](#) class instance. This method is provided just for convenience, it has the same functionality as the [LvRenderer::Close\(\)](#) static method.

#### Parameters

<i>pRenderer</i>	A pointer to the <a href="#">LvRenderer</a> instance, obtained from the <a href="#">LvRenderer::Open()</a> function. This pointer is set to NULL after close.
------------------	---

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### See also

[LvRenderer::Close\(\)](#).

### 5.18.2.5 LvStatus LvStream::FlushQueue ( LvQueueOperation Operation )

Moves the buffers according to the [LvQueueOperation](#) specified.

#### Parameters

<i>Operation</i>	One of the <a href="#">LvQueueOperation</a> .
------------------	---

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

### 5.18.2.6 LvStatus LvStream::GetBufferAt ( uint32\_t BufferIndex, LvBuffer \*& pBuffer )

Returns the buffer instance at given index.



## Parameters

<i>BufferIndex</i>	Zero-based index.
<i>pBuffer</i>	In this parameter the pointer to <a href="#">LvBuffer</a> instance is returned.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.18.2.7 LvHStream LvStream::GetHandle ( )

Returns a handle of the Stream (used in the Plain C API), associated with this class.

## Returns

The Plain C API handle.

5.18.2.8 static LvStatus LvStream::Open ( LvDevice \* pDevice, const char \* pStreamId, LvStream \*& pStream )  
[static]

Creates the [LvStream](#) class instance, associated with the device.

## Parameters

<i>pDevice</i>	A pointer to the <a href="#">LvDevice</a> instance, obtained from the <a href="#">LvDevice::Open()</a> function.
<i>pStreamId</i>	A string ID of the stream, obtained from <a href="#">LvDevice::GetStreamId()</a> . If an empty string is used, the first found stream is opened. This is usually the image data stream.
<i>pStream</i>	In this parameter the pointer to the <a href="#">LvStream</a> instance is returned.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## See also

[LvDevice::OpenStream\(\)](#).

## 5.18.2.9 LvStatus LvStream::OpenBuffer ( void \* pDataPointer, size\_t DataSize, void \* pUserPointer, uint32\_t Options, LvBuffer \*&amp; pBuffer )

Creates the [LvBuffer](#) class instance. On the GenTL level it corresponds to [DSAnnounceBuffer\(\)](#) or [DSAllocAndAnnounceBuffer\(\)](#). This method is provided just for convenience, it has the same functionality as the [LvBuffer::Open\(\)](#) static method.

## Parameters

<i>pDataPointer</i>	Pointer to image data buffer. This can be supplied by the application (in such case the <a href="#">DataSize</a> must be set to the actual size of the buffer), or can be left NULL - in such case the buffer is allocated by SynView.
---------------------	--

<i>DataSize</i>	Size of the buffer supplied, or 0 if the <i>pDataPointer</i> is NULL.
<i>pUserPointer</i>	A user pointer, which is then passed back in the <a href="#">LvEventCallbackNewBufferFunc()</a> . It enables the application to reference some own data structure associated with the buffer.
<i>Options</i>	Currently unused, must be 0.
<i>pBuffer</i>	To this parameter the pointer to the <a href="#">LvBuffer</a> instance is returned.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**See also**

[LvBuffer::Open\(\)](#).

**5.18.2.10 LvStatus LvStream::OpenEvent ( LvEventType EventType, LvEvent \*& pEvent )**

Creates the [LvEvent](#) class instance, owned by the Stream. This method is provided just for convenience, it has the same functionality as the [LvEvent::Open\(\)](#) static method.

**Parameters**

<i>EventType</i>	One of the <a href="#">LvEventType</a> .
<i>pEvent</i>	To this parameter the Event class instance is stored.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**See also**

[LvEvent::Open\(\)](#).

**5.18.2.11 LvStatus LvStream::OpenRenderer ( LvRenderer \*& pRenderer )**

Creates the [LvRenderer](#) class instance for image display. The renderer attempts to load the `sv.synview.display` library. In case of SynView installation in an operating system without possibility to graphically display (for example Linux without XWindows), the load of this library fails and the calls to `Renderer` functions will return errors. This method is provided just for convenience, it has the same functionality as the [LvRenderer::Open\(\)](#) static method.

**Parameters**

<i>pRenderer</i>	In this parameter the pointer to the <a href="#">LvRenderer</a> is returned.
------------------	--

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**See also**

[LvRenderer::Open\(\)](#).

**5.18.2.12 LvStatus LvStream::Start ( uint32\_t StartFlags = 0x00000000, uint32\_t ImagesToAcquire = 0xFFFFFFFF )**

Starts the stream. This function need not be used on the image stream, where it is called automatically in the [LvDevice::AcquisitionStart\(\)](#) function.

## Parameters

<i>StartFlags</i>	One of the GroupSynView_StreamStartFlags.
<i>ImagesToAcquire</i>	Number of images to acquire.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.18.2.13 LvStatus LvStream::Stop ( uint32\_t StopFlags = 0x00000000 )**

Stops the stream. This function need not be used on the image stream, where it is called automatically in the [LvDevice::AcquisitionStop\(\)](#) function.

## Parameters

<i>StopFlags</i>	One of the GroupSynView_StreamStopFlags.
------------------	--

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.19 SynView LvBuffer methods

### Functions

- static [LvStatus LvBuffer::Open](#) ([LvStream](#) \*pStream, void \*pDataPointer, [size\\_t](#) DataSize, void \*pUser←→ Pointer, [uint32\\_t](#) Options, [LvBuffer](#) \*&pBuffer)
- static [LvStatus LvBuffer::Close](#) ([LvBuffer](#) \*&pBuffer)
- [LvStatus LvBuffer::AttachProcessBuffer](#) (void \*pDataPointer, [size\\_t](#) DataSize)
- [LvStatus LvBuffer::Queue](#) ()
- [LvStatus LvBuffer::ParseChunkData](#) (bool UpdateLayout=false)
- [LvStatus LvBuffer::SaveImageToBmpFile](#) (const char \*pFileName)
- [LvStatus LvBuffer::SaveImageToJpgFile](#) (const char \*pFileName, [uint32\\_t](#) Quality)
- [LvStatus LvBuffer::SaveImageToTifFile](#) (const char \*pFileName, [uint32\\_t](#) Options=0)
- [LvStatus LvBuffer::GetImgInfo](#) ([LviImgInfo](#) &ImgInfo, [uint32\\_t](#) Options=0)
- [LvStatus LvBuffer::GetLastPaintRect](#) ([int32\\_t](#) \*pX, [int32\\_t](#) \*pY, [int32\\_t](#) \*pWidth, [int32\\_t](#) \*pHeight)
- [LvStatus LvBuffer::UniCalculateWhiteBalance](#) ()
- [LvHBuffer LvBuffer::GetHandle](#) ()
- void \* [LvBuffer::GetUserPtr](#) ()

### 5.19.1 Detailed Description

### 5.19.2 Function Documentation

#### 5.19.2.1 [LvStatus LvBuffer::AttachProcessBuffer](#) ( void \* *pDataPointer*, [size\\_t](#) *DataSize* )

Attaches a process buffer to a buffer. The process buffer may be needed for software processing, for example Bayer decoding, if the device hardware is not capable of it. The process buffer can be either supplied by the application by this function, or allocated automatically by SynView, upon need.

#### Parameters

<i>pDataPointer</i>	Pointer to the supplied buffer.
<i>DataSize</i>	Size of the buffer.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.19.2.2 static [LvStatus LvBuffer::Close](#) ( [LvBuffer](#) \*& *pBuffer* ) [static]

Deletes the [LvBuffer](#) class instance. On the GenTL level it corresponds to the DSRevokeBuffer() function.

#### Parameters

<i>pBuffer</i>	A pointer to the <a href="#">LvBuffer</a> instance, obtained from the <a href="#">LvBuffer::Open()</a> function. This pointer is assigned NULL after the operation.
----------------	---

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### See also

[LvStream::CloseBuffer\(\)](#).

## 5.19.2.3 LvHBuffer LvBuffer::GetHandle ( )

Returns a handle of the Buffer (used in the Plain C API), associated with this class.

## Returns

The Plain C API handle.

5.19.2.4 LvStatus LvBuffer::GetImgInfo ( LvImgInfo & *imgInfo*, uint32\_t *Options* = 0 )

Fills the [LvImgInfo](#) structure for the image in the buffer. This simplifies a direct use of the [SynView Image Processing Library](#). If the image is processed, the image info points to the processed image, otherwise it points to the original image.

## Parameters

<i>imgInfo</i>	The <a href="#">imgInfo</a> structure, to which are the image parameters stored.
<i>Options</i>	Currently unused, must be 0.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.19.2.5 LvStatus LvBuffer::GetLastPaintRect ( int32\_t \* *pX*, int32\_t \* *pY*, int32\_t \* *pWidth*, int32\_t \* *pHeight* )

Returns the rectangle to which the buffer was last painted. This is useful namely in case you have a tile mode and want to identify the buffer according a mouse click location. If the buffer was not yet painted by the renderer, the returned values are 0.

## Parameters

<i>pX</i>	Pointer to X offset in pixels.
<i>pY</i>	Pointer to Y offset in pixels.
<i>pWidth</i>	Pointer to Width in pixels.
<i>pHeight</i>	Pointer to Height in pixels.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.19.2.6 void\* LvBuffer::GetUserPtr ( )

Returns a pointer to the Buffer (used in the Plain C API), associated with this class.

## Returns

void pointer.

5.19.2.7 static LvStatus LvBuffer::Open ( LvStream \* *pStream*, void \* *pDataPointer*, size\_t *DataSize*, void \* *pUserPointer*, uint32\_t *Options*, LvBuffer \*& *pBuffer* ) [static]

Creates the [LvBuffer](#) class instance. On the GenTL level it corresponds to [DSAnnounceBuffer\(\)](#) or [DSAllocAndAnnounceBuffer\(\)](#).

## Parameters

<i>pStream</i>	A pointer to the <a href="#">LvStream</a> instance, obtained from the <a href="#">LvStream::Open()</a> function.
<i>pDataPointer</i>	Pointer to image data buffer. This can be supplied by the application (in such case the <code>DataSize</code> must be set to the actual size of the buffer), or can be left NULL - in such case the buffer is allocated by SynView.
<i>DataSize</i>	Size of the buffer supplied, or 0 if the <i>pDataPointer</i> is NULL.
<i>pUserPointer</i>	A user pointer, which is then passed back in the <a href="#">LvEventCallbackNewBufferFunct()</a> . It enables the application to reference some own data structure associated with the buffer.
<i>Options</i>	Currently unused, must be 0.
<i>pBuffer</i>	To this parameter the pointer to the <a href="#">LvBuffer</a> instance is returned.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## See also

[LvStream::OpenBuffer\(\)](#).

#### 5.19.2.8 [LvStatus](#) [LvBuffer::ParseChunkData](#) ( `bool UpdateLayout = false` )

Parses the chunk data of the image. The chunk data are then accessible as device remote features (for example [LvDevice\\_ChunkTimestamp](#)).

## Parameters

<i>UpdateLayout</i>	If set to true, the layout of chunk data is decoded. If set to false, the data are only read from already decoded layout - this is faster. Usually, the layout of the chunk data is constant, so it needs to be decoded only at first call of this function.
---------------------	--

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.19.2.9 [LvStatus](#) [LvBuffer::Queue](#) ( )

Puts the buffer to the input buffer pool. This is an important part of the image handling loop: after the buffer with the acquired image is passed to the application, the application must return it to the input buffer pool by this function after processing.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.19.2.10 [LvStatus](#) [LvBuffer::SaveImageToBmpFile](#) ( `const char * pFileName` )

Saves the image to a file in Windows BMP format. If the image is in the pixel format not compatible with the BMP format, it is automatically converted.

## Parameters

<i>pFileName</i>	The file name. Be sure to specify it with the full path.
------------------	--

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.19.2.11 LvStatus LvBuffer::SaveImageToJpgFile ( const char \* pFileName, uint32\_t Quality )**

Saves the image to a file in JPEG format. If the image is in the pixel format not compatible with the JPEG format, it is automatically converted.

**Parameters**

<i>pFileName</i>	The file name. Be sure to specify it with the full path.
<i>Quality</i>	The quality factor in range from from 1 to 100. The higher is the factor, the higher is the quality and lower the compression.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.19.2.12 LvStatus LvBuffer::SaveImageToTifFile ( const char \* pFileName, uint32\_t Options = 0 )**

Saves the image to a file in the TIFF format. If the image is in the pixel format not compatible with the TIF format, it is automatically converted.

**Parameters**

<i>pFileName</i>	The file name. Be sure to specify it with the full path.
<i>Options</i>	Options for saved pixel format. The <a href="#">LvipOption_TiffConvertTo16Bit</a> flag can be used there.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.19.2.13 LvStatus LvBuffer::UniCalculateWhiteBalance ( )**

Calculates white balance factors from the current image.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.20 SynView LvEvent methods

### Functions

- static `LvStatus LvEvent::Open (LvSystem *pSystem, LvEventType EventType, LvEvent *&pEvent)`
- static `LvStatus LvEvent::Open (LvDevice *pDevice, LvEventType EventType, LvEvent *&pEvent)`
- static `LvStatus LvEvent::Open (LvStream *pStream, LvEventType EventType, LvEvent *&pEvent)`
- static `LvStatus LvEvent::Close (LvEvent *&pEvent)`
- `LvStatus LvEvent::Kill ()`
- `LvStatus LvEvent::Flush ()`
- `LvStatus LvEvent::WaitAndGetData (void *pBuffer, size_t *pSize, uint32_t Timeout=0xFFFFFFFF)`
- `LvStatus LvEvent::WaitAndGetNewBuffer (LvBuffer *&pBuffer, uint32_t Timeout=0xFFFFFFFF)`
- `LvStatus LvEvent::GetDataInfo (void *pInBuffer, size_t InSize, LvEventDataInfo Info, void *pBuffer, size_t *pSize, LvInfoDataType *pType=NULL, int32_t Param=0)`
- `LvStatus LvEvent::PutData (void *pBuffer, size_t Size)`
- `LvStatus LvEvent::SetCallback (LvEventCallbackFunc pFunction, void *pUserParam)`
- `LvStatus LvEvent::SetCallbackNewBuffer (LvEventCallbackNewBufferFunc pFunction, void *pUserParam)`
- `LvStatus LvEvent::StartThread ()`
- `LvStatus LvEvent::StopThread ()`
- `bool LvEvent::CallbackMustExit ()`
- `LvHEvent LvEvent::GetHandle ()`

### 5.20.1 Detailed Description

### 5.20.2 Function Documentation

#### 5.20.2.1 `bool LvEvent::CallbackMustExit ( )`

Indicates that the callback function for the thread of the Event of the `LvEventType_NewBuffer` needs to exit as soon as possible

#### Returns

Returns the must exit status of the callback function of `LvEventType_NewBuffer`

#### 5.20.2.2 `static LvStatus LvEvent::Close ( LvEvent *& pEvent ) [static]`

Deletes the `LvEvent` class instance.

#### Parameters

<i>pEvent</i>	A pointer to the <code>LvEvent</code> instance, obtained from the <code>LvEvent::Open()</code> function. This pointer is assigned NULL after the operation.
---------------	---

#### Returns

Returns the `LvStatus` value indicating the success of the requested operation. See `LvStatus definitions`.

#### See also

`LvSystem::CloseEvent()`, `LvDevice::CloseEvent()`, `LvStream::CloseEvent()`.



## 5.20.2.3 LvStatus LvEvent::Flush ( )

Discards all buffers in the output buffer queue (waiting to be delivered to the application).

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.20.2.4 LvStatus LvEvent::GetDataInfo ( void \* *pInBuffer*, size\_t *InSize*, LvEventDataInfo *Info*, void \* *pBuffer*, size\_t \* *pSize*, LvInfoDataType \* *pType* = NULL, int32\_t *Param* = 0 )

Enables to parse the buffer from [LvEvent::WaitAndGetData](#).

## Parameters

<i>pInBuffer</i>	Pointer to a buffer containing event data. This value must not be NULL.
<i>InSize</i>	Size of the provided <i>pInBuffer</i> in bytes.
<i>Info</i>	One of the <a href="#">LvEventDataInfo</a> .
<i>pBuffer</i>	Pointer to a user allocated buffer to receive the requested information. If this parameter is NULL, <i>pSize</i> will contain the minimal size of <i>pBuffer</i> in bytes. If the <i>pType</i> is a string, the size includes the terminating 0.
<i>pSize</i>	Size of the buffer is returned in this parameter.
<i>pType</i>	One of the <a href="#">LvInfoDataType</a> .
<i>Param</i>	Additional parameter, if used, its role is explained by the <a href="#">LvEventDataInfo</a> .

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.20.2.5 LvHEvent LvEvent::GetHandle ( )

Returns a handle of the Event (used in the Plain C API), associated with this class.

## Returns

The Plain C API handle.

## 5.20.2.6 LvStatus LvEvent::Kill ( )

Terminates a single wait in the [LvEvent::WaitAndGetData\(\)](#) function.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.20.2.7 static LvStatus LvEvent::Open ( LvSystem \* *pSystem*, LvEventType *EventType*, LvEvent \*& *pEvent* )  
[static]

Creates the [LvEvent](#) class instance for specified [LvSystem](#) module.

## Parameters

<i>pSystem</i>	Owner <a href="#">LvSystem</a> instance.
<i>EventType</i>	One of the <a href="#">LvEventType</a> .
<i>pEvent</i>	In this parameter a pointer to <a href="#">LvEvent</a> instance is returned.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## See also

[LvSystem::OpenEvent\(\)](#).

**5.20.2.8** `static LvStatus LvEvent::Open ( LvDevice * pDevice, LvEventType EventType, LvEvent *& pEvent )`  
`[static]`

Creates the [LvEvent](#) class instance for specified [LvDevice](#) module.

## Parameters

<i>pDevice</i>	Owner <a href="#">LvDevice</a> instance.
<i>EventType</i>	One of the <a href="#">LvEventType</a> .
<i>pEvent</i>	In this parameter a pointer to <a href="#">LvEvent</a> instance is returned.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## See also

[LvDevice::OpenEvent\(\)](#).

**5.20.2.9** `static LvStatus LvEvent::Open ( LvStream * pStream, LvEventType EventType, LvEvent *& pEvent )`  
`[static]`

Creates the [LvEvent](#) class instance for specified [LvStream](#) module.

## Parameters

<i>pStream</i>	Owner <a href="#">LvStream</a> instance.
<i>EventType</i>	One of the <a href="#">LvEventType</a> .
<i>pEvent</i>	In this parameter a pointer to <a href="#">LvEvent</a> instance is returned.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## See also

[LvStream::OpenEvent\(\)](#).

**5.20.2.10** `LvStatus LvEvent::PutData ( void * pBuffer, size_t Size )`

Puts a new event to Event output queue. This function can be used only for user-defined events.

## Parameters

<i>pBuffer</i>	Pointer to event data.
<i>Size</i>	Size of the event data.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.20.2.11 **LvStatus LvEvent::SetCallback ( LvEventCallbackFunct pFunction, void \* pUserParam )**

Specifies a callback function for the event thread. Note that the callback function cannot be a method of a class.

## Parameters

<i>pFunction</i>	The callback function in the forms of <a href="#">LvEventCallbackFunct</a> .
<i>pUserParam</i>	User parameter, which will be passed to each callback call.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.20.2.12 **LvStatus LvEvent::SetCallbackNewBuffer ( LvEventCallbackNewBufferFunct pFunction, void \* pUserParam )**

Specifies a callback function for the thread of the Event of the [LvEventType\\_NewBuffer](#). Once the application specifies this callback, it becomes responsible for returning the image buffers to the input buffer pool. Note that the callback function cannot be a method of a class.

## Parameters

<i>pFunction</i>	The callback function in the forms of <a href="#">LvEventCallbackNewBufferFunct</a> .
<i>pUserParam</i>	User parameter, which will be passed to each callback call.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.20.2.13 **LvStatus LvEvent::StartThread ( )**

Starts an internal thread, which waits for events and passes them to specified callback function. When the thread is started, the application must no longer call the [LvEvent::WaitAndGetData\(\)](#) or [LvEvent::WaitAndGetNewBufer\(\)](#) functions - this is called internally in the thread and upon return from this function a callback function is called.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.20.2.14 **LvStatus LvEvent::StopThread ( )**

Stops the event internal thread.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.20.2.15 LvStatus LvEvent::WaitAndGetData ( void \* *pBuffer*, size\_t \* *pSize*, uint32\_t *Timeout* = 0xFFFFFFFF )

Waits for the event and gets its data in one atomic operation. Use this function only for events other than LvEvent↔Type\_NewBuffer, for the the LvEventType\_NewBuffer event type use the [LvEvent::WaitAndGetNewBuffer\(\)](#) function instead. Do not use this function if you use the callback - see [LvEvent::SetCallback\(\)](#) or [LvEvent::SetCallbackNew↔Buffer\(\)](#).

##### Parameters

<i>pBuffer</i>	Pointer to a user allocated buffer to receive the event data. The buffer can be parsed by the <a href="#">LvEvent::GetDataInfo()</a> function.
<i>pSize</i>	Size of the buffer must be specified in this parameter and after the function returns, the actual size is returned in this parameter.
<i>Timeout</i>	The wait timeout in milliseconds. The value 0xFFFFFFFF is considered as infinite. Note that you can also kill waiting from another thread using the <a href="#">LvEvent::Kill()</a> function.

##### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.20.2.16 LvStatus LvEvent::WaitAndGetNewBuffer ( LvBuffer \*& *pBuffer*, uint32\_t *Timeout* = 0xFFFFFFFF )

Waits for the event and gets its data in one atomic operation. Use this function only for events of the LvEventType↔\_NewBuffer type. Do not use this function if you use the callback - see [LvEvent::SetCallback\(\)](#) or [LvEvent::Set↔CallbackNewBuffer\(\)](#).

##### Parameters

<i>pBuffer</i>	The pointer to the received <a href="#">LvBuffer</a> instance is returned in this parameter.
<i>Timeout</i>	The wait timeout in milliseconds. The value 0xFFFFFFFF is considered as infinite. Note that you can also kill waiting from another thread using the <a href="#">LvEvent::Kill()</a> function.

##### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.21 SynView LvRenderer methods

### Functions

- static [LvStatus LvRenderer::Open](#) ([LvStream](#) \*pStream, [LvRenderer](#) \*&pRenderer)
- static [LvStatus LvRenderer::Close](#) ([LvRenderer](#) \*&pRenderer)
- [LvStatus LvRenderer::SetWindow](#) (void \*pDisplay, int64\_t hWindow)
- [LvStatus LvRenderer::DisplayImage](#) ([LvBuffer](#) \*pBuffer, uint32\_t RenderFlags=0)
- [LvStatus LvRenderer::Repaint](#) (uint32\_t RenderFlags=0)
- [LvHRenderer LvRenderer::GetHandle](#) ( )

### 5.21.1 Detailed Description

### 5.21.2 Function Documentation

#### 5.21.2.1 static [LvStatus LvRenderer::Close](#) ( [LvRenderer](#) \*& *pRenderer* ) [static]

Deletes the [LvRenderer](#) class instance.

#### Parameters

<i>pRenderer</i>	A pointer to the <a href="#">LvRenderer</a> instance, obtained from the <a href="#">LvRenderer::Open()</a> function. This pointer is set to NULL after close.
------------------	---

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### See also

[LvStream::CloseRenderer\(\)](#).

#### 5.21.2.2 [LvStatus LvRenderer::DisplayImage](#) ( [LvBuffer](#) \* *pBuffer*, uint32\_t *RenderFlags* = 0 )

Displays the image. The image display mode is set by [Renderer](#) features, see [LvRendererFtr](#).

#### Parameters

<i>pBuffer</i>	Pointer to the <a href="#">LvBuffer</a> to be displayed.
<i>RenderFlags</i>	Zero or a combination of <a href="#">LvRenderFlags</a> .

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.21.2.3 [LvHRenderer LvRenderer::GetHandle](#) ( )

Returns a handle of the [Renderer](#) (used in the Plain C API), associated with this class.

#### Returns

The Plain C API handle.

5.21.2.4 `static LvStatus LvRenderer::Open ( LvStream * pStream, LvRenderer *& pRenderer ) [static]`

Creates the [LvRenderer](#) class instance for image display. The renderer attempts to load the `sv.synview.display` library. In case of SynView installation in an operating system without possibility to graphically display (for example Linux without XWindows), the load of this library fails and the calls to `Renderer` functions will return errors.

## Parameters

<i>pStream</i>	A pointer to the <a href="#">LvStream</a> instance, obtained from the <a href="#">LvStream::Open()</a> function.
<i>pRenderer</i>	In this parameter the pointer to the <a href="#">LvRenderer</a> is returned.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## See also

[LvStream::OpenRenderer\(\)](#).

5.21.2.5 [LvStatus](#) [LvRenderer::Repaint \( uint32\\_t RenderFlags = 0 \)](#)

Repaints the contents of the display window. In order to be able to repaint, all images to be displayed must be still held by the application, i.e. must not be returned to the input buffer pool. See also [LvStream\\_LvPostponeQueue↔Buffers](#) feature. A typical usage of this function is in the WM\_PAINT handler in a Windows application.

## Parameters

<i>RenderFlags</i>	Zero or a combination of <a href="#">LvRenderFlags</a> .
--------------------	--

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.21.2.6 [LvStatus](#) [LvRenderer::SetWindow \( void \\* pDisplay, int64\\_t hWindow \)](#)

Sets the target window, in which the renderer has to display. Note that the application itself assure any repainting (when the window need to be repainted due to a movement of overlapping) - use [LvRenderer::Repaint\(\)](#) in such case.

## Parameters

<i>pDisplay</i>	Pointer to the display. It is defined as void* in order to make SynView header files independent on XWindows.
<i>hWindow</i>	Handle to the window.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## 5.22 SynView LvModule methods

### Functions

- [LvStatus LvModule::GetNumFeatures](#) ([LvFtrGroup](#) FtrGroup, [uint32\\_t](#) \*pNumFeatures)
- [LvStatus LvModule::GetFeatureAt](#) ([LvFtrGroup](#) FtrGroup, [uint32\\_t](#) Index, [LvFeature](#) \*pFeature, [uint32\\_t](#) \*pLevel=NULL)
- [LvStatus LvModule::GetFeatureByName](#) ([LvFtrGroup](#) FtrGroup, [const char](#) \*pName, [LvFeature](#) \*pFeature)
- [bool LvModule::IsImplemented](#) ([LvFeature](#) Feature)
- [bool LvModule::IsImplementedByName](#) ([LvEnum](#) FeatureGroup, [const char](#) \*pName)
- [bool LvModule::IsAvailable](#) ([LvFeature](#) Feature)
- [bool LvModule::IsAvailableByName](#) ([LvEnum](#) FeatureGroup, [const char](#) \*pName)
- [bool LvModule::IsReadable](#) ([LvFeature](#) Feature)
- [bool LvModule::IsWritable](#) ([LvFeature](#) Feature)
- [bool LvModule::IsAvailableEnumEntry](#) ([LvFeature](#) Feature, [LvEnum](#) EnumEntry)
- [bool LvModule::IsImplementedEnumEntry](#) ([LvFeature](#) Feature, [LvEnum](#) EnumEntry)
- [LvStatus LvModule::GetType](#) ([LvFeature](#) Feature, [LvFtrType](#) \*pFtrType, [LvFtrGui](#) \*pFtrGui=NULL, [LvFtrGroup](#) \*pFtrGroup=NULL)
- [LvStatus LvModule::GetBool](#) ([LvFeature](#) Feature, [bool](#) \*pValue)
- [LvStatus LvModule::SetBool](#) ([LvFeature](#) Feature, [bool](#) Value)
- [LvStatus LvModule::GetInt32](#) ([LvFeature](#) Feature, [int32\\_t](#) \*pValue)
- [LvStatus LvModule::SetInt32](#) ([LvFeature](#) Feature, [int32\\_t](#) Value)
- [LvStatus LvModule::GetInt32Range](#) ([LvFeature](#) Feature, [int32\\_t](#) \*pMinValue, [int32\\_t](#) \*pMaxValue, [int32\\_t](#) \*pIncrement)
- [LvStatus LvModule::GetInt64](#) ([LvFeature](#) Feature, [int64\\_t](#) \*pValue)
- [LvStatus LvModule::SetInt64](#) ([LvFeature](#) Feature, [int64\\_t](#) Value)
- [LvStatus LvModule::GetInt64Range](#) ([LvFeature](#) Feature, [int64\\_t](#) \*pMinValue, [int64\\_t](#) \*pMaxValue, [int64\\_t](#) \*pIncrement)
- [LvStatus LvModule::GetInt](#) ([LvFeature](#) Feature, [int64\\_t](#) \*pValue)
- [LvStatus LvModule::SetInt](#) ([LvFeature](#) Feature, [int64\\_t](#) Value)
- [LvStatus LvModule::GetIntRange](#) ([LvFeature](#) Feature, [int64\\_t](#) \*pMinValue, [int64\\_t](#) \*pMaxValue, [int64\\_t](#) \*pIncrement)
- [LvStatus LvModule::GetFloat](#) ([LvFeature](#) Feature, [double](#) \*pValue)
- [LvStatus LvModule::SetFloat](#) ([LvFeature](#) Feature, [double](#) Value)
- [LvStatus LvModule::GetFloatRange](#) ([LvFeature](#) Feature, [double](#) \*pMinValue, [double](#) \*pMaxValue, [double](#) \*pIncrement=NULL)
- [LvStatus LvModule::GetString](#) ([LvFeature](#) Feature, [char](#) \*pValue, [size\\_t](#) Size)
- [LvStatus LvModule::GetStringSize](#) ([LvFeature](#) Feature, [size\\_t](#) \*pSize)
- [LvStatus LvModule::GetString](#) ([LvFeature](#) Feature, [std::string](#) &sValue)
- [LvStatus LvModule::SetString](#) ([LvFeature](#) Feature, [const char](#) \*pValue)
- [LvStatus LvModule::GetBuffer](#) ([LvFeature](#) Feature, [void](#) \*pBuffer, [size\\_t](#) Size)
- [LvStatus LvModule::GetBufferSize](#) ([LvFeature](#) Feature, [size\\_t](#) \*pSize)
- [LvStatus LvModule::SetBuffer](#) ([LvFeature](#) Feature, [void](#) \*pBuffer, [size\\_t](#) Size)
- [LvStatus LvModule::GetPtr](#) ([LvFeature](#) Feature, [void](#) \*\*ppValue)
- [LvStatus LvModule::SetPtr](#) ([LvFeature](#) Feature, [void](#) \*pValue)
- [LvStatus LvModule::GetEnum](#) ([LvFeature](#) Feature, [LvEnum](#) \*pValue)
- [LvStatus LvModule::SetEnum](#) ([LvFeature](#) Feature, [LvEnum](#) Value)
- [LvStatus LvModule::GetEnumStr](#) ([LvFeature](#) Feature, [char](#) \*pSymbolicName, [size\\_t](#) Size)
- [LvStatus LvModule::GetEnumStr](#) ([LvFeature](#) Feature, [std::string](#) &sSymbolicName)
- [LvStatus LvModule::SetEnumStr](#) ([LvFeature](#) Feature, [const char](#) \*pSymbolicName)
- [LvStatus LvModule::GetEnumValByStr](#) ([LvFeature](#) Feature, [const char](#) \*pSymbolicName, [LvEnum](#) \*pValue, [LvFtrAccess](#) \*pFtrAccess=NULL)
- [LvStatus LvModule::GetEnumStrByVal](#) ([LvFeature](#) Feature, [LvEnum](#) Value, [char](#) \*pSymbolicName, [size\\_t](#) SymbolicNameSize, [LvFtrAccess](#) \*pFtrAccess=NULL)



- `LvStatus LvModule::GetEnumStrByVal (LvFeature Feature, LvEnum Value, std::string &sSymbolicName, LvFtrAccess *pFtrAccess=NULL)`
- `LvStatus LvModule::CmdExecute (LvFeature Feature, uint32_t Timeout=0)`
- `LvStatus LvModule::CmdIsDone (LvFeature Feature, bool *plsDone)`
- `LvStatus LvModule::GetAccess (LvFeature Feature, LvFtrAccess *pFtrAccess)`
- `LvStatus LvModule::GetVisibility (LvFeature Feature, LvFtrVisibility *pFtrVisibility)`
- `LvStatus LvModule::GetInfo (LvFeature Feature, LvFtrInfo FtrInfo, int32_t *pInfo, int32_t Param=0)`
- `LvStatus LvModule::GetInfoStr (LvFeature Feature, LvFtrInfo FtrInfo, char *pInfoStr, size_t Size, int32_t Param=0)`
- `LvStatus LvModule::GetInfoStrSize (LvFeature Feature, LvFtrInfo FtrInfo, size_t *pSize, int32_t Param=0)`
- `LvStatus LvModule::GetInfoStr (LvFeature Feature, LvFtrInfo FtrInfo, std::string &sInfoStr, int32_t Param=0)`
- `LvStatus LvModule::RegisterFeatureCallback (LvFeature Feature, LvFeatureCallbackFunc pFunction, void *pUserParam=NULL, void *pFeatureParam=NULL)`
- `LvStatus LvModule::StartPollingThread (uint32_t PollingTime=1000, bool PollChildren=false)`
- `LvStatus LvModule::StopPollingThread ()`
- `LvStatus LvModule::Poll ()`

## Variables

- `LvHModule LvModule::m_hModule`

### 5.22.1 Detailed Description

### 5.22.2 Function Documentation

#### 5.22.2.1 `LvStatus LvModule::CmdExecute ( LvFeature Feature, uint32_t Timeout = 0 )`

Executes a command.

#### Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <code>LvModule::GetFeatureByName()</code> function.
<i>Timeout</i>	If greater than 0, the <code>LvModule::CmdIsDone()</code> is called in a loop to wait for the command completion, until the <code>LvModule::CmdIsDone()</code> returns true or the Timeout (in milliseconds) expires. If set to 0, no wait is done.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.22.2.2 `LvStatus LvModule::CmdIsDone ( LvFeature Feature, bool * plsDone )`

Checks if the command execution has completed.

#### Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <code>LvModule::GetFeatureByName()</code> function.
<i>plsDone</i>	In this parameter is returned true, if the command is completed, otherwise false.

#### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

### 5.22.2.3 LvStatus LvModule::GetAccess ( LvFeature *Feature*, LvFtrAccess \* *pFtrAccess* )

Gets the access mode of the feature.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pFtrAccess</i>	The access is returned in this parameter. One of the <a href="#">LvFtrAccess</a> .

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.4 [LvStatus](#) LvModule::GetBool ( [LvFeature](#) *Feature*, [bool](#) \* *pValue* )

Gets a Boolean value.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pValue</i>	The bool value is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.5 [LvStatus](#) LvModule::GetBuffer ( [LvFeature](#) *Feature*, [void](#) \* *pBuffer*, [size\\_t](#) *Size* )

Gets a block of data.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pBuffer</i>	Pointer to a buffer, to which the data will be stored.
<i>Size</i>	Size of the buffer.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.6 [LvStatus](#) LvModule::GetBufferSize ( [LvFeature](#) *Feature*, [size\\_t](#) \* *pSize* )

Gets the block data size.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pSize</i>	The needed size of the buffer is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.7 [LvStatus](#) LvModule::GetEnum ( [LvFeature](#) *Feature*, [LvEnum](#) \* *pValue* )

Gets the SynView constant for the enumeration entry, if exists. If does not exist, you must work with the string enumeration entry value.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pValue</i>	SynView constant for the enum entry is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.8 `LvStatus LvModule::GetEnumStr ( LvFeature Feature, char * pSymbolicName, size_t Size )`

Gets the enumeration entry as a string (symbolic name). It is not possible to get the needed size for this single feature, instead, it is possible to get the maximum size of the all enum values of this feature, by the [LvModule::GetInfo\(LvFtrInfo\\_EnumEntryNameMaxSize\)](#) function.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pSymbolicName</i>	A pointer to a string buffer, where the symbolic name will be returned.
<i>Size</i>	Size of the buffer.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.9 `LvStatus LvModule::GetEnumStr ( LvFeature Feature, std::string & sSymbolicName )`

Gets the enumeration entry as a standard string (symbolic name).

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>sSymbolicName</i>	A string, where the symbolic name will be returned.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.10 `LvStatus LvModule::GetEnumStrByVal ( LvFeature Feature, LvEnum Value, char * pSymbolicName, size_t SymbolicNameSize, LvFtrAccess * pFtrAccess = NULL )`

Returns a string symbolic name of the enum entry for the SynView constant.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>Value</i>	The SynView constant for the enum entry.
<i>pSymbolicName</i>	Pointer to string buffer, where the symbolic name is returned. Can be NULL.

<i>SymbolicName</i> ↔ <i>Size</i>	Size of pSymbolicName buffer.
<i>pFtrAccess</i>	The access mode of the enum entry is returned in this parameter - one of <a href="#">LvFtrAccess</a> . Can be NULL.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.22.2.11 `LvStatus LvModule::GetEnumStrByVal ( LvFeature Feature, LvEnum Value, std::string & sSymbolicName, LvFtrAccess * pFtrAccess = NULL )`

Returns a string symbolic name of the enum entry for the SynView constant.

**Parameters**

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>Value</i>	The SynView constant for the enum entry.
<i>sSymbolicName</i>	In this parameter the symbolic name is returned.
<i>pFtrAccess</i>	The access mode of the enum entry is returned in this parameter - one of <a href="#">LvFtrAccess</a> . Can be NULL.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.22.2.12 `LvStatus LvModule::GetEnumValByStr ( LvFeature Feature, const char * pSymbolicName, LvEnum * pValue, LvFtrAccess * pFtrAccess = NULL )`

Gets the SynView constant for the enumeration entry, if exists.

**Parameters**

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pSymbolicName</i>	A string with symbolic name of the enum entry.
<i>pValue</i>	The SynView constant for the enum entry is returned in this parameter. If the SynView constant does not exist for this enumeration entry, 0 is returned (no error in indicated).
<i>pFtrAccess</i>	The feature access is returned in this parameter - one of <a href="#">LvFtrAccess</a> . Can be NULL.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.22.2.13 `LvStatus LvModule::GetFeatureAt ( LvFtrGroup FtrGroup, uint32_t Index, LvFeature * pFeature, uint32_t * pLevel = NULL )`

Returns the feature ID at specified position. Can be used to iterate all the features in a list.

**Parameters**

<i>FtrGroup</i>	One of the <a href="#">LvFtrGroup</a> .
<i>Index</i>	Zero based index of the feature in the list.
<i>pFeature</i>	Feature ID is returned in this parameter.
<i>pLevel</i>	Feature Level expressing its position in the tree is returned in this parameter. The base level has value 1.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.22.2.14 LvStatus LvModule::GetFeatureByName ( LvFtrGroup *FtrGroup*, const char \* *pName*, LvFeature \* *pFeature* )**

Returns a feature ID based on the feature name. This function is a substantial function for the generic approach to the feature - by this function you can get the ID of any existing feature, that means also for those, for which a SynView constant is not defined. Be sure to check the success of this function - if the feature is not mandatory, it may not exist.

**Parameters**

<i>FtrGroup</i>	One of the <a href="#">LvFtrGroup</a> .
<i>pName</i>	Name of the feature.
<i>pFeature</i>	Feature ID is returned in this parameter.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.22.2.15 LvStatus LvModule::GetFloat ( LvFeature *Feature*, double \* *pValue* )**

Gets a float value.

**Parameters**

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pValue</i>	The float value is returned in this parameter.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.22.2.16 LvStatus LvModule::GetFloatRange ( LvFeature *Feature*, double \* *pMinValue*, double \* *pMaxValue*, double \* *pIncrement* = NULL )**

Returns a range of a float feature.

**Parameters**

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pMinValue</i>	The minimum value is returned in this parameter. Can be NULL.

<i>pMaxValue</i>	The maximum value is returned in this parameter. Can be NULL.
<i>pIncrement</i>	The increment value is returned in this parameter. If the increment is not defined, 0 is returned. Can be NULL.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.22.2.17 LvStatus LvModule::GetInfo ( LvFeature *Feature*, LvFtrInfo *FtrInfo*, int32\_t \* *pInfo*, int32\_t *Param* = 0 )

Gets an info in form of a 32-bit integer value.

**Parameters**

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>FtrInfo</i>	One of the <a href="#">LvFtrInfo</a> .
<i>pInfo</i>	The value is returned in this parameter.
<i>Param</i>	Additional parameter, required by some types of info.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.22.2.18 LvStatus LvModule::GetInfoStr ( LvFeature *Feature*, LvFtrInfo *FtrInfo*, char \* *pInfoStr*, size\_t *Size*, int32\_t *Param* = 0 )

Gets an info in form of a string value.

**Parameters**

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>FtrInfo</i>	One of the <a href="#">LvFtrInfo</a> .
<i>pInfoStr</i>	The string value is returned in this parameter.
<i>Size</i>	Size of the buffer (to which pInfoStr points).
<i>Param</i>	Additional parameter, required by some types of info.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.22.2.19 LvStatus LvModule::GetInfoStr ( LvFeature *Feature*, LvFtrInfo *FtrInfo*, std::string & *sInfoStr*, int32\_t *Param* = 0 )

Gets an info in form of a string value.

**Parameters**

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
----------------	---

<i>FtrInfo</i>	One of the <a href="#">LvFtrInfo</a> .
<i>sInfoStr</i>	The string value is returned in this parameter.
<i>Param</i>	Additional parameter, required by some types of info.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.22.2.20 `LvStatus LvModule::GetInfoStrSize ( LvFeature Feature, LvFtrInfo FtrInfo, size_t * pSize, int32_t Param = 0 )`

Gets a buffer size needed for an info in form of a string value.

**Parameters**

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>FtrInfo</i>	One of the <a href="#">LvFtrInfo</a> .
<i>pSize</i>	Size of the buffer is returned in this parameter.
<i>Param</i>	Additional parameter, required by some types of info.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.22.2.21 `LvStatus LvModule::GetInt ( LvFeature Feature, int64_t * pValue )`

Gets a 64-bit integer value.

**Parameters**

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pValue</i>	The integer value is returned in this parameter.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**Note**

This function is equal to the [LvModule::GetInt64\(\)](#) function.

#### 5.22.2.22 `LvStatus LvModule::GetInt32 ( LvFeature Feature, int32_t * pValue )`

Gets a 32-bit integer value.

**Parameters**

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
----------------	---



<i>pValue</i>	The integer value is returned in this parameter.
---------------	--

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**Note**

The value is internally kept always as a 64-bit value; the functions for setting and getting a 32-bit value are provided just for convenience.

#### 5.22.2.23 **LvStatus LvModule::GetInt32Range ( LvFeature *Feature*, int32\_t \* *pMinValue*, int32\_t \* *pMaxValue*, int32\_t \* *pIncrement* )**

Returns a range and increment of an 32-bit integer feature.

**Parameters**

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pMinValue</i>	The minimum value is returned in this parameter. Can be NULL.
<i>pMaxValue</i>	The maximum value is returned in this parameter. Can be NULL.
<i>pIncrement</i>	The increment value is returned in this parameter. Can be NULL.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**Note**

The value is internally kept always as a 64-bit value; the functions for setting and getting a 32-bit value are provided just for convenience.

#### 5.22.2.24 **LvStatus LvModule::GetInt64 ( LvFeature *Feature*, int64\_t \* *pValue* )**

Gets a 64-bit integer value.

**Parameters**

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pValue</i>	The integer value is returned in this parameter.

**Returns**

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**Note**

This function is equal to the [LvModule::GetInt\(\)](#) function.

#### 5.22.2.25 **LvStatus LvModule::GetInt64Range ( LvFeature *Feature*, int64\_t \* *pMinValue*, int64\_t \* *pMaxValue*, int64\_t \* *pIncrement* )**

Returns a range and increment of an 64-bit integer feature.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pMinValue</i>	The minimum value is returned in this parameter. Can be NULL.
<i>pMaxValue</i>	The maximum value is returned in this parameter. Can be NULL.
<i>pIncrement</i>	The increment value is returned in this parameter. Can be NULL.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

This function is equal to the [LvModule::GetIntRange\(\)](#) function.

**5.22.2.26** `LvStatus LvModule::GetIntRange ( LvFeature Feature, int64_t * pMinValue, int64_t * pMaxValue, int64_t * pIncrement )`

Returns a range and increment of an 64-bit integer feature.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pMinValue</i>	The minimum value is returned in this parameter. Can be NULL.
<i>pMaxValue</i>	The maximum value is returned in this parameter. Can be NULL.
<i>pIncrement</i>	The increment value is returned in this parameter. Can be NULL.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

This function is equal to the [LvModule::GetInt64Range\(\)](#) function.

**5.22.2.27** `LvStatus LvModule::GetNumFeatures ( LvFtrGroup FtrGroup, uint32_t * pNumFeatures )`

Returns a number of features for specified group. This is useful for building a list of all available features (like the tree in lv.explorer).

## Parameters

<i>FtrGroup</i>	One of the <a href="#">LvFtrGroup</a> .
<i>pNumFeatures</i>	The number of features is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

**5.22.2.28** `LvStatus LvModule::GetPtr ( LvFeature Feature, void ** ppValue )`

Gets a pointer.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>ppValue</i>	The pointer is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.29 `LvStatus LvModule::GetString ( LvFeature Feature, char * pValue, size_t Size )`

Gets a string value. If you need first to get the string size, use the [LvModule::GetStringSize\(\)](#) function.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pValue</i>	Pointer to a null-terminated string buffer.
<i>Size</i>	Size of the buffer.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.30 `LvStatus LvModule::GetString ( LvFeature Feature, std::string & sValue )`

Gets a string value as `std::string`.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>sValue</i>	In this parametr the string value is returned.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.31 `LvStatus LvModule::GetStringSize ( LvFeature Feature, size_t * pSize )`

Gets a buffer size needed for a string.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pSize</i>	Size of the buffer (including space for terminating zero) is returned in this parameter.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.32 `LvStatus LvModule::GetType ( LvFeature Feature, LvFtrType * pFtrType, LvFtrGui * pFtrGui = NULL, LvFtrGroup * pFtrGroup = NULL )`

Returns the feature type, GUI representation and group.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pFtrType</i>	The feature type is returned in this parameter. The returned value is one of the <a href="#">LvFtrType</a> . Can be NULL.
<i>pFtrGui</i>	The feature GUI representation is returned in this parameter. The returned value is one of the <a href="#">LvFtrGui</a> . Can be NULL.
<i>pFtrGroup</i>	The feature group, to which the feature belongs. The returned value is one of the <a href="#">LvFtrGroup</a> . Can be NULL.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.33 `LvStatus LvModule::GetVisibility ( LvFeature Feature, LvFtrVisibility * pFtrVisibility )`

Gets the feature visibility (beginner-expert-guru).

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pFtrVisibility</i>	The visibility is returned in this parameter. One of the <a href="#">LvFtrVisibility</a> .

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.34 `bool LvModule::IsAvailable ( LvFeature Feature )`

A helper function, allowing simply to determine, if a feature is available. It is a wrapper around the [LvModule::GetAccess\(\)](#) function.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
----------------	---

## Returns

If the feature is available, returns true, otherwise false.

5.22.2.35 `bool LvModule::IsAvailableByName ( LvEnum FeatureGroup, const char * pName )`

A helper function, allowing simply to determine, if a feature is available. It is a wrapper around the [LvModule::GetAccess\(\)](#) and [LvModule::GetFeatureByName\(\)](#) functions.

## Parameters

<i>FeatureGroup</i>	One of the <a href="#">LvFtrGroup</a> .
<i>pName</i>	Name of the feature.

## Returns

If the feature is available, returns true, otherwise false.

5.22.2.36 `bool LvModule::IsAvailableEnumEntry ( LvFeature Feature, LvEnum EnumEntry )`

A helper function, allowing simply to determine, if an enum entry of an enum feature is available.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>EnumEntry</i>	The SynView constant for the enum entry.

## Returns

If the enum entry is available, returns true, otherwise false.

5.22.2.37 `bool LvModule::IsImplemented ( LvFeature Feature )`

A helper function, allowing simply to determine, if a feature is implemented. It is a wrapper around the [LvModule::GetAccess\(\)](#) function.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
----------------	---

## Returns

If the feature is implemented, returns true, otherwise false.

5.22.2.38 `bool LvModule::IsImplementedByName ( LvEnum FeatureGroup, const char * pName )`

A helper function, allowing simply to determine, if a feature is implemented. It is a wrapper around the [LvModule::GetAccess\(\)](#) and [LvModule::GetFeatureByName\(\)](#) functions.

## Parameters

<i>FeatureGroup</i>	One of the <a href="#">LvFtrGroup</a> .
<i>pName</i>	Name of the feature.

## Returns

If the feature is implemented, returns true, otherwise false.

5.22.2.39 `bool LvModule::IsImplementedEnumEntry ( LvFeature Feature, LvEnum EnumEntry )`

A helper function, allowing simply to determine, if an enum entry of an enum feature is implemented.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>EnumEntry</i>	The SynView constant for the enum entry.

## Returns

If the enum entry is implemented, returns true, otherwise false.

5.22.2.40 `bool LvModule::IsReadable ( LvFeature Feature )`

A helper function, allowing simply to determine, if a feature is readable. It is a wrapper around the [LvModule::GetAccess\(\)](#) function.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
----------------	---

## Returns

If the feature is readable, returns true, otherwise false.

5.22.2.41 `bool LvModule::IsWritable ( LvFeature Feature )`

A helper function, allowing simply to determine, if a feature is writable. It is a wrapper around the [LvModule::GetAccess\(\)](#) function.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
----------------	---

## Returns

If the feature is writable, returns true, otherwise false.

5.22.2.42 `LvStatus LvModule::Poll ( )`

Polls all the non-cached features of the module. If the feature polling interval expires, the value is read and the feature callback is called.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.43 `LvStatus LvModule::RegisterFeatureCallback ( LvFeature Feature, LvFeatureCallbackFunct pFunction, void * pUserParam = NULL, void * pFeatureParam = NULL )`

Registers or unregisters a callback function for the feature. This callback is produced by GenApi when a feature changes its value or status. The application should process this callback fast. Note that the callback can be called also from another thread - see [LvEventType\\_FeatureDevEvent](#). Important note: The feature callback function should never set any other feature. Doing so can lead to recursions, which would be probably hard to diagnose and could cause unexpected behavior.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pFunction</i>	The callback function in the form of <a href="#">LvFeatureCallbackFunct</a> . If you want to unregister the function, use NULL at this parameter.
<i>pUserParam</i>	User parameter, which will be passed to each callback call.
<i>pFeatureParam</i>	Second user parameter, which will be passed to each callback call.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.44 `LvStatus LvModule::SetBool ( LvFeature Feature, bool Value )`

Sets a Boolean value.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>Value</i>	Value to be set.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.45 `LvStatus LvModule::SetBuffer ( LvFeature Feature, void * pBuffer, size_t Size )`

Sets a block of data.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pBuffer</i>	Pointer to the data.
<i>Size</i>	Size of the data.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.46 `LvStatus LvModule::SetEnum ( LvFeature Feature, LvEnum Value )`

Sets the enumeration entry by the SynView constant. If the SynView constant is not defined for the feature, then use [LvModule::SetEnumStr\(\)](#) to set the enum entry by a string.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>Value</i>	SynView constant for the requested enumeration entry.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.47 `LvStatus LvModule::SetEnumStr ( LvFeature Feature, const char * pSymbolicName )`

Sets enumeration entry by its string symbolic name.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pSymbolicName</i>	A string with the symbolic name of the enumeration entry.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.48 `LvStatus LvModule::SetFloat ( LvFeature Feature, double Value )`

Sets a float value.



## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>Value</i>	The value to be set.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

5.22.2.49 [LvStatus](#) [LvModule::SetInt](#) ( [LvFeature](#) *Feature*, [int64\\_t](#) *Value* )

Sets a 64-bit integer value.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>Value</i>	Value to be set.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

This function is equal to the [LvModule::SetInt64\(\)](#) function.

5.22.2.50 [LvStatus](#) [LvModule::SetInt32](#) ( [LvFeature](#) *Feature*, [int32\\_t](#) *Value* )

Sets a 32-bit value.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>Value</i>	Value to be set.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

The value is internally kept always as a 64-bit value; the functions for setting and getting a 32-bit value are provided just for convenience.

5.22.2.51 [LvStatus](#) [LvModule::SetInt64](#) ( [LvFeature](#) *Feature*, [int64\\_t](#) *Value* )

Sets a 64-bit integer value.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>Value</i>	Value to be set.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

## Note

This function is equal to the [LvModule::SetInt\(\)](#) function.

#### 5.22.2.52 `LvStatus LvModule::SetPtr ( LvFeature Feature, void * pValue )`

Sets a pointer.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pValue</i>	The pointer to be set.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.22.2.53 `LvStatus LvModule::SetString ( LvFeature Feature, const char * pValue )`

Sets a string value.

## Parameters

<i>Feature</i>	The feature ID - use a symbolic constant (one of the <a href="#">Features</a> ) or an ID obtained by the <a href="#">LvModule::GetFeatureByName()</a> function.
<i>pValue</i>	The string value (null-terminated).

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.22.2.54 `LvStatus LvModule::StartPollingThread ( uint32_t PollingTime = 1000, bool PollChildren = false )`

Starts a thread, which in a loop polls the non-cached features. If the feature polling interval expires, the value is read and the feature callback is called.

## Parameters

<i>PollingTime</i>	A time in milliseconds between 2 calls to poll the features.
<i>PollChildren</i>	If set to true, also the features in all children modules are polled. For example, if your application uses only one System module, then it is a parent of all other modules, so the polling will be propagated to all modules from a single thread. If a module has started own polling thread, then it is excluded from the propagating.

## Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

#### 5.22.2.55 LvStatus LvModule::StopPollingThread ( )

Stops the polling thread. See [LvModule::StartPollingThread\(\)](#) for details.

##### Returns

Returns the [LvStatus](#) value indicating the success of the requested operation. See [LvStatus definitions](#).

### 5.22.3 Variable Documentation

#### 5.22.3.1 LvHModule LvModule::m\_hModule [protected]

The base class handle.

## 5.23 SynView

### Modules

- [SynView Plain C API functions](#)
- [SynView C++ API functions](#)
- [SynView defines and typedefs](#)
- [SynView enumerations](#)
- [SynView Image Processing Library](#)
- [SynView INI file API](#)

### 5.23.1 Detailed Description

## 5.24 SynView defines and typedefs

### Modules

- [LvStreamStart\(\)](#) flags definitions
- [LvStreamStop\(\)](#) flags definitions
- [LvDeviceUniSetLut\(\)](#) and [LvDeviceUniGetLut\(\)](#) flags definitions
- [LvSaveFlag](#) definitions
- [LvPixelFormat](#) definitions
- [LvStatus](#) definitions

### Macros

- `#define LV_DLEENTRY`
- `#define LVIP_DLEENTRY`

### Typedefs

- `typedef uint32_t LvHModule`
- `typedef void(* LvEventCallbackFunct)(void *pBuffer, size_t Size, void *pUserParam)`
- `typedef void(* LvEventCallbackNewBufferFunct)(LvHBuffer hBuffer, void *pUserPointer, void *pUserParam)`
- `typedef void(* LvFeatureCallbackFunct)(void *pUserParam, void *pFeatureParam, const char *pName)`

#### 5.24.1 Detailed Description

#### 5.24.2 Macro Definition Documentation

##### 5.24.2.1 `#define LV_DLEENTRY`

Typedef for the API functions calling convention and export/import type.

##### 5.24.2.2 `#define LVIP_DLEENTRY`

Typedef for the API functions calling convention and export/import type.

#### 5.24.3 Typedef Documentation

##### 5.24.3.1 `typedef void(* LvEventCallbackFunct)(void *pBuffer, size_t Size, void *pUserParam)`

Prototype for the general callback function, which can be registered at the Event, using the [LvEventSetCallback\(\)](#) function or [LvEvent::SetCallback\(\)](#) class. IMPORTANT: the function must have the LV\_STDC calling convention.

##### Parameters

<i>pBuffer</i>	Pointer to buffer, extracted from the output queue.
<i>Size</i>	Buffer size.
<i>pUserParam</i>	User parameter, supplied in the <a href="#">LvEventSetCallback()</a> function or <a href="#">LvEvent::SetCallback()</a> . It enables the application to distinguish from which object the callback was called in case the same callback function is shared by multiple Event modules.

### 5.24.3.2 typedef void(\* LvEventCallbackNewBufferFunc)(LvHBuffer hBuffer, void \*pUserPointer, void \*pUserParam)

Prototype for the new image callback function, which can be registered at the Event, using the [LvEventSetCallbackNewBuffer\(\)](#) function or [LvEvent::SetCallbackNewBuffer\(\)](#) class. IMPORTANT: the function must have the LV\_STDC calling convention.

#### Parameters

<i>hBuffer</i>	Handle to <a href="#">LvBuffer</a> , extracted from the output queue.
<i>pUserPointer</i>	The pUserPointer of the <a href="#">LvBuffer</a> is passed here. In the C++ wrapper class this is used to give direct pointer to the <a href="#">LvBuffer</a> class instance.
<i>pUserParam</i>	User parameter, supplied in the <a href="#">LvEventSetCallbackNewBuffer()</a> function or <a href="#">LvEvent::SetCallbackNewBuffer()</a> . It enables the application to distinguish from which object the callback was called in case the same callback function is shared by multiple Event modules.

### 5.24.3.3 typedef void(\* LvFeatureCallbackFunc)(void \*pUserParam, void \*pFeatureParam, const char \*pName)

Prototype for the feature updated callback function, which can be registered using the [LvRegisterFeatureCallback\(\)](#) function. IMPORTANT: the function must have the LV\_STDC calling convention.

#### Parameters

<i>pUserParam</i>	User pUserParam, supplied in the <a href="#">LvRegisterFeatureCallback()</a> function. It can be used to distinguish from which object the callback was called in case the same callback function is shared by multiple Event modules.
<i>pFeatureParam</i>	The pFeatureParam passed in the <a href="#">LvRegisterFeatureCallback()</a> . It is usually used to identify the feature, which has changed.
<i>pName</i>	The string ID of the feature.

### 5.24.3.4 typedef uint32\_t LvHModule

Base typedef for a handle to a module.

## 5.25 SynView enumerations

### Modules

- [Features](#)
- [Enumeration entries](#)

### Enumerations

- enum `LvLiblInfo` {  
`LvInfo_BinPath`, `LvInfo_AppDataPath`, `LvInfo_UserDataPath`, `LvInfo_CfgPath`,  
`LvInfo_InstPath`, `LvInfo_IniFile`, `LvInfo_BuildDate` }
- enum `LvFtrGroup` {  
`LvFtrGroup_DeviceRemote`, `LvFtrGroup_SystemGtl`, `LvFtrGroup_InterfaceGtl`, `LvFtrGroup_DeviceGtl`,  
`LvFtrGroup_StreamGtl`, `LvFtrGroup_BufferGtl`, `LvFtrGroup_SystemLocal`, `LvFtrGroup_InterfaceLocal`,  
`LvFtrGroup_DeviceLocal`, `LvFtrGroup_StreamLocal`, `LvFtrGroup_BufferLocal`, `LvFtrGroup_RendererLocal`,  
`LvFtrGroup_EventLocal`, `LvFtrGroup_BufferItemsGtl`, `LvFtrGroup_EventItemsGtl`, `LvFtrGroup_System`↵  
`Hidden`,  
`LvFtrGroup_InterfaceHidden`, `LvFtrGroup_DeviceHidden`, `LvFtrGroup_StreamHidden`, `LvFtrGroup_Buffer`↵  
`Hidden`,  
`LvFtrGroup_RendererHidden`, `LvFtrGroup_EventHidden` }
- enum `LvFtrType` {  
`LvFtrType_Integer`, `LvFtrType_Float`, `LvFtrType_String`, `LvFtrType_Enumeration`,  
`LvFtrType_Boolean`, `LvFtrType_Command`, `LvFtrType_Category`, `LvFtrType_StringList`,  
`LvFtrType_Pointer`, `LvFtrType_Buffer`, `LvFtrType_Other` }
- enum `LvFtrGui` {  
`LvFtrGui_IntEdit`, `LvFtrGui_IntEditHex`, `LvFtrGui_IntSlider`, `LvFtrGui_IntSliderLog`,  
`LvFtrGui_FloatEdit`, `LvFtrGui_FloatSlider`, `LvFtrGui_FloatSliderLog`, `LvFtrGui_Label`,  
`LvFtrGui_StringEdit`, `LvFtrGui_CheckBox`, `LvFtrGui_ComboBox`, `LvFtrGui_Button`,  
`LvFtrGui_IpV4Address`, `LvFtrGui_IpMacAddress`, `LvFtrGui_Undefined` }
- enum `LvFtrVisibility` { `LvFtrVisibility_Beginner`, `LvFtrVisibility_Expert`, `LvFtrVisibility_Guru`, `LvFtrVisibility`↵  
`Invisible` }
- enum `LvFtrAccess` {  
`LvFtrAccess_NotImplemented`, `LvFtrAccess_NotAvailable`, `LvFtrAccess_WriteOnly`, `LvFtrAccess_ReadOnly`,  
`LvFtrAccess_ReadWrite` }
- enum `LvFtrInfo` {  
`LvFtrInfo_IsStreamable`, `LvFtrInfo_IsWrapped`, `LvFtrInfo_IsSelector`, `LvFtrInfo_IsCached`,  
`LvFtrInfo_PollingTime`, `LvFtrInfo_Name`, `LvFtrInfo_DisplayName`, `LvFtrInfo_Description`,  
`LvFtrInfo_PhysicalUnits`, `LvFtrInfo_ToolTip`, `LvFtrInfo_SymbolicConst`, `LvFtrInfo_SymbolicEnumConst`,  
`LvFtrInfo_SelectedFeatures`, `LvFtrInfo_SelectingFeatures`, `LvFtrInfo_SymbolicGroupConst`, `LvFtrInfo`↵  
`ModuleName`,  
`LvFtrInfo_FitsTo32Bit`, `LvFtrInfo_TakeAsReadOnly`, `LvFtrInfo_EnumEntryName`, `LvFtrInfo_EnumEntry`↵  
`DisplayName`,  
`LvFtrInfo_EnumEntryDescription`, `LvFtrInfo_EnumEntryToolTip`, `LvFtrInfo_EnumEntryAccess`, `LvFtrInfo`↵  
`EnumEntryValue`,  
`LvFtrInfo_EnumEntryCount`, `LvFtrInfo_EnumEntryNameMaxSize`, `LvFtrInfo_InterfaceID`, `LvFtrInfo`↵  
`InterfaceDisplayName`,  
`LvFtrInfo_InterfaceTIType`, `LvFtrInfo_DeviceID`, `LvFtrInfo_DeviceVendor`, `LvFtrInfo_DeviceModel`,  
`LvFtrInfo_DeviceTIType`, `LvFtrInfo_DeviceDisplayName`, `LvFtrInfo_DeviceAccessStatus` }
- enum `LvInfoDataType` {  
`LvInfoDataType_Unknown`, `LvInfoDataType_String`, `LvInfoDataType_StringList`, `LvInfoDataType_Int16`,  
`LvInfoDataType_UInt16`, `LvInfoDataType_Int32`, `LvInfoDataType_UInt32`, `LvInfoDataType_Int64`,  
`LvInfoDataType_UInt64`, `LvInfoDataType_Float64`, `LvInfoDataType_Ptr`, `LvInfoDataType_Bool`,  
`LvInfoDataType_SizeT`, `LvInfoDataType_Buffer` }
- enum `LvQueueOperation` {  
`LvQueueOperation_InputToOutput`, `LvQueueOperation_OutputDiscard`, `LvQueueOperation_AllToInput`, `Lv`↵

- QueueOperation\_UnqueuedToInput,  
LvQueueOperation\_AIIDiscard }
- enum LvEventType {  
LvEventType\_Error, LvEventType\_NewBuffer, LvEventType\_FeatureInvalidate, LvEventType\_FeatureChange,  
LvEventType\_FeatureDevEvent, **LvEventType\_RemoteDevice**, LvEventType\_Module }
- enum LvEventDataInfo { LvEventDataInfo\_Id, LvEventDataInfo\_Value }
- enum LvRenderFlags { LvRenderFlags\_RepaintBackground, LvRenderFlags\_DontPaintIncomplete, LvRenderFlags\_IgnoreInvalidWinHandle }
- enum LvFindBy {  
LvFindBy\_UserID, LvFindBy\_VendorName, LvFindBy\_ModelName, LvFindBy\_TLType,  
LvFindBy\_DisplayName, LvFindBy\_GevIPAddress, LvFindBy\_GevMACAddress, LvFindBy\_SerialNumber,  
LvFindBy\_Any }

### 5.25.1 Detailed Description

### 5.25.2 Enumeration Type Documentation

#### 5.25.2.1 enum LvEventDataInfo

LvEventDataInfo constants. Define values for the info specification in the [LvEventGetDataInfo\(\)](#) function.

#### Enumerator

**LvEventDataInfo\_Id** Represents the GenTL EVENT\_DATA\_ID - Event ID. See [LvEventType](#) for the explanation, what this ID means according to the event type.

**LvEventDataInfo\_Value** Represents the GenTL EVENT\_DATA\_VALUE - Event Data. See [LvEventType](#) for the explanation, what this data means according to the event type.

#### 5.25.2.2 enum LvEventType

LvEventType constants. Currently only the [LvEventType\\_NewBuffer](#) is supported by SynView.

#### Enumerator

**LvEventType\_Error** Represents the GenTL EVENT\_ERROR - Notification on module errors. For this type of event the [LvEventDataInfo\\_Id](#) is [LvInfoDataType\\_Int32](#) and [LvEventDataInfo\\_Value](#) is [LvInfoDataType\\_String](#).

**LvEventType\_NewBuffer** Represents the GenTL EVENT\_NEW\_BUFFER - Notification on newly filled buffers placed to the output queue. For this type of event the [LvEventDataInfo\\_Id](#) is [LvInfoDataType\\_Ptr](#) (GenTL Buffer handle) and [LvEventDataInfo\\_Value](#) is [LvInfoDataType\\_Ptr](#) (Private pointer).

**LvEventType\_FeatureInvalidate** Represents the GenTL EVENT\_FEATURE\_INVALIDATE - Notification if a feature was changed by the GenTL Producer library and thus needs to be invalidated in the GenICam GenApi instance using the module.

**LvEventType\_FeatureChange** Represents the GenTL EVENT\_FEATURE\_CHANGE - Notification if the GenTL Producer library wants to manually set a feature in the GenICam GenApi instance using the module.

**LvEventType\_FeatureDevEvent** Represents the GenTL EVENT\_FEATURE\_DEVEVENT - Notification if the GenTL Producer wants to inform the GenICam GenApi instance of the remote device that a GenApi compatible event was fired. This event is processed internally in SynView API - it is converted into the feature change callback - see the [LvModule::RegisterFeatureCallback\(\)](#) function. However, the thread which checks the GenTL event and converts it into the callbacks must be started explicitly by the application - see the [LvEventStartThread\(\)](#) function. This event type can be opened only on the Device module.

**LvEventType\_Module** Represents the GenTL EVENT\_MODULE - Notification if the GenTL Producer wants to inform the GenICam GenApi instance of the module that a GenApi compatible event was fired.



## 5.25.2.3 enum LvFindBy

Enum values for the [LvSystemFindInterface\(\)](#) and [LvInterfaceFindDevice\(\)](#) functions.

## Enumerator

**LvFindBy\_UserID** Can be used in the [LvInterfaceFindDevice\(\)](#) for finding the device by its User ID (nickname).

**LvFindBy\_VendorName** Can be used in the [LvInterfaceFindDevice\(\)](#) for finding the device by its vendor name.

**LvFindBy\_ModelName** Can be used in the [LvInterfaceFindDevice\(\)](#) for finding the device by its model name.

**LvFindBy\_TLType** Can be used in the [LvSystemFindInterface\(\)](#) or [LvInterfaceFindDevice\(\)](#) for finding the interface or device by its Transport Layer type. The search string can be then one of the following:

- "GEV" for GigE Vision,
- "CL" for Camera Link,
- "IIDC" for IIDC 1394,
- "UVC" for USB video class devices,
- "Custom" for not defined ones (for example the New Electronic Technology CorSight streaming device).

**LvFindBy\_DisplayName** Can be used in the [LvSystemFindInterface\(\)](#) or [LvInterfaceFindDevice\(\)](#) for finding the interface or device by its display name.

**LvFindBy\_GevIPAddress** Can be used in the [LvSystemFindInterface\(\)](#) or [LvInterfaceFindDevice\(\)](#) for finding the interface or device by its IP address (in case of the interface, it is the default IP address of the NIC).

**LvFindBy\_GevMACAddress** Can be used in the [LvInterfaceFindDevice\(\)](#) for finding the device by its model name.

**LvFindBy\_SerialNumber** Can be used in the [LvInterfaceFindDevice\(\)](#) for finding the device by its serial number.

**LvFindBy\_Any** Tries to find the string in all available IDs (UserID, VendorName, ModelName...).

## 5.25.2.4 enum LvFtrAccess

LvFtrAccess constants. Define the current feature access mode. Used in the [LvGetAccess\(\)](#). Also used for enumeration features in functions [LvGetEnumValByStr\(\)](#) and [LvGetEnumStrByVal\(\)](#).

## Enumerator

**LvFtrAccess\_NotImplemented** The feature is not implemented at all.

**LvFtrAccess\_NotAvailable** The feature is implemented, but under the current conditions is not available.

**LvFtrAccess\_WriteOnly** The feature is available and is write only.

**LvFtrAccess\_ReadOnly** The feature is available and is read only.

**LvFtrAccess\_ReadWrite** The feature is available and is fully accessible.

## 5.25.2.5 enum LvFtrGroup

LvFtrGroup constants. Define the group of features. The group is composed of the module and the feature origin. The richest set is belonging to the Device module:

- Device remote features are those, which are provided by the device itself through GenICam GenApi.
- Device GenTL features are those, which are provided by the GenTL library through GenICam GenApi.

- Device local features are those, which are implemented directly in the SynView library. Used in [LvGetNumFeatures\(\)](#), [LvGetFeatureAt\(\)](#), [LvGetFeatureByName\(\)](#).

#### Enumerator

- LvFtrGroup\_DeviceRemote** Device remote features obtained from the device GenApi node tree.
- LvFtrGroup\_SystemGtl** System features obtained from the GenTL GenApi node tree.
- LvFtrGroup\_InterfaceGtl** Interface features obtained from the GenTL GenApi node tree.
- LvFtrGroup\_DeviceGtl** Device features obtained from the GenTL GenApi node tree.
- LvFtrGroup\_StreamGtl** Stream features obtained from the GenTL GenApi node tree.
- LvFtrGroup\_BufferGtl** Buffer features obtained from the GenTL GenApi node tree.
- LvFtrGroup\_SystemLocal** System local features, implemented in SynView.
- LvFtrGroup\_InterfaceLocal** Interface local features, implemented in SynView.
- LvFtrGroup\_DeviceLocal** Device local features, implemented in SynView.
- LvFtrGroup\_StreamLocal** Stream local features, implemented in SynView.
- LvFtrGroup\_BufferLocal** Buffer local features, implemented in SynView.
- LvFtrGroup\_RendererLocal** Renderer local features, implemented in SynView.
- LvFtrGroup\_EventLocal** Event local features, implemented in SynView.
- LvFtrGroup\_BufferItemsGtl** Obsolete - will be removed. Buffer local GenTL features obtained from the GenTL plain C API.
- LvFtrGroup\_EventItemsGtl** Obsolete - will be removed. Event local GenTL features obtained from the GenTL plain C API.
- LvFtrGroup\_SystemHidden** System hidden features. Do not use, reserved for special purposes.
- LvFtrGroup\_InterfaceHidden** Interface hidden features. Do not use, reserved for special purposes.
- LvFtrGroup\_DeviceHidden** Device hidden features. Do not use, reserved for special purposes.
- LvFtrGroup\_StreamHidden** Stream hidden features. Do not use, reserved for special purposes.
- LvFtrGroup\_BufferHidden** Buffer hidden features. Do not use, reserved for special purposes.
- LvFtrGroup\_RendererHidden** Renderer hidden features. Do not use, reserved for special purposes.
- LvFtrGroup\_EventHidden** Event hidden features. Do not use, reserved for special purposes.

#### 5.25.2.6 enum LvFtrGui

LvFtrGui constants. Define the recommended GUI representation of the feature. Used in the [LvGetType\(\)](#) function.

#### Enumerator

- LvFtrGui\_IntEdit** The recommended representation is an edit box with a decimal value. Used by [LvFtrType\\_Integer](#).
- LvFtrGui\_IntEditHex** The recommended representation is an edit box with a hexadecimal value. Used by [LvFtrType\\_Integer](#).
- LvFtrGui\_IntSlider** The recommended representation is a linear slider. Used by [LvFtrType\\_Integer](#).
- LvFtrGui\_IntSliderLog** The recommended representation is a logarithmic slider. Used by [LvFtrType\\_Integer](#).
- LvFtrGui\_FloatEdit** The recommended representation is an edit box. Used by [LvFtrType\\_Float](#).
- LvFtrGui\_FloatSlider** The recommended representation is a linear slider. Used by [LvFtrType\\_Float](#).
- LvFtrGui\_FloatSliderLog** The recommended representation is a logarithmic slider. Used by [LvFtrType\\_Float](#).
- LvFtrGui\_Label** The recommended representation is read-only label. Used by [LvFtrType\\_Category](#).

- LvFtrGui\_StringEdit** The recommended representation is an edit box for a string. Used by [LvFtrType\\_String](#).
- LvFtrGui\_CheckBox** The recommended representation is a check box. Used by [LvFtrType\\_Boolean](#).
- LvFtrGui\_ComboBox** The recommended representation is a combo box. Used by [LvFtrType\\_Boolean](#).
- LvFtrGui\_Button** The recommended representation is a button. Used by [LvFtrType\\_Command](#).
- LvFtrGui\_IpV4Address** The recommended representation is an edit box for a string with an IP address in the form N.N.N.N. Used by [LvFtrType\\_Integer](#).
- LvFtrGui\_IpMacAddress** The recommended representation is an edit box for a string with a MAC address in the form XX:XX:XX:XX:XX:XX. Used by [LvFtrType\\_Integer](#).
- LvFtrGui\_Undefined** The recommended representation is not defined.

#### 5.25.2.7 enum LvFtrInfo

LvFtrInfo constants. Define the info type when querying for feature info by the [LvGetInfo\(\)](#) and [LvGetInfoStr\(\)](#) functions.

##### Enumerator

- LvFtrInfo\_IsStreamable** Returns 1 if the feature has the Streamable attribute set. To be used in the [LvGetInfo\(\)](#) function.
- LvFtrInfo\_IsWrapped** Returns 1 if the feature should not be used directly, because SynView provides for this functionality a native API. For example the AcquisitionStart and AcquisitionStop device remote features are wrapped by additional functionality in SynView (for example locking TL params before the AcquisitionStart command is issued). To be used in the [LvGetInfo\(\)](#) function.
- LvFtrInfo\_IsSelector** Returns 1 if the feature is a selector, that means subsequent features are indexed by it. To be used in the [LvGetInfo\(\)](#) function.
- LvFtrInfo\_IsCached** Returns 1 if the feature is cached. To be used in the [LvGetInfo\(\)](#) function.
- LvFtrInfo\_PollingTime** Returns the polling time for a non-cached feature. If the feature is dependent on other non-cached features, the returned polling time is the minimum found. The polling time defines recommended time to update the non-cached feature. For example the LvDevice\_DeviceTemperature is a typical non-cached feature - it changes independently and as it changes slowly, the recommended polling time might be 10000 = 10 seconds, i.e. the application, which displays the temperature, should update it on screen every 10 seconds. The returned value -1 means the polling time is not defined. To be used in the [LvGetInfo\(\)](#) function.
- LvFtrInfo\_Name** Returns the feature Name. Do not confuse it with the DisplayName - the Name is the string identifier, by which the feature can be identified and a numeric ID can be obtained for further actions (generic feature access). To be used in the [LvGetInfoStr\(\)](#) function.
- LvFtrInfo\_DisplayName** Returns the feature Display name for representation in GUI. To be used in the [LvGetInfoStr\(\)](#) function.
- LvFtrInfo\_Description** Returns the feature Description text. To be used in the [LvGetInfoStr\(\)](#) function.
- LvFtrInfo\_PhysicalUnits** Returns the feature Physical units, if defined. To be used in the [LvGetInfoStr\(\)](#) function.
- LvFtrInfo\_ToolTip** Returns the feature Tooltip (a short description to be used in the GUI). To be used in the [LvGetInfoStr\(\)](#) function.
- LvFtrInfo\_SymbolicConst** Returns the SynView symbolic constant of the feature, as a string (utilized in the Source code generator). To be used in the [LvGetInfoStr\(\)](#) function.
- LvFtrInfo\_SymbolicEnumConst** Returns the SynView symbolic constant of the enumeration feature, as a string (utilized in the Source code generator). To be used in the [LvGetInfoStr\(\)](#) function.
- LvFtrInfo\_SelectedFeatures** Returns the string ID of selected features belonging under this selector. Param = index (utilized in the Source code generator). To be used in the [LvGetInfoStr\(\)](#) function.
- LvFtrInfo\_SelectingFeatures** Returns the string ID of selecting features under which this feature belongs. Param = index (utilized in the Source code generator). To be used in the [LvGetInfoStr\(\)](#) function.

- LvFtrInfo\_SymbolicGroupConst** Returns the SynView symbolic constant for the feature group, to which the feature belongs, as a string (utilized in the Source code generator). To be used in the [LvGetInfoStr\(\)](#) function.
- LvFtrInfo\_ModuleName** Returns the string indicating the type of module, to which the feature belongs, for example "System", "Interface", "Device", ... (utilized in the Source code generator). To be used in the [LvGetInfoStr\(\)](#) function.
- LvFtrInfo\_FitsTo32Bit** Returns 1 if for this feature can be safely used 32-bit integer instead of 64-bit (if the feature is of the LvFtrType\_Integer type). This info is utilized in the source code generator. To be used in the [LvGetInfo\(\)](#) function.
- LvFtrInfo\_TakeAsReadOnly** Returns 1 if this feature is either permanently read-only (cannot become read-write depending on other features), or the feature is writable, but it is not usual to set its value from code. This info is utilized in the source code generator. To be used in the [LvGetInfo\(\)](#) function.
- LvFtrInfo\_EnumEntryName** Returns the symbolic name of the enum entry. To be used in the [LvGetInfoStr\(\)](#) function. The Param specifies a zero based index of the entry or the SynView enum entry constant. You can obtain the number of entries by the [LvGetInfo\(\)](#) function with the [LvFtrInfo\\_EnumEntryCount](#) parameter. If the Param is set to [LV\\_ENUMENTRY\\_CURRENT](#), the returned info is for the currently selected enum entry.
- LvFtrInfo\_EnumEntryDisplayName** Returns the display name of the enum entry. To be used in the [LvGetInfoStr\(\)](#) function. The Param specifies a zero based index of the entry or the SynView enum entry constant. You can obtain the number of entries by the [LvGetInfo\(\)](#) function with the [LvFtrInfo\\_EnumEntryCount](#) parameter. If the Param is set to [LV\\_ENUMENTRY\\_CURRENT](#), the returned info is for the currently selected enum entry.
- LvFtrInfo\_EnumEntryDescription** Returns the description of the enum entry. To be used in the [LvGetInfoStr\(\)](#) function. The Param specifies a zero based index of the entry or the SynView enum entry constant. You can obtain the number of entries by the [LvGetInfo\(\)](#) function with the [LvFtrInfo\\_EnumEntryCount](#) parameter. If the Param is set to [LV\\_ENUMENTRY\\_CURRENT](#), the returned info is for the currently selected enum entry.
- LvFtrInfo\_EnumEntryToolTip** Returns the tooltip of the enum entry. To be used in the [LvGetInfoStr\(\)](#) function. The Param specifies a zero based index of the entry or the SynView enum entry constant. You can obtain the number of entries by the [LvGetInfo\(\)](#) function with the [LvFtrInfo\\_EnumEntryCount](#) parameter. If the Param is set to [LV\\_ENUMENTRY\\_CURRENT](#), the returned info is for the currently selected enum entry.
- LvFtrInfo\_EnumEntryAccess** Returns the access of the enum entry (one of the [LvFtrAccess](#) constants). To be used in the [LvGetInfo\(\)](#) function. The Param specifies a zero based index of the entry or the SynView enum entry constant. You can obtain the number of entries by the [LvGetInfo\(\)](#) function with the [LvFtrInfo\\_EnumEntryCount](#) parameter. If the Param is set to [LV\\_ENUMENTRY\\_CURRENT](#), the returned info is for the currently selected enum entry.
- LvFtrInfo\_EnumEntryValue** Returns the SynView constant for the enum entry (if exists). To be used in the [LvGetInfo\(\)](#) function. The Param specifies a zero based index of the entry. You can obtain the number of entries by the [LvGetInfo\(\)](#) function with the [LvFtrInfo\\_EnumEntryCount](#) parameter. If the Param is set to [LV\\_ENUMENTRY\\_CURRENT](#), the returned info is for the currently selected enum entry.
- LvFtrInfo\_EnumEntryCount** Returns the number of enum entries for the enum. To be used in the [LvGetInfo\(\)](#) function.
- LvFtrInfo\_EnumEntryNameMaxSize** Returns the maximum string size needed (including terminating zero) for any entry name of the enum To be used in the [LvGetInfo\(\)](#) function.
- LvFtrInfo\_InterfaceID** Returns the string ID of the interface. Param = interface index. This constant can be used only in the [LvSystem](#) module for enumerating unopened interfaces ([LvGetInfoStr\(\)](#) function, as the Feature use [LvSystem\\_Info](#)).
- LvFtrInfo\_InterfaceDisplayName** Returns the Display name of the interface. Param = interface index. This constant can be used only in the [LvSystem](#) module for enumerating unopened interfaces ([LvGetInfoStr\(\)](#) function, as the Feature use [LvSystem\\_Info](#)).
- LvFtrInfo\_InterfaceTlType** Returns the interface Transport layer type. Param = interface index. This constant can be used only in the [LvSystem](#) module for enumerating unopened interfaces ([LvGetInfoStr\(\)](#) function, as the Feature use [LvSystem\\_Info](#)). For example a standard interface TL type is "GEV" for GigE-Vision devices.

**LvFtrInfo\_DeviceID** Returns the string ID of the device. Param = device index. This constant can be used only in the [LvInterface](#) module for enumerating unopened devices ([LvGetInfoStr\(\)](#) function, as the Feature use [LvInterface\\_Info](#)).

**LvFtrInfo\_DeviceVendor** Returns the Vendor name of the device. Param = device index. This constant can be used only in the [LvInterface](#) module for enumerating unopened devices ([LvGetInfoStr\(\)](#) function, as the Feature use [LvInterface\\_Info](#)).

**LvFtrInfo\_DeviceModel** Returns the Model name of the device. Param = device index. This constant can be used only in the [LvInterface](#) module for enumerating unopened devices ([LvGetInfoStr\(\)](#) function, as the Feature use [LvInterface\\_Info](#)).

**LvFtrInfo\_DeviceTType** Returns the Transport layer type of the device. Param = device index. This constant can be used only in the [LvInterface](#) module for enumerating unopened devices ([LvGetInfoStr\(\)](#) function, as the Feature use [LvInterface\\_Info](#)).

**LvFtrInfo\_DeviceDisplayName** Returns the Display name the device. Param = device index. This constant can be used only in the [LvInterface](#) module for enumerating unopened devices ([LvGetInfoStr\(\)](#) function, as the Feature use [LvInterface\\_Info](#)).

**LvFtrInfo\_DeviceAccessStatus** Returns the the device access. Param = device index. The returned value is one of the [LvDeviceAccessStatus](#) constants. Can be used only in the [LvInterface](#) module for enumerating unopened devices ([LvGetInfo\(\)](#) function, as the Feature use [LvInterface\\_Info](#)).

#### 5.25.2.8 enum LvFtrType

LvFtrType constants. Define the type of the feature. Used in the [LvGetType\(\)](#) function.

Enumerator

**LvFtrType\_Integer** Integer type, use [LvGetInt32\(\)](#), [LvSetInt32\(\)](#), [LvGetInt64\(\)](#), [LvSetInt64\(\)](#) to get/set a value.

**LvFtrType\_Float** Float type, use [LvGetFloat\(\)](#) and [LvSetFloat\(\)](#) to get/set a value.

**LvFtrType\_String** String type, use [LvGetString\(\)](#) and [LvSetString\(\)](#) to get/set a value.

**LvFtrType\_Enumeration** Enumeration type, use [LvGetEnum\(\)](#), [LvSetEnum\(\)](#), [LvGetEnumStr\(\)](#) and [LvSetEnumStr\(\)](#) to get/set a value.

**LvFtrType\_Boolean** Boolean type, use [LvGetInt32\(\)](#) and [LvSetInt32\(\)](#) to get/set a value.

**LvFtrType\_Command** Command type, use [LvCmdExecute\(\)](#) and [LvCmdIsDone\(\)](#) to execute and check.

**LvFtrType\_Category** Category type, used in the tree of feaures build.

**LvFtrType\_StringList** String list type (multiple strings in one, separated by terminating 0), use [LvGetString\(\)](#) and [LvSetString\(\)](#) to get/set a value.

**LvFtrType\_Pointer** Pointer type, use [LvGetPtr\(\)](#) and [LvSetPtr\(\)](#) to get/set a value.

**LvFtrType\_Buffer** Buffer type (in GenICam it corresponds with the Register type), use [LvGetBuffer\(\)](#) and [LvSetBuffer\(\)](#) to get/set a value. Do not confuse it with the [LvBuffer](#) module.

**LvFtrType\_Other** Unknown type, cannot be accessed.

#### 5.25.2.9 enum LvFtrVisibility

LvFtrVisibility constants. Define the visibility level of the feature. Used in [LvGetVisibility\(\)](#). Should be used for displaying the feature tree (or list).

Enumerator

**LvFtrVisibility\_Beginner** Beginner level - the feature should be displayed always.

**LvFtrVisibility\_Expert** Expert level - the feature should be displayed if at least the Expert level is selected.

**LvFtrVisibility\_Guru** Guru level - the feature should be displayed if at least the Guru level is selected.

**LvFtrVisibility\_Invisible** Invisible - the feature should not be displayed.

### 5.25.2.10 enum LvInfoDataType

LvInfoDataType constants. The enum is used only by the [LvEventGetDataInfo\(\)](#) function - this function follows the GenTL EventGetDataInfo() function, which uses different data types, than the GenApi.

#### Enumerator

- LvInfoDataType\_Unknown** Represents the GenTL INFO\_DATATYPE\_UNKNOWN info - Unknown data type.
- LvInfoDataType\_String** Represents the GenTL INFO\_DATATYPE\_STRING info - 0-terminated C string (ASCII encoded).
- LvInfoDataType\_StringList** Represents the GenTL INFO\_DATATYPE\_STRINGLIST info - Concatenated INFO\_DATATYPE\_STRING list. End of list is signaled with an additional 0.
- LvInfoDataType\_Int16** Represents the GenTL INFO\_DATATYPE\_INT16 info - Signed 16 bit integer.
- LvInfoDataType\_UInt16** Represents the GenTL INFO\_DATATYPE\_UINT16 info - unsigned 16 bit integer.
- LvInfoDataType\_Int32** Represents the GenTL INFO\_DATATYPE\_INT32 info - signed 32 bit integer.
- LvInfoDataType\_UInt32** Represents the GenTL INFO\_DATATYPE\_UINT32 info - unsigned 32 bit integer.
- LvInfoDataType\_Int64** Represents the GenTL INFO\_DATATYPE\_INT64 info - signed 64 bit integer.
- LvInfoDataType\_UInt64** Represents the GenTL INFO\_DATATYPE\_UINT64 info - unsigned 64 bit integer.
- LvInfoDataType\_Float64** Represents the GenTL INFO\_DATATYPE\_FLOAT64 info - Signed 64 bit floating point number.
- LvInfoDataType\_Ptr** Represents the GenTL INFO\_DATATYPE\_PTR info - Pointer type (void\*). Size is platform dependent (32 bit on 32 bit platforms).
- LvInfoDataType\_Bool** Represents the GenTL INFO\_DATATYPE\_BOOL8 info - Boolean value occupying 8 bit. 0 for false and anything for true.
- LvInfoDataType\_SizeT** Represents the GenTL INFO\_DATATYPE\_SIZE\_T info - Platform dependent unsigned integer (32 bit on 32 bit platforms).
- LvInfoDataType\_Buffer** Represents the GenTL INFO\_DATATYPE\_BUFFER info - Like the INFO\_DATATYPE\_STRING but with arbitrary data and no 0 termination.

### 5.25.2.11 enum LvLibInfo

Enum values for the Info parameter of the [LvGetLibInfo\(\)](#), [LvGetLibInfoStr\(\)](#) and [LvGetLibInfoStrSize\(\)](#) functions.

#### Enumerator

- LvInfo\_BinPath** Returns the full path to the SynView binaries (applications and libraries - in Windows the Bin folder of SynView). LvFtrType\_String.
- LvInfo\_AppDataPath** Returns the full path to the SynView application data. This folder may be different from the BinPath, for example in Windows Vista the BinPath is write protected, while AppDataPath is at the read-write location and contains files like sv.synview.log etc. LvFtrType\_String.
- LvInfo\_UserDataPath** Returns the full path to the SynView user data. In Windows this is equal to AppDataPath. LvFtrType\_String.
- LvInfo\_CfgPath** Returns the full path to the SynView config data. In Windows this is equal to AppDataPath. LvFtrType\_String.
- LvInfo\_InstPath** Returns the full path to the SynView installation root folder. LvFtrType\_String.
- LvInfo\_IniFile** Returns the full path to the lv.SynView.ini file. LvFtrType\_String.
- LvInfo\_BuildDate** Returns the build date of the library.



## 5.25.2.12 enum LvQueueOperation

LvQueueOperation constants. Define enum values for the [LvStreamFlushQueue\(\)](#) function.

## Enumerator

**LvQueueOperation\_InputToOutput** Represents the GenTL ACQ\_QUEUE\_INPUT\_TO\_OUTPUT. Flushes the input pool to the output queue and if necessary adds entries in the LvEventType\_NewBuffer event data queue.

**LvQueueOperation\_OutputDiscard** Represents the GenTL ACQ\_QUEUE\_OUTPUT\_DISCARD. Discards all buffers in the output queue and if necessary removes the entries from the event data queue.

**LvQueueOperation\_AllToInput** Represents the GenTL ACQ\_QUEUE\_ALL\_TO\_INPUT. Puts all buffers in the input pool. Even those in the output queue and discard entries in the event data queue.

**LvQueueOperation\_UnqueuedToInput** Represents the GenTL ACQ\_QUEUE\_UNQUEUED\_TO\_INPUT. Puts all buffers that are not in the input pool or the output queue in the input pool.

**LvQueueOperation\_AllDiscard** Represents the GenTL ACQ\_QUEUE\_ALL\_DISCARD. Discards all buffers in the input pool and output queue.

## 5.25.2.13 enum LvRenderFlags

The flags passed as parameter to the functions [LvRendererDisplayImage\(\)](#) and [LvRendererRepaint\(\)](#).

## Enumerator

**LvRenderFlags\_RepaintBackground** Before painting the image, the window background is repainted. This is done automatically whenever the change of the window size is detected, or display mode is switched. You can also call [LvRendererDisplayImage\(\)](#) with 0 as the buffer handle and this flag just to erase image painting area.

**LvRenderFlags\_DontPaintIncomplete** If the buffer LvBuffer\_IsIncomplete feature is true, it is not painted. The IsIncomplete feature indicates the contents of the buffer is a mixture of new and old image data, typically it happens when some packets with image data from a GigE camera are lost. If this flag is set simply the paint or repaint of such buffer is skipped, leaving whatever was before on the screen.

**LvRenderFlags\_IgnoreInvalidWinHandle** This flag has a meaning only for the [LvRendererCanDisplayImage\(\)](#) function. If used, this function will not return an error if the window handle was not yet assigned by the [LvRendererSetWindow\(\)](#) function. This can be utilized for checking if the image is displayable before the display window is actually used.

## 5.26 SynView Image Processing Library

### Modules

- [SynView Image Processing Library defines, typedefs and enums](#)
- [SynView Image Processing Library functions](#)
- [SynView Image Processing Library LvStatus definitions](#)

### 5.26.1 Detailed Description



## 5.27 SynView Image Processing Library defines, typedefs and enums

### Modules

- [Definitions for Enumeration Entry Info](#)

### Classes

- struct [LvIpImgInfo](#)

### Macros

- `#define LVIP_LUT_BAYER`
- `#define LVIP_LUT_BAYER_16`

### Enumerations

- enum [LvIpImgAttr](#) {  
[LvIpImgAttr\\_BottomUp](#), [LvIpImgAttr\\_DWordAligned](#), [LvIpImgAttr\\_QWordAligned](#), [LvIpImgAttr\\_SSEAligned](#),  
[LvIpImgAttr\\_NotDataOwner](#) }
- enum [LvIpOption](#) {  
[LvIpOption\\_ReallocateDst](#), [LvIpOption\\_TiffConvertTo16Bit](#), [LvIpOption\\_BmpForceTopDown](#), [LvIpOption\\_↔  
BmpForceBottomUp](#),  
[LvIpOption\\_JpegConvertToBgr](#), [LvIpOption\\_JpegReadHeaderOnly](#), [LvIpOption\\_WbCorrectFactors](#) }
- enum [LvIpLutType](#) {  
[LvIpLutType\\_Uni](#), [LvIpLutType\\_8Bit](#), [LvIpLutType\\_10Bit](#), [LvIpLutType\\_12Bit](#),  
[LvIpLutType\\_UniBayer](#), [LvIpLutType\\_8BitBayer](#), [LvIpLutType\\_10BitBayer](#), [LvIpLutType\\_12BitBayer](#),  
[LvIpLutType\\_UniBayer16](#), [LvIpLutType\\_10BitBayer16](#), [LvIpLutType\\_12BitBayer16](#) }
- enum [LvIpColor](#) { [LvIpColor\\_None](#) }
- enum [LvIpTextAttr](#) {  
[LvIpTextAttr\\_Bold](#), [LvIpTextAttr\\_Italic](#), [LvIpTextAttr\\_Underline](#), [LvIpTextAttr\\_Strikeout](#),  
[LvIpTextAttr\\_Nonantialiased](#), [LvIpTextAttr\\_Shadow](#), [LvIpTextAttr\\_Outline](#), [LvIpTextAttr\\_ShadowRB](#),  
[LvIpTextAttr\\_ShadowRT](#), [LvIpTextAttr\\_ShadowLB](#), [LvIpTextAttr\\_ShadowLT](#), [LvIpTextAttr\\_ShadowB](#),  
[LvIpTextAttr\\_ShadowT](#), [LvIpTextAttr\\_ShadowR](#), [LvIpTextAttr\\_ShadowL](#) }

#### 5.27.1 Detailed Description

#### 5.27.2 Macro Definition Documentation

##### 5.27.2.1 `#define LVIP_LUT_BAYER`

If the LUT is to be used in [Bayer decoding/encoding functions](#), this attribute is to be OR-ed to the [LvIpLutType](#) specification in the [LvIpAllocateLut\(\)](#) function Bayer LUT requires bigger size - is needed for the bilinear interpolation methods and for 10- and 12-bit source formats.

##### 5.27.2.2 `#define LVIP_LUT_BAYER_16`

Bayer16 is a subset of [LVIP\\_LUT\\_BAYER](#), suitable for all 10- and 12-bit decoding, with the exception of [LvIpBd↔  
BilinearInterpolation\(\)](#) function.

### 5.27.3 Enumeration Type Documentation

#### 5.27.3.1 enum LvipColor

Color definitions for the Overlay functions.

Enumerator

**LvipColor\_None** Defines a non-color. This is useful for the transparent color - specifying the transparent color as LvipColor\_None in LvipSetOverlayTransparentColor() switches off overlay transparency.

#### 5.27.3.2 enum LvipImgAttr

Image attributes. Flags to be used in the Attributes of the [LvipImgInfo](#) structure.

Enumerator

**LvipImgAttr\_BottomUp** Lines in the image buffer are ordered from the bottom line to the top line, so the image bufer begins with the bottom line.

**LvipImgAttr\_DWordAligned** The line increment is aligned to double word (32 bits). This is required for example by the Windows Device Independent Bitmap format (DIB, BMP) This attribute is used only in the [LvipInitImgInfo\(\)](#) function (which can be called as a result of the LvipOption\_ReallocateDst attribute).

**LvipImgAttr\_QWordAligned** The line increment is aligned to quad word (64 bits). This attribute is used in the [LvipInitImgInfo\(\)](#) function (which can be called as a result of the LvipOption\_ReallocateDst attribute).

**LvipImgAttr\_SSEAligned** The line increment is aligned to SSE words (128 bits). This attribute is used in the [LvipInitImgInfo\(\)](#) function (which can be called as a result of the LvipOption\_ReallocateDst attribute).

**LvipImgAttr\_NotDataOwner** The [LvipImgInfo](#) is not the owner of image data, so the [LvipDeallocateImageData\(\)](#) function will not attempt to deallocate the image data. This attribute is used when the image data are owned by another [LvipImgInfo](#) or belonging to other code, for example when working directly with the image in the DMA buffer. Note that [LvipDeallocateImageData\(\)](#) may be called from other functions, for example, when you use the [LvipOption\\_ReallocateDst](#) attribute.

#### 5.27.3.3 enum LvipLutType

LUT type - to be used in the [LvipAllocateLut\(\)](#) function.

Enumerator

**LvipLutType\_Uni** LUT which internally contains 3 LUTs: 8-bit, 10-bit and 12-bit. All the LUTs are kept synchronized.

**LvipLutType\_8Bit** 8-bit LUT type, used for images with [LvPixelFormat\\_Mono8](#).

**LvipLutType\_10Bit** 10-bit LUT type, used for images with [LvPixelFormat\\_Mono10](#).

**LvipLutType\_12Bit** 12-bit LUT type, used for images with [LvPixelFormat\\_Mono12](#)

**LvipLutType\_UniBayer** LvipLutType\_Uni type with the [LVIP\\_LUT\\_BAYER](#).

**LvipLutType\_8BitBayer** LvipLutType\_8Bit type with the [LVIP\\_LUT\\_BAYER](#).

**LvipLutType\_10BitBayer** LvipLutType\_10Bit type with the [LVIP\\_LUT\\_BAYER](#).

**LvipLutType\_12BitBayer** LvipLutType\_12Bit type with the [LVIP\\_LUT\\_BAYER](#).

**LvipLutType\_UniBayer16** LvipLutType\_Uni type with the [LVIP\\_LUT\\_BAYER\\_16](#).

**LvipLutType\_10BitBayer16** LvipLutType\_10Bit type with the [LVIP\\_LUT\\_BAYER\\_16](#).

**LvipLutType\_12BitBayer16** LvipLutType\_12Bit type with the [LVIP\\_LUT\\_BAYER\\_16](#).

## 5.27.3.4 enum LvipOption

Options for image processing functions in the Options parameter.

## Enumerator

**LvipOption\_ReallocateDst** The destination image data can be reallocated if it is needed. If the function stores a result of the operation to the destination image buffer, it first checks if the destination [LvipImgInfo](#) has appropriate parameters and the buffer(s) allocated. If not and this attribute is specified, it adapts the parameters of the [LvipImgInfo](#) and reallocates the buffer as needed. If this attribute is not specified, the function returns an error in case of mismatch.

**LvipOption\_TiffConvertTo16Bit** The attribute will force conversion of the image to 16-bit mono format, if it is in 9- to 15-bit mono format. This can be used when saving mono image to TIFF by the [LvipSaveToTiff\(\)](#) function, as many software packages do not understand mono TIFF if it is in 9- to 15-bit mono format.

**LvipOption\_BmpForceTopDown** The BMP file will be read to the top-down line layout. This attribute is used in the [LvipLoadFromBmp\(\)](#) and [LvipSaveToBmp\(\)](#) functions, as the BMP format can be either in the bottom-up line layout or in the top-down line layout.

**LvipOption\_BmpForceBottomUp** The BMP file will be read to the bottom-up line layout. This attribute is used in the [LvipLoadFromBmp\(\)](#) and [LvipSaveToBmp\(\)](#) functions, as the BMP format can be either in the bottom-up line layout or in the top-down line layout.

**LvipOption\_JpegConvertToBgr** The color JPEG images are stored in RGB format (24-bit). With this option the pixel format will be reversed to the BGR format in the [LvipLoadFromJpg\(\)](#) function.

**LvipOption\_JpegReadHeaderOnly** The JPEG image data will not be read, only the header will be read. This enables to allocate the image buffer and then read the full image.

**LvipOption\_WbCorrectFactors** This attribute can be used in the [LvipCalcWbFactors\(\)](#) function. If present, it is assumed that the white balance is calculated from the image to which were applied white balancing factors passed as input parameters. Thus only a correction is calculated and the existing factors are modified.

## 5.27.3.5 enum LvipTextAttr

Text attributes definitions for the Overlay functions.

## Enumerator

**LvipTextAttr\_Bold** Bold text. Text attribute for the [LvipSetOverlayTextParams\(\)](#) function: Bold text

**LvipTextAttr\_Italic** Italics text. Text attribute for the [LvipSetOverlayTextParams\(\)](#) function: Italics text

**LvipTextAttr\_Underline** Underlined text. Text attribute for the [LvipSetOverlayTextParams\(\)](#) function: Underlined text

**LvipTextAttr\_Strikeout** Strikeout text. Text attribute for the [LvipSetOverlayTextParams\(\)](#) function: Strikeout text

**LvipTextAttr\_Nonantialiased** Text antialiasing off. Text attribute for the [LvipSetOverlayTextParams\(\)](#) function: Text antialiasing will be switched off - this is useful for text on transparent background, where antialiasing (like ClearType) can make undesirable effects.

**LvipTextAttr\_Shadow** Text with a 1 pixel shadow. Text attribute for the [LvipSetOverlayTextParams\(\)](#) function: Text with a 1 pixel shadow at right-bottom direction.

**LvipTextAttr\_Outline** Text with a 1 pixel outline. Text attribute for the [LvipSetOverlayTextParams\(\)](#) function: Text with a 1 pixel outline around the letters. This is useful namely for the text on transparent background - by adding the outline of different color, then the text is readable even if the background become of the same color, as the text.

**LvipTextAttr\_ShadowRB** Text with a 1 pixel shadow at right-bottom direction. Text attribute for the [LvipSetOverlayTextParams\(\)](#) function: Text with a 1 pixel shadow at right-bottom direction (equal to [LvipTextAttr\\_Shadow](#) constant). This constant can be combined with other [LvipTextAttr\\_ShadowXX](#) constants.

- LvipTextAttr\_ShadowRT*** Text with a 1 pixel shadow at right-top direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at right-top direction. This constant can be combined with other LVIP\_TEXTATTR\_SHADOW\_x constants.
- LvipTextAttr\_ShadowLB*** Text with a 1 pixel shadow at left-bottom direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at left-bottom direction. This constant can be combined with other LVIP\_TEXTATTR\_SHADOW\_x constants.
- LvipTextAttr\_ShadowLT*** Text with a 1 pixel shadow at left-top direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at left-top direction. This constant can be combined with other LVIP\_TEXTATTR\_SHADOW\_x constants.
- LvipTextAttr\_ShadowB*** Text with a 1 pixel shadow at bottom direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at bottom direction. This constant can be combined with other LVIP\_TEXTATTR\_SHADOW\_x constants.
- LvipTextAttr\_ShadowT*** Text with a 1 pixel shadow at top direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at top direction. This constant can be combined with other LVIP\_TEXTATTR\_SHADOW\_x constants.
- LvipTextAttr\_ShadowR*** Text with a 1 pixel shadow at right direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at right direction. This constant can be combined with other LVIP\_TEXTATTR\_SHADOW\_x constants.
- LvipTextAttr\_ShadowL*** Text with a 1 pixel shadow at left direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at left direction. This constant can be combined with other LVIP\_TEXTATTR\_SHADOW\_x constants.

## 5.28 Definitions for Enumeration Entry Info

### Macros

- `#define LV_ENUMENTRY_CURRENT`

### Typedefs

- typedef `LvHModule LvHSystem`
- typedef `LvHModule LvHInterface`
- typedef `LvHModule LvHDevice`
- typedef `LvHModule LvHStream`
- typedef `LvHModule LvHEvent`
- typedef `LvHModule LvHRenderer`
- typedef `LvHModule LvHBuffer`
- typedef `uint32_t LvHOverlay`
- typedef `uint32_t LvFeature`
- typedef `uint32_t LvEnum`

#### 5.28.1 Detailed Description

#### 5.28.2 Macro Definition Documentation

##### 5.28.2.1 `#define LV_ENUMENTRY_CURRENT`

If used as Param of the `LvGetInfo()`, `LvGetInfoStr()` and `LvGetInfoStrSize()` the returned value is for the current enum entry.

#### 5.28.3 Typedef Documentation

##### 5.28.3.1 typedef `uint32_t LvEnum`

Base typedef for the entry of the enumeration item.

##### 5.28.3.2 typedef `uint32_t LvFeature`

Base typedef for the ID of the feature.

##### 5.28.3.3 typedef `LvHModule LvHBuffer`

Typedef for a handle to the Buffer module.

##### 5.28.3.4 typedef `LvHModule LvHDevice`

Typedef for a handle to the Device module.

##### 5.28.3.5 typedef `LvHModule LvHEvent`

Typedef for a handle to the Event module.

**5.28.3.6 typedef LvHModule LvHInterface**

Typedef for a handle to the Interface module.

**5.28.3.7 typedef uint32\_t LvHOverlay**

Typedef for a handle to the overlay.

**5.28.3.8 typedef LvHModule LvHRenderer**

Typedef for a handle to the Renderer module.

**5.28.3.9 typedef LvHModule LvHStream**

Typedef for a handle to the Stream module.

**5.28.3.10 typedef LvHModule LvHSystem**

Typedef for a handle to the System module.

## 5.29 Features

### Enumerations

- enum LvSystemFtr {  
LvSystem\_TLVendorName, LvSystem\_TLModelName, LvSystem\_TLID, LvSystem\_TLVersion,  
LvSystem\_TLPath, LvSystem\_TLType, LvSystem\_GenTLVersionMajor, LvSystem\_GenTLVersionMinor,  
LvSystem\_GevVersionMajor, LvSystem\_GevVersionMinor, LvSystem\_InterfaceUpdateList, LvSystem\_↵  
InterfaceSelector,  
LvSystem\_InterfaceID, LvSystem\_GevInterfaceMACAddress, LvSystem\_GevInterfaceDefaultIPAddress,  
LvSystem\_GevInterfaceDefaultSubnetMask,  
LvSystem\_GevInterfaceDefaultGateway, LvSystem\_LvSystemDisplayName, LvSystem\_Info }
- enum LvInterfaceFtr {  
LvInterface\_InterfaceID, LvInterface\_InterfaceType, LvInterface\_GevInterfaceGatewaySelector, LvInterface\_↵  
\_GevInterfaceGateway,  
LvInterface\_GevMACAddress, LvInterface\_GevInterfaceSubnetSelector, LvInterface\_GevInterfaceSubnetI↵  
PAddress, LvInterface\_GevInterfaceSubnetMask,  
LvInterface\_DeviceUpdateList, LvInterface\_DeviceSelector, LvInterface\_DeviceID, LvInterface\_Device\_↵  
VendorName,  
LvInterface\_DeviceModelName, LvInterface\_DeviceAccessStatus, LvInterface\_GevDeviceIPAddress, Lv\_↵  
Interface\_GevDeviceSubnetMask,  
LvInterface\_GevDeviceMACAddress, LvInterface\_LvDeviceUserID, LvInterface\_LvDeviceSerialNumber,  
LvInterface\_LvInterfaceDisplayName,  
LvInterface\_Info }
- enum LvDeviceFtr {  
LvDevice\_DeviceVendorName, LvDevice\_DeviceModelName, LvDevice\_DeviceManufacturerInfo, Lv\_↵  
Device\_DeviceVersion,  
LvDevice\_DeviceFirmwareVersion, LvDevice\_LvRecoveryFirmwareVersion, LvDevice\_DeviceSerialNumber,  
LvDevice\_DeviceUserID,  
LvDevice\_LvSensorID, LvDevice\_LvGrabberID, LvDevice\_DeviceScanType, LvDevice\_DeviceRegisters\_↵  
StreamingStart,  
LvDevice\_DeviceRegistersStreamingEnd, LvDevice\_DeviceRegistersCheck, LvDevice\_DeviceRegisters\_↵  
Valid, LvDevice\_DeviceReset,  
LvDevice\_DeviceClockSelector, LvDevice\_DeviceClockFrequency, LvDevice\_DeviceTemperatureSelector,  
LvDevice\_DeviceTemperature,  
LvDevice\_LvDeviceUpTime, LvDevice\_LvDeviceType, LvDevice\_SensorWidth, LvDevice\_SensorHeight,  
LvDevice\_WidthMax, LvDevice\_HeightMax, LvDevice\_Width, LvDevice\_Height,  
LvDevice\_OffsetX, LvDevice\_OffsetY, LvDevice\_PixelFormat, LvDevice\_BinningHorizontal,  
LvDevice\_BinningVertical, LvDevice\_DecimationHorizontal, LvDevice\_DecimationVertical, LvDevice\_LvA\_↵  
OIMode,  
LvDevice\_LvReadoutWidth, LvDevice\_LvReadoutHeight, LvDevice\_LvReadoutOffsetX, LvDevice\_Lv\_↵  
ReadoutOffsetY,  
LvDevice\_LvVariablePayloadSize, LvDevice\_AcquisitionMode, LvDevice\_TriggerSelector, LvDevice\_↵  
TriggerMode,  
LvDevice\_TriggerSoftware, LvDevice\_TriggerSource, LvDevice\_TriggerActivation, LvDevice\_TriggerDelay,  
LvDevice\_TriggerDivider, LvDevice\_LvTriggerCaching, LvDevice\_ExposureMode, LvDevice\_LvLong\_↵  
RangeExposureMode,  
LvDevice\_LvGlobalResetMode, LvDevice\_ExposureTime, LvDevice\_ExposureAuto, LvDevice\_Lv\_↵  
AcquisitionFrameRateControlMode,  
LvDevice\_AcquisitionFrameRate, LvDevice\_LineSelector, LvDevice\_LineMode, LvDevice\_LineFormat,  
LvDevice\_LineSource, LvDevice\_LineInverter, LvDevice\_LineStatus, LvDevice\_LineStatusAll,  
LvDevice\_UserOutputSelector, LvDevice\_UserOutputValue, LvDevice\_UserOutputValueAll, LvDevice\_↵  
UserOutputValueAllMask,  
LvDevice\_CounterSelector, LvDevice\_LvCounterMode, LvDevice\_CounterEventSource, LvDevice\_↵  
CounterReset,  
LvDevice\_CounterValue, LvDevice\_CounterDuration, LvDevice\_TimerSelector, LvDevice\_TimerDuration,  
LvDevice\_TimerDelay, LvDevice\_TimerTriggerSource, LvDevice\_LvSpecialPurposeTriggerSelector, Lv\_↵

Device\_LvSpecialPurposeTriggerSource,  
 LvDevice\_LvSpecialPurposeTriggerActivation, LvDevice\_LvSpecialPurposeTriggerSoftware, LvDevice\_Lv↔  
 ImageStampsResetMask, LvDevice\_LvImageStampSelector,  
 LvDevice\_LvImageStampResetEnable, LvDevice\_LvBootSwitch, LvDevice\_LvBayerDecoderAlgorithm, Lv↔  
 Device\_LvBayerDecoderThreshold,  
 LvDevice\_LvWatchdogEnable, LvDevice\_LvWatchdogTimerDuration, LvDevice\_LvWatchdogTimerReset,  
 LvDevice\_LvWatchdogFailed,  
 LvDevice\_GainSelector, LvDevice\_Gain, LvDevice\_GainAuto, LvDevice\_BlackLevelSelector,  
 LvDevice\_BlackLevel, LvDevice\_BlackLevelAuto, LvDevice\_ColorTransformationSelector, LvDevice\_Color↔  
 TransformationEnable,  
 LvDevice\_ColorTransformationValueSelector, LvDevice\_ColorTransformationValue, LvDevice\_LvExternal↔  
 DeviceControlMode, LvDevice\_LvExternalADCSelector,  
 LvDevice\_LvExternalADCValue, LvDevice\_LvPowerSwitchCurrentAction, LvDevice\_LvPowerSwitch↔  
 Selector, LvDevice\_LvPowerSwitchBoundADC,  
 LvDevice\_LvPowerSwitchDrive, LvDevice\_LvPowerSwitchPulsePlus, LvDevice\_LvPowerSwitchPulseMinus,  
 LvDevice\_LvLensControlCalibrate,  
 LvDevice\_LvLensControlMinusEnd, LvDevice\_LvLensControlPlusEnd, LvDevice\_LvLensControlPulse↔  
 Period, LvDevice\_LvLensControlDutyCycle,  
 LvDevice\_LvLensControlTargetApproach, LvDevice\_LvLensControlNrSlowSteps, LvDevice\_LvLens↔  
 ControlTargetPosition, LvDevice\_LvLensControlAdjustPosition,  
 LvDevice\_LvPowerSwitchPulseDuration, LvDevice\_LvLensControlMinCalibrationRange, LvDevice\_Lv↔  
 LensControlCalibrateAll, LvDevice\_LUTSelector,  
 LvDevice\_LUTEnable, LvDevice\_LUTIndex, LvDevice\_LUTValue, LvDevice\_LUTValueAll,  
 LvDevice\_PayloadSize, LvDevice\_GevVersionMajor, LvDevice\_GevVersionMinor, LvDevice\_GevDevice↔  
 ModelsBigEndian,  
 LvDevice\_GevDeviceModeCharacterSet, LvDevice\_GevInterfaceSelector, LvDevice\_GevMACAddress, Lv↔  
 Device\_GevSupportedOptionSelector,  
 LvDevice\_GevSupportedOption, LvDevice\_GevCurrentIPConfigurationLLA, LvDevice\_GevCurrentIP↔  
 ConfigurationDHCP, LvDevice\_GevCurrentIPConfigurationPersistentIP,  
 LvDevice\_GevCurrentIPAddress, LvDevice\_GevCurrentSubnetMask, LvDevice\_GevCurrentDefaultGateway,  
 LvDevice\_GevPersistentIPAddress,  
 LvDevice\_GevPersistentSubnetMask, LvDevice\_GevPersistentDefaultGateway, LvDevice\_GevNumberof↔  
 Interfaces, LvDevice\_GevMessageChannelCount,  
 LvDevice\_GevStreamChannelCount, LvDevice\_GevHeartbeatTimeout, LvDevice\_GevTimestampTick↔  
 Frequency, LvDevice\_GevTimestampControlLatch,  
 LvDevice\_GevTimestampControlReset, LvDevice\_GevTimestampControlLatchReset, LvDevice\_Gev↔  
 TimestampValue, LvDevice\_GevCCP,  
 LvDevice\_GevStreamChannelSelector, LvDevice\_GevSCPIInterfaceIndex, LvDevice\_GevSCPHostPort, Lv↔  
 Device\_GevSCPSFireTestPacket,  
 LvDevice\_GevSCPSDoNotFragment, LvDevice\_GevSCPSBigEndian, LvDevice\_GevSCSPacketSize, Lv↔  
 Device\_GevSCPD,  
 LvDevice\_GevSCDA, LvDevice\_GevLinkSpeed, LvDevice\_UserSetSelector, LvDevice\_UserSetLoad,  
 LvDevice\_UserSetSave, LvDevice\_UserSetDefaultSelector, LvDevice\_ChunkModeActive, LvDevice\_↔  
 ChunkSelector,  
 LvDevice\_ChunkEnable, LvDevice\_ChunkOffsetX, LvDevice\_ChunkOffsetY, LvDevice\_ChunkWidth,  
 LvDevice\_ChunkHeight, LvDevice\_ChunkPixelFormat, LvDevice\_ChunkLinePitch, LvDevice\_ChunkFrame↔  
 ID,  
 LvDevice\_ChunkTimestamp, LvDevice\_ChunkExposureTime, LvDevice\_ChunkGainSelector, LvDevice\_↔  
 ChunkGain,  
 LvDevice\_ChunkBlackLevel, LvDevice\_ChunkLineStatusAll, LvDevice\_ChunkLvExternalADCSelector, Lv↔  
 Device\_ChunkLvExternalADCValue,  
 LvDevice\_EventSelector, LvDevice\_EventNotification, LvDevice\_LvSmartAppID, LvDevice\_LvSmartApp↔  
 Int1,  
 LvDevice\_LvSmartAppInt2, LvDevice\_LvSmartAppInt3, LvDevice\_LvSmartAppInt4, LvDevice\_LvSmart↔  
 AppInt5,  
 LvDevice\_LvSmartAppInt6, LvDevice\_LvSmartAppInt7, LvDevice\_LvSmartAppInt8, LvDevice\_LvSmart↔  
 AppInt9,  
 LvDevice\_LvSmartAppInt10, LvDevice\_LvSmartAppInt11, LvDevice\_LvSmartAppInt12, LvDevice\_Lv↔



SmartAppInt13,  
 LvDevice\_LvSmartAppInt14, LvDevice\_LvSmartAppInt15, LvDevice\_LvSmartAppInt16, LvDevice\_Lv↔  
 SmartAppInt17,  
 LvDevice\_LvSmartAppInt18, LvDevice\_LvSmartAppInt19, LvDevice\_LvSmartAppInt20, LvDevice\_Lv↔  
 SmartAppInt21,  
 LvDevice\_LvSmartAppInt22, LvDevice\_LvSmartAppInt23, LvDevice\_LvSmartAppInt24, LvDevice\_Lv↔  
 SmartAppInt25,  
 LvDevice\_LvSmartAppInt26, LvDevice\_LvSmartAppInt27, LvDevice\_LvSmartAppInt28, LvDevice\_Lv↔  
 SmartAppInt29,  
 LvDevice\_LvSmartAppInt30, LvDevice\_LvSmartAppInt31, LvDevice\_LvSmartAppInt32, LvDevice\_Lv↔  
 SmartAppUint1,  
 LvDevice\_LvSmartAppUint2, LvDevice\_LvSmartAppUint3, LvDevice\_LvSmartAppUint4, LvDevice\_Lv↔  
 SmartAppUint5,  
 LvDevice\_LvSmartAppUint6, LvDevice\_LvSmartAppUint7, LvDevice\_LvSmartAppUint8, LvDevice\_Lv↔  
 SmartAppUint9,  
 LvDevice\_LvSmartAppUint10, LvDevice\_LvSmartAppUint11, LvDevice\_LvSmartAppUint12, LvDevice\_Lv↔  
 SmartAppUint13,  
 LvDevice\_LvSmartAppUint14, LvDevice\_LvSmartAppUint15, LvDevice\_LvSmartAppUint16, LvDevice\_Lv↔  
 SmartAppUint17,  
 LvDevice\_LvSmartAppUint18, LvDevice\_LvSmartAppUint19, LvDevice\_LvSmartAppUint20, LvDevice\_Lv↔  
 SmartAppUint21,  
 LvDevice\_LvSmartAppUint22, LvDevice\_LvSmartAppUint23, LvDevice\_LvSmartAppUint24, LvDevice\_Lv↔  
 SmartAppUint25,  
 LvDevice\_LvSmartAppUint26, LvDevice\_LvSmartAppUint27, LvDevice\_LvSmartAppUint28, LvDevice\_Lv↔  
 SmartAppUint29,  
 LvDevice\_LvSmartAppUint30, LvDevice\_LvSmartAppUint31, LvDevice\_LvSmartAppUint32, LvDevice\_Lv↔  
 SmartAppAsciiCmdString,  
 LvDevice\_LvSmartAppAsciiCmdExecute, LvDevice\_LvSmartAppAsciiCmdFeedback, LvDevice\_LvSmart↔  
 AppAsciiCmdRetCode, LvDevice\_LvSmartAppPath,  
 LvDevice\_LvSmartAppStart, LvDevice\_EventLvLog, LvDevice\_EventLvLogTimestamp, LvDevice\_EventLv↔  
 LogMessage,  
 LvDevice\_EventLvSmartAppLog, LvDevice\_EventLvSmartAppLogTimestamp, LvDevice\_EventLvSmart↔  
 AppLogMessage, LvDevice\_LvSerialPortBaudRate,  
 LvDevice\_LvSerialPortParity, LvDevice\_LvSerialPortDataBits, LvDevice\_LvSerialPortStopBits, LvDevice\_↔  
 LvSerialPortTimeout,  
 LvDevice\_LvSerialPortEOTMarker, LvDevice\_LvSerialPortMaxResponseLength, LvDevice\_LvSerialPort↔  
 CommandString, LvDevice\_LvSerialPortCommandSend,  
 LvDevice\_LvSerialPortCommandResponse, LvDevice\_LvSerialPortCommandStatus, LvDevice\_LvSmart↔  
 AppExitEvent, LvDevice\_LvWatchdogTimerValue,  
 LvDevice\_LvLensControlInvertedPolarity, LvDevice\_GevMCPHostPort, LvDevice\_GevMCDA, LvDevice\_↔  
 GevMCTT,  
 LvDevice\_GevMCRC, LvDevice\_ChunkLvSmartAppString, LvDevice\_ChunkLvSmartAppIntSelector, Lv↔  
 Device\_ChunkLvSmartAppInt,  
 LvDevice\_ChunkLvSmartAppUintSelector, LvDevice\_ChunkLvSmartAppUint, LvDevice\_ChunkLvSmart↔  
 AppRegister, LvDevice\_EventLvSmartAppString,  
 LvDevice\_EventLvSmartAppStringTimestamp, LvDevice\_EventLvSmartAppStringValue, LvDevice\_Event↔  
 LvSmartAppInt, LvDevice\_EventLvSmartAppIntTimestamp,  
 LvDevice\_EventLvSmartAppIntSelector, LvDevice\_EventLvSmartAppIntValue, LvDevice\_EventLvSmart↔  
 AppUint, LvDevice\_EventLvSmartAppUintTimestamp,  
 LvDevice\_EventLvSmartAppUintSelector, LvDevice\_EventLvSmartAppUintValue, LvDevice\_EventLv↔  
 SmartAppRegister, LvDevice\_EventLvSmartAppRegisterTimestamp,  
 LvDevice\_EventLvSmartAppRegisterValue, LvDevice\_DeviceSFNCVersionMajor, LvDevice\_DeviceSFNC↔  
 VersionMinor, LvDevice\_DeviceSFNCVersionSubMinor,  
 LvDevice\_LvLineDebounceDuration, LvDevice\_ActionDeviceKey, LvDevice\_ActionSelector, LvDevice\_↔  
 ActionGroupKey,  
 LvDevice\_ActionGroupMask, LvDevice\_LvLensControlCalibrationStatus, LvDevice\_LvLUTMode, Lv↔  
 Device\_BalanceRatioSelector,  
 LvDevice\_BalanceRatio, LvDevice\_BalanceWhiteAuto, LvDevice\_GevDeviceClass, LvDevice\_GevIP↔

```

ConfigurationStatus,
LvDevice_GevDiscoveryAckDelay, LvDevice_GevGVCPExtendedStatusCodes, LvDevice_GevGVCP↔
PendingAck, LvDevice_GevGVCPHeartbeatDisable,
LvDevice_GevGVCPPendingTimeout, LvDevice_GevPrimaryApplicationSwitchoverKey, LvDevice_Gev↔
PrimaryApplicationSocket, LvDevice_GevPrimaryApplicationIPAddress,
LvDevice_GevMCSP, LvDevice_GevSCCFGUnconditionalStreaming, LvDevice_GevSCCFGExtended↔
ChunkData, LvDevice_GevSCPDirection,
LvDevice_GevSCSP, LvDevice_ChunkLvTriggerDelayed, LvDevice_EventLvTriggerDropped, LvDevice_↔
EventLvTriggerDroppedTimestamp,
LvDevice_LvStrobeEnable, LvDevice_LvStrobeDurationMode, LvDevice_LvStrobeDuration, LvDevice_Lv↔
StrobeDelay,
LvDevice_LvStrobeBrightness, LvDevice_LvStrobeDropMode, LvDevice_LvLUTReset, LvDevice_Chunk↔
LvStrobeDropped,
LvDevice_ReverseX, LvDevice_ReverseY, LvDevice_RegionSelector, LvDevice_RegionMode,
LvDevice_RegionDestination, LvDevice_AcquisitionFrameCount, LvDevice_AcquisitionBurstFrameCount,
LvDevice_LvCustomID,
LvDevice_LvCustomInfo, LvDevice_LvCustomRegMode, LvDevice_LvCustomRegAddr, LvDevice_Lv↔
CustomRegData,
LvDevice_LvCustomRegMux, LvDevice_LinePitch, LvDevice_ChunkLvFrameAbort, LvDevice_ChunkLv↔
TriggerDropped,
LvDevice_ChunkLvTriggerError, LvDevice_ChunkLvEncoderPosition, LvDevice_ChunkLvEncoderRotation,
LvDevice_RegionNumber,
LvDevice_UserSetDefault, LvDevice_LvLineDebounceMode, LvDevice_Gamma, LvDevice_LvCustomReg1,
LvDevice_LvCustomReg2, LvDevice_LvCustomReg3, LvDevice_LvCustomReg4, LvDevice_LvCustom↔
Reg5,
LvDevice_LvCustomReg6, LvDevice_LvCustomReg7, LvDevice_LvCustomReg8, LvDevice_LvCustom↔
Reg9,
LvDevice_LvCustomReg10, LvDevice_LvCustomReg11, LvDevice_LvCustomReg12, LvDevice_LvCustom↔
Reg13,
LvDevice_LvCustomReg14, LvDevice_LvCustomReg15, LvDevice_LvCustomReg16, LvDevice_LvCustom↔
Bypass,
LvDevice_DeviceID, LvDevice_DeviceType, LvDevice_GevDeviceIPAddress, LvDevice_GevDeviceSubnet↔
Mask,
LvDevice_GevDeviceMACAddress, LvDevice_GevDeviceGateway, LvDevice_LvGevDeviceStream↔
CaptureMode, LvDevice_StreamSelector,
LvDevice_StreamID, LvDevice_DeviceEndianessMechanism, LvDevice_LvGevFindMaxPacketSize, Lv↔
Device_LvGevPacketSizeValue,
LvDevice_LvGevTestPacketSize, LvDevice_LvGevPacketSizeTestSuccess, LvDevice_LvGevCCTT, Lv↔
Device_LvGevCCRC,
LvDevice_LvCCStatus, LvDevice_LvDeviceDisplayName, LvDevice_LvDevicesAcquiring, LvDevice_Lv↔
UniProcessMode,
LvDevice_LvUniProcessEnableInPlace, LvDevice_LvUniPixelFormat, LvDevice_LvUniProcessPayloadSize,
LvDevice_LvUniLinePitch,
LvDevice_LvUniBayerDecoderAlgorithm, LvDevice_LvUniBrightness, LvDevice_LvUniContrast, LvDevice_↔
LvUniGamma,
LvDevice_LvUniBalanceRatioSelector, LvDevice_LvUniBalanceRatio, LvDevice_LvUniBalanceWhiteAuto,
LvDevice_LvUniBalanceWhiteReset,
LvDevice_LvUniColorTransformationSelector, LvDevice_LvUniColorTransformationEnable, LvDevice_Lv↔
UniColorTransformationValueSelector, LvDevice_LvUniColorTransformationValue,
LvDevice_LvUniSaturation, LvDevice_LvUniProcessExecution, LvDevice_LvUniLUTMode, LvDevice_Lv↔
UniLUTSelector,
LvDevice_LvUniLUTEnable, LvDevice_LvUniLUTIndex, LvDevice_LvUniLUTValue, LvDevice_LvUniLUT↔
ValueAll,
LvDevice_LvUniColorTransformationMode, LvDevice_LvDeviceExpiringDate, LvDevice_Info }

```

- enum LvStreamFtr {

```

LvStream_StreamID, LvStream_StreamAnnouncedBufferCount, LvStream_StreamAcquisitionMode↔
Selector, LvStream_StreamAnnounceBufferMinimum,
LvStream_StreamType, LvStream_LvStreamDisplayName, LvStream_Lv↔

```

- PostponeQueueBuffers,
- LvStream\_LvAwaitDeliveryLimit, LvStream\_LvAutoAllocateProcessBuffers, LvStream\_LvPreallocateProcessBuffers, LvStream\_LvNumDelivered,
- LvStream\_LvNumUnderrun, LvStream\_LvNumAnnounced, LvStream\_LvNumQueued, LvStream\_LvNumAwaitDelivery,
- LvStream\_LvIsGrabbing, LvStream\_LvNumAborted, LvStream\_LvNumStarted, LvStream\_Info }
- enum LvRendererFtr {
  - LvRenderer\_LvAutoDisplay, LvRenderer\_LvRenderType, LvRenderer\_LvOffsetX, LvRenderer\_LvOffsetY,
  - LvRenderer\_LvWidth, LvRenderer\_LvHeight, LvRenderer\_LvIgnoreAspectRatio, LvRenderer\_LvDisableScaleUp,
  - LvRenderer\_LvDisableScaleDown, LvRenderer\_LvCenterImage, LvRenderer\_LvNumberOfTiles, LvRenderer\_LvColumns,
  - LvRenderer\_LvRows, LvRenderer\_LvTileGap, LvRenderer\_LvAutoTileCalculation, LvRenderer\_LvShowPixel,
  - LvRenderer\_LvPixelCoordinateX, LvRenderer\_LvPixelCoordinateY, LvRenderer\_LvPixelVal1, LvRenderer\_LvPixelVal2,
  - LvRenderer\_LvPixelVal3, LvRenderer\_LvShowHistogram, LvRenderer\_Info }
- enum LvEventFtr { LvEvent\_EventType, LvEvent\_NumInQueue, LvEvent\_NumFired }
- enum LvBufferFtr {
  - LvBuffer\_Base, LvBuffer\_Size, LvBuffer\_UserPtr, LvBuffer\_TimeStamp,
  - LvBuffer\_NewData, LvBuffer\_IsQueued, LvBuffer\_IsAcquiring, LvBuffer\_IsIncomplete,
  - LvBuffer\_TType, LvBuffer\_SizeFilled, LvBuffer\_Width, LvBuffer\_Height,
  - LvBuffer\_XOffset, LvBuffer\_YOffset, LvBuffer\_XPadding, LvBuffer\_YPadding,
  - LvBuffer\_FrameId, LvBuffer\_ImagePresent, LvBuffer\_ImageOffset, LvBuffer\_PayloadType,
  - LvBuffer\_PixelFormat, LvBuffer\_PixelFormatNameSpace, LvBuffer\_DeliveredImageHeight, LvBuffer\_DeliveredChunkPayloadSize,
  - LvBuffer\_ChunkLayoutId, LvBuffer\_FileName, LvBuffer\_UniBase, LvBuffer\_UniSize,
  - LvBuffer\_ProcessBase, LvBuffer\_ProcessSize, LvBuffer\_ExecProcess, LvBuffer\_UniImageOffset }

## 5.29.1 Detailed Description

### 5.29.2 Enumeration Type Documentation

#### 5.29.2.1 enum LvBufferFtr

LvBufferFtr constants.

#### Enumerator

**LvBuffer\_Base** Represents the GenTL BUFFER\_INFO\_BASE info - Base address of the buffer memory. LvFtrType\_Pointer.

**LvBuffer\_Size** Represents the GenTL BUFFER\_INFO\_SIZE info - Size of the buffer in bytes. LvFtrType\_Integer.

**LvBuffer\_UserPtr** Represents the GenTL BUFFER\_INFO\_USER\_PTR info - The user pointer (supplied by the application when the buffer was allocated). LvFtrType\_Pointer. Note: This pointer should not be used in the C++ API and .Net Class Library, where this pointer is utilized internally for the [LvBuffer](#) class instance. The actual User pointer is available by the [LvBuffer::GetUserPtr\(\)](#) function.

**LvBuffer\_TimeStamp** Represents the GenTL BUFFER\_INFO\_TIMESTAMP info - Timestamp the buffer was acquired. The unit is device/implementation dependent. LvFtrType\_Integer.

**LvBuffer\_NewData** Represents the GenTL BUFFER\_INFO\_NEW\_DATA info - Flag to indicate that the buffer contains new data since the last call. LvFtrType\_Boolean.

**LvBuffer\_IsQueued** Represents the GenTL BUFFER\_INFO\_IS\_QUEUED info - Flag to indicate if the buffer is in the input pool or output queue. LvFtrType\_Boolean.

**LvBuffer\_IsAcquiring** Represents the GenTL BUFFER\_INFO\_ISACQUIRING info - Flag to indicate that the buffer is currently being filled with data. LvFtrType\_Boolean.

- LvBuffer\_IsIncomplete** Represents the GenTL BUFFER\_INFO\_ISINCOMPLETE info - Flag to indicate that a buffer was filled, but an error occurred during that process. LvFtrType\_Boolean.
- LvBuffer\_TIType** Represents the GenTL BUFFER\_INFO\_TLTYPE info - Transport layer technologies that are supported. LvFtrType\_String.
- LvBuffer\_SizeFilled** Represents the GenTL BUFFER\_INFO\_SIZE\_FILLED info - Number of bytes written into the buffer last time it has been filled. This value is reset to 0 when the buffer is placed into the Input Buffer Pool. LvFtrType\_Integer.
- LvBuffer\_Width** Represents the GenTL 1.2 BUFFER\_INFO\_WIDTH info - Width of the data in the buffer in number of pixels. This information refers for example to the width entry in the GigE Vision image stream data leader. LvFtrType\_Integer.
- LvBuffer\_Height** Represents the GenTL 1.2 BUFFER\_INFO\_HEIGHT info - Height of the data in the buffer in number of pixels as configured. For variable size images this is the max Height of the buffer. For example this information refers to the height entry in the GigE Vision image stream data leader. LvFtrType\_Integer.
- LvBuffer\_XOffset** Represents the GenTL 1.2 BUFFER\_INFO\_XOFFSET info - XOffset of the data in the buffer in number of pixels from the image origin to handle areas of interest. This information refers for example to the information provided in the GigE Vision image stream data leader. LvFtrType\_Integer.
- LvBuffer\_YOffset** Represents the GenTL 1.2 BUFFER\_INFO\_YOFFSET info - YOffset of the data in the buffer in number of lines from the image origin to handle areas of interest. This information refers for example to the information provided in the GigE Vision image stream data leader. LvFtrType\_Integer.
- LvBuffer\_XPadding** Represents the GenTL 1.2 BUFFER\_INFO\_XPADDING info - XPadding of the data in the buffer in number of bytes. This information refers for example to the information provided in the GigE Vision image stream data leader. LvFtrType\_Integer.
- LvBuffer\_YPadding** Represents the GenTL 1.2 BUFFER\_INFO\_YPADDING info - YPadding of the data in the buffer in number of bytes. This information refers for example to the information provided in the GigE Vision image stream data leader. LvFtrType\_Integer.
- LvBuffer\_FrameId** Represents the GenTL 1.2 BUFFER\_INFO\_FRAMEID info - A sequentially incremented number of the frame. This information refers for example to the information provided in the GigE Vision image stream block id. The wrap around of this number is transportation technology dependent. For GigE Vision it is (so far) 16bit wrapping to 1. LvFtrType\_Integer.
- LvBuffer\_ImagePresent** Represents the GenTL 1.2 BUFFER\_INFO\_IMAGEPRESENT info - Flag to indicate if the current data in the buffer contains image data. This information refers for example to the information provided in the GigE Vision image stream data leader. LvFtrType\_Boolean.
- LvBuffer\_ImageOffset** Represents the GenTL 1.2 BUFFER\_INFO\_IMAGEOFFSET info - Offset of the image data from the beginning of the delivered buffer in bytes. Applies for example when delivering the image as part of chunk data or on technologies requiring specific buffer alignment. LvFtrType\_Integer.
- LvBuffer\_PayloadType** Represents the GenTL 1.2 BUFFER\_INFO\_PAYLOADTYPE info - Payload type of the data. This information refers to the constants defined in GenTL PAYLOADTYPE\_IDS (UNKNOWN=0, IMAGE=1, RAW\_DATA=2, FILE=3, CHUNK\_DATA=4, CUSTOM=1000) LvFtrType\_Integer.
- LvBuffer\_PixelFormat** Represents the GenTL 1.2 BUFFER\_INFO\_PIXELFORMAT info - This information refers for example to the information provided in the GigE Vision image stream data leader. The interpretation of the pixel format depends on the namespace the pixel format belongs to. This can be inquired using the LvBuffer\_PixelFormatNameSpace feature. LvFtrType\_Integer.
- LvBuffer\_PixelFormatNameSpace** Represents the GenTL 1.2 BUFFER\_INFO\_PIXELFORMAT\_NAMESPAC↵  
PACE info - This information refers to the constants defined in GenTL 1.2 PIXELFORMAT\_NAMESPAC↵  
E\_IDS to allow interpretation of LvBuffer\_PixelFormat (UNKNOWN=0, GEV=1, IIDC=2, CUSTOM=1000). LvFtrType\_Integer.
- LvBuffer\_DeliveredImageHeight** Represents the GenTL 1.2 BUFFER\_INFO\_DELIVERED\_IMAGEHEIG↵  
HT info - The number of lines in the current buffer as delivered by the transport mechanism. For area scan type images this is usually the number of lines configured in the device. For variable size linescan images this number may be lower than the configured image height. This information refers for example to the information provided in the GigE Vision image stream data trailer. LvFtrType\_Integer.
- LvBuffer\_DeliveredChunkPayloadSize** Represents the GenTL 1.2 BUFFER\_INFO\_DELIVERED\_CHUN↵  
KPAYLOADSIZE info - This information refers for example to the information provided in the GigE Vision image stream data leader. LvFtrType\_Integer.

**LvBuffer\_ChunkLayoutId** Represents the GenTL 1.2 BUFFER\_INFO\_CHUNKLAYOUTID info - This information refers for example to the information provided in the GigE Vision image stream data leader. The chunk layout id serves as an indicator that the chunk layout has changed and the application should re-parse the chunk layout in the buffer. When a chunk layout (availability or position of individual chunks) changes since the last buffer delivered by the device through the same stream, the device MUST change the chunk layout id. As long as the chunk layout remains stable, the camera MUST keep the chunk layout id intact. When switching back to a layout, which was already used before, the camera can use the same id again or use a new id. A chunk layout id value of 0 is invalid. It is reserved for use by cameras not supporting the layout id functionality. LvFtrType\_Integer.

**LvBuffer\_FileName** Represents the GenTL 1.2 BUFFER\_INFO\_FILENAME info - This information refers for example to the information provided in the GigE Vision image stream data leader. For other technologies this is to be implemented accordingly. Since this is GigE Vision related information and the filename in GigE Vision is UTF8 coded, this filename is also UTF8 coded. LvFtrType\_Integer.

**LvBuffer\_UniBase** Unified base address of the buffer. If the image was processed to the output buffer, the pointer to the output buffer is returned, otherwise the pointer to the acquisition buffer is returned. This enables to write simple universal code for image handling. LvFtrType\_Pointer. SynView feature.

**LvBuffer\_UniSize** Size of the buffer returned on [LvBuffer\\_UniBase](#) call. LvFtrType\_Integer. SynView feature.

**LvBuffer\_ProcessBase** Pointer to the process buffer, attached to this acquisition buffer. LvFtrType\_Pointer. SynView feature.

**LvBuffer\_ProcessSize** Size of the process buffer, attached to this acquisition buffer. LvFtrType\_Integer. SynView feature.

**LvBuffer\_ExecProcess** Executes the SW image processing of the buffer. To be used when the [LvDevice↔\\_LvUniProcessExecution](#) is set to [LvUniProcessExecution\\_OnExplicitRequest](#). LvFtrType\_Command. SynView feature.

**LvBuffer\_UniImageOffset** Unified image offset. If the image was processed to the output buffer, the image offset to the output buffer is returned, otherwise the image offset to the acquisition buffer is returned. This enables to write simple universal code for image handling. LvFtrType\_Integer. SynView feature.

### 5.29.2.2 enum LvDeviceFtr

LvDeviceFtr constants.

Enumerator

**LvDevice\_DeviceVendorName** Name of the manufacturer of the device. LvFtrType\_String. Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

**LvDevice\_DeviceModelName** Model name of the device. LvFtrType\_String. Device remote feature ([LvFtr↔\\_Group\\_DeviceRemote](#)).

**LvDevice\_DeviceManufacturerInfo** Manufacturer information about the device. LvFtrType\_String. Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

**LvDevice\_DeviceVersion** Version of the device. LvFtrType\_String. Device remote feature ([LvFtrGroup\\_↔\\_DeviceRemote](#)).

**LvDevice\_DeviceFirmwareVersion** Version of the firmware loaded in the device. LvFtrType\_String. Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

**LvDevice\_LvRecoveryFirmwareVersion** String that indicates the version of the firmware and software to which the device would recover. LvFtrType\_String. Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

**LvDevice\_DeviceSerialNumber** Device identifier (serial number). LvFtrType\_String. Note: This feature is called DeviceID in the SFNC, but we use rather the DeviceSerialNumber in order not to confuse it with the GenTL DeviceID, which is used for the device opening. Device remote feature ([LvFtrGroup\\_Device↔\\_Remote](#)).

**LvDevice\_DeviceUserID** User-programmable device identifier. LvFtrType\_String. Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

- LvDevice\_LvSensorID** Serial number of the sensor board. [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvGrabberID** Serial number of the grabber board. [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceScanType** Scan type of the sensor. [LvFtrType\\_Enumeration](#). Values: [LvDeviceScanType](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceRegistersStreamingStart** Prepare the device for registers streaming without checking for consistency. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceRegistersStreamingEnd** Announce the end of registers streaming. This will do a register set validation for consistency and activate it. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceRegistersCheck** Perform the validation of the current register set for consistency. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceRegistersValid** Reports if the current register set is valid and consistent. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceReset** Resets the device and to put it in its power up state. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceClockSelector** Selects a device clock frequency to be configured. [LvFtrType\\_Enumeration](#). Values: [LvDeviceClockSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceClockFrequency** Frequency of the selected clock in Hz. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceTemperatureSelector** Selects the location within the device, where the temperature will be measured. [LvFtrType\\_Enumeration](#). Values: [LvDeviceTemperatureSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceTemperature** Current temperature at the selected location in degrees of Celcius [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvDeviceUpTime** Current up-time of the device in milliseconds. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvDeviceType** String that indicates the basic type of the device. [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_SensorWidth** Effective width of the sensor in pixels. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_SensorHeight** Effective height of the sensor in pixels. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_WidthMax** Maximum width of the image in pixels. The dimension is calculated after applying horizontal binning, decimation or readout width. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_HeightMax** Maximum height of the image in pixels. The dimension is calculated after applying vertical binning, decimation or readout height. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_Width** Image width provided by the device in pixels. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_Height** Image height provided by the device in pixels. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_OffsetX** Horizontal offset from the origin of the AOI (area of interest) in pixels. The AOI is applied to the result of binning and or decimation. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_OffsetY** Vertical offset from the origin of the AOI (area of interest) in pixels. The AOI is applied to the result of binning and or decimation. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_PixelFormat** Pixel format provided by the device. The feature combines pixel coding, size and color filter attributes. [LvFtrType\\_Enumeration](#). Values: see [LvPixelFormat](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).



- LvDevice\_BinningHorizontal** Horizontal binning, number of horizontal pixels to combine together. This increases the intensity (and S/N ratio) of the pixels and reduces the horizontal resolution (width) of the image. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_BinningVertical** Vertical binning, number of vertical pixels to combine together. This increases the intensity (and S/N ratio) of the pixels and reduces the vertical resolution (height) of the image. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DecimationHorizontal** Horizontal decimation (sub-sampling) of the image. This reduces the horizontal resolution (width) of the image by the specified factor. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DecimationVertical** Vertical decimation (sub-sampling) of the image. This reduces the vertical resolution (height) of the image by the specified factor. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvAOIMode** Selects the mode of controlling the area of interest [LvFtrType\\_Enumeration](#). Values: [LvAOIMode](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvReadoutWidth** Width of the sensor-side area of interest in pixels. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvReadoutHeight** Height of the sensor-side area of interest in pixels. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvReadoutOffsetX** X offset (left offset) for the sensor-side area of interest in pixels. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvReadoutOffsetY** Y offset (top offset) for the sensor-side area of interest in pixels. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvVariablePayloadSize** This flag controls, whether the payload size can change during acquisition. When set, the image dimensions and other parameters can vary during acquisition. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_AcquisitionMode** Sets the acquisition mode of the device. It defines mainly the number of frames to capture during an acquisition and the way the acquisition stops. [LvFtrType\\_Enumeration](#). Values: [LvAcquisitionMode](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_TriggerSelector** Selects the type of trigger to configure. [LvFtrType\\_Enumeration](#). Values: [LvTriggerSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_TriggerMode** Controls if the selected trigger is active. [LvFtrType\\_Enumeration](#). Values: [LvTriggerMode](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_TriggerSoftware** Generates a software trigger when trigger source is set to 'software' or any physical line. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_TriggerSource** Specifies the internal signal or physical input line to use as the trigger source. [LvFtrType\\_Enumeration](#). Values: [LvTriggerSource](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_TriggerActivation** Activation mode of the trigger - specifies which edge of the signal is active. [LvFtrType\\_Enumeration](#). Values: [LvTriggerActivation](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_TriggerDelay** Trigger delay in microseconds, specifies a delay introduced between the trigger reception and its actual activation. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_TriggerDivider** Used to divide the number of incoming trigger pulses by an integer factor. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvTriggerCaching** Sets the caching mode for the selected trigger. The feature controls how early triggers are treated by the device. [LvFtrType\\_Enumeration](#). Values: [LvTriggerCaching](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ExposureMode** Controls the exposure (shutter) mode applied for each acquisition. [LvFtrType\\_Enumeration](#). Values: [LvExposureMode](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLongRangeExposureMode** Switches to mode with wider range of exposure times, but slightly higher jitter. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

- LvDevice\_LvGlobalResetMode** Switches to mode with wider range of exposure times, but slightly higher jitter. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ExposureTime** Exposure time in microseconds. The feature controls how long are the pixels exposed to illumination. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ExposureAuto** Selects the automatic exposure mode. [LvFtrType\\_Enumeration](#). Values: [LvExposureAuto](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvAcquisitionFrameRateControlMode** Switches the acquisition frame rate control on or off. The camera might internally switch to different working mode, which can decrease the maximum frame rate. [LvFtrType\\_Enumeration](#). Values: [LvAcquisitionFrameRateControlMode](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_AcquisitionFrameRate** Acquisition frame rate in frames per second (Hz) - the frequency with which the image frames are captured. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LineSelector** Selects the I/O line for querying and configuration. Note that to use given line to drive a device feature (trigger, counter, etc.), source of the given feature has to refer to the line. [LvFtrType\\_Enumeration](#). Values: [LvLineSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LineMode** Line mode - controls, whether given line is used as signal input or output. [LvFtrType\\_Enumeration](#). Values: [LvLineMode](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LineFormat** This feature controls the current electrical format of the selected physical input or output Line. [LvFtrType\\_Enumeration](#). Values: [LvLineFormat](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LineSource** Selects a device internal signal that should drive the output signal of the selected line. LineMode must be Output. Not applicable for input lines. [LvFtrType\\_Enumeration](#). Values: [LvLineSource](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LineInverter** Inverts the signal output on the selected line. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LineStatus** Reports the current status of the selected line. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LineStatusAll** Bit field indicating status of all i/o lines. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_UserOutputSelector** Selects the user output for querying and configuration. [LvFtrType\\_Enumeration](#). Values: [LvUserOutputSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_UserOutputValue** Reports the current status of the selected user output. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_UserOutputValueAll** Bit field indicating status of all user outputs. Only the bits defined in the User Output Value All Mask are used, the others are ignored. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_UserOutputValueAllMask** Mask for the User Output Value All bitfield - defines which bits are used to change a user output value and which are ignored. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_CounterSelector** Selects which counter to configure. [LvFtrType\\_Enumeration](#). Values: [LvCounterSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCounterMode** Selects working mode of the selected counter. [LvFtrType\\_Enumeration](#). Values: [LvCounterMode](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_CounterEventSource** Internal device signal incrementing the selected counter. [LvFtrType\\_Enumeration](#). Values: [LvCounterEventSource](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_CounterReset** This command resets the selected counter [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_CounterValue** Reads or sets the current value of the selected counter. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).



- LvDevice\_CounterDuration** Duration (or number of events) before the counter end event is generated and the counter expires. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_TimerSelector** Selects which timer to configure. [LvFtrType\\_Enumeration](#). Values: [LvTimerSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_TimerDuration** Sets the duration (in microseconds) of the timer active pulse. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_TimerDelay** Sets the delay (in microseconds) applied between activating the timer and issuing the timer active signal. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_TimerTriggerSource** Internal device signal activating the selected timer. [LvFtrType\\_Enumeration](#). Values: [LvTimerTriggerSource](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSpecialPurposeTriggerSelector** Selects the special purpose trigger type to configure. [LvFtrType\\_Enumeration](#). Values: [LvSpecialPurposeTriggerSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSpecialPurposeTriggerSource** Specifies the internal signal or physical input line to use as the trigger source. [LvFtrType\\_Enumeration](#). Values: [LvSpecialPurposeTriggerSource](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSpecialPurposeTriggerActivation** Activation mode of the trigger - specifies which edge of the signal is active. [LvFtrType\\_Enumeration](#). Values: [LvSpecialPurposeTriggerActivation](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSpecialPurposeTriggerSoftware** Generates a software trigger for the selected trigger action when trigger source is set to 'software' or any physical line. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvImageStampsResetMask** A single bitfield that selects which features will be reset by the timestamp reset trigger in one access. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvImageStampSelector** Selects an image stamp type for configuration. [LvFtrType\\_Enumeration](#). Values: [LvImageStampSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvImageStampResetEnable** Enables/disables the reset trigger functionality for the selected image stamp type. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvBootSwitch** Selects the firmware type to load on next boot. [LvFtrType\\_Enumeration](#). Values: [LvBootSwitch](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvBayerDecoderAlgorithm** Selects the algorithm used by the Bayer decoder. [LvFtrType\\_Enumeration](#). Values: [LvBayerDecoderAlgorithm](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvBayerDecoderThreshold** Sets the threshold controlling the performance of the variable gradient Bayer decoder algorithm. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvWatchdogEnable** Enables the watchdog reset function. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvWatchdogTimerDuration** When watchdog is enabled, the device reboots when the timeout specified expires. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvWatchdogTimerReset** Resets the watchdog timer, the watchdog starts counting the specified timeout again. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvWatchdogFailed** Signals that the last device reboot was initiated by the watchdog function. After reading, reset this flag explicitly, it wouldn't be affected by a 'warm' system reboot. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GainSelector** Selects which gain type to configure. [LvFtrType\\_Enumeration](#). Values: [LvGainSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_Gain** Gain value for the selected gain type in dB. This is an amplification factor applied to the video signal. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GainAuto** Controls the automatic gain control (AGC) mode. [LvFtrType\\_Enumeration](#). Values: [LvGainAuto](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

- LvDevice\_BlackLevelSelector** Selects which black level type to configure. [LvFtrType\\_Enumeration](#). Values: [LvBlackLevelSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_BlackLevel** Controls the analog black level. This represents a DC offset applied to the video signal. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_BlackLevelAuto** Controls the automatic black level mode. [LvFtrType\\_Enumeration](#). Values: [LvBlackLevelAuto](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ColorTransformationSelector** Selects which color transformation module is controlled by the color transformation features. It also gives particular meaning to individual color transformation gains. [LvFtrType\\_Enumeration](#). Values: [LvColorTransformationSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ColorTransformationEnable** Activates the selected Color Transformation module. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ColorTransformationValueSelector** Selects the gain factor or offset of the transformation matrix to configure [LvFtrType\\_Enumeration](#). Values: [LvColorTransformationValueSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ColorTransformationValue** Value of the selected color transformation matrix entry. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvExternalDeviceControlMode** Selects the operation mode of external device control. [LvFtrType\\_Enumeration](#). Values: [LvExternalDeviceControlMode](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvExternalADCSelector** Selects the external ADC to configure. [LvFtrType\\_Enumeration](#). Values: [LvExternalADCSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvExternalADCValue** Reads the value of the selected external ADC. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvPowerSwitchCurrentAction** Reports the automated action currently performed by a power switch. [LvFtrType\\_Enumeration](#). Values: [LvPowerSwitchCurrentAction](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvPowerSwitchSelector** Selects the power switch to configure. [LvFtrType\\_Enumeration](#). Values: [LvPowerSwitchSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvPowerSwitchBoundADC** Sets an external ADC to the selected power switch. The bound pair will work together during the automatic operation. [LvFtrType\\_Enumeration](#). Values: [LvPowerSwitchBoundADC](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvPowerSwitchDrive** Drives the selected power switch with desired polarity. [LvFtrType\\_Enumeration](#). Values: [LvPowerSwitchDrive](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvPowerSwitchPulsePlus** Pulses the selected power switch with plus polarity. Available in the automatic operation mode. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvPowerSwitchPulseMinus** Pulses the selected power switch with minus polarity. Available in the automatic operation mode. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLensControlCalibrate** Starts an automatic calibration on the selected power switch and bounded ADCs. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLensControlMinusEnd** Represents the calibrated minimal ADC achievable by driving the power switch's with minus polarity (plus if the polarity is inverted). [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLensControlPlusEnd** Represents the calibrated maximal ADC achievable by driving the power switch's with plus polarity (minus if the polarity is inverted). [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLensControlPulsePeriod** Represents the calibrated slow motion pulse period for the selected power switch, in microseconds. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLensControlDutyCycle** Represents the calibrated slow motion duty cycle for the selected power switch (in %). Defines how much of the pulse period is the power switch actually active. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

- LvDevice\_LvLensControlTargetApproach** Selects how the target lens position should be approached. [LvFtrType\\_Enumeration](#). Values: [LvLensControlTargetApproach](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLensControlNrSlowSteps** Sets the number of slow steps required before reaching the target position [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLensControlTargetPosition** Sets the target position (value) of the ADC bound to the selected power switch [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLensControlAdjustPosition** Adjusts the required target position (value) of the ADC bound to the selected power switch. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvPowerSwitchPulseDuration** Duration (in microseconds) of the pulses issued at the power switch. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLensControlMinCalibrationRange** Minimum value range that has to be reached on the external ADC to count the calibration as valid. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLensControlCalibrateAll** Starts an automatic calibration on the active power switches and bounded ADCs. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LUTSelector** Selects which LUT to configure. [LvFtrType\\_Enumeration](#). Values: [LvLUTSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LUTEnable** Activates the selected LUT.. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LUTIndex** Index of the element to access in the selected LUT [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LUTValue** Value of the element for the current index in the selected LUT. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LUTValueAll** This register accesses the entire content of the selected LUT in one chunk access. [LvFtrType\\_Buffer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_PayloadSize** Provides the number of bytes transferred for each image by the device, including image and chunk data. The value defines the required size of the target buffer used for acquisition. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevVersionMajor** Major version of the GigE Vision specification implemented by the device. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevVersionMinor** Minor version of the GigE Vision specification implemented by the device. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevDeviceModelsBigEndian** Endianness of the device registers. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevDeviceModeCharacterSet** Character set used by all the strings of the device registers. [LvFtrType\\_Enumeration](#). Values: [LvGevDeviceModeCharacterSet](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevInterfaceSelector** Selects which physical network interface to control. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevMACAddress** MAC address of the network interface. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSupportedOptionSelector** Selects the GEV option to interrogate for existing support. [LvFtrType\\_Enumeration](#). Values: [LvGevSupportedOptionSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSupportedOption** Returns if the selected GEV option is supported. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevCurrentIPConfigurationLLA** Indicates if Link Local Address IP configuration scheme is activated on the given network interface. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevCurrentIPConfigurationDHCP** Indicates if DHCP IP configuration scheme is activated on the given network interface. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

- LvDevice\_GevCurrentIPConfigurationPersistentIP** Indicates if persistent IP configuration scheme is activated on the given network interface. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevCurrentIPAddress** Reports the IP address for the given network interface. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevCurrentSubnetMask** Provides the subnet mask of the given interface. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevCurrentDefaultGateway** Indicates the default gateway IP address to be used on the given network interface. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevPersistentIPAddress** Indicates the persistent IP address for this network interface. It is only used when the device boots with the persistent IP configuration scheme. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevPersistentSubnetMask** Indicates the persistent subnet mask associated with the persistent IP address on this network interface. It is only used when the device boots with the Persistent IP configuration scheme. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevPersistentDefaultGateway** Indicates the persistent default gateway for this network interface. It is only used when the device boots with the persistent IP configuration scheme. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevNumberOfInterfaces** Indicates the number of physical network interfaces supported by this device. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevMessageChannelCount** Indicates the number of message channels supported by this device. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevStreamChannelCount** Indicates the number of stream channels supported by this device. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevHeartbeatTimeout** Indicates the current heartbeat timeout in milliseconds. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevTimestampTickFrequency** Indicates the number of timestamp ticks during 1 second (frequency in Hz). [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevTimestampControlLatch** Latches current timestamp counter into [GevTimestampValue](#). [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevTimestampControlReset** Resets the Timestamp counter to 0. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevTimestampControlLatchReset** Reset and latch in a single command. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevTimestampValue** Returns the latched 64-bit value of the timestamp counter. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevCCP** Controls the device access privilege of an application. [LvFtrType\\_Enumeration](#). Values: [LvGevCCP](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevStreamChannelSelector** Selects the stream channel to control. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSCPIInterfaceIndex** Index of network interface to use. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSCPHostPort** Indicates the port to which the device must send data stream. Setting this value to 0 closes the stream channel. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSCPSFireTestPacket** Sends a test packet. When this feature is set, the device will fire one test packet. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSCPSDoNotFragment** The state of this feature is copied into the "do not fragment" bit of IP header of each stream packet. It can be used by the application to prevent IP fragmentation of packets on the stream channel. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSCPSBigEndian** Specifies the stream packet size in bytes to send on this channel. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

- LvDevice\_GevSCSPacketSize** Specifies the stream packet size in bytes to send on this channel. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSCPD** Indicates the delay (in timestamp counter unit, which is currently a microsecond) to insert between each packet for this stream channel. This can be used as a crude flow-control mechanism if the application or the network infrastructure cannot keep up with the packets coming from the device. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSCDA** Indicates the destination IP address for this stream channel. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevLinkSpeed** Indicates the speed of transmission negotiated by the given network interface in Mb/s. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_UserSetSelector** Selects the feature configuration user set to load, save or configure. [LvFtrType\\_Enumeration](#). Values: [LvUserSetSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_UserSetLoad** Loads the selected user configuration set and makes it active [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_UserSetSave** Saves the current device configuration into the selected user configuration set. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_UserSetDefaultSelector** Selects the default feature configuration set to be loaded and activated upon camera boot or reset. [LvFtrType\\_Enumeration](#). Values: [LvUserSetDefaultSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkModeActive** Activates the chunk mode, ie. inclusion of chunk data in the payload data. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkSelector** Selects the chunk to configure. [LvFtrType\\_Enumeration](#). Values: [LvChunkSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkEnable** Enables the inclusion of the selected chunk in the payload data. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkOffsetX** X offset applied the image included in the payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkOffsetY** Y offset applied the image included in the payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkWidth** Width of the image included in the payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkHeight** Height of the image included in the payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkPixelFormat** Pixel format of the image included in the payload. [LvFtrType\\_Enumeration](#). Values: see [LvPixelFormat](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLinePitch** Line pitch of the image included in the payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkFrameID** Frame id of the image included in the payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkTimestamp** Timestamp associated with the image included in the payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkExposureTime** Exposure time used to acquire the image included in the payload. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkGainSelector** Selects the gain type to be reported in chunk data. [LvFtrType\\_Enumeration](#). Values: [LvChunkGainSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkGain** Gain used to acquire the image included in the payload. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkBlackLevel** Black level used to acquire the image included in the payload. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLineStatusAll** Bit field indicating status of all i/o lines at the time the image included in the payload was acquired. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).



- LvDevice\_ChunkLvExternalADCSelector** Selects the external ADC to be reported in chunk data. [LvFtrType\\_Enumeration](#). Values: [LvChunkLvExternalADCSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvExternalADCValue** Reads the value of the selected external ADC at time of acquisition of the image included in the payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventSelector** Selects which event to signal to the host application. [LvFtrType\\_Enumeration](#). Values: [LvEventSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventNotification** Activate or deactivate the notification to the host application of the selected event occurrence. [LvFtrType\\_Enumeration](#). Values: [LvEventNotification](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppID** ID string the smart application [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt1** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt2** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt3** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt4** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt5** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt6** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt7** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt8** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt9** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt10** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt11** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt12** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt13** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt14** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt15** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt16** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt17** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt18** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppInt19** Generic signed integer register controlling a smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).



- LvDevice\_LvSmartAppUint13** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint14** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint15** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint16** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint17** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint18** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint19** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint20** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint21** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint22** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint23** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint24** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint25** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint26** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint27** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint28** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint29** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint30** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint31** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppUint32** Generic unsigned integer register controlling a smart application. [LvFtrType↔\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppAsciiCmdString** Characters of the ASCII command for the smart application. [Lv↔FtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppAsciiCmdExecute** Executes the ASCII command for the smart application. [LvFtr↔Type\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppAsciiCmdFeedback** Response to the ASCII command for the smart application. [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppAsciiCmdRetCode** Numeric return code of the ASCII command for the smart application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppPath** Path of the smart application to be started [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).



- LvDevice\_LvSmartAppStart** Starts the smart application defined by the path. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvLog** Returns the unique identifier of the log type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvLogTimestamp** Returns the timestamp of the log event. It can be used to determine precisely when the event occurred. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvLogMessage** The log message coming with the event [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppLog** Returns the unique identifier of the smart application log type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppLogTimestamp** Returns the timestamp of the smart application Smart Application Log Event. It can be used to determine precisely when the event occurred. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppLogMessage** The smart application log message coming with the event. [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSerialPortBaudRate** Baud rate used for the serial port communication. [LvFtrType\\_Enumeration](#). Values: [LvSerialPortBaudRate](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSerialPortParity** Parity used for the serial port communication. [LvFtrType\\_Enumeration](#). Values: [LvSerialPortParity](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSerialPortDataBits** Data bits per character for the serial port communication. [LvFtrType\\_Enumeration](#). Values: [LvSerialPortDataBits](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSerialPortStopBits** Stop bits per character for the serial port communication. [LvFtrType\\_Enumeration](#). Values: [LvSerialPortStopBits](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSerialPortTimeout** Timeout value used to finish waiting for command response [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSerialPortEOTMarker** Short string (or single character) marking end of transmission. [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSerialPortMaxResponseLength** Maximal expected length of the command response. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSerialPortCommandString** String of the ASCII command to be sent over the serial port. [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSerialPortCommandSend** Sends the ASCII command over the serial port Command. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSerialPortCommandResponse** Response to the ASCII command sent over the serial port StringReg. [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSerialPortCommandStatus** Status code indicating success of the last command. Values: [LvSerialPortCommandStatus](#). [LvFtrType\\_Enumeration](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvSmartAppExitEvent** Sends an exit event to the running smart application. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvWatchdogTimerValue** Current watchdog timer value - reports the current value, after which the timer expires and the device reboots. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLensControllInvertedPolarity** Indicates if the lens is wired with inverted polarity, meaning that driving the power switch to the plus side decreases the external ADC feedback. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

- LvDevice\_GevMCPHostPort** Controls the port to which the device must send messages. Setting this value to 0 closes the message channel. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevMCDA** Controls the destination IP address for the message channel. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevMCTT** Provides the transmission timeout value in milliseconds. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevMCRC** Controls the number of retransmissions allowed when a message channel message times out. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvSmartAppString** The smart application string related to the delivered payload. [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvSmartAppIntSelector** Selects one of the signed integer values related to the delivered payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvSmartAppInt** The selected smart application signed integer related to the delivered payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvSmartAppUIntSelector** Selects one of the unsigned integer values related to the delivered payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvSmartAppUInt** The selected smart application unsigned integer related to the delivered payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvSmartAppRegister** The smart application raw register related to the delivered payload. [LvFtrType\\_Buffer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppString** Returns the unique identifier of the smart application string type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppStringTimestamp** Returns the timestamp of the smart application string event. It can be used to determine precisely when the event occurred. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppStringValue** The smart application string value coming with the event. [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppInt** Returns the unique identifier of the smart application signed integer type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppIntTimestamp** Returns the timestamp of the smart application signed integer event. It can be used to determine precisely when the event occurred. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppIntSelector** Selects one of the signed integer values coming with the event. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppIntValue** Value of the selected signed integer coming with the event. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppUInt** Returns the unique identifier of the smart application unsigned integer type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppUIntTimestamp** Returns the timestamp of the smart application unsigned integer event. It can be used to determine precisely when the event occurred. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppUIntSelector** Selects one of the unsigned integer values coming with the event. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppUIntValue** Value of the selected unsigned integer coming with the event. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

- LvDevice\_EventLvSmartAppRegister** Returns the unique identifier of the smart application raw register type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppRegisterTimestamp** Returns the timestamp of the smart application raw register event. It can be used to determine precisely when the event occurred. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvSmartAppRegisterValue** The smart application raw register value coming with the event Register. [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceSFNCVersionMajor** Major version of the Standard Feature Naming Convention that was used to create the device's XML. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceSFNCVersionMinor** Minor version of the Standard Feature Naming Convention that was used to create the device's XML. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceSFNCVersionSubMinor** Sub-minor version of Standard Feature Naming Convention that was used to create the device's XML. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLineDebounceDuration** Sets the duration (in microseconds) of the line debounce period. Value of 0.0 switches the debouncer off. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ActionDeviceKey** Provides the device key that allows the device to check the validity of action commands. The device internal assertion of an action signal is only authorized if the ActionDeviceKey and the action device key value in the protocol message are equal. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ActionSelector** Selects to which action signal further action settings apply. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ActionGroupKey** Provides the key that the device will use to validate the action on reception of the action protocol message. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ActionGroupMask** Provides the mask that the device will use to validate the action on reception of the action protocol message. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLensControlCalibrationStatus** Reports current calibration status of the selected power switch and its bound ADC. The status is computed from the current ADC range, no matter if it is a result of calibration operation or configured manually. [LvFtrType\\_Enumeration](#). Values: [LvLensControlCalibrationStatus](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLUTMode** Selects the LUT control mode. [LvFtrType\\_Enumeration](#). Values: [LvLUTMode](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_BalanceRatioSelector** Selects which color channel to configure for white-balancing. [LvFtrType\\_Enumeration](#). Values: [LvBalanceRatioSelector](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_BalanceRatio** Controls white balance ratio coefficient to be applied on the selected color channel. Note that the white balance functionality is implemented using the LUT. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_BalanceWhiteAuto** Controls the mode for automatic white balancing between the color channels. The white balancing ratios are automatically adjusted. Note that the white balance functionality is implemented using the LUT. [LvFtrType\\_Enumeration](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevDeviceClass** Returns the class of the device. [LvFtrType\\_Enumeration](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevIPConfigurationStatus** Reports the current IP configuration status, ie. the method through which the network interface was configured. [LvFtrType\\_Enumeration](#). Values: [LvGevIPConfigurationStatus](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

- LvDevice\_GevDiscoveryAckDelay** Indicates the maximum randomized delay the device will wait to acknowledge a discovery command. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevGVCPExtendedStatusCodes** Enables the generation of extended status codes. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevGVCPPendingAck** Enables the generation of PENDING\_ACK. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevGVCPHeartbeatDisable** Disables the GVCP heartbeat. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevGVCPPendingTimeout** Indicates the longest GVCP command execution time before a device returns a PENDING\_ACK. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevPrimaryApplicationSwitchoverKey** Controls the key to use to authenticate primary application switchover requests. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevPrimaryApplicationSocket** Returns the UDP source port of the primary application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevPrimaryApplicationIPAddress** Returns the address of the primary application. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevMCSP** This feature indicates the source port for the message channel. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSCCFGUnconditionalStreaming** Enables the camera to continue to stream, for this stream channel, if its control channel is closed or regardless of the reception of any ICMP messages (such as destination unreachable messages). [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSCCFGExtendedChunkData** Enables cameras to use the extended chunk data payload type for this stream channel. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSCPDiretion** Reports the direction of the stream channel. [LvFtrType\\_Enumeration](#). Values: [LvGevSCPDiretion](#) Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_GevSCSP** Indicates the source port of the stream channel. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvTriggerDelayed** Flag indicating if the trigger was delayed when acquiring the image included in the payload. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvTriggerDropped** Returns the unique identifier of the dropped trigger type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_EventLvTriggerDroppedTimestamp** Returns the timestamp of the dropped trigger event. It can be used to determine precisely when the event occurred. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvStrobeEnable** Selects the LED clusters of the strobe light that should be enabled. [LvFtrType\\_Enumeration](#). Values: [LvStrobeEnable](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvStrobeDurationMode** Controls the way how the maximum time of strobe duration is calculated. [LvFtrType\\_Enumeration](#). Values: [LvStrobeDurationMode](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvStrobeDuration** Duration of the strobe pulse in usec. The maximum time depends on the setting of Strobe Duration Mode feature. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvStrobeDelay** A delay before the strobe pulse starts after frame trigger is applied in usec. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvStrobeBrightness** Brightness (in %) of the strobe light. Allows to lower the full brightness of the strobe. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvStrobeDropMode** Sets mode of handling of strobes not matching the device hardware constraints. If a strobe is required (activated by a frame trigger) before the strobe device is ready, the strobe must be dropped or delayed. [LvFtrType\\_Enumeration](#). Values: [LvStrobeDropMode](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).

- LvDevice\_LvLUTReset** Resets the LUT settings. [LvFtrType\\_Command](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvStrobeDropped** Flag indicating if the configured strobe was dropped when acquiring the image included in the payload. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ReverseX** Flip horizontally the image sent by the device. The AOI is applied after the flipping. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ReverseY** Flip vertically the image sent by the device. The AOI is applied after the flipping. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_RegionSelector** Selects the region of interest to control. The RegionSelector feature allows devices that are able to extract multiple regions out of an image, to configure the features of those individual regions independently. [LvFtrType\\_Enumeration](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_RegionMode** Controls if the selected Region of interest is active and streaming. [LvFtrType\\_Enumeration](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_RegionDestination** Control the destination of the selected region. [LvFtrType\\_Enumeration](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_AcquisitionFrameCount** Number of frames to acquire in MultiFrame Acquisition mode. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_AcquisitionBurstFrameCount** Number of frames to acquire for each FrameBurstStart trigger. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomID** Revision number of the custom module. [LvFtrType\\_String](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomInfo** Info register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomRegMode** Controls the way of addressing a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomRegAddr** Defines the address of a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomRegData** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomRegMux** Defines the address and transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LinePitch** Total number of bytes between 2 successive lines. This feature is used to facilitate alignment of image data. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvFrameAbort** Flag indicating if a frame was dropped when acquiring the image included in the payload. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvTriggerDropped** Flag indicating if a trigger was dropped when acquiring the image included in the payload. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvTriggerError** Flag indicating if a mismatch between trigger and sensor data was detected when acquiring the image included in the payload. [LvFtrType\\_Boolean](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvEncoderPosition** Encoder position generating the trigger for the image included in the payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_ChunkLvEncoderRotation** Encoder rotation generating the trigger for the image included in the payload. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_RegionNumber** Defines the number of regions (Sequence of exposures). [LvFtrType\\_Enumeration](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_UserSetDefault** Selects the default feature configuration set to be loaded and activated upon camera boot or reset. [LvFtrType\\_Enumeration](#). Values: [LvUserSetDefault](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvLineDebounceMode** Sets the mode of the line debounce filter. [LvFtrType\\_Enumeration](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).



- LvDevice\_Gamma** Sets the gamma value. [LvFtrType\\_Float](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg1** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg2** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg3** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg4** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg5** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg6** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg7** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg8** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg9** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg10** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg11** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg12** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg13** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg14** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg15** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomReg16** Transfers data to and from a register in the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_LvCustomBypass** Bypass the custom module. [LvFtrType\\_Integer](#). Device remote feature ([LvFtrGroup\\_DeviceRemote](#)).
- LvDevice\_DeviceID** Device ID. [LvFtrType\\_String](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_DeviceType** Device type. [LvFtrType\\_Enumeration](#). Values: [LvDeviceType](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_GevDeviceIPAddress** Device IP address. [LvFtrType\\_Integer](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_GevDeviceSubnetMask** Device subnet mask. [LvFtrType\\_Integer](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_GevDeviceMACAddress** Device MAC address. [LvFtrType\\_Integer](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_GevDeviceGateway** Device gateway. [LvFtrType\\_Integer](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_LvGevDeviceStreamCaptureMode** Device stream capture mode. Controls, how the GVSP (image) stream is processed by the host machining. [LvFtrType\\_Enumeration](#). Values: [LvGevDeviceStreamCaptureMode](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).

- LvDevice\_StreamSelector** Stream selector. [LvFtrType\\_Integer](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_StreamID** Stream ID. [LvFtrType\\_String](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_DeviceEndiannessMechanism** Identifies the endianness mode. This feature should be set to 'standard' for all GigE Vision remote devices based on GenICam schema version 1.1 (and newer). The value 'legacy' is intended for devices using GenICam schema version 1.0. Values: [LvDeviceEndiannessMechanism](#). [LvFtrType\\_Enumeration](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_LvGevFindMaxPacketSize** Determine the maximal usable packet size for streaming. The procedure is already applied automatically when opening the device. Do not use during active streaming. [LvFtrType\\_Command](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_LvGevPacketSizeValue** Streaming packet size to be verified using the Test Packet Size command. Do not use during active streaming. [LvFtrType\\_Integer](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_LvGevTestPacketSize** Test if the selected packet size is suitable for streaming in current network configuration. [LvFtrType\\_Command](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_LvGevPacketSizeTestSuccess** Reports success of the last packet size test command. [LvFtrType\\_Boolean](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_LvGevCCTT** Controls the GigE Vision control channel transmission timeout value in milliseconds. When it expires, the acknowledge from the device will be considered as missing and the command eventually sent again. [LvFtrType\\_Command](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_LvGevCCRC** Controls the number of GigE Vision control channel command retransmissions allowed when a control channel command times out. [LvFtrType\\_Command](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_LvCCStatus** Control channel status. [LvFtrType\\_Integer](#). Device GenTL feature ([LvFtrGroup\\_DeviceGtl](#)).
- LvDevice\_LvDeviceDisplayName** User readable name of the device. [LvFtrType\\_String](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvDevicesAcquiring** Returns true if the acquisition was started. Note that this feature does not tell whether the images are actually delivered to the output buffer queue; it simply informs that your application is between the AcquisitionStart and AcquisitionStop actions. [LvFtrType\\_Boolean](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniProcessMode** The UniProcessing provides unified API for image preprocessing, which is done either on the device itself, if it is supported by the hardware, or by software, if not. The preprocessing includes Bayer decoding or pixel format conversion, application of LUT and Color Correction. [LvFtrType\\_Enumeration](#). Values: [LvUniProcessMode](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniProcessEnableInPlace** If possible, the software image preprocessing will be preferably done in the same image (not to another buffer). This is possible only in case the preprocessing does not change the pixel format, that means the [LvUniPixelFormat](#) must be equal to [PixelFormat](#) (for example the Bayer decoding is not done by software). [LvFtrType\\_Boolean](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniPixelFormat** If the image preprocessing is enabled, this is the desired pixel format, to which the image is to be converted. Only monochrome and RGB/BGR color pixel formats are supported. The processing chain is set so that:
- if the [PixelFormat](#) is undecoded Bayer, the Bayer decoding to desired [LvUniPixelFormat](#) is included
  - otherwise if the [PixelFormat](#) is not equal to [LvUniPixelFormat](#), a pixel format conversion is included. [LvFtrType\\_Enumeration](#). Values: see [LvPixelFormat](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniProcessPayloadSize** Returns the size needed for the processing output buffer, which is to be used when the in-place processing is not enabled or not possible. Normally is this buffer allocated automatically for each acquisition buffer, but your application can also provide your own buffers and this feature gives the size of the buffers needed. [LvFtrType\\_Integer](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniLinePitch** The line increment of the process buffer, if the processing is active, or of the source buffer, if processing is not active. To access the image regardless whether it was processed to the process buffer or not, you need 5 independent values:

- Pointer to the data - use `LvUniBase` feature of the Buffer, which points either to the acquired image (if no processing was done), or to the processed image (if it was processed).
- Width and height - these are the same for the acquired and processed image, so use the Width and Height from the remote device, or `ChunkWidth` and `ChunkHeight` if these can change during the acquisition.
- Pixel format - use `LvUniPixelFormat` - if this is different from the `PixelFormat` then processing is done, so `LvUniPixelFormat` is always correct.
- Line pitch - use `LvUniLinePitch`, which returns proper line pitch of the buffer, to which the `LvUniBase` pointer points. Note that the above is valid only in case the processing can be successfully done (for example the source image is not in unsupported `PixelFormat`) and is not disabled (for example by `LvUniProcessExecution=OnExplicitRequest`). `LvFtrType_Integer`. Device local feature (`LvFtrGroup↔_DeviceLocal`).

**`LvDevice_LvUniBayerDecoderAlgorithm`** Selects the Bayer array decoding method for the software processing. This does not apply to the hardware Bayer decoding on the device, which is usually fixed to one method. `LvFtrType_Enumeration`. Values: `LvBayerDecoderAlgorithm`. Device local feature (`LvFtrGroup↔_DeviceLocal`).

**`LvDevice_LvUniBrightness`** Brightness of the image. It is realized by the LUT. Values under 1.0 means darker than original, above 1.0 lighter than the original. The `LvUniLUTMode` must be Generated, in order to enable this feature. `LvFtrType_Float`. Device local feature (`LvFtrGroup_DeviceLocal`).

**`LvDevice_LvUniContrast`** Contrast of the image. It is realized by the LUT. Values under 1.0 means lower contrast than original, above 1.0 higher contrast than the original. The `LvUniLUTMode` must be Generated, in order to enable this feature. `LvFtrType_Float`. Device local feature (`LvFtrGroup_DeviceLocal`).

**`LvDevice_LvUniGamma`** Gamma correction of the image. It is realized by the LUT. Values under 1.0 make the middle tones darker, above 1.0 lighter. The `LvUniLUTMode` must be Generated, in order to enable this feature. `LvFtrType_Float`. Device local feature (`LvFtrGroup_DeviceLocal`).

**`LvDevice_LvUniBalanceRatioSelector`** Selects which color channel will be accessed by the `LvUniBalance↔Ratio` feature. The `LvUniLUTMode` must be Generated, in order to enable this feature. `LvFtrType_↔Enumeration`. Values: `LvUniBalanceRatioSelector`. Device local feature (`LvFtrGroup_DeviceLocal`).

**`LvDevice_LvUniBalanceRatio`** The white balance factor to be applied on the selected color channel. The selected color channel of all pixels will be multiplied by this value (not directly, but through the precalculated LUT). If the value is < 1.0, the saturated pixels will become gray (white is no more white). Thus it is better if all 3 factors are greater than or equal to 1.0. `LvFtrType_Float`. Device local feature (`LvFtrGroup_↔DeviceLocal`).

**`LvDevice_LvUniBalanceWhiteAuto`** Selects the action for automatic white balance calculation. Currently only the option `Once` is available. Setting this option causes the following:

- if there is already acquired image available, the white balance factors are calculated from this image and LUT is updated to reflect the changes
- if there is no image acquired yet, an internal flag is set and the calculation is done when the image is acquired. Note that the enumeration is self-clearing, that means its value is automatically changed to `Off`, when the white balance calculation is finished. The newly calculated white balance is applied to newly acquired images, not to the existing ones, unless you explicitly call the `ExecProcess` command for the already acquired buffers. At the time of calculation the camera should look at a neutral grey (not white) object, which should fill the whole image area. Making white balance from normal image can bring less satisfactory results. `LvFtrType_Enumeration`. Values: `LvUniBalance↔WhiteAuto`. Device local feature (`LvFtrGroup_DeviceLocal`).

**`LvDevice_LvUniBalanceWhiteReset`** Sets all the white balance factors (`LvUniBalanceRatio`) to 1. The advantage of this feature in comparison with setting the 3 factors to 1 is that the LUT is updated only once, so it is faster. `LvFtrType_Command`. Device local feature (`LvFtrGroup_DeviceLocal`).

**`LvDevice_LvUniColorTransformationSelector`** Selects which color transformation module is controlled by the color transformation features. It also gives particular meaning to individual color transformation gains. `LvFtrType_Enumeration`. Values: `LvUniColorTransformationSelector`. Device local feature (`LvFtrGroup_↔DeviceLocal`).



- LvDevice\_LvUniColorTransformationEnable** Enables the Color Transformation in the processing. When disabled, the Color Transformation matrix does not lose its values; when enabling it, the original values are retained. [LvFtrType\\_Boolean](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniColorTransformationValueSelector** Selects the cell of the Color Transformation matrix to be accessed by [LvUniColorTransformationValue](#). [LvFtrType\\_Enumeration](#). Values: [LvColorTransformationValueSelector](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniColorTransformationValue** The value of the selected cell of the Color Transformation matrix. [LvFtrType\\_Float](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniSaturation** Sets the Color Correction matrix according to specified saturation. The saturation set to 0 causes a conversion to greyscale, 1.0 leaves the image identical, 2.0 emphasizes the colors. [LvFtrType\\_Float](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniProcessExecution** Defines the point, when the software image processing of the buffer is done. You may need to define this point in case you do not need all the images to be processed. Note that this applies only to the software processing; the hardware processing is done on the remote device always. [LvFtrType\\_Enumeration](#). Values: [LvUniProcessExecution](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniLUTMode** Selects the LUT control mode. The mode determines, if the LUT can be directly modified by the application, or if the LUT is to be reserved for implementation of white balance, gamma, brightness and contrast - in such case the LUT is filled with precalculated values by SynView library and cannot be directly modified. [LvFtrType\\_Enumeration](#). Values: [LvUniLUTMode](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniLUTSelector** This selector selects for which LUT is applied [LvUniLUTIndex](#)/[LvUniLUTValue](#). In case of monochrome image the LUT has only one array = Luminance. In case of color images, the LUT consists of 3 arrays, for Red, Green and Blue. [LvFtrType\\_Enumeration](#). Values: [LvUniLUTSelector](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniLUTEnable** Enables the LUT in the processing. When disabled, the LUT does not lose its values, the disabled LUT is substituted by a linear LUT, and when enabling the LUT, the original values are retained. [LvFtrType\\_Boolean](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniLUTIndex** Index of the element to be accessed in the selected LUT via the [LvUniLUTValue](#) feature. Note that accessing the whole LUT by this approach can be very time consuming, namely on GigE cameras. If possible, it is better to use the [LvUniLUTValueAll](#) or SynView dedicated LUT functions. [LvFtrType\\_Integer](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniLUTValue** Value of the element for the current [LvUniLUTIndex](#) in the selected LUT. Note that accessing the whole LUT by this approach can be very time consuming, namely on GigE cameras. If possible, it is better to use the [LvUniLUTValueAll](#) or SynView dedicated LUT functions. [LvFtrType\\_Integer](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniLUTValueAll** This feature enables to get/set the entire content of the selected LUT in one block access. Beware that the LUT buffer structure is vendor and model dependent, so take care if your application is expected to work with various types of devices or devices from various vendors. [LvFtrType\\_Buffer](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvUniColorTransformationMode** Selects the Color Transformation matrix control mode. The mode determines, if the matrix can be directly modified by the application, or if the matrix is to be reserved for implementation of the Saturation or other higher level features - in such case the matrix is filled with precalculated values by SynView library and cannot be directly modified. [LvFtrType\\_Enumeration](#). Values: [LvUniColorTransformationMode](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_LvDeviceExpiringDate** expiring date of the device. [LvFtrType\\_String](#). Device local feature ([LvFtrGroup\\_DeviceLocal](#)).
- LvDevice\_Info** Constant to be used in [LvGetInfo\(\)](#) and [LvGetInfoStr\(\)](#) to obtain various info about the device.

### 5.29.2.3 enum LvEventFtr

LvEventFtr constants.

## Enumerator

- LvEvent\_EventType** Represents the GenTL EVENT\_EVENT\_TYPE info - The event type. LvFtrType\_↔ Integer.
- LvEvent\_NumInQueue** Represents the GenTL EVENT\_NUM\_IN\_QUEUE info - Number of events in the event data queue. LvFtrType\_Integer.
- LvEvent\_NumFired** Represents the GenTL EVENT\_NUM\_FIRED info - Number of events, that were fired since the creation of the event. LvFtrType\_Integer

## 5.29.2.4 enum LvInterfaceFtr

LvInterfaceFtr constants.

## Enumerator

- LvInterface\_InterfaceID** Interface ID. LvFtrType\_String. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_InterfaceType** Interface type. LvFtrType\_Enumeration. Values: LvInterfaceType. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_GevInterfaceGatewaySelector** Interface gateway selector. LvFtrType\_Integer. Interface Gen↔ TL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_GevInterfaceGateway** Interface gateway. LvFtrType\_Integer. Depends on LvInterface\_Gev↔ InterfaceGatewaySelector. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_GevMACAddress** Interface MAC address. LvFtrType\_Integer. Interface GenTL feature (LvFtr↔ Group\_InterfaceGtl).
- LvInterface\_GevInterfaceSubnetSelector** Interface subnet selector. LvFtrType\_Integer. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_GevInterfaceSubnetIPAddress** Interface subnet IP address. LvFtrType\_Integer. Depends on LvInterface\_GevInterfaceSubnetSelector. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_GevInterfaceSubnetMask** Interface subnet mask. LvFtrType\_Integer. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_DeviceUpdateList** Update internal list of devices. LvFtrType\_Command. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_DeviceSelector** Device selector. LvFtrType\_Integer. Interface GenTL feature (LvFtrGroup\_↔ InterfaceGtl).
- LvInterface\_DeviceID** Device ID. LvFtrType\_String. Depends on LvInterface\_DeviceSelector. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_DeviceVendorName** Device vendor name. LvFtrType\_String. Depends on LvInterface\_↔ DeviceSelector. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_DeviceModelName** Device Model name. LvFtrType\_String. Depends on LvInterface\_Device↔ Selector. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_DeviceAccessStatus** Device access status. LvFtrType\_Enumeration. Values: LvDevice↔ AccessStatus. Depends on LvInterface\_DeviceSelector. Interface GenTL feature (LvFtrGroup\_↔ InterfaceGtl).
- LvInterface\_GevDeviceIPAddress** Device IP address. LvFtrType\_Integer. Depends on LvInterface\_↔ DeviceSelector. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_GevDeviceSubnetMask** Device subnet mask. LvFtrType\_Integer. Depends on LvInterface\_↔ DeviceSelector. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_GevDeviceMACAddress** Device MAC address. LvFtrType\_Integer. Depends on LvInterface\_↔ DeviceSelector. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_LvDeviceUserID** Device User ID. LvFtrType\_String. Depends on LvInterface\_DeviceSelector. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).
- LvInterface\_LvDeviceSerialNumber** Device identifier (serial number). LvFtrType\_String. Depends on Lv↔ Interface\_DeviceSelector. Interface GenTL feature (LvFtrGroup\_InterfaceGtl).

**LvInterface\_LvInterfaceDisplayName** User readable name of the interface. [LvFtrType\\_String](#). Interface local feature ([LvFtrGroup\\_InterfaceLocal](#)).

**LvInterface\_Info** Constant to be used in [LvGetInfo\(\)](#) and [LvGetInfoStr\(\)](#) to obtain various info about the interface.

### 5.29.2.5 enum LvRendererFtr

LvRendererFtr constants.

#### Enumerator

**LvRenderer\_LvAutoDisplay** If set, the image is automatically displayed before it is passed to the supplied callback. This is functional only in case the Event thread is started. [LvFtrType\\_Boolean](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvRenderType** Controls way how the acquired images are rendered on the screen. Note that all the Scale- options require scaling capability of the display and might not be supported in all operating systems. [LvFtrType\\_Enumeration](#). Values: [LvRenderType](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvOffsetX** Sets the horizontal offset of the image to be rendered, i.e. the distance from the left edge of the display window. [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvOffsetY** Sets the vertical offset of the image to be rendered, i.e. the distance from the top edge of the display window. [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvWidth** Sets the width of the rectangle to which the image is to be rendered. Note that if the [LvIgnoreAspectRatio](#) feature is False, the real image width can be smaller, in order to keep the aspect ratio. [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvHeight** Sets the height of the rectangle to which the image is to be rendered. Note that if the [LvIgnoreAspectRatio](#) feature is False, the real image height can be smaller, in order to keep the aspect ratio. [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvIgnoreAspectRatio** Allows to ignore the original aspect ratio while rendering the image, so the image can be scaled up/down in one dimension with different factor than in the other dimension. [LvFtrType\\_Boolean](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvDisableScaleUp** Disables scaling the image up. [LvFtrType\\_Boolean](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvDisableScaleDown** Disables scaling the image down. [LvFtrType\\_Boolean](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvCenterImage** If the image is smaller than required window client size or the specified rectangle, the image is placed to the center. [LvFtrType\\_Boolean](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvNumberOfTiles** Sets the number of tiles used for image rendering. Note that for the tile repaint is needed that the corresponding buffers are still in the application ownership; once the buffer is placed to the input buffer pool, it should not be accessed for paint anymore (see also [LvPostponeQueueBuffers](#)). [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvColumns** Sets the number of columns used for image rendering. When the value is 0, the number of columns is calculated automatically. [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvRows** Sets the number of rows used for image rendering. When the value is 0, the number of rows is calculated automatically. [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvTileGap** Sets the gap between the tiles in pixels. [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

**LvRenderer\_LvAutoTileCalculation** When set to True, the tile sizes and positions are calculated automatically. When the [LvColumns](#) and/or [LvRows](#) are 0, also the number of columns and/or rows is calculated automatically. [LvFtrType\\_Boolean](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).

- LvRenderer\_LvShowPixel** Activate the Pixel Coordinate section for getting pixel values at given coordinates. [LvFtrType\\_Boolean](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).
- LvRenderer\_LvPixelCoordinateX** Sets the horizontal coordinate of the pixel for which the value should be retrieved. [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).
- LvRenderer\_LvPixelCoordinateY** Sets the vertical coordinate of the pixel for which the value should be retrieved. [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).
- LvRenderer\_LvPixelVal1** First Value of the retrieved Pixel. [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).
- LvRenderer\_LvPixelVal2** Second Value of the retrieved Pixel. [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).
- LvRenderer\_LvPixelVal3** Third Value of the retrieved Pixel. [LvFtrType\\_Integer](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).
- LvRenderer\_LvShowHistogram** Activate a histogram window. [LvFtrType\\_Boolean](#). Renderer local feature ([LvFtrGroup\\_RendererLocal](#)).
- LvRenderer\_Info** Constant to be used in [LvGetInfo\(\)](#) and [LvGetInfoStr\(\)](#) to obtain various info about the device.

#### 5.29.2.6 enum LvStreamFtr

LvStreamFtr constants.

Enumerator

- LvStream\_StreamID** Stream ID. [LvFtrType\\_String](#). Stream GenTL feature ([LvFtrGroup\\_StreamGtl](#)).
- LvStream\_StreamAnnouncedBufferCount** Number of buffers announced for the data stream. [LvFtrType\\_Integer](#). Stream GenTL feature ([LvFtrGroup\\_StreamGtl](#)).
- LvStream\_StreamAcquisitionModeSelector** Selects desired acquisition mode. [LvFtrType\\_Enumeration](#). Values: [LvStreamAcquisitionModeSelector](#). Stream GenTL feature ([LvFtrGroup\\_StreamGtl](#)).
- LvStream\_StreamAnnounceBufferMinimum** Minimum number of buffers to be announced for selected acquisition mode. [LvFtrType\\_Integer](#). Stream GenTL feature ([LvFtrGroup\\_StreamGtl](#)).
- LvStream\_StreamType** Stream type. [LvFtrType\\_Enumeration](#). Values: [LvStreamType](#). Stream GenTL feature ([LvFtrGroup\\_StreamGtl](#)).
- LvStream\_LvStreamDisplayName** Returns the display name of the stream. [LvFtrType\\_String](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).
- LvStream\_LvCalcPayloadSize** Returns the payload size (size of buffer to hold the image data). If the payload size is not provided by the stream or device, it is calculated, so this feature returns always a valid value. [LvFtrType\\_Integer](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).
- LvStream\_LvPostponeQueueBuffers** Number of buffers to be kept postponed before returning to the input buffer pool. [LvFtrType\\_Integer](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).
- LvStream\_LvAwaitDeliveryLimit** Limit for images in the output buffer. Applicable only if the event thread is running - then if there is more than this number of buffers in the output queue, the oldest buffers are discarded and returned to input buffer pool. This is useful in case the application is not able to process all the images in time. [LvFtrType\\_Integer](#). If the value is 0, this limit is inactive. Stream local feature ([LvFtrGroup\\_StreamLocal](#)).
- LvStream\_LvAutoAllocateProcessBuffers** Enable the auto allocation of process buffers. The process buffers are allocated only if they are needed for the image processing or conversion. You can disable the automatic buffer allocation and provide own buffers, using the [LvBufferAttachProcessBuffer\(\)](#) function. [LvFtrType\\_Boolean](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).
- LvStream\_LvPreallocateProcessBuffers** Preallocates all the process buffers, even if it is not yet sure if they will be needed. With this command you can avoid time delays when allocating the buffers during the acquisition. [LvFtrType\\_Command](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).

**LvStream\_LvNumDelivered** Number of acquired frames since last acquisition start. It is equivalent to the GenTL STREAM\_INFO\_NUM\_DELIVERED info. [LvFtrType\\_Integer](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).

**LvStream\_LvNumUnderrun** Number of lost frames due to input buffer pool underrun since stream open. It is equivalent to the GenTL STREAM\_INFO\_NUM\_UNDERRUN info. [LvFtrType\\_Integer](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).

**LvStream\_LvNumAnnounced** Number of announced buffers. It is equivalent to the GenTL STREAM\_INFO\_NUM\_ANNOUNCED info. [LvFtrType\\_Integer](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).

**LvStream\_LvNumQueued** Number of buffers currently in the input pool. It is equivalent to the GenTL STREAM\_INFO\_NUM\_QUEUED info. [LvFtrType\\_Integer](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).

**LvStream\_LvNumAwaitDelivery** Number of buffers currently in the output queue. It is equivalent to the GenTL STREAM\_INFO\_NUM\_AWAIT\_DELIVERY info. [LvFtrType\\_Integer](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).

**LvStream\_LvIsGrabbing** Flag indicating whether the acquisition engine is started or not. This is independent from the acquisition status of the remote device. It is equivalent to the GenTL STREAM\_INFO\_IS\_GRABBING info. [LvFtrType\\_Boolean](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).

**LvStream\_LvNumAborted** Number of aborted frames since last acquisition start. [LvFtrType\\_Integer](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).

**LvStream\_LvNumStarted** Number of started frames since stream open. It is equivalent to the GenTL STREAM\_INFO\_NUM\_STARTED info. [LvFtrType\\_Integer](#). Stream local feature ([LvFtrGroup\\_StreamLocal](#)).

**LvStream\_Info** Constant to be used in [LvGetInfo\(\)](#) and [LvGetInfoStr\(\)](#) to obtain various info about the stream.

#### 5.29.2.7 enum LvSystemFtr

LvSystemFtr constants.

Enumerator

**LvSystem\_TLVendorName** GenTL producer vendor name. [LvFtrType\\_String](#). System GenTL feature ([LvFtrGroup\\_SystemGtl](#)).

**LvSystem\_TLModelName** GenTL producer model name. [LvFtrType\\_String](#). System GenTL feature ([LvFtrGroup\\_SystemGtl](#)).

**LvSystem\_TLID** GenTL producer ID. [LvFtrType\\_String](#). System GenTL feature ([LvFtrGroup\\_SystemGtl](#)).

**LvSystem\_TLVersion** GenTL producer version. [LvFtrType\\_String](#). System GenTL feature ([LvFtrGroup\\_SystemGtl](#)).

**LvSystem\_TLPath** GenTL producer path. [LvFtrType\\_String](#). System GenTL feature ([LvFtrGroup\\_SystemGtl](#)).

**LvSystem\_TLType** GenTL producer type. [LvFtrType\\_Enumeration](#). Values: [LvTLType](#). System GenTL feature ([LvFtrGroup\\_SystemGtl](#)).

**LvSystem\_GenTLVersionMajor** GenTL version major. [LvFtrType\\_Integer](#). System GenTL feature ([LvFtrGroup\\_SystemGtl](#)).

**LvSystem\_GenTLVersionMinor** GenTL version minor. [LvFtrType\\_Integer](#). System GenTL feature ([LvFtrGroup\\_SystemGtl](#)).

**LvSystem\_GevVersionMajor** GigE Vision version major. [LvFtrType\\_Integer](#). System GenTL feature ([LvFtrGroup\\_SystemGtl](#)).

**LvSystem\_GevVersionMinor** GigE Vision version minor. [LvFtrType\\_Integer](#). System GenTL feature ([LvFtrGroup\\_SystemGtl](#)).

**LvSystem\_InterfaceUpdateList** Update internal list of interfaces. [LvFtrType\\_Command](#). System GenTL feature ([LvFtrGroup\\_SystemGtl](#)).

- LvSystem\_InterfaceSelector** Interface selector. [LvFtrType\\_Integer](#). System GenTL feature ([LvFtrGroup\\_↔SystemGtl](#)).
- LvSystem\_InterfaceID** Interface ID. [LvFtrType\\_String](#). Depends on [LvSystem\\_InterfaceSelector](#). System GenTL feature ([LvFtrGroup\\_SystemGtl](#)).
- LvSystem\_GevInterfaceMACAddress** Interface MAC address. [LvFtrType\\_Integer](#). Depends on [LvSystem\\_↔InterfaceSelector](#). System GenTL feature ([LvFtrGroup\\_InterfaceGtl](#)).
- LvSystem\_GevInterfaceDefaultIPAddress** Interface default IP address. [LvFtrType\\_Integer](#). Depends on [LvSystem\\_InterfaceSelector](#). System GenTL feature ([LvFtrGroup\\_InterfaceGtl](#)).
- LvSystem\_GevInterfaceDefaultSubnetMask** Interface default subnet mask. [LvFtrType\\_Integer](#). Depends on [LvSystem\\_InterfaceSelector](#). System GenTL feature ([LvFtrGroup\\_InterfaceGtl](#)).
- LvSystem\_GevInterfaceDefaultGateway** Interface default gateway. [LvFtrType\\_Integer](#). Depends on [Lv\\_↔System\\_InterfaceSelector](#). System GenTL feature ([LvFtrGroup\\_InterfaceGtl](#)).
- LvSystem\_LvSystemDisplayName** User readable name of the system. [LvFtrType\\_String](#). System local feature ([LvFtrGroup\\_SystemLocal](#)).
- LvSystem\_Info** Constant to be used in [LvGetInfo\(\)](#) and [LvGetInfoStr\(\)](#) to obtain various info about the system.



## 5.30 Enumeration entries

### Enumerations

- enum `LvPixelFormat` {  
`LvPixelFormat_Mono8` = 0x01080001, `LvPixelFormat_Mono8S` = 0x01080002, `LvPixelFormat_Mono10` = 0x01100003, `LvPixelFormat_Mono10Packed` = 0x010C0004,  
`LvPixelFormat_Mono12` = 0x01100005, `LvPixelFormat_Mono12Packed` = 0x010C0006, `LvPixelFormat_Mono14` = 0x01100025, `LvPixelFormat_Mono16` = 0x01100007,  
`LvPixelFormat_BayerGR8` = 0x01080008, `LvPixelFormat_BayerRG8` = 0x01080009, `LvPixelFormat_BayerGB8` = 0x0108000A, `LvPixelFormat_BayerBG8` = 0x0108000B,  
`LvPixelFormat_BayerGR10` = 0x0110000C, `LvPixelFormat_BayerRG10` = 0x0110000D, `LvPixelFormat_BayerGB10` = 0x0110000E, `LvPixelFormat_BayerBG10` = 0x0110000F,  
`LvPixelFormat_BayerGR12` = 0x01100010, `LvPixelFormat_BayerRG12` = 0x01100011, `LvPixelFormat_BayerGB12` = 0x01100012, `LvPixelFormat_BayerBG12` = 0x01100013,  
`LvPixelFormat_BayerGR10Packed` = 0x010C0026, `LvPixelFormat_BayerRG10Packed` = 0x010C0027, `LvPixelFormat_BayerGB10Packed` = 0x010C0028, `LvPixelFormat_BayerBG10Packed` = 0x010C0029,  
`LvPixelFormat_BayerGR12Packed` = 0x010C002A, `LvPixelFormat_BayerRG12Packed` = 0x010C002B, `LvPixelFormat_BayerGB12Packed` = 0x010C002C, `LvPixelFormat_BayerBG12Packed` = 0x010C002D,  
`LvPixelFormat_BayerGR16` = 0x0110002E, `LvPixelFormat_BayerRG16` = 0x0110002F, `LvPixelFormat_BayerGB16` = 0x01100030, `LvPixelFormat_BayerBG16` = 0x01100031,  
`LvPixelFormat_RGB8` = 0x02180014, `LvPixelFormat_BGR8` = 0x02180015, `LvPixelFormat_RGBA8` = 0x02200016, `LvPixelFormat_BGRA8` = 0x02200017,  
`LvPixelFormat_RGB10` = 0x02300018, `LvPixelFormat_BGR10` = 0x02300019, `LvPixelFormat_RGB12` = 0x0230001A, `LvPixelFormat_BGR12` = 0x0230001B,  
`LvPixelFormat_RGB16` = 0x02300033, `LvPixelFormat_RGB10V1Packed` = 0x0220001C, `LvPixelFormat_RGB10P32` = 0x0220001D, `LvPixelFormat_RGB12V1Packed` = 0x02240034,  
`LvPixelFormat_RGB565P` = 0x02100035, `LvPixelFormat_BGR565P` = 0x02100036, `LvPixelFormat_YUV411_8` = 0x020C001E, `LvPixelFormat_YUV422_8_UYVY` = 0x0210001F,  
`LvPixelFormat_YUV8` = 0x02180020, `LvPixelFormat_YUV422_8` = 0x02100032, `LvPixelFormat_YCbCr422_8` = 0x0210003B, `LvPixelFormat_YCbCr601_422_8` = 0x0210003E,  
`LvPixelFormat_YCbCr601_422_8_CbYCrY` = 0x02100044, `LvPixelFormat_RGB8_Planar` = 0x02180021, `LvPixelFormat_RGB10_Planar` = 0x02300022, `LvPixelFormat_RGB12_Planar` = 0x02300023,  
`LvPixelFormat_RGB16_Planar` = 0x02300024, `LvPixelFormat_BGR555P` = 0x021000E1 }
- enum `LvDeviceAccess` { `LvDeviceAccess_None` = 1, `LvDeviceAccess_ReadOnly` = 2, `LvDeviceAccess_Control` = 3, `LvDeviceAccess_Exclusive` = 4 }
- enum `LvDeviceAccessStatus` { `LvDeviceAccessStatus_Unknown` = 0, `LvDeviceAccessStatus_ReadWrite` = 1, `LvDeviceAccessStatus_ReadOnly` = 2, `LvDeviceAccessStatus_NoAccess` = 3 }
- enum `LvDeviceScanType` { `LvDeviceScanType_Areascan`, `LvDeviceScanType_Linescan` }
- enum `LvDeviceClockSelector` { `LvDeviceClockSelector_SensorDigitization` }
- enum `LvDeviceTemperatureSelector` { `LvDeviceTemperatureSelector_Sensor`, `LvDeviceTemperatureSelector_Mainboard` }
- enum `LvAOIMode` { `LvAOIMode_Automatic`, `LvAOIMode_ClipOnTransfer`, `LvAOIMode_Manual` }
- enum `LvAcquisitionMode` { `LvAcquisitionMode_SingleFrame`, `LvAcquisitionMode_MultiFrame`, `LvAcquisitionMode_Continuous` }
- enum `LvExposureAuto` { `LvExposureAuto_Off`, `LvExposureAuto_Once`, `LvExposureAuto_Continuous` }
- enum `LvTriggerSelector` { `LvTriggerSelector_FrameStart`, `LvTriggerSelector_FrameBurstStart`, `LvTriggerSelector_LineStart`, `LvTriggerSelector_VirtualFrameActive` }
- enum `LvTriggerMode` { `LvTriggerMode_Off`, `LvTriggerMode_On` }
- enum `LvTriggerSource` {

- ```

LvTriggerSource_Line1, LvTriggerSource_Line2, LvTriggerSource_Line3, LvTriggerSource_Line4,
LvTriggerSource_Line5, LvTriggerSource_Line6, LvTriggerSource_Line7, LvTriggerSource_Line8,
LvTriggerSource_Line17, LvTriggerSource_Line18, LvTriggerSource_Line19, LvTriggerSource_Line20,
LvTriggerSource_Line21, LvTriggerSource_Line22, LvTriggerSource_Line23, LvTriggerSource_Line24,
LvTriggerSource_Software, LvTriggerSource_Action1, LvTriggerSource_Action2, LvTriggerSource_Action3,
LvTriggerSource_Action4, LvTriggerSource_Action5, LvTriggerSource_Action6, LvTriggerSource_Action7,
LvTriggerSource_Action8, LvTriggerSource_Quad, LvTriggerSource_Counter1, LvTriggerSource_Counter2,
LvTriggerSource_Counter3, LvTriggerSource_Counter4, LvTriggerSource_Timer1, LvTriggerSource_Timer2,
LvTriggerSource_Timer3, LvTriggerSource_Timer4, LvTriggerSource_UserOutput1, LvTriggerSource_User↵
Output2,
LvTriggerSource_UserOutput3, LvTriggerSource_UserOutput4, LvTriggerSource_UserOutput5, LvTrigger↵
Source_UserOutput6,
LvTriggerSource_UserOutput7, LvTriggerSource_UserOutput8 }

```
- enum LvTriggerActivation {
 

```

LvTriggerActivation_RisingEdge, LvTriggerActivation_FallingEdge, LvTriggerActivation_AnyEdge, Lv↵
TriggerActivation_LevelHigh,
LvTriggerActivation_LevelLow }

```
  - enum LvTriggerCaching { LvTriggerCaching\_Cache, LvTriggerCaching\_Drop }
  - enum LvExposureMode { LvExposureMode\_Timed }
  - enum LvAcquisitionFrameRateControlMode { LvAcquisitionFrameRateControlMode\_Off, LvAcquisition↵
FrameRateControlMode\_On }
  - enum LvLineSelector {
 

```

LvLineSelector_Line1, LvLineSelector_Line2, LvLineSelector_Line3, LvLineSelector_Line4,
LvLineSelector_Line5, LvLineSelector_Line6, LvLineSelector_Line7, LvLineSelector_Line8,
LvLineSelector_Line9, LvLineSelector_Line10, LvLineSelector_Line11, LvLineSelector_Line12,
LvLineSelector_Line13, LvLineSelector_Line14, LvLineSelector_Line15, LvLineSelector_Line16,
LvLineSelector_Line17, LvLineSelector_Line18, LvLineSelector_Line19, LvLineSelector_Line20,
LvLineSelector_Line21, LvLineSelector_Line22, LvLineSelector_Line23, LvLineSelector_Line24,
LvLineSelector_Line25, LvLineSelector_Line26, LvLineSelector_Line27, LvLineSelector_Line28,
LvLineSelector_Line29, LvLineSelector_Line30, LvLineSelector_Line31, LvLineSelector_Line32 }

```
  - enum LvLineMode { LvLineMode\_Input, LvLineMode\_Output }
  - enum LvLineFormat {
 

```

LvLineFormat_NoConnect, LvLineFormat_TriState, LvLineFormat_TTL, LvLineFormat_LVDS,
LvLineFormat_RS422, LvLineFormat_OptoCoupled }

```
  - enum LvLineSource {
 

```

LvLineSource_Off, LvLineSource_ExposureActive, LvLineSource_Timer1Active, LvLineSource_Timer2↵
Active,
LvLineSource_Timer3Active, LvLineSource_Timer4Active, LvLineSource_UserOutput1, LvLineSource_↵
UserOutput2,
LvLineSource_UserOutput3, LvLineSource_UserOutput4, LvLineSource_UserOutput5, LvLineSource_↵
UserOutput6,
LvLineSource_UserOutput7, LvLineSource_UserOutput8, LvLineSource_Counter1Active, LvLineSource_↵
Counter2Active,
LvLineSource_Counter3Active, LvLineSource_Counter4Active }

```
  - enum LvLineDebounceMode { LvLineDebounceMode\_Debounce, LvLineDebounceMode\_Deglintch }
  - enum LvCounterSelector { LvCounterSelector\_Counter1, LvCounterSelector\_Counter2, LvCounterSelector\_↵
\_Counter3, LvCounterSelector\_Counter4 }
  - enum LvCounterMode { LvCounterMode\_Autoreset }
  - enum LvCounterEventSource {
 

```

LvCounterEventSource_Off, LvCounterEventSource_FrameTrigger, LvCounterEventSource_TimerTick, Lv↵
CounterEventSource_Line1,
LvCounterEventSource_Line2, LvCounterEventSource_Line3, LvCounterEventSource_Line4, LvCounter↵
EventSource_Line17,
LvCounterEventSource_Line18 }

```
  - enum LvTimerSelector { LvTimerSelector\_Timer1, LvTimerSelector\_Timer2, LvTimerSelector\_Timer3, Lv↵
TimerSelector\_Timer4 }



- enum LvTimerTriggerSource {  
LvTimerTriggerSource\_Off, LvTimerTriggerSource\_FrameTrigger, LvTimerTriggerSource\_Counter1End, LvTimerTriggerSource\_Counter2End,  
LvTimerTriggerSource\_Counter3End, LvTimerTriggerSource\_Counter4End, LvTimerTriggerSource\_UserOutput1, LvTimerTriggerSource\_UserOutput2,  
LvTimerTriggerSource\_UserOutput3, LvTimerTriggerSource\_UserOutput4, LvTimerTriggerSource\_UserOutput5, LvTimerTriggerSource\_UserOutput6,  
LvTimerTriggerSource\_UserOutput7, LvTimerTriggerSource\_UserOutput8 }
- enum LvSpecialPurposeTriggerSelector { LvSpecialPurposeTriggerSelector\_ImageStampsReset }
- enum LvSpecialPurposeTriggerSource {  
LvSpecialPurposeTriggerSource\_Off, LvSpecialPurposeTriggerSource\_Line1, LvSpecialPurposeTriggerSource\_Line2, LvSpecialPurposeTriggerSource\_Line3,  
LvSpecialPurposeTriggerSource\_Line4, LvSpecialPurposeTriggerSource\_Line5, LvSpecialPurposeTriggerSource\_Line6, LvSpecialPurposeTriggerSource\_Line7,  
LvSpecialPurposeTriggerSource\_Line8, LvSpecialPurposeTriggerSource\_Line17, LvSpecialPurposeTriggerSource\_Line18, LvSpecialPurposeTriggerSource\_Line19,  
LvSpecialPurposeTriggerSource\_Line20, LvSpecialPurposeTriggerSource\_Line21, LvSpecialPurposeTriggerSource\_Line22, LvSpecialPurposeTriggerSource\_Line23,  
LvSpecialPurposeTriggerSource\_Line24, LvSpecialPurposeTriggerSource\_Action1, LvSpecialPurposeTriggerSource\_Action2, LvSpecialPurposeTriggerSource\_Action3,  
LvSpecialPurposeTriggerSource\_Action4, LvSpecialPurposeTriggerSource\_Action5, LvSpecialPurposeTriggerSource\_Action6, LvSpecialPurposeTriggerSource\_Action7,  
LvSpecialPurposeTriggerSource\_Action8 }
- enum LvSpecialPurposeTriggerActivation { LvSpecialPurposeTriggerActivation\_RisingEdge, LvSpecialPurposeTriggerActivation\_FallingEdge }
- enum LvImageStampSelector { LvImageStampSelector\_Timestamp, LvImageStampSelector\_FrameID }
- enum LvBootSwitch { LvBootSwitch\_PureGEV, LvBootSwitch\_Legacy }
- enum LvGainSelector { LvGainSelector\_All, LvGainSelector\_AnalogAll, LvGainSelector\_DigitalAll }
- enum LvGainAuto { LvGainAuto\_Off, LvGainAuto\_Once, LvGainAuto\_Continuous }
- enum LvBlackLevelSelector { LvBlackLevelSelector\_All, LvBlackLevelSelector\_Tap1, LvBlackLevelSelector\_Tap2 }
- enum LvBlackLevelAuto { LvBlackLevelAuto\_Off, LvBlackLevelAuto\_Once, LvBlackLevelAuto\_Continuous }
- enum LvColorTransformationSelector { LvColorTransformationSelector\_RGBtoRGB, LvColorTransformationSelector\_RGBtoYUV }
- enum LvColorTransformationValueSelector {  
LvColorTransformationValueSelector\_Gain00, LvColorTransformationValueSelector\_Gain01, LvColorTransformationValueSelector\_Gain02, LvColorTransformationValueSelector\_Gain10,  
LvColorTransformationValueSelector\_Gain11, LvColorTransformationValueSelector\_Gain12, LvColorTransformationValueSelector\_Gain20, LvColorTransformationValueSelector\_Gain21,  
LvColorTransformationValueSelector\_Gain22, LvColorTransformationValueSelector\_Offset0, LvColorTransformationValueSelector\_Offset1, LvColorTransformationValueSelector\_Offset2 }
- enum LvExternalDeviceControlMode { LvExternalDeviceControlMode\_Custom }
- enum LvExternalADCSelector { LvExternalADCSelector\_ExternalADC1, LvExternalADCSelector\_ExternalADC2, LvExternalADCSelector\_ExternalADC3, LvExternalADCSelector\_ExternalADC4 }
- enum LvPowerSwitchCurrentAction {  
LvPowerSwitchCurrentAction\_Idle, LvPowerSwitchCurrentAction\_Pulse, LvPowerSwitchCurrentAction\_Calibrate, LvPowerSwitchCurrentAction\_AdjustPosition,  
LvPowerSwitchCurrentAction\_Drive }
- enum LvPowerSwitchSelector { LvPowerSwitchSelector\_PowerSwitch1, LvPowerSwitchSelector\_PowerSwitch2, LvPowerSwitchSelector\_PowerSwitch3, LvPowerSwitchSelector\_PowerSwitch4 }
- enum LvPowerSwitchDrive { LvPowerSwitchDrive\_Off, LvPowerSwitchDrive\_Plus, LvPowerSwitchDrive\_Minus }
- enum LvPowerSwitchDriveAll { LvPowerSwitchDriveAll\_Off, LvPowerSwitchDriveAll\_Plus, LvPowerSwitchDriveAll\_Minus }
- enum LvPowerSwitchBoundADC {  
LvPowerSwitchBoundADC\_None, LvPowerSwitchBoundADC\_ExternalADC1, LvPowerSwitchBoundADC\_ExternalADC2, LvPowerSwitchBoundADC\_ExternalADC3,  
LvPowerSwitchBoundADC\_ExternalADC4 }

- enum LvLensControlTargetApproach { LvLensControlTargetApproach\_Direct, LvLensControlTargetApproach\_FromPlus, LvLensControlTargetApproach\_FromMinus }
- enum LvLUTSelector { LvLUTSelector\_Luminance, LvLUTSelector\_Red, LvLUTSelector\_Green, LvLUTSelector\_Blue }
- enum LvGevDeviceModeCharacterSet { LvGevDeviceModeCharacterSet\_UTF8 }
- enum LvGevSupportedOptionSelector { LvGevSupportedOptionSelector\_IPConfigurationLLA, LvGevSupportedOptionSelector\_IPConfigurationDHCP, LvGevSupportedOptionSelector\_IPConfigurationPersistentIP, LvGevSupportedOptionSelector\_CommandsConcatenation, LvGevSupportedOptionSelector\_WriteMem, LvGevSupportedOptionSelector\_PacketResend, LvGevSupportedOptionSelector\_Event, LvGevSupportedOptionSelector\_EventData, LvGevSupportedOptionSelector\_PendingAck, LvGevSupportedOptionSelector\_Action, LvGevSupportedOptionSelector\_PrimaryApplicationSwitchover, LvGevSupportedOptionSelector\_ExtendedStatusCodes, LvGevSupportedOptionSelector\_DiscoveryAckDelayWritable, LvGevSupportedOptionSelector\_DiscoveryAckDelay, LvGevSupportedOptionSelector\_TestData, LvGevSupportedOptionSelector\_ManifestTable, LvGevSupportedOptionSelector\_CCPApplicationSocket, LvGevSupportedOptionSelector\_LinkSpeed, LvGevSupportedOptionSelector\_HeartbeatDisable, LvGevSupportedOptionSelector\_SerialNumber, LvGevSupportedOptionSelector\_UserDefinedName, LvGevSupportedOptionSelector\_StreamChannelSourceSocket, LvGevSupportedOptionSelector\_StreamChannel0ExtendedChunkData, LvGevSupportedOptionSelector\_StreamChannel0UnconditionalStreaming, LvGevSupportedOptionSelector\_StreamChannel0IPReassembly, LvGevSupportedOptionSelector\_StreamChannel0BigAndLittleEndian, LvGevSupportedOptionSelector\_MessageChannelSourceSocket }
- enum LvGevCCP { LvGevCCP\_OpenAccess, LvGevCCP\_ExclusiveAccess, LvGevCCP\_ControlAccess, LvGevCCP\_ControlAccessSwitchoverActive }
- enum LvUserSetSelector { LvUserSetSelector\_Default, LvUserSetSelector\_UserSet1, LvUserSetSelector\_UserSet2, LvUserSetSelector\_UserSet3, LvUserSetSelector\_UserSet4 }
- enum LvUserSetDefaultSelector { LvUserSetDefaultSelector\_Default, LvUserSetDefaultSelector\_UserSet1, LvUserSetDefaultSelector\_UserSet2, LvUserSetDefaultSelector\_UserSet3, LvUserSetDefaultSelector\_UserSet4, LvUserSetDefaultSelector\_None }
- enum LvUserSetDefault { LvUserSetDefault\_Default, LvUserSetDefault\_UserSet1, LvUserSetDefault\_UserSet2, LvUserSetDefault\_UserSet3, LvUserSetDefault\_UserSet4, LvUserSetDefault\_None }
- enum LvChunkSelector { LvChunkSelector\_OffsetX, LvChunkSelector\_OffsetY, LvChunkSelector\_Width, LvChunkSelector\_Height, LvChunkSelector\_PixelFormat, LvChunkSelector\_LinePitch, LvChunkSelector\_FrameID, LvChunkSelector\_Timestamp, LvChunkSelector\_ExposureTime, LvChunkSelector\_Gain, LvChunkSelector\_LineStatusAll, LvChunkSelector\_BlackLevel, LvChunkSelector\_LvExternalADCValue, LvChunkSelector\_LvSmartAppString, LvChunkSelector\_LvSmartAppInt, LvChunkSelector\_LvSmartAppUInt, LvChunkSelector\_LvSmartAppRegister, LvChunkSelector\_LvTriggerDelayed, LvChunkSelector\_LvStrobeDropped, LvChunkSelector\_LvFrameAbort, LvChunkSelector\_LvTriggerDropped, LvChunkSelector\_LvTriggerError, LvChunkSelector\_LvEncoderPosition, LvChunkSelector\_LvEncoderRotation, LvChunkSelector\_LvVirtFrmTriggerAbort, LvChunkSelector\_LvVirtFrmTriggerDrop, LvChunkSelector\_LvVirtFrmTriggerDelay, LvChunkSelector\_LvVirtFrmFirst, LvChunkSelector\_LvVirtFrmLength }
- enum LvChunkGainSelector { LvChunkGainSelector\_AnalogAll, LvChunkGainSelector\_DigitalAll }
- enum LvEventSelector { LvEventSelector\_LvLog, LvEventSelector\_LvSmartAppLog, LvEventSelector\_LvSmartAppString, LvEventSelector\_LvSmartAppInt, LvEventSelector\_LvSmartAppUInt, LvEventSelector\_LvSmartAppRegister, LvEventSelector\_LvTriggerDropped }
- enum LvEventNotification { LvEventNotification\_Off, LvEventNotification\_On }

- enum `LvTLType` { `LvTLType_Mixed`, `LvTLType_Custom`, `LvTLType_GEV` }
- enum `LvInterfaceType` { `LvInterfaceType_Custom`, `LvInterfaceType_GEV`, `LvInterfaceType_U3V` }
- enum `LvDeviceType` { `LvDeviceType_Custom`, `LvDeviceType_GEV`, `LvDeviceType_U3V` }
- enum `LvGevDeviceStreamCaptureMode` { `LvGevDeviceStreamCaptureMode_SystemDefault`, `LvGevDeviceStreamCaptureMode_Socket`, `LvGevDeviceStreamCaptureMode_FilterDriver` }
- enum `LvStreamAcquisitionModeSelector` { `LvStreamAcquisitionModeSelector_Default` }
- enum `LvStreamType` { `LvStreamType_Custom`, `LvStreamType_GEV`, `LvStreamType_U3V` }
- enum `LvUniProcessMode` { `LvUniProcessMode_HwOnly`, `LvUniProcessMode_SwOnly`, `LvUniProcessMode_Auto`, `LvUniProcessMode_Off` }
- enum `LvBayerDecoderAlgorithm` { `LvBayerDecoderAlgorithm_NearestNeighbour`, `LvBayerDecoderAlgorithm_BilinearInterpolation`, `LvBayerDecoderAlgorithm_BilinearColorCorrection`, `LvBayerDecoderAlgorithm_PixelGrouping`, `LvBayerDecoderAlgorithm_VariableGradient` }
- enum `LvUniBalanceRatioSelector` { `LvUniBalanceRatioSelector_Red`, `LvUniBalanceRatioSelector_Green`, `LvUniBalanceRatioSelector_Blue` }
- enum `LvUniBalanceWhiteAuto` { `LvUniBalanceWhiteAuto_Off`, `LvUniBalanceWhiteAuto_Once` }
- enum `LvUniColorTransformationSelector` { `LvUniColorTransformationSelector_RGBtoRGB` }
- enum `LvUniColorTransformationValueSelector` { `LvUniColorTransformationValueSelector_Gain00`, `LvUniColorTransformationValueSelector_Gain01`, `LvUniColorTransformationValueSelector_Gain02`, `LvUniColorTransformationValueSelector_Gain10`, `LvUniColorTransformationValueSelector_Gain11`, `LvUniColorTransformationValueSelector_Gain12`, `LvUniColorTransformationValueSelector_Gain20`, `LvUniColorTransformationValueSelector_Gain21`, `LvUniColorTransformationValueSelector_Gain22` }
- enum `LvRenderType` { `LvRenderType_FullSize`, `LvRenderType_ScaleToFit`, `LvRenderType_ScaleToSize`, `LvRenderType_ScaleToTiles` }
- enum `LvSerialPortBaudRate` { `LvSerialPortBaudRate_Baud2400`, `LvSerialPortBaudRate_Baud4800`, `LvSerialPortBaudRate_Baud9600`, `LvSerialPortBaudRate_Baud14400`, `LvSerialPortBaudRate_Baud19200`, `LvSerialPortBaudRate_Baud38400`, `LvSerialPortBaudRate_Baud57600`, `LvSerialPortBaudRate_Baud115200` }
- enum `LvSerialPortParity` { `LvSerialPortParity_None`, `LvSerialPortParity_Odd`, `LvSerialPortParity_Even` }
- enum `LvSerialPortDataBits` { `LvSerialPortDataBits_DataBits7`, `LvSerialPortDataBits_DataBits8` }
- enum `LvSerialPortStopBits` { `LvSerialPortStopBits_StopBits1`, `LvSerialPortStopBits_StopBits1dot5`, `LvSerialPortStopBits_StopBits2` }
- enum `LvSerialPortCommandStatus` { `LvSerialPortCommandStatus_Success`, `LvSerialPortCommandStatus_Timeout`, `LvSerialPortCommandStatus_PortBusy`, `LvSerialPortCommandStatus_CommunicationError`, `LvSerialPortCommandStatus_FrameError`, `LvSerialPortCommandStatus_ParityError`, `LvSerialPortCommandStatus_Overflow` }
- enum `LvChunkLvExternalADCSelector` { `LvChunkLvExternalADCSelector_ExternalADC1`, `LvChunkLvExternalADCSelector_ExternalADC2`, `LvChunkLvExternalADCSelector_ExternalADC3`, `LvChunkLvExternalADCSelector_ExternalADC4` }
- enum `LvUserOutputSelector` { `LvUserOutputSelector_UserOutput1`, `LvUserOutputSelector_UserOutput2`, `LvUserOutputSelector_UserOutput3`, `LvUserOutputSelector_UserOutput4`, `LvUserOutputSelector_UserOutput5`, `LvUserOutputSelector_UserOutput6`, `LvUserOutputSelector_UserOutput7`, `LvUserOutputSelector_UserOutput8` }
- enum `LvUniProcessExecution` { `LvUniProcessExecution_OnBufferPtrQuery`, `LvUniProcessExecution_OnPopFromQueue`, `LvUniProcessExecution_OnExplicitRequest` }
- enum `LvLensControlCalibrationStatus` { `LvLensControlCalibrationStatus_Invalid`, `LvLensControlCalibrationStatus_Valid` }
- enum `LvLUTMode` { `LvLUTMode_Direct`, `LvLUTMode_BalanceWhite` }
- enum `LvBalanceRatioSelector` { `LvBalanceRatioSelector_Red`, `LvBalanceRatioSelector_Green`, `LvBalanceRatioSelector_Blue` }
- enum `LvBalanceWhiteAuto` { `LvBalanceWhiteAuto_Off`, `LvBalanceWhiteAuto_Once`, `LvBalanceWhiteAuto_Continuous` }
- enum `LvGevDeviceClass` { `LvGevDeviceClass_Transmitter` }

- enum `LvGevIPConfigurationStatus` {  
`LvGevIPConfigurationStatus_None`, `LvGevIPConfigurationStatus_PersistentIP`, `LvGevIPConfigurationStatus_DHCP`, `LvGevIPConfigurationStatus_LLA`,  
`LvGevIPConfigurationStatus_ForceIP` }
- enum `LvGevSCPDirection` { `LvGevSCPDirection_Transmitter` }
- enum `LvDeviceEndiannessMechanism` { `LvDeviceEndiannessMechanism_Legacy`, `LvDeviceEndiannessMechanism_Standard` }
- enum `LvUniLUTMode` { `LvUniLUTMode_Direct`, `LvUniLUTMode_Generated` }
- enum `LvUniLUTSelector` { `LvUniLUTSelector_Luminance`, `LvUniLUTSelector_Red`, `LvUniLUTSelector_Green`, `LvUniLUTSelector_Blue` }
- enum `LvUniColorTransformationMode` { `LvUniColorTransformationMode_Direct`, `LvUniColorTransformationMode_Generated` }
- enum `LvStrobeEnable` { `LvStrobeEnable_Off`, `LvStrobeEnable_AllClusters`, `LvStrobeEnable_LEDCluster1`, `LvStrobeEnable_LEDCluster2` }
- enum `LvStrobeDurationMode` { `LvStrobeDurationMode_FrameRateRelated`, `LvStrobeDurationMode_Free` }
- enum `LvStrobeDropMode` { `LvStrobeDropMode_DropStrobe`, `LvStrobeDropMode_DelayFrame` }
- enum `LvRegionSelector` { `LvRegionSelector_Region0`, `LvRegionSelector_Region1`, `LvRegionSelector_Region2`, `LvRegionSelector_Region3` }

### 5.30.1 Detailed Description

### 5.30.2 Enumeration Type Documentation

#### 5.30.2.1 enum `LvAcquisitionFrameRateControlMode`

Enum values for the `LvDevice_LvAcquisitionFrameRateControlMode` feature.

Enumerator

**`LvAcquisitionFrameRateControlMode_Off`** Disables frame rate control - the camera operates at maximum frame rate

**`LvAcquisitionFrameRateControlMode_On`** Enables frame rate control - the rate can be explicitly adjusted

#### 5.30.2.2 enum `LvAcquisitionMode`

Enum values for the `LvDevice_AcquisitionMode` feature.

Enumerator

**`LvAcquisitionMode_SingleFrame`** Single frame acquisition - after acquisition starts, single frame is acquired and acquisition stops.

**`LvAcquisitionMode_MultiFrame`** Multiple frame acquisition - after acquisition starts, specified number of frames is acquired and acquisition stops.

**`LvAcquisitionMode_Continuous`** Continuous acquisition - after starting, the acquisition is active until explicitly stopped.

#### 5.30.2.3 enum `LvAOIMode`

Enum values for the `LvDevice_LvAOIMode` feature.

Enumerator

**`LvAOIMode_Automatic`** Camera automatically applies as much of the desired AOI setting on the sensor and the rest is cut on transfer

**`LvAOIMode_ClipOnTransfer`** The AOI is applied before the transfer, in camera memory

**`LvAOIMode_Manual`** Fine control of separate AOI setting on the sensor and before the transfer

5.30.2.4 enum `LvBalanceRatioSelector`

Enum values for the `LvDevice_BalanceRatioSelector` feature.

## Enumerator

**`LvBalanceRatioSelector_Red`** Balance ratio will be applied to the red channel.

**`LvBalanceRatioSelector_Green`** Balance ratio will be applied to the green channel.

**`LvBalanceRatioSelector_Blue`** Balance ratio will be applied to the blue channel.

5.30.2.5 enum `LvBalanceWhiteAuto`

Enum values for the `LvDevice_BalanceWhiteAuto` feature.

## Enumerator

**`LvBalanceWhiteAuto_Off`** Automatic white balance mode off - the automatic white balance is not applied.

**`LvBalanceWhiteAuto_Once`** Automatic white balance mode once - the white balance factors are once adjusted, then switches the enumeration back to the Off value.

**`LvBalanceWhiteAuto_Continuous`** Automatic white balance mode continuous - the white balance is continuously auto-adjusted.

5.30.2.6 enum `LvBayerDecoderAlgorithm`

Enum values for the `LvDevice_LvBayerDecoderAlgorithm` and `LvDevice_LvUniBayerDecoderAlgorithm` feature.

## Enumerator

**`LvBayerDecoderAlgorithm_NearestNeighbour`** Nearest neighbour algorithm - Fastest decoding, giving the worst results, enables also decoding to a monochrome pixel format.

**`LvBayerDecoderAlgorithm_BilinearInterpolation`** Bilinear interpolation algorithm - Fast common decoding, enables also decoding to a monochrome pixel format.

**`LvBayerDecoderAlgorithm_BilinearColorCorrection`** Bilinear color correction algorithm - Decoding with quick enhancements on edges.

**`LvBayerDecoderAlgorithm_PixelGrouping`** Pixel grouping algorithm - Slower decoding, giving very good results.

**`LvBayerDecoderAlgorithm_VariableGradient`** Variable gradient algorithm - Slowest decoding, giving the best results.

5.30.2.7 enum `LvBlackLevelAuto`

Enum values for the `LvDevice_BlackLevelAuto` feature.

## Enumerator

**`LvBlackLevelAuto_Off`** Automatic black level mode off - the black level value is controlled 'manually'.

**`LvBlackLevelAuto_Once`** Automatic black level mode 'once' - the black level value is calculated and applied once and the feature switches back to 'off' (manual mode).

**`LvBlackLevelAuto_Continuous`** Continuous automatic black level mode - the automatic black level is applied continuously.

### 5.30.2.8 enum LvBlackLevelSelector

Enum values for the [LvDevice\\_BlackLevelSelector](#) feature.

Enumerator

**LvBlackLevelSelector\_All** Apply black level on all channels and taps.

**LvBlackLevelSelector\_Tap1** Apply black level on tap1.

**LvBlackLevelSelector\_Tap2** Apply black level on tap2.

### 5.30.2.9 enum LvBootSwitch

Enum values for the [LvDevice\\_LvBootSwitch](#) feature.

Enumerator

**LvBootSwitch\_PureGEV** Selects the pure GigE Vision mode strictly following the GigE Vision specification

**LvBootSwitch\_Legacy** Selects the legacy mode allowing dual operation through GigE Vision or custom protocol.

### 5.30.2.10 enum LvChunkGainSelector

Enum values for the [LvDevice\\_ChunkGainSelector](#) feature.

Enumerator

**LvChunkGainSelector\_AnalogAll** Analog gain.

**LvChunkGainSelector\_DigitalAll** Digital gain.

### 5.30.2.11 enum LvChunkLvExternalADCSelector

Enum values for the [LvDevice\\_ChunkLvExternalADCSelector](#) feature.

Enumerator

**LvChunkLvExternalADCSelector\_ExternalADC1** External ADC 1.

**LvChunkLvExternalADCSelector\_ExternalADC2** External ADC 2.

**LvChunkLvExternalADCSelector\_ExternalADC3** External ADC 3.

**LvChunkLvExternalADCSelector\_ExternalADC4** External ADC 4.

### 5.30.2.12 enum LvChunkSelector

Enum values for the [LvDevice\\_ChunkSelector](#) feature.

Enumerator

**LvChunkSelector\_OffsetX** Selects the X offset chunk for configuration.

**LvChunkSelector\_OffsetY** Selects the Y offset chunk for configuration.

**LvChunkSelector\_Width** Selects the width chunk for configuration.

**LvChunkSelector\_Height** Selects the height chunk for configuration.

**LvChunkSelector\_PixelFormat** Selects the pixel format chunk for configuration.

**LvChunkSelector\_LinePitch** Selects the line pitch chunk for configuration.

- LvChunkSelector\_FrameID*** Selects the frame id chunk for configuration.
- LvChunkSelector\_Timestamp*** Selects the time stamp chunk for configuration.
- LvChunkSelector\_ExposureTime*** Selects the exposure time chunk for configuration.
- LvChunkSelector\_Gain*** Selects the gain chunk for configuration.
- LvChunkSelector\_LineStatusAll*** Selects the line status all chunk for configuration.
- LvChunkSelector\_BlackLevel*** Selects the black level chunk for configuration.
- LvChunkSelector\_LvExternalADCValue*** Selects the external ADC chunk for configuration.
- LvChunkSelector\_LvSmartAppString*** Selects the smart application string chunk for configuration.
- LvChunkSelector\_LvSmartAppInt*** Selects the smart application signed integer chunk for configuration.
- LvChunkSelector\_LvSmartAppUInt*** Selects the smart application unsigned integer chunk for configuration.
- LvChunkSelector\_LvSmartAppRegister*** Selects the smart application raw register chunk for configuration.
- LvChunkSelector\_LvTriggerDelayed*** Selects the trigger delayed chunk for configuration.
- LvChunkSelector\_LvStrobeDropped*** Selects the strobe dropped chunk for configuration.
- LvChunkSelector\_LvFrameAbort*** Selects the frame abort chunk for configuration.
- LvChunkSelector\_LvTriggerDropped*** Selects the trigger dropped chunk for configuration.
- LvChunkSelector\_LvTriggerError*** Selects the trigger error chunk for configuration.
- LvChunkSelector\_LvEncoderPosition*** Selects the encoder position chunk for configuration.
- LvChunkSelector\_LvEncoderRotation*** Selects the encoder rotation chunk for configuration.
- LvChunkSelector\_LvVirtFrmTriggerAbort*** Selects the frame trigger abort chunk for configuration.
- LvChunkSelector\_LvVirtFrmTriggerDrop*** Selects the frame trigger drop chunk for configuration.
- LvChunkSelector\_LvVirtFrmTriggerDelay*** Selects the frame trigger delay chunk for configuration.
- LvChunkSelector\_LvVirtFrmFirst*** Selects the first virtual frame chunk for configuration.
- LvChunkSelector\_LvVirtFrmLength*** Selects the virtual frame length chunk for configuration.

#### 5.30.2.13 enum LvColorTransformationSelector

Enum values for the [LvDevice\\_ColorTransformationSelector](#) feature.

Enumerator

- LvColorTransformationSelector\_RGBtoRGB*** RGB to RGB matrix transformation
- LvColorTransformationSelector\_RGBtoYUV*** RGB to YUV matrix transformation

#### 5.30.2.14 enum LvColorTransformationValueSelector

Enum values for the [LvDevice\\_ColorTransformationValueSelector](#) feature.

Enumerator

- LvColorTransformationValueSelector\_Gain00*** Selects the gain 00 (RR, red-red) entry of the color transformation matrix.
- LvColorTransformationValueSelector\_Gain01*** Selects the gain 01 (RG, red-green) entry of the color transformation matrix.
- LvColorTransformationValueSelector\_Gain02*** Selects the gain 02 (RB, red-blue) entry of the color transformation matrix.
- LvColorTransformationValueSelector\_Gain10*** Selects the gain 10 (GR, green-red) entry of the color transformation matrix.



- LvColorTransformationValueSelector\_Gain11*** Selects the gain 11 (GG, green-green) entry of the color transformation matrix.
- LvColorTransformationValueSelector\_Gain12*** Selects the gain 12 (GB, green-blue) entry of the color transformation matrix.
- LvColorTransformationValueSelector\_Gain20*** Selects the gain 20 (BR, blue-red) entry of the color transformation matrix.
- LvColorTransformationValueSelector\_Gain21*** Selects the gain 21 (BG, blue-green) entry of the color transformation matrix.
- LvColorTransformationValueSelector\_Gain22*** Selects the gain 22 (BB, blue-blue) entry of the color transformation matrix.
- LvColorTransformationValueSelector\_Offset0*** Selects the offset 0 (red) entry of the color transformation matrix.
- LvColorTransformationValueSelector\_Offset1*** Selects the offset 1 (green) entry of the color transformation matrix.
- LvColorTransformationValueSelector\_Offset2*** Selects the offset 2 (blue) entry of the color transformation matrix.

#### 5.30.2.15 enum `LvCounterEventSource`

Enum values for the [LvDevice\\_CounterEventSource](#) feature.

##### Enumerator

- LvCounterEventSource\_Off*** Switches counter event signal off - no signal will be incrementing the counter
- LvCounterEventSource\_FrameTrigger*** Switches counter event signal to frame trigger - activation of the frame trigger internal signal (before counting down eventual trigger delay) increments the counter.
- LvCounterEventSource\_TimerTick*** Switches counter event signal to timer tick - 1MHz clock increments the counter.
- LvCounterEventSource\_Line1*** Switches counter event signal to line 1 (optocoupler input) - active edge of line 1 increments the counter.
- LvCounterEventSource\_Line2*** Switches counter event signal to line 2 (optocoupler input) - active edge of line 2 increments the counter.
- LvCounterEventSource\_Line3*** Switches counter event signal to line 3 (optocoupler input) - active edge of line 3 increments the counter.
- LvCounterEventSource\_Line4*** Switches counter event signal to line 4 (optocoupler input) - active edge of line 4 increments the counter.
- LvCounterEventSource\_Line17*** Switches counter event signal to line 17 (TTL input) - active edge of line 17 increments the counter.
- LvCounterEventSource\_Line18*** Switches counter event signal to line 18 (TTL input) - active edge of line 18 increments the counter.

#### 5.30.2.16 enum `LvCounterMode`

Enum values for the [LvDevice\\_LvCounterMode](#) feature.

##### Enumerator

- LvCounterMode\_Autoreset*** Automatic reset mode. Once completed, the counter automatically resets itself and starts counting again.



## 5.30.2.17 enum LvCounterSelector

Enum values for the [LvDevice\\_CounterSelector](#) feature.

## Enumerator

**LvCounterSelector\_Counter1** Selects counter 1 for configuration.

**LvCounterSelector\_Counter2** Selects counter 2 for configuration.

**LvCounterSelector\_Counter3** Selects counter 3 for configuration.

**LvCounterSelector\_Counter4** Selects counter 4 for configuration.

## 5.30.2.18 enum LvDeviceAccess

This enum is used for opening the Device - see [LvDeviceOpen\(\)](#).

## Enumerator

**LvDeviceAccess\_None** Represents the GenTL DEVICE\_ACCESS\_NONE. This either means that the Device is not open because it was not opened before or the access to it was denied.

**LvDeviceAccess\_ReadOnly** Represents the GenTL DEVICE\_ACCESS\_READONLY. Open the Device read only. All Port functions can only read from the Device.

**LvDeviceAccess\_Control** Represents the GenTL DEVICE\_ACCESS\_CONTROL. Open the Device in a way that other hosts/processes can have read only access to the Device. Device access level is read/write for this process.

**LvDeviceAccess\_Exclusive** Represents the GenTL DEVICE\_ACCESS\_EXCLUSIVE. Open the Device in a way that only this host/process can have access to the Device. Device access level is read/write for this process.

## 5.30.2.19 enum LvDeviceAccessStatus

Values for the [LvFtrInfo\\_DeviceAccessStatus](#) and [LvInterface\\_DeviceAccessStatus](#) features.

## Enumerator

**LvDeviceAccessStatus\_Unknown** Represents the GenTL DEVICE\_ACCESS\_STATUS\_UNKNOWN. The current availability of the Device is unknown.

**LvDeviceAccessStatus\_ReadWrite** Represents the GenTL DEVICE\_ACCESS\_STATUS\_READWRITE - The Device is available for Read/Write.

**LvDeviceAccessStatus\_ReadOnly** Represents the GenTL DEVICE\_ACCESS\_STATUS\_READONLY - The Device is available only for Read access (cannot be controlled).

**LvDeviceAccessStatus\_NoAccess** Represents the GenTL DEVICE\_ACCESS\_STATUS\_NOACCESS - The Device is not available either because it is already open or because it is not reachable.

## 5.30.2.20 enum LvDeviceClockSelector

Enum values for the [LvDevice\\_DeviceClockSelector](#) feature.

## Enumerator

**LvDeviceClockSelector\_SensorDigitization** Sensor digitization clock.

### 5.30.2.21 enum LvDeviceEndiannessMechanism

Enum values for the [LvDevice\\_DeviceEndiannessMechanism](#) feature.

Enumerator

**LvDeviceEndiannessMechanism\_Legacy** Legacy endianness handling mode, intended for GigE Vision remote devices using GenICam schema version 1.0.

**LvDeviceEndiannessMechanism\_Standard** Standard endianness handling mode, intended for GigE Vision remote devices using GenICam schema version 1.1 and newer.

### 5.30.2.22 enum LvDeviceScanType

Enum values for the [LvDevice\\_DeviceScanType](#) feature.

Enumerator

**LvDeviceScanType\_Areascan** Indicates area scan sensor.

**LvDeviceScanType\_Linescan** Indicates linr scan sensor.

### 5.30.2.23 enum LvDeviceTemperatureSelector

Enum values for the [LvDevice\\_DeviceTemperatureSelector](#) feature.

Enumerator

**LvDeviceTemperatureSelector\_Sensor** Temperature on sensor

**LvDeviceTemperatureSelector\_Mainboard** Temperature on main board

### 5.30.2.24 enum LvDeviceType

Enum values for the [LvDevice\\_DeviceType](#) feature.

Enumerator

**LvDeviceType\_Custom** Device based on a custom technology.

**LvDeviceType\_GEV** GigE Vision compatible device.

**LvDeviceType\_U3V** USB3 Vision compatible device.

### 5.30.2.25 enum LvEventNotification

Enum values for the [LvDevice\\_EventNotification](#) feature.

Enumerator

**LvEventNotification\_Off** The notifications for the selected event are deactivated.

**LvEventNotification\_On** The notifications for the selected event are activated.

5.30.2.26 enum `LvEventSelector`

Enum values for the `LvDevice_EventSelector` feature.

## Enumerator

**`LvEventSelector_LvLog`** This enumeration value selects the log event for configuration.

**`LvEventSelector_LvSmartAppLog`** This enumeration value selects the smart application log event for configuration.

**`LvEventSelector_LvSmartAppString`** This enumeration value selects the smart application string event for configuration.

**`LvEventSelector_LvSmartAppInt`** This enumeration value selects the smart application signed integer event for configuration.

**`LvEventSelector_LvSmartAppUint`** This enumeration value selects the smart application unsigned integer event for configuration.

**`LvEventSelector_LvSmartAppRegister`** This enumeration value selects the smart application raw register event for configuration.

**`LvEventSelector_LvTriggerDropped`** This enumeration value selects the dropped trigger event for configuration.

5.30.2.27 enum `LvExposureAuto`

Enum values for the `LvDevice_ExposureAuto` feature.

## Enumerator

**`LvExposureAuto_Off`** Automatic exposure mode off - the automatic exposure is not applied.

**`LvExposureAuto_Once`** Automatic exposure mode once - the exposure time is once adjusted, then switches back to off.

**`LvExposureAuto_Continuous`** Automatic exposure mode continuous - the exposure time is continuously auto-adjusted.

5.30.2.28 enum `LvExposureMode`

Enum values for the `LvDevice_ExposureMode` feature.

## Enumerator

**`LvExposureMode_Timed`** Timed exposure mode - the exposure time is controlled by corresponding feature.

5.30.2.29 enum `LvExternalADCSelector`

Enum values for the `LvDevice_LvExternalADCSelector` feature.

## Enumerator

**`LvExternalADCSelector_ExternalADC1`** Selects external ADC 1 for configuration.

**`LvExternalADCSelector_ExternalADC2`** Selects external ADC 2 for configuration.

**`LvExternalADCSelector_ExternalADC3`** Selects external ADC 3 for configuration.

**`LvExternalADCSelector_ExternalADC4`** Selects external ADC 4 for configuration.

### 5.30.2.30 enum LvExternalDeviceControlMode

Enum values for the [LvDevice\\_LvExternalDeviceControlMode](#) feature.

Enumerator

***LvExternalDeviceControlMode\_Custom*** Selects the custom mode.

### 5.30.2.31 enum LvGainAuto

Enum values for the [LvDevice\\_GainAuto](#) feature.

Enumerator

***LvGainAuto\_Off*** Automatic gain mode off - the gain value is controlled 'manually'.

***LvGainAuto\_Once*** Automatic gain mode 'once' - the gain value is calculated and applied once and the feature switches back to 'off' (manual mode).

***LvGainAuto\_Continuous*** Continuous automatic gain mode - the AGC is applied continuously.

### 5.30.2.32 enum LvGainSelector

Enum values for the [LvDevice\\_GainSelector](#) feature.

Enumerator

***LvGainSelector\_All*** Apply gain on all channels and taps.

***LvGainSelector\_AnalogAll*** Apply analog gain.

***LvGainSelector\_DigitalAll*** Apply digital gain.

### 5.30.2.33 enum LvGevCCP

Enum values for the [LvDevice\\_GevCCP](#) feature.

Enumerator

***LvGevCCP\_OpenAccess*** Sets the control channel privilege feature to open.

***LvGevCCP\_ExclusiveAccess*** Sets the control channel privilege feature to exclusive.

***LvGevCCP\_ControlAccess*** Sets the control channel privilege feature to control.

***LvGevCCP\_ControlAccessSwitchoverActive*** Sets the control channel privilege feature to control with switchover active.

### 5.30.2.34 enum LvGevDeviceClass

Enum values for the [LvDevice\\_GevDeviceClass](#) feature.

Enumerator

***LvGevDeviceClass\_Transmitter*** Indicates that the device is a GigE Vision transmitter.

### 5.30.2.35 enum LvGevDeviceModeCharacterSet

Enum values for the [LvDevice\\_GevDeviceModeCharacterSet](#) feature.

Enumerator

***LvGevDeviceModeCharacterSet\_UTF8*** Indicates that the camera uses the UTF8 character set.

5.30.2.36 enum `LvGevDeviceStreamCaptureMode`

Enum values for the `LvDevice_LvGevDeviceStreamCaptureMode` feature.

## Enumerator

**`LvGevDeviceStreamCaptureMode_SystemDefault`** System default mode, configurable in the ini file.

**`LvGevDeviceStreamCaptureMode_Socket`** Socket mode, the GVSP stream is processed through the socket interface (regular operating system networks stack).

**`LvGevDeviceStreamCaptureMode_FilterDriver`** Filter driver mode, the GVSP stream is processed through the filter driver interface (bypassing operating system network stack).

5.30.2.37 enum `LvGevIPConfigurationStatus`

Enum values for the `LvDevice_GevIPConfigurationStatus` feature.

## Enumerator

**`LvGevIPConfigurationStatus_None`** No IP configuration method was executed or it is not known.

**`LvGevIPConfigurationStatus_PersistentIP`** The current device IP configuration was obtained through persistent IP.

**`LvGevIPConfigurationStatus_DHCP`** The current device IP configuration was obtained through DHCP.

**`LvGevIPConfigurationStatus_LLA`** The current device IP configuration was obtained through LLA.

**`LvGevIPConfigurationStatus_ForceIP`** The current device IP configuration was obtained through ForceIP.

5.30.2.38 enum `LvGevSCPDirection`

Enum values for the `LvDevice_GevSCPDirection` feature.

## Enumerator

**`LvGevSCPDirection_Transmitter`** Indicates that the stream channel is a transmitter.

5.30.2.39 enum `LvGevSupportedOptionSelector`

Enum values for the `LvDevice_GevSupportedOptionSelector` feature.

## Enumerator

**`LvGevSupportedOptionSelector_IPConfigurationLLA`** Indicates whether the (first) network interface supports auto IP addressing (also known as LLA).

**`LvGevSupportedOptionSelector_IPConfigurationDHCP`** Indicates whether the (first) network interface supports DHCP IP addressing.

**`LvGevSupportedOptionSelector_IPConfigurationPersistentIP`** Indicates whether the (first) network interface supports fixed IP addressing (also known as persistent IP addressing).

**`LvGevSupportedOptionSelector_CommandsConcatenation`** Indicates whether command concatenation is supported by the device.

**`LvGevSupportedOptionSelector_WriteMem`** Indicates whether write memory scheme is supported by the device.

**`LvGevSupportedOptionSelector_PacketResend`** Indicates whether packet resend is supported by the device.

**`LvGevSupportedOptionSelector_Event`** Indicates whether event (message channel) is supported by the device.

- LvGevSupportedOptionSelector\_EventData*** Indicates whether eventdata (message channel) is supported by the device.
- LvGevSupportedOptionSelector\_PendingAck*** Indicates whether pending acknowledge is supported by the device.
- LvGevSupportedOptionSelector\_Action*** Indicates whether action commands are supported by the device.
- LvGevSupportedOptionSelector\_PrimaryApplicationSwitchover*** Indicates whether primary application switchover is supported by the device.
- LvGevSupportedOptionSelector\_ExtendedStatusCodes*** Indicates whether extended GigE Vision status codes are supported by the device.
- LvGevSupportedOptionSelector\_DiscoveryAckDelayWritable*** Indicates whether writable discovery acknowledge delay is supported by the device.
- LvGevSupportedOptionSelector\_DiscoveryAckDelay*** Indicates whether discovery acknowledge delay is supported by the device.
- LvGevSupportedOptionSelector\_TestData*** Indicates whether test data is supported by the device.
- LvGevSupportedOptionSelector\_ManifestTable*** Indicates whether manifest table is supported by the device.
- LvGevSupportedOptionSelector\_CCPApplicationSocket*** Indicates whether the primary application port and IP address features are supported by the device.
- LvGevSupportedOptionSelector\_LinkSpeed*** Indicates whether link speed feature is supported by the device.
- LvGevSupportedOptionSelector\_HeartbeatDisable*** Indicates whether heartbeat disabling is supported by the device.
- LvGevSupportedOptionSelector\_SerialNumber*** Indicates whether serial number feature is supported by the device.
- LvGevSupportedOptionSelector\_UserDefinedName*** Indicates whether user defined name is supported by the device.
- LvGevSupportedOptionSelector\_StreamChannelSourceSocket*** Indicates whether the stream channel source port feature is supported by the device.
- LvGevSupportedOptionSelector\_StreamChannel0ExtendedChunkData*** Indicates whether the extended chunk data is supported by the device.
- LvGevSupportedOptionSelector\_StreamChannel0UnconditionalStreaming*** Indicates whether the unconditional streaming is supported by the device.
- LvGevSupportedOptionSelector\_StreamChannel0IPReassembly*** Indicates whether the reassembly of fragmented IP packets is supported by the device.
- LvGevSupportedOptionSelector\_StreamChannel0BigAndLittleEndian*** Indicates whether the big and little endian stream channel is supported by the device.
- LvGevSupportedOptionSelector\_MessageChannelSourceSocket*** Indicates whether the message channel source port feature is supported by the device.

#### 5.30.2.40 enum LvImageStampSelector

Enum values for the [LvDevice\\_LvImageStampSelector](#) feature.

##### Enumerator

- LvImageStampSelector\_Timestamp*** Selects the flag controlling reset of the image timestamp
- LvImageStampSelector\_FrameID*** Selects the flag controlling reset of the image frame ID

5.30.2.41 enum `LvInterfaceType`

Enum values for the `LvInterface_InterfaceType` feature.

Enumerator

**`LvInterfaceType_Custom`** Interface supporting a custom technology devices.

**`LvInterfaceType_GEV`** Interface supporting GigE Vision devices.

**`LvInterfaceType_U3V`** Interface supporting USB3 Vision devices.

5.30.2.42 enum `LvLensControlCalibrationStatus`

Enum values for the `LvDevice_LvLensControlCalibrationStatus` feature.

Enumerator

**`LvLensControlCalibrationStatus_Invalid`** Current calibration parameters are invalid

**`LvLensControlCalibrationStatus_Valid`** Current calibration parameters are valid

5.30.2.43 enum `LvLensControlTargetApproach`

Enum values for the `LvDevice_LvLensControlTargetApproach` feature.

Enumerator

**`LvLensControlTargetApproach_Direct`** Approaches the target position directly, no matter from which side.

**`LvLensControlTargetApproach_FromPlus`** Approaches the target position always from the plus side to improve accuracy.

**`LvLensControlTargetApproach_FromMinus`** Approaches the target position always from the minus side to improve accuracy.

5.30.2.44 enum `LvLineDebounceMode`

Enum values for the `LvDevice_LineMode` feature.

Enumerator

**`LvLineDebounceMode_Debounce`** The line debounce filter is used in debounce mode.

**`LvLineDebounceMode_Deglitch`** The line debounce filter is used in deglitch mode.

5.30.2.45 enum `LvLineFormat`

Enum values for the `LvDevice_LineFormat` feature.

Enumerator

**`LvLineFormat_NoConnect`** Not connected line.

**`LvLineFormat_TriState`** The Line is currently in Tri-state mode (Not driven).

**`LvLineFormat_TTL`** The Line is currently accepting or sending TTL level signals.

**`LvLineFormat_LVDS`** The Line is currently accepting or sending LVDS level signals.

**`LvLineFormat_RS422`** The Line is currently accepting or sending RS-422 level signals.

**`LvLineFormat_OptoCoupled`** Optically isolated line (optocoupler).

#### 5.30.2.46 enum LvLineMode

Enum values for the [LvDevice\\_LineMode](#) feature.

##### Enumerator

***LvLineMode\_Input*** The line is used as signal input.

***LvLineMode\_Output*** The line is used as signal output.

#### 5.30.2.47 enum LvLineSelector

Enum values for the [LvDevice\\_LineSelector](#) feature.

##### Enumerator

***LvLineSelector\_Line1*** Selects device's logical line 1 (optocoupler input).

***LvLineSelector\_Line2*** Selects device's logical line 2 (optocoupler input).

***LvLineSelector\_Line3*** Selects device's logical line 3 (optocoupler input).

***LvLineSelector\_Line4*** Selects device's logical line 4 (optocoupler input).

***LvLineSelector\_Line5*** Selects device's logical line 5.

***LvLineSelector\_Line6*** Selects device's logical line 6.

***LvLineSelector\_Line7*** Selects device's logical line 7.

***LvLineSelector\_Line8*** Selects device's logical line 8.

***LvLineSelector\_Line9*** Selects device's logical line 9 (optocoupler output).

***LvLineSelector\_Line10*** Selects device's logical line 10 (optocoupler output).

***LvLineSelector\_Line11*** Selects device's logical line 11 (optocoupler output).

***LvLineSelector\_Line12*** Selects device's logical line 12 (optocoupler output).

***LvLineSelector\_Line13*** Selects device's logical line 13.

***LvLineSelector\_Line14*** Selects device's logical line 14.

***LvLineSelector\_Line15*** Selects device's logical line 15.

***LvLineSelector\_Line16*** Selects device's logical line 16.

***LvLineSelector\_Line17*** Selects device's logical line 17 (TTL input).

***LvLineSelector\_Line18*** Selects device's logical line 18 (TTL input).

***LvLineSelector\_Line19*** Selects device's logical line 19.

***LvLineSelector\_Line20*** Selects device's logical line 20.

***LvLineSelector\_Line21*** Selects device's logical line 21.

***LvLineSelector\_Line22*** Selects device's logical line 22.

***LvLineSelector\_Line23*** Selects device's logical line 23.

***LvLineSelector\_Line24*** Selects device's logical line 24.

***LvLineSelector\_Line25*** Selects device's logical line 25 (TTL output).

***LvLineSelector\_Line26*** Selects device's logical line 26 (TTL output).

***LvLineSelector\_Line27*** Selects device's logical line 27.

***LvLineSelector\_Line28*** Selects device's logical line 28.

***LvLineSelector\_Line29*** Selects device's logical line 29.

***LvLineSelector\_Line30*** Selects device's logical line 30.

***LvLineSelector\_Line31*** Selects device's logical line 31.

***LvLineSelector\_Line32*** Selects device's logical line 32.



## 5.30.2.48 enum LvLineSource

Enum values for the [LvDevice\\_LineSource](#) feature.

## Enumerator

**LvLineSource\_Off** Switches the line source off. This disables the line output (disconnects the line).

**LvLineSource\_ExposureActive** Selects exposure active signal as line source.

**LvLineSource\_Timer1Active** Selects timer 1 active signal as line source.

**LvLineSource\_Timer2Active** Selects timer 2 active signal as line source.

**LvLineSource\_Timer3Active** Selects timer 3 active signal as line source.

**LvLineSource\_Timer4Active** Selects timer 4 active signal as line source.

**LvLineSource\_UserOutput1** Selects user output 1 value signal as line source.

**LvLineSource\_UserOutput2** Selects user output 2 value signal as line source.

**LvLineSource\_UserOutput3** Selects user output 3 value signal as line source.

**LvLineSource\_UserOutput4** Selects user output 4 value signal as line source.

**LvLineSource\_UserOutput5** Selects user output 5 value signal as line source.

**LvLineSource\_UserOutput6** Selects user output 6 value signal as line source.

**LvLineSource\_UserOutput7** Selects user output 7 value signal as line source.

**LvLineSource\_UserOutput8** Selects user output 8 value signal as line source.

**LvLineSource\_Counter1Active** Selects counter 1 active signal as line source.

**LvLineSource\_Counter2Active** Selects counter 2 active signal as line source.

**LvLineSource\_Counter3Active** Selects counter 3 active signal as line source.

**LvLineSource\_Counter4Active** Selects counter 4 active signal as line source.

## 5.30.2.49 enum LvLUTMode

Enum values for the [LvDevice\\_LvLUTMode](#) feature.

## Enumerator

**LvLUTMode\_Direct** In this mode the LUT is controlled directly.

**LvLUTMode\_BalanceWhite** In this mode the LUT is controlled through the higher level features, such as brightness, contrast, gamma or white balance.

## 5.30.2.50 enum LvLUTSelector

Enum values for the [LvDevice\\_LUTSelector](#) feature.

## Enumerator

**LvLUTSelector\_Luminance** Selects the luminance LUT for configuration.

**LvLUTSelector\_Red** Selects the red LUT for configuration.

**LvLUTSelector\_Green** Selects the green LUT for configuration.

**LvLUTSelector\_Blue** Selects the blue LUT for configuration.

5.30.2.51 enum `LvPixelFormat`

`LvPixelFormat` constants Enum for the `LvDevice_PixelFormat`, `LvDevice_ChunkPixelFormat` and `LvDevice_LvUniPixelFormat` features. The Pixel format constants are defined by the GenICam standard. The value consists of 3 parts:

- byte 1 - color/mono/custom
  - byte 2 - bits per pixel
  - byte 3 and 4 - ID of the pixel format
- Exceptions are:
- `LvPixelFormat_BGR555Packed` - used only in the Image Processing Library (in conversions for display).

## Enumerator

**`LvPixelFormat_Mono8`** Monochrome 8-bit. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY8BIT | 0x0001`).

**`LvPixelFormat_Mono8S`** Monochrome 8-bit signed. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY8BIT | 0x0002`).

**`LvPixelFormat_Mono10`** Monochrome 10-bit. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0003`).

**`LvPixelFormat_Mono10Packed`** Monochrome 10-bit packed. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY12BIT | 0x0004`).

**`LvPixelFormat_Mono12`** Monochrome 12-bit. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0005`).

**`LvPixelFormat_Mono12Packed`** Monochrome 12-bit packed. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY12BIT | 0x0006`).

**`LvPixelFormat_Mono14`** Monochrome 14-bit. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0025`).

**`LvPixelFormat_Mono16`** Monochrome 16-bit. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0007`).

**`LvPixelFormat_BayerGR8`** Undecoded 8-bit Bayer array with the GR array position. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY8BIT | 0x0008`).

**`LvPixelFormat_BayerRG8`** Undecoded 8-bit Bayer array with the RG array position. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY8BIT | 0x0009`).

**`LvPixelFormat_BayerGB8`** Undecoded 8-bit Bayer array with the GB array position. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY8BIT | 0x000A`).

**`LvPixelFormat_BayerBG8`** Undecoded 8-bit Bayer array with the BG array position. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY8BIT | 0x000B`).

**`LvPixelFormat_BayerGR10`** Undecoded 10-bit Bayer array with the GR array position. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x000C`).

**`LvPixelFormat_BayerRG10`** Undecoded 10-bit Bayer array with the RG array position. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x000D`).

**`LvPixelFormat_BayerGB10`** Undecoded 10-bit Bayer array with the GB array position. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x000E`).

**`LvPixelFormat_BayerBG10`** Undecoded 10-bit Bayer array with the BG array position. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x000F`).

**`LvPixelFormat_BayerGR12`** Undecoded 12-bit Bayer array with the GR array position. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0010`).

**`LvPixelFormat_BayerRG12`** Undecoded 12-bit Bayer array with the RG array position. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0011`).

**`LvPixelFormat_BayerGB12`** Undecoded 12-bit Bayer array with the GB array position. Defined as (`LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0012`).

- LvPixelFormat\_BayerBG12** Undecoded 12-bit Bayer array with the BG array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY16BIT | 0x0013).
- LvPixelFormat\_BayerGR10Packed** Undecoded 10-bit packed Bayer array with the GR array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY12BIT | 0x0026).
- LvPixelFormat\_BayerRG10Packed** Undecoded 10-bit packed Bayer array with the RG array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY12BIT | 0x0027).
- LvPixelFormat\_BayerGB10Packed** Undecoded 10-bit packed Bayer array with the GB array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY12BIT | 0x0028).
- LvPixelFormat\_BayerBG10Packed** Undecoded 10-bit packed Bayer array with the BG array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY12BIT | 0x0029).
- LvPixelFormat\_BayerGR12Packed** Undecoded 12-bit packed Bayer array with the GR array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY12BIT | 0x002A).
- LvPixelFormat\_BayerRG12Packed** Undecoded 12-bit packed Bayer array with the RG array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY12BIT | 0x002B).
- LvPixelFormat\_BayerGB12Packed** Undecoded 12-bit packed Bayer array with the GB array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY12BIT | 0x002C).
- LvPixelFormat\_BayerBG12Packed** Undecoded 10-bit packed Bayer array with the BG array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY12BIT | 0x002D).
- LvPixelFormat\_BayerGR16** Undecoded 16-bit Bayer array with the GR array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY16BIT | 0x002E).
- LvPixelFormat\_BayerRG16** Undecoded 16-bit Bayer array with the RG array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY16BIT | 0x002F).
- LvPixelFormat\_BayerGB16** Undecoded 16-bit Bayer array with the GB array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY16BIT | 0x0030).
- LvPixelFormat\_BayerBG16** Undecoded 16-bit Bayer array with the BG array position. Defined as (LV\_PIXEL\_MONO | LV\_PIXEL\_OCCUPY16BIT | 0x0031).
- LvPixelFormat\_RGB8** RGB 24-bit packed (3x8 bits). Defined as (LV\_PIXEL\_COLOR | LV\_PIXEL\_OCCUPY24BIT | 0x0014).
- LvPixelFormat\_BGR8** BGR 24-bit packed (3x8 bits). Defined as (LV\_PIXEL\_COLOR | LV\_PIXEL\_OCCUPY24BIT | 0x0015).
- LvPixelFormat\_RGBA8** RGB 32-bit packed (3x8 bits + 1x8 bits alpha). Defined as (LV\_PIXEL\_COLOR | LV\_PIXEL\_OCCUPY32BIT | 0x0016).
- LvPixelFormat\_BGRA8** BGR 32-bit packed (3x8 bits + 1x8 bits alpha). Defined as (LV\_PIXEL\_COLOR | LV\_PIXEL\_OCCUPY32BIT | 0x0017).
- LvPixelFormat\_RGB10** RGB 48-bit packed (3x10 bits). Defined as (LV\_PIXEL\_COLOR | LV\_PIXEL\_OCCUPY48BIT | 0x0018).
- LvPixelFormat\_BGR10** BGR 48-bit packed (3x10 bits). Defined as (LV\_PIXEL\_COLOR | LV\_PIXEL\_OCCUPY48BIT | 0x0019).
- LvPixelFormat\_RGB12** RGB 48-bit packed (3x12 bits). Defined as (LV\_PIXEL\_COLOR | LV\_PIXEL\_OCCUPY48BIT | 0x001A).
- LvPixelFormat\_BGR12** BGR 48-bit packed (3x12 bits). Defined as (LV\_PIXEL\_COLOR | LV\_PIXEL\_OCCUPY48BIT | 0x001B).
- LvPixelFormat\_RGB16** RGB 48-bit packed (3x16 bits). Defined as (LV\_PIXEL\_COLOR | LV\_PIXEL\_OCCUPY48BIT | 0x0033).
- LvPixelFormat\_RGB10V1Packed** RGB 32-bit packed (3x10 bits). Defined as (LV\_PIXEL\_COLOR | LV\_PIXEL\_OCCUPY32BIT | 0x001C).
- LvPixelFormat\_RGB10P32** RGB 32-bit packed (3x10 bits). Defined as (LV\_PIXEL\_COLOR | LV\_PIXEL\_OCCUPY32BIT | 0x001D).
- LvPixelFormat\_RGB12V1Packed** RGB 36-bit packed (3x12 bits). Defined as (LV\_PIXEL\_COLOR | LV\_PIXEL\_OCCUPY36BIT | 0x0034).

- LvPixelFormat\_RGB565P** RGB 16-bit packed (5,6,5 bits). Defined as (LV\_PIX\_COLOR | LV\_PIX\_OCCU←PY16BIT | 0x0035).
- LvPixelFormat\_BGR565P** BGR 16-bit packed (5,6,5 bits). Defined as (LV\_PIX\_COLOR | LV\_PIX\_OCCU←PY16BIT | 0x0036).
- LvPixelFormat\_YUV411\_8** YUV 4-1-1 Packed. Defined as (LV\_PIX\_COLOR | LV\_PIX\_OCCUPY12BIT | 0x001E).
- LvPixelFormat\_YUV422\_8\_UYVY** YUV 4-2-2 UYVY Packed. Defined as (LV\_PIX\_COLOR | LV\_PIX\_OC←CUPY16BIT | 0x001F).
- LvPixelFormat\_YUV8** YUV 4-4-4 Packed. Defined as (LV\_PIX\_COLOR | LV\_PIX\_OCCUPY24BIT | 0x0020).
- LvPixelFormat\_YUV422\_8** YUV 4-2-2 YUYV Packed. Defined as (LV\_PIX\_COLOR | LV\_PIX\_OCCUPY16←BIT | 0x0032).
- LvPixelFormat\_YCbCr422\_8** YCbCr 4-2-2 Packed. Defined as (LV\_PIX\_COLOR | LV\_PIX\_OCCUPY16BIT | 0x003B).
- LvPixelFormat\_YCbCr601\_422\_8** YCbCr 4-2-2 Packed. Defined as (LV\_PIX\_COLOR | LV\_PIX\_OCCUP←Y16BIT | 0x003E).
- LvPixelFormat\_YCbCr601\_422\_8\_CbYCrY** YCbCr 4-2-2 Packed. Defined as (LV\_PIX\_COLOR | LV\_PIX←\_OCCUPY16BIT | 0x0044).
- LvPixelFormat\_RGB8\_Planar** RGB 8-bit planar. Defined as (LV\_PIX\_COLOR | LV\_PIX\_OCCUPY24BIT | 0x0021).
- LvPixelFormat\_RGB10\_Planar** RGB 10-bit planar. Defined as (LV\_PIX\_COLOR | LV\_PIX\_OCCUPY48BIT | 0x0022).
- LvPixelFormat\_RGB12\_Planar** RGB 12-bit planar. Defined as (LV\_PIX\_COLOR | LV\_PIX\_OCCUPY48BIT | 0x0023).
- LvPixelFormat\_RGB16\_Planar** RGB 16-bit planar. Defined as (LV\_PIX\_COLOR | LV\_PIX\_OCCUPY48BIT | 0x0024).
- LvPixelFormat\_BGR555P** RGB 15-bit packed (3x5 bits). Defined as (LV\_PIX\_COLOR | LV\_PIX\_OCCUP←Y16BIT | 0x00E1). Not a standard GenICam format, used only in the image processing library.

### 5.30.2.52 enum LvPowerSwitchBoundADC

Enum values for the [LvDevice\\_LvPowerSwitchBoundADC](#) feature.

Enumerator

- LvPowerSwitchBoundADC\_None** Binds no external ADC to the power switch
- LvPowerSwitchBoundADC\_ExternalADC1** Binds external ADC 1 to the power switch
- LvPowerSwitchBoundADC\_ExternalADC2** Binds external ADC 2 to the power switch
- LvPowerSwitchBoundADC\_ExternalADC3** Binds external ADC 3 to the power switch
- LvPowerSwitchBoundADC\_ExternalADC4** Binds external ADC 4 to the power switch

### 5.30.2.53 enum LvPowerSwitchCurrentAction

Enum values for the [LvDevice\\_LvPowerSwitchCurrentAction](#) feature.

Enumerator

- LvPowerSwitchCurrentAction\_Idle** Reports that all power switches are idle
- LvPowerSwitchCurrentAction\_Pulse** Reports that a pulse command is pending
- LvPowerSwitchCurrentAction\_Calibrate** Reports that a calibration is pending
- LvPowerSwitchCurrentAction\_AdjustPosition** Reports that a position adjustment is pending
- LvPowerSwitchCurrentAction\_Drive** Reports that a power switch drive operation is pending

## 5.30.2.54 enum LvPowerSwitchDrive

Enum values for the [LvDevice\\_LvPowerSwitchDrive](#) feature.

Enumerator

- LvPowerSwitchDrive\_Off** Switches the selected power switch off.
- LvPowerSwitchDrive\_Plus** Switches the selected power switch to plus polarity.
- LvPowerSwitchDrive\_Minus** Switches the selected power switch to minus polarity.

## 5.30.2.55 enum LvPowerSwitchDriveAll

Enum values for the [LvPowerSwitchDriveAll](#) feature.

Enumerator

- LvPowerSwitchDriveAll\_Off** Switches the active power switches off
- LvPowerSwitchDriveAll\_Plus** Switches the active power switches to plus polarity
- LvPowerSwitchDriveAll\_Minus** Switches the active power switches to minus polarity

## 5.30.2.56 enum LvPowerSwitchSelector

Enum values for the [LvDevice\\_LvPowerSwitchSelector](#) feature.

Enumerator

- LvPowerSwitchSelector\_PowerSwitch1** Selects power switch 1 for configuration.
- LvPowerSwitchSelector\_PowerSwitch2** Selects power switch 2 for configuration.
- LvPowerSwitchSelector\_PowerSwitch3** Selects power switch 3 for configuration.
- LvPowerSwitchSelector\_PowerSwitch4** Selects power switch 4 for configuration.

## 5.30.2.57 enum LvRegionSelector

Enum values for the [LvDevice\\_RegionSelector](#) feature.

Enumerator

- LvRegionSelector\_Region0** Selects region 0 for configuration.
- LvRegionSelector\_Region1** Selects region 1 for configuration.
- LvRegionSelector\_Region2** Selects region 2 for configuration.
- LvRegionSelector\_Region3** Selects region 3 for configuration.

## 5.30.2.58 enum LvRenderType

Enum values for the [LvRenderer\\_LvRenderType](#) feature.

Enumerator

- LvRenderType\_FullSize** Renders the acquired image in full size.
- LvRenderType\_ScaleToFit** Renders the acquired image to fit into the window.
- LvRenderType\_ScaleToSize** Renders the acquired image scaled to required size.
- LvRenderType\_ScaleToTiles** Renders the acquired images in tiles.

#### 5.30.2.59 enum LvSerialPortBaudRate

Enum values for the [LvDevice\\_LvSerialPortBaudRate](#) feature.

##### Enumerator

- LvSerialPortBaudRate\_Baud2400*** Baud rate of 2400 bauds.
- LvSerialPortBaudRate\_Baud4800*** Baud rate of 4800 bauds.
- LvSerialPortBaudRate\_Baud9600*** Baud rate of 9600 bauds.
- LvSerialPortBaudRate\_Baud14400*** Baud rate of 14400 bauds.
- LvSerialPortBaudRate\_Baud19200*** Baud rate of 19200 bauds.
- LvSerialPortBaudRate\_Baud38400*** Baud rate of 38400 bauds.
- LvSerialPortBaudRate\_Baud57600*** Baud rate of 57600 bauds.
- LvSerialPortBaudRate\_Baud115200*** Baud rate of 115200 bauds.

#### 5.30.2.60 enum LvSerialPortCommandStatus

Enum values for the [LvDevice\\_LvSerialPortCommandStatus](#) feature.

##### Enumerator

- LvSerialPortCommandStatus\_Success*** Last command was successfully transferred.
- LvSerialPortCommandStatus\_Timeout*** Last command ended with timeout (depending on configuration this might be problem or not).
- LvSerialPortCommandStatus\_PortBusy*** Last command failed: port busy.
- LvSerialPortCommandStatus\_CommunicationError*** Last command failed: generic communication error.
- LvSerialPortCommandStatus\_FrameError*** Last command failed: frame error.
- LvSerialPortCommandStatus\_ParityError*** Last command failed: parity error.
- LvSerialPortCommandStatus\_Overflow*** Last command failed: overflow.

#### 5.30.2.61 enum LvSerialPortDataBits

Enum values for the [LvDevice\\_LvSerialPortDataBits](#) feature.

##### Enumerator

- LvSerialPortDataBits\_DataBits7*** 7 data bits per character.
- LvSerialPortDataBits\_DataBits8*** 8 data bits per character.

#### 5.30.2.62 enum LvSerialPortParity

Enum values for the [LvDevice\\_LvSerialPortParity](#) feature.

##### Enumerator

- LvSerialPortParity\_None*** Parity method 'none', parity bit not used.
- LvSerialPortParity\_Odd*** Parity method 'odd', odd number of set bits in each character.
- LvSerialPortParity\_Even*** Parity method 'even', even number of set bits in each character.

## 5.30.2.63 enum LvSerialPortStopBits

Enum values for the [LvDevice\\_LvSerialPortStopBits](#) feature.

## Enumerator

**LvSerialPortStopBits\_StopBits1** 1 stop bit per character.

**LvSerialPortStopBits\_StopBits1dot5** 1.5 stop bit per character.

**LvSerialPortStopBits\_StopBits2** 2 stop bits per character.

## 5.30.2.64 enum LvSpecialPurposeTriggerActivation

Enum values for the [LvDevice\\_LvSpecialPurposeTriggerActivation](#) feature.

## Enumerator

**LvSpecialPurposeTriggerActivation\_RisingEdge** Selects the trigger signal's rising edge as active.

**LvSpecialPurposeTriggerActivation\_FallingEdge** Selects the trigger signal's falling edge as active

## 5.30.2.65 enum LvSpecialPurposeTriggerSelector

Enum values for the [LvDevice\\_LvSpecialPurposeTriggerSelector](#) feature.

## Enumerator

**LvSpecialPurposeTriggerSelector\_ImageStampsReset** Timestamps reset trigger - controls reset of timestamp, frame ID and other image stamps.

## 5.30.2.66 enum LvSpecialPurposeTriggerSource

Enum values for the [LvDevice\\_LvSpecialPurposeTriggerSource](#) feature.

## Enumerator

**LvSpecialPurposeTriggerSource\_Off** Sets trigger source off - it can be still be issued by an explicit software trigger

**LvSpecialPurposeTriggerSource\_Line1** Sets the signal source for the selected trigger to line 1 (optocoupler input).

**LvSpecialPurposeTriggerSource\_Line2** Sets the signal source for the selected trigger to line 2 (optocoupler input).

**LvSpecialPurposeTriggerSource\_Line3** Sets the signal source for the selected trigger to line 3 (optocoupler input).

**LvSpecialPurposeTriggerSource\_Line4** Sets the signal source for the selected trigger to line 4 (optocoupler input).

**LvSpecialPurposeTriggerSource\_Line5** Sets the signal source for the selected trigger to line 5.

**LvSpecialPurposeTriggerSource\_Line6** Sets the signal source for the selected trigger to line 6.

**LvSpecialPurposeTriggerSource\_Line7** Sets the signal source for the selected trigger to line 7.

**LvSpecialPurposeTriggerSource\_Line8** Sets the signal source for the selected trigger to line 8.

**LvSpecialPurposeTriggerSource\_Line17** Sets the signal source for the selected trigger to line 17 (TT↔L input).

**LvSpecialPurposeTriggerSource\_Line18** Sets the signal source for the selected trigger to line 18 (TT↔L input).

- LvSpecialPurposeTriggerSource\_Line19*** Sets the signal source for the selected trigger to line 19.
- LvSpecialPurposeTriggerSource\_Line20*** Sets the signal source for the selected trigger to line 20.
- LvSpecialPurposeTriggerSource\_Line21*** Sets the signal source for the selected trigger to line 21.
- LvSpecialPurposeTriggerSource\_Line22*** Sets the signal source for the selected trigger to line 22.
- LvSpecialPurposeTriggerSource\_Line23*** Sets the signal source for the selected trigger to line 23.
- LvSpecialPurposeTriggerSource\_Line24*** Sets the signal source for the selected trigger to line 24.
- LvSpecialPurposeTriggerSource\_Action1*** Sets the signal source for the selected trigger to action signal 1.
- LvSpecialPurposeTriggerSource\_Action2*** Sets the signal source for the selected trigger to action signal 2.
- LvSpecialPurposeTriggerSource\_Action3*** Sets the signal source for the selected trigger to action signal 3.
- LvSpecialPurposeTriggerSource\_Action4*** Sets the signal source for the selected trigger to action signal 4.
- LvSpecialPurposeTriggerSource\_Action5*** Sets the signal source for the selected trigger to action signal 5.
- LvSpecialPurposeTriggerSource\_Action6*** Sets the signal source for the selected trigger to action signal 6.
- LvSpecialPurposeTriggerSource\_Action7*** Sets the signal source for the selected trigger to action signal 7.
- LvSpecialPurposeTriggerSource\_Action8*** Sets the signal source for the selected trigger to action signal 8.

#### 5.30.2.67 enum LvStreamAcquisitionModeSelector

Enum values for the [LvStream\\_StreamAcquisitionModeSelector](#) feature.

Enumerator

***LvStreamAcquisitionModeSelector\_Default*** Default acquisition mode.

#### 5.30.2.68 enum LvStreamType

Enum values for the [LvStream\\_StreamType](#) feature.

Enumerator

***LvStreamType\_Custom*** Stream belonging to a custom technology device.

***LvStreamType\_GEV*** Stream belonging to a GigE Vision compatible device.

***LvStreamType\_U3V*** Stream belonging to a USB3 Vision compatible device.

#### 5.30.2.69 enum LvStrobeDropMode

Enum values for the [LvDevice\\_LvStrobeDropMode](#) feature.

Enumerator

***LvStrobeDropMode\_DropStrobe*** Strobe drop mode 'drop' - the strobe is dropped, image is acquired without the strobe.

***LvStrobeDropMode\_DelayFrame*** Strobe drop mode 'delay' - the frame acquisition is delayed, until the strobe can be issued.



5.30.2.70 enum `LvStrobeDurationMode`

Enum values for the `LvDevice_LvStrobeDurationMode` feature.

## Enumerator

**`LvStrobeDurationMode_FrameRateRelated`** The maximum strobe duration depends on the maximum frame rate of the camera. For very fast sensors the max. strobe duration time, dependent on the specification of the LEDs used, cannot be applied in full length, as the recovery time may become too short. The calculation is done automatically depending on the LEDs used and the max. frame rate of the camera in its actual mode of operation. Such calculation also includes boosted frame rates e.g. when the camera is in partial scanning and/or binning mode.

**`LvStrobeDurationMode_Free`** The maximum strobe duration depends on the maximum allowed ON-time and the minimum required recovery time of the LEDs used. The user can program the strobe duration free, according to his request, but must be aware himself about the relation of strobe ON-time and recovery time. An automatic protection circuit in HW drops a strobe in case the proper relation of ON-time to recovery time is not guaranteed. In such case a related error code is returned by the SW (frame message or otherwise)

5.30.2.71 enum `LvStrobeEnable`

Enum values for the `LvDevice_LvStrobeEnable` feature.

## Enumerator

**`LvStrobeEnable_Off`** Switches the strobe off.

**`LvStrobeEnable_AllClusters`** Switches on all LED clusters of the strobe light. On strobe lights possessing just a single LED cluster this cluster is switched on.

**`LvStrobeEnable_LEDCluster1`** Switches on LED cluster 1 only. The strobe will use just the LEDs in this cluster.

**`LvStrobeEnable_LEDCluster2`** Switches on LED cluster 2 only. The strobe will use just the LEDs in this cluster.

5.30.2.72 enum `LvTimerSelector`

Enum values for the `LvDevice_TimerSelector` feature.

## Enumerator

**`LvTimerSelector_Timer1`** Selects timer 1 for configuration.

**`LvTimerSelector_Timer2`** Selects timer 2 for configuration.

**`LvTimerSelector_Timer3`** Selects timer 3 for configuration.

**`LvTimerSelector_Timer4`** Selects timer 4 for configuration.

5.30.2.73 enum `LvTimerTriggerSource`

Enum values for the `LvDevice_TimerTriggerSource` feature.

## Enumerator

**`LvTimerTriggerSource_Off`** Switches timer trigger signal off - no signal will be firing the timer

**`LvTimerTriggerSource_FrameTrigger`** Switches timer trigger signal to frame trigger - activation of the frame trigger internal signal (before counting down eventual trigger delay) activates the timer.

- LvTimerTriggerSource\_Counter1End** Switches timer trigger signal to counter 1 end - expiration of counter 1 activates the timer.
- LvTimerTriggerSource\_Counter2End** Switches timer trigger signal to counter 2 end - expiration of counter 2 activates the timer.
- LvTimerTriggerSource\_Counter3End** Switches timer trigger signal to counter 3 end - expiration of counter 3 activates the timer.
- LvTimerTriggerSource\_Counter4End** Switches timer trigger signal to counter 4 end - expiration of counter 4 activates the timer.
- LvTimerTriggerSource\_UserOutput1** Switches timer trigger signal to user output 1 - activation of user output 1 activates the timer.
- LvTimerTriggerSource\_UserOutput2** Switches timer trigger signal to user output 2 - activation of user output 2 activates the timer.
- LvTimerTriggerSource\_UserOutput3** Switches timer trigger signal to user output 3 - activation of user output 3 activates the timer.
- LvTimerTriggerSource\_UserOutput4** Switches timer trigger signal to user output 4 - activation of user output 4 activates the timer.
- LvTimerTriggerSource\_UserOutput5** Switches timer trigger signal to user output 5 - activation of user output 5 activates the timer.
- LvTimerTriggerSource\_UserOutput6** Switches timer trigger signal to user output 6 - activation of user output 6 activates the timer.
- LvTimerTriggerSource\_UserOutput7** Switches timer trigger signal to user output 7 - activation of user output 7 activates the timer.
- LvTimerTriggerSource\_UserOutput8** Switches timer trigger signal to user output 8 - activation of user output 8 activates the timer.

#### 5.30.2.74 enum LvTLType

Enum values for the [LvSystem\\_TLType](#) feature.

Enumerator

- LvTLType\_Mixed** GenTL producer supporting mixed technologies.
- LvTLType\_Custom** GenTL producer supporting a custom technology devices.
- LvTLType\_GEV** GenTL producer supporting GigE Vision devices.

#### 5.30.2.75 enum LvTriggerActivation

Enum values for the [LvDevice\\_TriggerActivation](#) feature.

Enumerator

- LvTriggerActivation\_RisingEdge** Selects the trigger signal's rising edge as active.
- LvTriggerActivation\_FallingEdge** Selects the trigger signal's falling edge as active
- LvTriggerActivation\_AnyEdge** Selects the trigger signal's any edge as active
- LvTriggerActivation\_LevelHigh** Selects the trigger signal's high level as active
- LvTriggerActivation\_LevelLow** Selects the trigger signal's low level as active

## 5.30.2.76 enum LvTriggerCaching

Enum values for the [LvDevice\\_LvTriggerCaching](#) feature.

## Enumerator

**LvTriggerCaching\_Cache** Trigger caching mode 'cache' - early triggers are cached and applied as soon as possible.

**LvTriggerCaching\_Drop** Trigger caching mode 'cache' - early triggers are dropped

## 5.30.2.77 enum LvTriggerMode

Enum values for the [LvDevice\\_TriggerMode](#) feature.

## Enumerator

**LvTriggerMode\_Off** Trigger mode off - disables selected trigger

**LvTriggerMode\_On** Trigger mode on - enables selected trigger.

## 5.30.2.78 enum LvTriggerSelector

Enum values for the [LvDevice\\_TriggerSelector](#) feature.

## Enumerator

**LvTriggerSelector\_FrameStart** Frame start trigger - controls a new frame acquisition.

**LvTriggerSelector\_FrameBurstStart** Frame burst start trigger - Selects a trigger starting the capture of the bursts of frames.

**LvTriggerSelector\_LineStart** Line start trigger - Selects a trigger starting the capture of one Line of a Frame.

**LvTriggerSelector\_VirtualFrameActive** Virtual frame active trigger - Selects a trigger activating the capture of virtual Frames.

## 5.30.2.79 enum LvTriggerSource

Enum values for the [LvDevice\\_TriggerSource](#) feature.

## Enumerator

**LvTriggerSource\_Line1** Sets the signal source for the selected trigger to line 1 (optocoupler input).

**LvTriggerSource\_Line2** Sets the signal source for the selected trigger to line 2 (optocoupler input).

**LvTriggerSource\_Line3** Sets the signal source for the selected trigger to line 3 (optocoupler input).

**LvTriggerSource\_Line4** Sets the signal source for the selected trigger to line 4 (optocoupler input).

**LvTriggerSource\_Line5** Sets the signal source for the selected trigger to line 5.

**LvTriggerSource\_Line6** Sets the signal source for the selected trigger to line 6.

**LvTriggerSource\_Line7** Sets the signal source for the selected trigger to line 7.

**LvTriggerSource\_Line8** Sets the signal source for the selected trigger to line 8.

**LvTriggerSource\_Line17** Sets the signal source for the selected trigger to line 17 (TTL input).

**LvTriggerSource\_Line18** Sets the signal source for the selected trigger to line 18 (TTL input).

**LvTriggerSource\_Line19** Sets the signal source for the selected trigger to line 19.

**LvTriggerSource\_Line20** Sets the signal source for the selected trigger to line 20.

- LvTriggerSource\_Line21*** Sets the signal source for the selected trigger to line 21.
- LvTriggerSource\_Line22*** Sets the signal source for the selected trigger to line 22.
- LvTriggerSource\_Line23*** Sets the signal source for the selected trigger to line 23.
- LvTriggerSource\_Line24*** Sets the signal source for the selected trigger to line 24.
- LvTriggerSource\_Software*** Sets the signal source for the selected trigger to software.
- LvTriggerSource\_Action1*** Sets the signal source for the selected trigger to action signal 1.
- LvTriggerSource\_Action2*** Sets the signal source for the selected trigger to action signal 2.
- LvTriggerSource\_Action3*** Sets the signal source for the selected trigger to action signal 3.
- LvTriggerSource\_Action4*** Sets the signal source for the selected trigger to action signal 4.
- LvTriggerSource\_Action5*** Sets the signal source for the selected trigger to action signal 5.
- LvTriggerSource\_Action6*** Sets the signal source for the selected trigger to action signal 6.
- LvTriggerSource\_Action7*** Sets the signal source for the selected trigger to action signal 7.
- LvTriggerSource\_Action8*** Sets the signal source for the selected trigger to action signal 8.
- LvTriggerSource\_Quad*** Sets the signal source for the selected trigger to quadrature decoder.
- LvTriggerSource\_Counter1*** Sets the signal source for the selected trigger to counter 1.
- LvTriggerSource\_Counter2*** Sets the signal source for the selected trigger to counter 2.
- LvTriggerSource\_Counter3*** Sets the signal source for the selected trigger to counter 3.
- LvTriggerSource\_Counter4*** Sets the signal source for the selected trigger to counter 4.
- LvTriggerSource\_Timer1*** Sets the signal source for the selected trigger to timer 1.
- LvTriggerSource\_Timer2*** Sets the signal source for the selected trigger to timer 2.
- LvTriggerSource\_Timer3*** Sets the signal source for the selected trigger to timer 3.
- LvTriggerSource\_Timer4*** Sets the signal source for the selected trigger to timer 4.
- LvTriggerSource\_UserOutput1*** Sets the signal source for the selected trigger to user output 1.
- LvTriggerSource\_UserOutput2*** Sets the signal source for the selected trigger to user output 2.
- LvTriggerSource\_UserOutput3*** Sets the signal source for the selected trigger to user output 3.
- LvTriggerSource\_UserOutput4*** Sets the signal source for the selected trigger to user output 4.
- LvTriggerSource\_UserOutput5*** Sets the signal source for the selected trigger to user output 5.
- LvTriggerSource\_UserOutput6*** Sets the signal source for the selected trigger to user output 6.
- LvTriggerSource\_UserOutput7*** Sets the signal source for the selected trigger to user output 7.
- LvTriggerSource\_UserOutput8*** Sets the signal source for the selected trigger to user output 8.

#### 5.30.2.80 enum LvUniBalanceRatioSelector

Enum values for the [LvDevice\\_LvUniBalanceRatioSelector](#) feature.

Enumerator

- LvUniBalanceRatioSelector\_Red*** Selects the red channel for configuration.
- LvUniBalanceRatioSelector\_Green*** Selects the green channel for configuration.
- LvUniBalanceRatioSelector\_Blue*** Selects the blue channel for configuration.

#### 5.30.2.81 enum LvUniBalanceWhiteAuto

Enum values for the [LvDevice\\_LvUniBalanceWhiteAuto](#) feature.

Enumerator

- LvUniBalanceWhiteAuto\_Off*** Automatic white balance mode off - the automatic white balance is not applied.
- LvUniBalanceWhiteAuto\_Once*** Automatic white balance mode once - the white balance factors are once adjusted, then switches back to off.

5.30.2.82 enum `LvUniColorTransformationMode`

Enum values for the `LvDevice_LvUniColorTransformationMode` feature.

## Enumerator

**`LvUniColorTransformationMode_Direct`** In this mode the Color Transformation matrix can be controlled directly.

**`LvUniColorTransformationMode_Generated`** In this mode the Color Transformation matrix is set through the higher level features, such as the Saturation.

5.30.2.83 enum `LvUniColorTransformationSelector`

Enum values for the `LvDevice_LvUniColorTransformationSelector` feature.

## Enumerator

**`LvUniColorTransformationSelector_RGBtoRGB`** RGB to RGB matrix transformation - currently the only Color Transformation matrix type.

5.30.2.84 enum `LvUniColorTransformationValueSelector`

Enum values for the `LvDevice_LvUniColorTransformationValueSelector` feature.

## Enumerator

**`LvUniColorTransformationValueSelector_Gain00`** Selects the gain 00 (RR, red-red) entry of the color transformation matrix.

**`LvUniColorTransformationValueSelector_Gain01`** Selects the gain 01 (RG, red-green) entry of the color transformation matrix.

**`LvUniColorTransformationValueSelector_Gain02`** Selects the gain 02 (RB, red-blue) entry of the color transformation matrix.

**`LvUniColorTransformationValueSelector_Gain10`** Selects the gain 10 (GR, green-red) entry of the color transformation matrix.

**`LvUniColorTransformationValueSelector_Gain11`** Selects the gain 11 (GG, green-green) entry of the color transformation matrix.

**`LvUniColorTransformationValueSelector_Gain12`** Selects the gain 12 (GB, green-blue) entry of the color transformation matrix.

**`LvUniColorTransformationValueSelector_Gain20`** Selects the gain 20 (BR, blue-red) entry of the color transformation matrix.

**`LvUniColorTransformationValueSelector_Gain21`** Selects the gain 21 (BG, blue-green) entry of the color transformation matrix.

**`LvUniColorTransformationValueSelector_Gain22`** Selects the gain 22 (BB, blue-blue) entry of the color transformation matrix.

5.30.2.85 enum `LvUniLUTMode`

Enum values for the `LvDevice_LvUniLUTMode` feature.

## Enumerator

**`LvUniLUTMode_Direct`** In this mode the LUT is controlled directly.

**`LvUniLUTMode_Generated`** In this mode the LUT is controlled through the higher level features, such as brightness, contrast, gamma or white balance.

### 5.30.2.86 enum LvUniLUTSelector

Enum values for the [LvDevice\\_LvUniLUTSelector](#) feature.

Enumerator

**LvUniLUTSelector\_Luminance** Selects the luminance LUT for configuration.

**LvUniLUTSelector\_Red** Selects the red LUT for configuration.

**LvUniLUTSelector\_Green** Selects the green LUT for configuration.

**LvUniLUTSelector\_Blue** Selects the blue LUT for configuration.

### 5.30.2.87 enum LvUniProcessExecution

Enum values for the [LvDevice\\_LvUniProcessExecution](#) feature.

Enumerator

**LvUniProcessExecution\_OnBufferPtrQuery** The SW image processing is delayed to the time the application asks for the [LvBuffer\\_UniBase](#) or [LvBuffer\\_ProcessBase](#) pointer or for the [LvIplmgInfo](#) data. This enables to the application to skip the processing in case it is not needed. If this is queried several times for the same image, the processing is done only once.

**LvUniProcessExecution\_OnPopFromQueue** The SW image processing is done always - at the moment the buffer is popped from the output buffer queue, before delivering it to the application.

**LvUniProcessExecution\_OnExplicitRequest** The SW processing is not done automatically, but must be explicitly done by the [ExecProcess](#) command of the Buffer.

### 5.30.2.88 enum LvUniProcessMode

Enum values for the [LvDevice\\_LvUniProcessMode](#) feature.

Enumerator

**LvUniProcessMode\_HwOnly** HwOnly - The processing is done only in case it is available directly on the hardware (device). The images will be delivered to the output buffer queue already processed.

**LvUniProcessMode\_SwOnly** SwOnly - The processing will be done by software even if the hardware could support the operation. The software processing is done when the buffer is passed to the output buffer queue (or later - see [LvUniProcessExecution](#)).

**LvUniProcessMode\_Auto** Auto - The processing will be done by hardware and by software will be processed only the part, which is not possible to do on hardware. Note that if the Bayer decoding is done by software (this happens when you select an undecoded Bayer pixel format as the device [PixelFormat](#)), the LUT must be then also done by software, even if it is available in hardware; that's because it must be applied after the Bayer decoding.

**LvUniProcessMode\_Off** Off - The automatic processing is not available. You can use the HW features (LUT etc.) directly.

### 5.30.2.89 enum LvUserOutputSelector

Enum values for the [LvDevice\\_UserOutputSelector](#) feature.

Enumerator

**LvUserOutputSelector\_UserOutput1** Selects user output 1.

**LvUserOutputSelector\_UserOutput2** Selects user output 2.

**LvUserOutputSelector\_UserOutput3** Selects user output 3.

- LvUserOutputSelector\_UserOutput4*** Selects user output 4.
- LvUserOutputSelector\_UserOutput5*** Selects user output 5.
- LvUserOutputSelector\_UserOutput6*** Selects user output 6.
- LvUserOutputSelector\_UserOutput7*** Selects user output 7.
- LvUserOutputSelector\_UserOutput8*** Selects user output 8.

#### 5.30.2.90 enum LvUserSetDefault

Enum values for the [LvDevice\\_UserSetDefault](#) feature.

##### Enumerator

- LvUserSetDefault\_Default*** Selects the default user set as the default startup set.
- LvUserSetDefault\_UserSet1*** Selects user set 1 as the default startup set.
- LvUserSetDefault\_UserSet2*** Selects user set 2 as the default startup set.
- LvUserSetDefault\_UserSet3*** Selects user set 3 as the default startup set.
- LvUserSetDefault\_UserSet4*** Selects user set 4 as the default startup set.
- LvUserSetDefault\_None*** When resetting/connecting the camera, no user set is applied, the last camera configuration remains. During camera boot, the default user set is applied.

#### 5.30.2.91 enum LvUserSetDefaultSelector

Enum values for the [LvDevice\\_UserSetDefaultSelector](#) feature.

##### Enumerator

- LvUserSetDefaultSelector\_Default*** Selects the default user set as the default startup set.
- LvUserSetDefaultSelector\_UserSet1*** Selects user set 1 as the default startup set.
- LvUserSetDefaultSelector\_UserSet2*** Selects user set 2 as the default startup set.
- LvUserSetDefaultSelector\_UserSet3*** Selects user set 3 as the default startup set.
- LvUserSetDefaultSelector\_UserSet4*** Selects user set 4 as the default startup set.
- LvUserSetDefaultSelector\_None*** When resetting/connecting the camera, no user set is applied, the last camera configuration remains. During camera boot, the default user set is applied.

#### 5.30.2.92 enum LvUserSetSelector

Enum values for the [LvDevice\\_UserSetSelector](#) feature.

##### Enumerator

- LvUserSetSelector\_Default*** Selects the default configuration set.
- LvUserSetSelector\_UserSet1*** Selects user set 1.
- LvUserSetSelector\_UserSet2*** Selects user set 2.
- LvUserSetSelector\_UserSet3*** Selects user set 3.
- LvUserSetSelector\_UserSet4*** Selects user set 4.

## 5.31 LvStreamStart() flags definitions

### Macros

- `#define LvStreamStartFlags_Default`

#### 5.31.1 Detailed Description

#### 5.31.2 Macro Definition Documentation

##### 5.31.2.1 `#define LvStreamStartFlags_Default`

Default stream start flag



## 5.32 LvStreamStop() flags definitions

### Macros

- `#define LvStreamStopFlags_Default`
- `#define LvStreamStopFlags_Kill`

#### 5.32.1 Detailed Description

#### 5.32.2 Macro Definition Documentation

##### 5.32.2.1 `#define LvStreamStopFlags_Default`

Stop the acquisition engine when the currently running tasks like filling a buffer are completed. This is the default.

##### 5.32.2.2 `#define LvStreamStopFlags_Kill`

Stop the acquisition engine immediately and leave buffers currently being filled in the Input Buffer Pool.

## 5.33 LvDeviceUniSetLut() and LvDeviceUniGetLut() flags definitions

### Macros

- `#define LvUniLutFlags_HwLut`

#### 5.33.1 Detailed Description

#### 5.33.2 Macro Definition Documentation

##### 5.33.2.1 `#define LvUniLutFlags_HwLut`

If present, the operation is done directly on HW LUT, passing the UniProcess mechanism.

## 5.34 LvSaveFlag definitions

### Macros

- `#define LvSaveFlag_RemoteFtr`
- `#define LvSaveFlag_LocalFtr`
- `#define LvSaveFlag_GenTIFtr`
- `#define LvSaveFlag_All`
- `#define LvSaveFlag_IgnoreVersion`
- `#define LvSaveFlag_IgnoreModel`

### 5.34.1 Detailed Description

### 5.34.2 Macro Definition Documentation

#### 5.34.2.1 `#define LvSaveFlag_All`

Save/load device all features (combines all flags above).

#### 5.34.2.2 `#define LvSaveFlag_GenTIFtr`

Save/load device GenTL XML features.

#### 5.34.2.3 `#define LvSaveFlag_IgnoreModel`

If specified, the remote device model check is not done when reading the file - the file is read even if it was created by different device model (this may lead to errors by some features).

#### 5.34.2.4 `#define LvSaveFlag_IgnoreVersion`

If specified, the remote device FW version check is not done when reading the file - the file is read even if it was created by device with a different FW version (this may lead to errors by some features).

#### 5.34.2.5 `#define LvSaveFlag_LocalFtr`

Save/load device local XML features.

#### 5.34.2.6 `#define LvSaveFlag_RemoteFtr`

Save/load device remote XML features.

## 5.35 LvPixelFormat definitions

### Macros

- `#define LV_PIX_MONO`
- `#define LV_PIX_COLOR`
- `#define LV_PIX_CUSTOM`
- `#define LV_PIX_COLOR_MASK`
- `#define LV_PIX_OCCUPY8BIT`
- `#define LV_PIX_OCCUPY12BIT`
- `#define LV_PIX_OCCUPY16BIT`
- `#define LV_PIX_OCCUPY24BIT`
- `#define LV_PIX_OCCUPY32BIT`
- `#define LV_PIX_OCCUPY36BIT`
- `#define LV_PIX_OCCUPY48BIT`
- `#define LV_PIX_EFFECTIVE_PIXEL_SIZE_MASK`
- `#define LV_PIX_EFFECTIVE_PIXEL_SIZE_SHIFT`
- `#define LvPixelFormat_Mono8Signed`
- `#define LvPixelFormat_RGB8Packed`
- `#define LvPixelFormat_BGR8Packed`
- `#define LvPixelFormat_RGBA8Packed`
- `#define LvPixelFormat_BGRA8Packed`
- `#define LvPixelFormat_RGB10Packed`
- `#define LvPixelFormat_BGR10Packed`
- `#define LvPixelFormat_RGB12Packed`
- `#define LvPixelFormat_BGR12Packed`
- `#define LvPixelFormat_RGB16Packed`
- `#define LvPixelFormat_RGB10V2Packed`
- `#define LvPixelFormat_RGB565Packed`
- `#define LvPixelFormat_BGR565Packed`
- `#define LvPixelFormat_YUV411Packed`
- `#define LvPixelFormat_YUV422Packed`
- `#define LvPixelFormat_YUV422UYVYPacked`
- `#define LvPixelFormat_YUV444Packed`
- `#define LvPixelFormat_RGB8Planar`
- `#define LvPixelFormat_RGB10Planar`
- `#define LvPixelFormat_RGB12Planar`
- `#define LvPixelFormat_RGB16Planar`
- `#define LvPixelFormat_Mono8s`
- `#define LvPixelFormat_RGBa8`
- `#define LvPixelFormat_BGRa8`
- `#define LvPixelFormat_RGB565p`
- `#define LvPixelFormat_BGR565p`
- `#define LvPixelFormat_RGB10p32`
- `#define LvPixelFormat_BGR555p`
- `#define LvPixelFormat_YUV411_8_UYYVYY`
- `#define LvPixelFormat_YUV8_UYV`

### 5.35.1 Detailed Description

### 5.35.2 Macro Definition Documentation

#### 5.35.2.1 `#define LV_PIX_COLOR`

PixelFormat component: The pixel format is color.

#### 5.35.2.2 #define LV\_PIX\_COLOR\_MASK

Mask for the color flag

#### 5.35.2.3 #define LV\_PIX\_CUSTOM

PixelFormat component: The pixel format is custom.

#### 5.35.2.4 #define LV\_PIX\_EFFECTIVE\_PIXEL\_SIZE\_MASK

Mask for the pixel size part.

#### 5.35.2.5 #define LV\_PIX\_EFFECTIVE\_PIXEL\_SIZE\_SHIFT

Shift for the pixel size part.

#### 5.35.2.6 #define LV\_PIX\_MONO

PixelFormat component: The pixel format is monochrome.

#### 5.35.2.7 #define LV\_PIX\_OCCUPY12BIT

PixelFormat component: One pixel occupies 12 bits.

#### 5.35.2.8 #define LV\_PIX\_OCCUPY16BIT

PixelFormat component: One pixel occupies 16 bits.

#### 5.35.2.9 #define LV\_PIX\_OCCUPY24BIT

PixelFormat component: One pixel occupies 24 bits.

#### 5.35.2.10 #define LV\_PIX\_OCCUPY32BIT

PixelFormat component: One pixel occupies 32 bits.

#### 5.35.2.11 #define LV\_PIX\_OCCUPY36BIT

PixelFormat component: One pixel occupies 36 bits.

#### 5.35.2.12 #define LV\_PIX\_OCCUPY48BIT

PixelFormat component: One pixel occupies 48 bits.

#### 5.35.2.13 #define LV\_PIX\_OCCUPY8BIT

PixelFormat component: One pixel occupies 8 bits.

5.35.2.14 `#define LvPixelFormat_BGR10Packed`

Alias for [LvPixelFormat\\_BGR10](#).

5.35.2.15 `#define LvPixelFormat_BGR12Packed`

Alias for [LvPixelFormat\\_BGR12](#).

5.35.2.16 `#define LvPixelFormat_BGR555p`

Alias for [LvPixelFormat\\_BGR555P](#).

5.35.2.17 `#define LvPixelFormat_BGR565p`

Alias for [LvPixelFormat\\_BGR565P](#).

5.35.2.18 `#define LvPixelFormat_BGR565Packed`

Alias for [LvPixelFormat\\_BGR565P](#).

5.35.2.19 `#define LvPixelFormat_BGR8Packed`

Alias for [LvPixelFormat\\_BGR8](#).

5.35.2.20 `#define LvPixelFormat_BGRa8`

Alias for [LvPixelFormat\\_BGRA8](#).

5.35.2.21 `#define LvPixelFormat_BGRA8Packed`

Alias for [LvPixelFormat\\_BGRA8](#).

5.35.2.22 `#define LvPixelFormat_Mono8s`

Alias for [LvPixelFormat\\_Mono8S](#).

5.35.2.23 `#define LvPixelFormat_Mono8Signed`

Alias for [LvPixelFormat\\_Mono8S](#).

5.35.2.24 `#define LvPixelFormat_RGB10p32`

Alias for [LvPixelFormat\\_RGB10P32](#).

5.35.2.25 `#define LvPixelFormat_RGB10Packed`

Alias for [LvPixelFormat\\_RGB10](#).

5.35.2.26 #define LvPixelFormat\_RGB10Planar

Alias for [LvPixelFormat\\_RGB10\\_Planar](#).

5.35.2.27 #define LvPixelFormat\_RGB10V2Packed

Alias for [LvPixelFormat\\_RGB10P32](#).

5.35.2.28 #define LvPixelFormat\_RGB12Packed

Alias for [LvPixelFormat\\_RGB12](#).

5.35.2.29 #define LvPixelFormat\_RGB12Planar

Alias for [LvPixelFormat\\_RGB12\\_Planar](#).

5.35.2.30 #define LvPixelFormat\_RGB16Packed

Alias for [LvPixelFormat\\_RGB16](#).

5.35.2.31 #define LvPixelFormat\_RGB16Planar

Alias for [LvPixelFormat\\_RGB16\\_Planar](#).

5.35.2.32 #define LvPixelFormat\_RGB565p

Alias for [LvPixelFormat\\_RGB565P](#).

5.35.2.33 #define LvPixelFormat\_RGB565Packed

Alias for [LvPixelFormat\\_RGB565P](#).

5.35.2.34 #define LvPixelFormat\_RGB8Packed

Alias for [LvPixelFormat\\_RGB8](#).

5.35.2.35 #define LvPixelFormat\_RGB8Planar

Alias for [LvPixelFormat\\_RGB8\\_Planar](#).

5.35.2.36 #define LvPixelFormat\_RGBa8

Alias for [LvPixelFormat\\_RGBA8](#).

5.35.2.37 #define LvPixelFormat\_RGBA8Packed

Alias for [LvPixelFormat\\_RGBA8](#).

5.35.2.38 `#define LvPixelFormat_YUV411_8_UYVY`

Alias for [LvPixelFormat\\_YUV411\\_8](#).

5.35.2.39 `#define LvPixelFormat_YUV411Packed`

Alias for [LvPixelFormat\\_YUV411\\_8](#).

5.35.2.40 `#define LvPixelFormat_YUV422Packed`

Alias for [LvPixelFormat\\_YUV422\\_8\\_UYVY](#).

5.35.2.41 `#define LvPixelFormat_YUV422UYVPacked`

Alias for [LvPixelFormat\\_YUV422\\_8](#).

5.35.2.42 `#define LvPixelFormat_YUV444Packed`

Alias for [LvPixelFormat\\_YUV8](#).

5.35.2.43 `#define LvPixelFormat_YUV8_UYV`

Alias for [LvPixelFormat\\_YUV8](#).



## 5.36 SynView Image Processing Library functions

### Modules

- [Common functions](#)
- [Image initialization functions](#)
- [Region of Interest \(ROI\) functions](#)
- [Lookup Table \(LUT\) functions](#)
- [Bayer decoding/encoding functions](#)
- [Rotation and line manipulation functions](#)
- [Pixel format conversion functions](#)
- [Saving/loading functions](#)
- [Overlay functions](#)
- [RGB color correction and convolution functions](#)
- [Shading correction functions](#)

### 5.36.1 Detailed Description

## 5.37 Common functions

### Functions

- LV\_EXTC LV\_DLLIMPORT void [LvipGetStatusMsg](#) ([LvStatus](#) TlStatus, char \*pMsg, size\_t MsgBufSize)

#### 5.37.1 Detailed Description

#### 5.37.2 Function Documentation

##### 5.37.2.1 LV\_EXTC LV\_DLLIMPORT void LvipGetStatusMsg ( LvStatus TlStatus, char \* pMsg, size\_t MsgBufSize )

Retrieves a text describing the status.

#### Parameters

|                   |                                          |
|-------------------|------------------------------------------|
| <i>TlStatus</i>   | Error status code.                       |
| <i>pMsg</i>       | Pointer to buffer for the error message. |
| <i>MsgBufSize</i> | Size of the buffer.                      |

## 5.38 Image initialization functions

### Functions

- LV\_EXTC LV\_DLLIMPORT [LvStatus LvIpInitImgInfo](#) ([LvIpImgInfo](#) \*pImgInfo, uint32\_t Width, uint32\_t Height, uint32\_t PixelFormat, uint32\_t Attributes)
- LV\_EXTC LV\_DLLIMPORT uint32\_t [LvIpGetImageDataSize](#) ([LvIpImgInfo](#) \*pImgInfo)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvIpAllocateImageData](#) ([LvIpImgInfo](#) \*pImgInfo)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvIpDeallocateImageData](#) ([LvIpImgInfo](#) \*pImgInfo)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvIpFillWithColor](#) ([LvIpImgInfo](#) \*pImgInfo, uint8\_t Red, uint8\_t Green, uint8\_t Blue, uint32\_t Options)

#### 5.38.1 Detailed Description

#### 5.38.2 Function Documentation

##### 5.38.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvIpAllocateImageData ( LvIpImgInfo \* pImgInfo )

Allocates appropriate space to pData or color planes, according to the Height and LineIncrement.

#### Parameters

|                 |                                                          |
|-----------------|----------------------------------------------------------|
| <i>pImgInfo</i> | pointer to the <a href="#">LvIpImgInfo</a> of the image. |
|-----------------|----------------------------------------------------------|

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

##### 5.38.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvIpDeallocateImageData ( LvIpImgInfo \* pImgInfo )

Deallocates the image data buffer(s) If the flags is not containing [LvIpImgAttr\\_NotDataOwner](#), deallocates pData or color planes and sets them to NULL.

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

##### 5.38.2.3 LV\_EXTC LV\_DLLIMPORT LvStatus LvIpFillWithColor ( LvIpImgInfo \* pImgInfo, uint8\_t Red, uint8\_t Green, uint8\_t Blue, uint32\_t Options )

Fills image data with specified color.

#### Parameters

|                 |                                                                                                              |
|-----------------|--------------------------------------------------------------------------------------------------------------|
| <i>pImgInfo</i> | pointer to <a href="#">LvIpImgInfo</a> structure, the data of which has to be filled with the selected color |
| <i>Red</i>      | 8bit Red value                                                                                               |
| <i>Green</i>    | 8bit Green value                                                                                             |
| <i>Blue</i>     | 8bit Blue value                                                                                              |
| <i>Options</i>  | Options reserved - should be set to 0.                                                                       |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.38.2.4 LV\_EXTC LV\_DLLIMPORT uint32\_t LvipGetImageDataSize ( LvipImgInfo \* *plmgInfo* )

Returns the data size required for the image. Expects the Height and LineIncrement are already calculated. In case of color planes returns the size of one plane

## Parameters

|                 |                                                         |
|-----------------|---------------------------------------------------------|
| <i>plmgInfo</i> | pointer to the <a href="#">LvplmgInfo</a> of the image. |
|-----------------|---------------------------------------------------------|

## Returns

The data size required for the image in bytes.

5.38.2.5 LV\_EXTC LV\_DLLIMPORT LvStatus LvplnitmgInfo ( [LvplmgInfo](#) \* *plmgInfo*, uint32\_t *Width*, uint32\_t *Height*, uint32\_t *PixelFormat*, uint32\_t *Attributes* )

Initializes the [LvplmgInfo](#) to specified values, calculates the line increment and sets pData to NULL (be sure to deallocate the image buffers if were allocated, before this function call). If pData of other owner is used, set the [LvplmgAttr\\_NotDataOwner](#) flag so that the data are not deallocated when [LvplDeallocatmgData\(\)](#) is applied to this [mgInfo](#).

## Parameters

|                    |                                                                            |
|--------------------|----------------------------------------------------------------------------|
| <i>plmgInfo</i>    | Pointer to <a href="#">LvplmgInfo</a> structure which is to be initialized |
| <i>Width</i>       | Width of image in pixels                                                   |
| <i>Height</i>      | Height of image in pixels                                                  |
| <i>PixelFormat</i> | Pixel format; one of the <a href="#">LvPixelFormat</a> .                   |
| <i>Attributes</i>  | Image attributes; OR-ed combination of the <a href="#">LvplmgAttr</a> .    |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

## 5.39 Region of Interest (ROI) functions

### Functions

- LV\_EXTC LV\_DLLIMPORT LvStatus LvipCopyArea (LvimgInfo \*pSrcimgInfo, LvimgInfo \*pDstimgInfo, int32\_t DstXOffset, int32\_t DstYOffset, uint32\_t DstWidth, uint32\_t DstHeight, uint32\_t Options)

#### 5.39.1 Detailed Description

#### 5.39.2 Function Documentation

5.39.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvipCopyArea ( LvimgInfo \* pSrcimgInfo, LvimgInfo \* pDstimgInfo, int32\_t DstXOffset, int32\_t DstYOffset, uint32\_t DstWidth, uint32\_t DstHeight, uint32\_t Options )

Extracts from the source bitmap a rectangle as destination bitmap. If the rectangle goes outside of the source image, the intersection rectangle is taken, that means the result width and/or height can be smaller than required. If the intersection is zero, the function returns returns [LVSTATUS\\_LVIP\\_DST\\_RECT\\_OUTSIDE\\_SRC](#).

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: No.

#### Parameters

|                    |                                                                                                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcimgInfo</i> | Source image info                                                                                                                                                                               |
| <i>pDstimgInfo</i> | Destination image info                                                                                                                                                                          |
| <i>DstXOffset</i>  | Left offset of the rectangle                                                                                                                                                                    |
| <i>DstYOffset</i>  | Upper offset of the rectangle                                                                                                                                                                   |
| <i>DstWidth</i>    | Width of area which has to be copied                                                                                                                                                            |
| <i>DstHeight</i>   | Height of area which has to be copied                                                                                                                                                           |
| <i>Options</i>     | Options - OR-ed combination of <a href="#">LvipOption</a> . If the LvipOption_ReallocateDst is used, then also can contain attributes from <a href="#">LvipimgAttr</a> for (re)allocated image. |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

## 5.40 Lookup Table (LUT) functions

### Functions

- LV\_EXTC LV\_DLLIMPORT LvStatus LvipAllocateLut (uint32\_t LutType)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipFreeLut (LvipHLut hLut)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipResetLut (LvipHLut hLut)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet8BitLut (LvipHLut hLut, uint8\_t \*pRed, uint8\_t \*pGreen, uint8\_t \*pBlue)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipGet8BitLut (LvipHLut hLut, uint8\_t \*pRed, uint8\_t \*pGreen, uint8\_t \*pBlue)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet10BitLut (LvipHLut hLut, uint16\_t \*pRed, uint16\_t \*pGreen, uint16\_t \*pBlue)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipGet10BitLut (LvipHLut hLut, uint16\_t \*pRed, uint16\_t \*pGreen, uint16\_t \*pBlue)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet12BitLut (LvipHLut hLut, uint16\_t \*pRed, uint16\_t \*pGreen, uint16\_t \*pBlue)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipGet12BitLut (LvipHLut hLut, uint16\_t \*pRed, uint16\_t \*pGreen, uint16\_t \*pBlue)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet8BitLutValue (LvipHLut hLut, LvEnum LutSelector, uint32\_t Index, uint8\_t Value)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipGet8BitLutValue (LvipHLut hLut, LvEnum LutSelector, uint32\_t Index, uint8\_t \*pValue)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet10BitLutValue (LvipHLut hLut, LvEnum LutSelector, uint32\_t Index, uint16\_t Value)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipGet10BitLutValue (LvipHLut hLut, LvEnum LutSelector, uint32\_t Index, uint16\_t \*pValue)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet12BitLutValue (LvipHLut hLut, LvEnum LutSelector, uint32\_t Index, uint16\_t Value)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipGet12BitLutValue (LvipHLut hLut, LvEnum LutSelector, uint32\_t Index, uint16\_t \*pValue)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipAddGammaToLut (uint32\_t Gamma, LvipHLut hLut)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipAddWbToLut (uint32\_t FactorRed, uint32\_t FactorGreen, uint32\_t FactorBlue, LvipHLut hLut)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipAddOffsetAndGainToLut (int32\_t Offset, int32\_t Gain, LvipHLut hLut)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipAddBrightnessAndContrastToLut (int32\_t Brightness, int32\_t Contrast, LvipHLut hLut)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipApplyLut (LvIpImgInfo \*pSrcImgInfo, LvIpImgInfo \*pDstImgInfo, LvipHLut hLut, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipCalcWbFactors (LvIpImgInfo \*pSrcImgInfo, uint32\_t \*pFactorRed, uint32\_t \*pFactorGreen, uint32\_t \*pFactorBlue, uint32\_t Options)

### 5.40.1 Detailed Description

### 5.40.2 Function Documentation

#### 5.40.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvipAddBrightnessAndContrastToLut ( int32\_t *Brightness*, int32\_t *Contrast*, LvipHLut *hLut* )

Adds brightness and contrast to LUT. Recalculates each value in the LUT table by adding the brightness and multiplying by contrast. This function is similar to the [LvipAddOffsetAndGainToLut\(\)](#) function, with the following 2 differences:

- The Brightness middle value is 1000, meaning no change. The Brightness 0 means black image and 2000 means fully white image, because subtracting or adding the 1000 means subtracting or adding the maximum pixel value.
- The Brightness factor is internally corrected in dependence on contrast. The Contrast is equivalent to Gain in the [LvipAddOffsetAndGainToLut\(\)](#) function. It is a factor multiplied by 1000, i.e. 1000 means 1.0 = no change. Can be also negative - 1000 makes inversion.

#### Parameters

|                   |                                                                                                       |
|-------------------|-------------------------------------------------------------------------------------------------------|
| <i>Brightness</i> | The Brightness to be added expressed in 1/1000 of the maximum pixel value. See the explanation above. |
| <i>Contrast</i>   | The Contrast factor multiplied by 1000.                                                               |
| <i>hLut</i>       | Handle to the LUT.                                                                                    |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure.

#### 5.40.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvipAddGammaToLut ( uint32\_t Gamma, LvipHLut hLut )

Adds gamma to LUT. Recalculates each value in the LUT table by applying the Gamma curve. Gamma is supplied multiplied by 1000, i.e. for gamma = 1.0 the passed value will be 1000. There is a possibility to do image lighter/darker using different gamma value. This gamma will be added to LUT and when the image is being transformed using any of function, add this LUT to this function as the last parameter.

#### Note

There is a need to have LUT - see [LvipAllocateLut\(\)](#) and its company.

#### Parameters

|              |                                                                                             |
|--------------|---------------------------------------------------------------------------------------------|
| <i>Gamma</i> | Minimal gamma value is 10 - it means that there is a need to enter gamma multiplies by 1000 |
| <i>hLut</i>  | Handle to LUT                                                                               |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.40.2.3 LV\_EXTC LV\_DLLIMPORT LvStatus LvipAddOffsetAndGainToLut ( int32\_t Offset, int32\_t Gain, LvipHLut hLut )

Adds offset and gain to LUT. Recalculates each value in the LUT table by adding the offset and multiplying by gain. The offset is in range -1000 to +1000, where 0 means no change and 1000 the maximum pixel value - adding 1000 will make the image fully white, adding -1000 will make it fully black. The offset is corresponding to Brightness - 1000, see [LvipAddBrightnessAndContrastToLut\(\)](#).

The gain is the gain factor multiplied by 1000, i.e. 1000 means 1.0 = no change. Can be also negative - 1000 makes inversion. It is equivalent to contrast.

#### Parameters

|               |                                                                                                   |
|---------------|---------------------------------------------------------------------------------------------------|
| <i>Offset</i> | The Offset to be added expressed in 1/1000 of the maximum pixel value. See the explanation above. |
|---------------|---------------------------------------------------------------------------------------------------|



|             |                                     |
|-------------|-------------------------------------|
| <i>Gain</i> | The Gain factor multiplied by 1000. |
| <i>hLut</i> | Handle to the LUT                   |

**Returns**

LVSTATUS\_OK in case of success, or error status in case of failure.

#### 5.40.2.4 LV\_EXTC LV\_DLLIMPORT LvStatus LvipAddWbToLut ( uint32\_t *FactorRed*, uint32\_t *FactorGreen*, uint32\_t *FactorBlue*, LvipHLut *hLut* )

Adds white balance to LUT. Recalculates each value in the LUT table by applying the white balance factors. The factors are supplied multiplied by 1000, i.e. for the factor = 1.0 the passed value will be 1000. See [LvipCalcWbFactors\(\)](#) for obtaining the WB factors from an image.

**Parameters**

|                    |                                                   |
|--------------------|---------------------------------------------------|
| <i>FactorRed</i>   | Red factor of white balance, multiplied by 1000   |
| <i>FactorGreen</i> | Green factor of white balance, multiplied by 1000 |
| <i>FactorBlue</i>  | Blue factor of white balance, multiplied by 1000  |
| <i>hLut</i>        | Handle to the LUT                                 |

**Returns**

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.40.2.5 LV\_EXTC LV\_DLLIMPORT LvipHLut LvipAllocateLut ( uint32\_t *LutType* )

Allocates the LUT.

**Parameters**

|                |                                                                                                                                               |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>LutType</i> | type of LUT which has to be allocated. One of <a href="#">LvipLutType</a> , this value could be optionally OR-ed with the LVIP_LUT_BAYER flag |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

handle to the allocated LUT

**Note**

LUT has to be freed up before end using [LvipFreeLut\(\)](#) function

#### 5.40.2.6 LV\_EXTC LV\_DLLIMPORT LvStatus LvipApplyLut ( LvipImgInfo \* *pSrcImgInfo*, LvipImgInfo \* *pDstImgInfo*, LvipHLut *hLut*, uint32\_t *Options* )

Apply LUT to source image and save it in the destination image. Applies the LUT to the image. Note that the LUT can be applied in other functions as well, which is faster than this separate processing.

Supported input pixel formats: 8-bit mono, 10-bit mono, 12-bit mono, 24-bit BGR, 32-bit BGR, 24-bit RGB, 32-bit RGB.

Supported output pixel formats: equal to the input pixel format.

Can be done in-place: Yes (DstImgInfo can be NULL).

## Parameters

|                    |                                                                                                                                                                                                              |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i> | Source image info                                                                                                                                                                                            |
| <i>pDstImgInfo</i> | Destination image info                                                                                                                                                                                       |
| <i>hLut</i>        | Handle to the LUT, which has to be applied                                                                                                                                                                   |
| <i>Options</i>     | Options - OR-ed combination of <a href="#">LvipOption</a> . If the <code>LvipOption_ReallocateDst</code> is used, then also can contain attributes from <a href="#">LvipImgAttr</a> for (re)allocated image. |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.40.2.7 LV\_EXTC LV\_DLLIMPORT LvStatus LvipCalcWbFactors ( LvipImgInfo \* pSrcImgInfo, uint32\_t \* pFactorRed, uint32\_t \* pFactorGreen, uint32\_t \* pFactorBlue, uint32\_t Options )

Calculates white balance factors. The image is expected to be obtained from camera pointed at a neutral grey area. The factor is a gain applied to each pixel component. The gain = 1.0 means no change. In order to avoid using float numbers, the factors are multiplied by 1000 and stored in uint32\_t. If the image pixel format is MONO, the image is expected to be Bayer Array encoded. The factors are normalized, so that all are >= 1.0. This assures the areas with saturated colors remain white.

The obtained factors could be used in the [LvipAddWbToLut\(\)](#) function.

## Note

If the `LvipOption_WbCorrectFactors` flag is used, it is assumed that the white balance is calculated from the image to which were applied white balancing factors passed as parameters. Thus only a correction is calculated and the existing factors are modified. This flag cannot be used on undecoded Bayer array image.

## Parameters

|                     |                                                                                                                                                                                                                                          |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i>  | Source image info from which the white balance has to be calculated                                                                                                                                                                      |
| <i>pFactorRed</i>   | Pointer to uint32_t variable to which will be saved the Red factor, multiplied by 1000. If the <code>LvipOption_WbCorrectFactors</code> flag is used, the variable should contain the factor already used for WB of the current image.   |
| <i>pFactorGreen</i> | Pointer to uint32_t variable to which will be saved the Green factor, multiplied by 1000. If the <code>LvipOption_WbCorrectFactors</code> flag is used, the variable should contain the factor already used for WB of the current image. |
| <i>pFactorBlue</i>  | Pointer to uint32_t variable to which will be saved the Blue factor, multiplied by 1000. If the <code>LvipOption_WbCorrectFactors</code> flag is used, the variable should contain the factor already used for WB of the current image.  |
| <i>Options</i>      | Options, see <a href="#">LvipOption_WbCorrectFactors</a>                                                                                                                                                                                 |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.40.2.8 LV\_EXTC LV\_DLLIMPORT LvStatus LvipFreeLut ( LvipHLut hLut )

Deallocates the LUT.

## Parameters

|             |                                                                                            |
|-------------|--------------------------------------------------------------------------------------------|
| <i>hLut</i> | Handle to LUT (which had been allocated by the <a href="#">LvipAllocateLut()</a> function) |
|-------------|--------------------------------------------------------------------------------------------|

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

```
5.40.2.9 LV_EXTC LV_DLLIMPORT LvStatus LvipGet10BitLut ( LvipHLut hLut, uint16_t * pRed, uint16_t * pGreen, uint16_t * pBlue )
```

Gets 10-bit LUT data. This function fills up supplied arrays with the current LUT data. It is useful for example after calling [LvipAddGammaToLut\(\)](#) or [LvipAddWbToLut\(\)](#) to get the values of current LUT.

## Parameters

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| <i>hLut</i>   | Handle to LUT                                                          |
| <i>pRed</i>   | pointer to an array of 1024 uint16_t values, will be filled with red   |
| <i>pGreen</i> | pointer to an array of 1024 uint16_t values, will be filled with green |
| <i>pBlue</i>  | pointer to an array of 1024 uint16_t values, will be filled with blue  |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.40.2.10 LV\_EXTC LV\_DLLIMPORT LvStatus LvipGet10BitLutValue ( LvipHLut *hLut*, LvEnum *LutSelector*, uint32\_t *Index*, uint16\_t \* *pValue* )

Gets 10-bit LUT value. This function reads one value from the LUT. Note that for reading the whole LUT a more effective function [LvipGet10BitLut\(\)](#) is available.

## Parameters

|                    |                                                                                          |
|--------------------|------------------------------------------------------------------------------------------|
| <i>hLut</i>        | Handle to LUT                                                                            |
| <i>LutSelector</i> | LUT selector (see LvLUTSelector). The Luminance LUT is actually stored in the Green one. |
| <i>Index</i>       | Value index in the LUT                                                                   |
| <i>pValue</i>      | Pointer to the variable which receives the value                                         |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.40.2.11 LV\_EXTC LV\_DLLIMPORT LvStatus LvipGet12BitLut ( LvipHLut *hLut*, uint16\_t \* *pRed*, uint16\_t \* *pGreen*, uint16\_t \* *pBlue* )

Gets 12-bit LUT data. This function fills up supplied arrays with the current LUT data. It is useful for example after calling [LvipAddGammaToLut\(\)](#) or [LvipAddWbToLut\(\)](#) to get the values of current LUT.

## Parameters

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| <i>hLut</i>   | Handle to LUT                                                          |
| <i>pRed</i>   | pointer to an array of 4096 uint16_t values, will be filled with red   |
| <i>pGreen</i> | pointer to an array of 4096 uint16_t values, will be filled with green |
| <i>pBlue</i>  | pointer to an array of 4096 uint16_t values, will be filled with blue  |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.40.2.12 LV\_EXTC LV\_DLLIMPORT LvStatus LvipGet12BitLutValue ( LvipHLut *hLut*, LvEnum *LutSelector*, uint32\_t *Index*, uint16\_t \* *pValue* )

Gets 12-bit LUT value. This function reads one value from the LUT. Note that for reading the whole LUT a more effective function [LvipGet12BitLut\(\)](#) is available.

## Parameters

|                    |                                                                                                           |
|--------------------|-----------------------------------------------------------------------------------------------------------|
| <i>hLut</i>        | Handle to LUT                                                                                             |
| <i>LutSelector</i> | LUT selector (see <a href="#">LvLUTSelector</a> ). The Luminance LUT is actually stored in the Green one. |
| <i>Index</i>       | Value index in the LUT                                                                                    |
| <i>pValue</i>      | Pointer to the variable which receives the value                                                          |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.40.2.13 LV\_EXTC LV\_DLLIMPORT LvStatus LvipGet8BitLut ( LvipHLut *hLut*, uint8\_t \* *pRed*, uint8\_t \* *pGreen*, uint8\_t \* *pBlue* )

Gets 8-bit LUT data. This function fills up supplied arrays with the current LUT data. It is useful for example after calling [LvipAddGammaToLut\(\)](#) or [LvipAddWbToLut\(\)](#) to get the values of current LUT.

## Parameters

|               |                                                                                           |
|---------------|-------------------------------------------------------------------------------------------|
| <i>hLut</i>   | Handle to LUT                                                                             |
| <i>pRed</i>   | pointer to an array of 256 uint8_t values, which will be filled with the Red LUT values   |
| <i>pGreen</i> | pointer to an array of 256 uint8_t values, which will be filled with the Green LUT values |
| <i>pBlue</i>  | pointer to an array of 256 uint8_t values, will be filled with the LUT values             |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.40.2.14 LV\_EXTC LV\_DLLIMPORT LvStatus LvipGet8BitLutValue ( LvipHLut *hLut*, LvEnum *LutSelector*, uint32\_t *Index*, uint8\_t \* *pValue* )

Gets one 8-bit LUT value. This function reads one value from the LUT. Note that for reading the whole LUT a more effective function [LvipGet8BitLut\(\)](#) is available.

## Parameters

|                    |                                                                                                           |
|--------------------|-----------------------------------------------------------------------------------------------------------|
| <i>hLut</i>        | Handle to LUT                                                                                             |
| <i>LutSelector</i> | LUT selector (see <a href="#">LvLUTSelector</a> ). The Luminance LUT is actually stored in the Green one. |
| <i>Index</i>       | Value index in the LUT                                                                                    |
| <i>pValue</i>      | Pointer to the variable which receives the value                                                          |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.40.2.15 LV\_EXTC LV\_DLLIMPORT LvStatus LvipResetLut ( LvipHLut *hLut* )

Resets the LUT data to the linear order.

## Parameters

|             |                                                                              |
|-------------|------------------------------------------------------------------------------|
| <i>hLut</i> | Handle to LUT allocated using the <a href="#">LvipAllocateLut()</a> function |
|-------------|------------------------------------------------------------------------------|

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.40.2.16 LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet10BitLut ( LvipHLut *hLut*, uint16\_t \* *pRed*, uint16\_t \* *pGreen*, uint16\_t \* *pBlue* )

Sets up 10-bit LUT data. Sets the LUT from 3 arrays of 1024 uint16\_t values with 10-bit values. For processing the monochrome images only the green is used.

#### Parameters

|               |                                                       |
|---------------|-------------------------------------------------------|
| <i>hLut</i>   | Handle to LUT                                         |
| <i>pRed</i>   | pointer to an array of 1024 uint16_t red LUT values   |
| <i>pGreen</i> | pointer to an array of 1024 uint16_t green LUT values |
| <i>pBlue</i>  | pointer to an array of 1024 uint16_t blue LUT values  |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.40.2.17 LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet10BitLutValue ( LvipHLut *hLut*, LvEnum *LutSelector*, uint32\_t *Index*, uint16\_t *Value* )

Sets one 10-bit LUT value. This function writes one value to the LUT. Note that for writing the whole LUT a more effective function [LvipSet10BitLut\(\)](#) is available.

#### Parameters

|                    |                                                                                          |
|--------------------|------------------------------------------------------------------------------------------|
| <i>hLut</i>        | Handle to LUT                                                                            |
| <i>LutSelector</i> | LUT selector (see LvLUTSelector). The Luminance LUT is actually stored in the Green one. |
| <i>Index</i>       | Value index in the LUT.                                                                  |
| <i>Value</i>       | The value.                                                                               |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.40.2.18 LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet12BitLut ( LvipHLut *hLut*, uint16\_t \* *pRed*, uint16\_t \* *pGreen*, uint16\_t \* *pBlue* )

Sets up 12-bit LUT data. Sets the LUT from 3 arrays of 4096 uint16\_t values with 12-bit values. For processing the monochrome images only the green is used.

#### Parameters

|               |                                                       |
|---------------|-------------------------------------------------------|
| <i>hLut</i>   | Handle to LUT                                         |
| <i>pRed</i>   | pointer to an array of 4096 uint16_t red LUT values   |
| <i>pGreen</i> | pointer to an array of 4096 uint16_t green LUT values |
| <i>pBlue</i>  | pointer to an array of 4096 uint16_t blue LUT values  |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.40.2.19 LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet12BitLutValue ( LvipHLut *hLut*, LvEnum *LutSelector*, uint32\_t *Index*, uint16\_t *Value* )

Sets one 12-bit LUT value. This function writes one value to the LUT. Note that for writing the whole LUT a more effective function [LvipSet12BitLut\(\)](#) is available.

## Parameters

|                    |                                                                                          |
|--------------------|------------------------------------------------------------------------------------------|
| <i>hLut</i>        | Handle to LUT                                                                            |
| <i>LutSelector</i> | LUT selector (see LvLUTSelector). The Luminance LUT is actually stored in the Green one. |
| <i>Index</i>       | Value index in the LUT.                                                                  |
| <i>Value</i>       | The value.                                                                               |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.40.2.20 LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet8BitLut ( LvipHLut *hLut*, uint8\_t \* *pRed*, uint8\_t \* *pGreen*, uint8\_t \* *pBlue* )

Sets up the 8-bit LUT data. Sets the LUT from 3 arrays of 256 uint8\_t values. For processing the monochrome images only the green is used.

## Parameters

|               |                                                                 |
|---------------|-----------------------------------------------------------------|
| <i>hLut</i>   | Handle to LUT                                                   |
| <i>pRed</i>   | pointer to an array of 256 uint8_t values with red LUT values   |
| <i>pGreen</i> | pointer to an array of 256 uint8_t values with green LUT values |
| <i>pBlue</i>  | pointer to an array of 256 uint8_t values with blue LUT values  |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.40.2.21 LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet8BitLutValue ( LvipHLut *hLut*, LvEnum *LutSelector*, uint32\_t *Index*, uint8\_t *Value* )

Sets one 8-bit LUT value. This function writes one value to the LUT. Note that for writing the whole LUT a more effective function [LvipSet8BitLut\(\)](#) is available.

## Parameters

|                    |                                                                                          |
|--------------------|------------------------------------------------------------------------------------------|
| <i>hLut</i>        | Handle to LUT                                                                            |
| <i>LutSelector</i> | LUT selector (see LvLUTSelector). The Luminance LUT is actually stored in the Green one. |
| <i>Index</i>       | Value index in the LUT.                                                                  |
| <i>Value</i>       | The value.                                                                               |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

## 5.41 Bayer decoding/encoding functions

### Functions

- LV\_EXTC LV\_DLLIMPORT [LvStatus LvIpBdShowMosaic](#) ([LvIpImgInfo](#) \*pSrcImgInfo, [LvIpImgInfo](#) \*pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvIpBdGreenToGreyscale](#) ([LvIpImgInfo](#) \*pSrcImgInfo, [LvIpImgInfo](#) \*pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvIpBdNearestNeighbour](#) ([LvIpImgInfo](#) \*pSrcImgInfo, [LvIpImgInfo](#) \*pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options, LvIpHLut hLut)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvIpBdBilinearInterpolation](#) ([LvIpImgInfo](#) \*pSrcImgInfo, [LvIpImgInfo](#) \*pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options, LvIpHLut hLut)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvIpBdBilinearColorCorrection](#) ([LvIpImgInfo](#) \*pSrcImgInfo, [LvIpImgInfo](#) \*pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvIpBdVariableGradients](#) ([LvIpImgInfo](#) \*pSrcImgInfo, [LvIpImgInfo](#) \*pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options, LvIpHLut hLut)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvIpBdPixelGrouping](#) ([LvIpImgInfo](#) \*pSrcImgInfo, [LvIpImgInfo](#) \*pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus LvIpBdEncodeToBayer](#) ([LvIpImgInfo](#) \*pSrcImgInfo, [LvIpImgInfo](#) \*pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options)

### 5.41.1 Detailed Description

### 5.41.2 Function Documentation

#### 5.41.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvIpBdBilinearColorCorrection ( LvIpImgInfo \* pSrcImgInfo, LvIpImgInfo \* pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options )

Bayer Decoding: The Bilinear interpolation with Linear Color Correction method The interpolation with Linear Color Correction (LCC) is another adaptive algorithm and optimized for images with edges in horizontal and vertical direction.

#### Note

This function does not support LUT due to the 2-pass algorithm

#### Parameters

|                       |                                                                                                                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i>    | Pointer to source image info                                                                                                                                                                                    |
| <i>pDstImgInfo</i>    | Pointer to destination image info                                                                                                                                                                               |
| <i>DstPixelFormat</i> | to which <a href="#">LvPixelFormat</a> has to be image converted                                                                                                                                                |
| <i>Options</i>        | Options - OR-ed combination of <a href="#">LvIpOption</a> . If the <a href="#">LvIpOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvIpImgAttr</a> for (re)allocated image. |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.41.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvIpBdBilinearInterpolation ( LvIpImgInfo \* pSrcImgInfo, LvIpImgInfo \* pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options, LvIpHLut hLut )

Bayer Decoding: The Bilinear Interpolation method The most commonly used method for fast Bayer decoding. For the color not directly available for the given pixel makes the linear interpolation between the 2 or 4 neighbouring pixels to get it. Gives good results with a high speed.

- Supported input pixel formats: 8-bit mono.



- Supported output pixel formats: 8-bit mono, 24-bit BGR, 32-bit BGR.
- Can be done in-place: No.

## Parameters

|                       |                                                                                                                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i>    | Bayer encoded source image info                                                                                                                                                                                 |
| <i>pDstImgInfo</i>    | Destination image info                                                                                                                                                                                          |
| <i>DstPixelFormat</i> | To which <a href="#">LvPixelFormat</a>                                                                                                                                                                          |
| <i>Options</i>        | Options - OR-ed combination of <a href="#">LvipOption</a> . If the <a href="#">LvipOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvIpImgAttr</a> for (re)allocated image. |
| <i>hLut</i>           | Handle to LUT                                                                                                                                                                                                   |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.41.2.3 `LV_EXTC LV_DLLIMPORT LvStatus LvipBdEncodeToBayer ( LvIpImgInfo * pSrcImgInfo, LvIpImgInfo * pDstImgInfo, uint32_t DstPixelFormat, uint32_t Options )`

This function encode an RGB image back to a Bayer encoded image. This function is generally for testing purposes.

## Parameters

|                       |                                                                                                                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i>    | RGB source image info                                                                                                                                                                                           |
| <i>pDstImgInfo</i>    | It will contain bayer encoded image                                                                                                                                                                             |
| <i>DstPixelFormat</i> | To which <a href="#">LvPixelFormat</a> convert                                                                                                                                                                  |
| <i>Options</i>        | Options - OR-ed combination of <a href="#">LvipOption</a> . If the <a href="#">LvipOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvIpImgAttr</a> for (re)allocated image. |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.41.2.4 `LV_EXTC LV_DLLIMPORT LvStatus LvipBdGreenToGreyscale ( LvIpImgInfo * pSrcImgInfo, LvIpImgInfo * pDstImgInfo, uint32_t DstPixelFormat, uint32_t Options )`

Bayer Decoding: Convert green to greyscale Converts fast but roughly the Bayer encoded image to a greyscale by using only the green pixels, which cover the half of all pixels. The other half is calculated by bilinear interpolation.

- Supported input pixel formats: 8-bit mono.
- Supported output pixel formats: 8-bit mono.
- Can be done in-place: No.

## Parameters

|                       |                                                                                                                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i>    | source image info                                                                                                                                                                                               |
| <i>pDstImgInfo</i>    | destination image info                                                                                                                                                                                          |
| <i>DstPixelFormat</i> | destination <a href="#">LvPixelFormat</a>                                                                                                                                                                       |
| <i>Options</i>        | Options - OR-ed combination of <a href="#">LvipOption</a> . If the <a href="#">LvipOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvIpImgAttr</a> for (re)allocated image. |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.41.2.5 LV\_EXTC LV\_DLLIMPORT LvStatus LvipBdNearestNeighbour ( LvplmgInfo \* pSrcImgInfo, LvplmgInfo \* pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options, LvipHLut hLut )

Bayer Decoding: The Nearest Neighbour method The fastest method for Bayer array decoding. It uses the nearest pixel with the required lense color to get the pixel value. Gives rough results.

- Supported input pixel formats: 8-bit mono.
- Supported output pixel formats: 8-bit mono, 24-bit BGR, 32-bit BGR.
- Can be done in-place: No.

##### Parameters

|                       |                                                                                                                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i>    | Bayer encoded source image info                                                                                                                                                                                 |
| <i>pDstImgInfo</i>    | Destination image info                                                                                                                                                                                          |
| <i>DstPixelFormat</i> | To which <a href="#">LvPixelFormat</a>                                                                                                                                                                          |
| <i>Options</i>        | Options - OR-ed combination of <a href="#">LvIpOption</a> . If the <a href="#">LvIpOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvIpImgAttr</a> for (re)allocated image. |
| <i>hLut</i>           | Handle to LUT                                                                                                                                                                                                   |

##### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.41.2.6 LV\_EXTC LV\_DLLIMPORT LvStatus LvipBdPixelGrouping ( LvplmgInfo \* pSrcImgInfo, LvplmgInfo \* pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options )

Bayer Decoding: The Pixel Grouping method. A method similar to the [LvipBdVariableGradients\(\)](#), but simplified and thus faster, still giving very good results.

- Supported input pixel formats: 8-bit mono.
- Supported output pixel formats: 24-bit BGR, 32-bit BGR.
- Can be done in-place: No.

##### Note

This function does not support LUT operations because of too high CPU load

##### Parameters

|                       |                                                                                                                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i>    | Bayer encoded source image info                                                                                                                                                                                 |
| <i>pDstImgInfo</i>    | Destination image info                                                                                                                                                                                          |
| <i>DstPixelFormat</i> | To which <a href="#">LvPixelFormat</a> convert                                                                                                                                                                  |
| <i>Options</i>        | Options - OR-ed combination of <a href="#">LvIpOption</a> . If the <a href="#">LvIpOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvIpImgAttr</a> for (re)allocated image. |

##### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.41.2.7 LV\_EXTC LV\_DLLIMPORT LvStatus LvipBdShowMosaic ( LvipImgInfo \* pSrcImgInfo, LvipImgInfo \* pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options )

Bayer Decoding: Show mosaic. This function converts the Bayer encoded image to RGB format, without decoding the color information, only showing in the color how the image is seen by the chip after the light goes through the color lenses. The purpose of this function is only to help in Bayer decoding algorithms investigations.

- Supported input pixel formats: 8-bit mono.
- Supported output pixel formats: 24-bit BGR, 32-bit BGR.
- Can be done in-place: No.

#### Parameters

|                       |                                                                                                                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i>    | Bayer encoded image                                                                                                                                                                                             |
| <i>pDstImgInfo</i>    | Where to save displayable image                                                                                                                                                                                 |
| <i>DstPixelFormat</i> | To which <a href="#">LvPixelFormat</a> convert bayer encoded image                                                                                                                                              |
| <i>Options</i>        | Options - OR-ed combination of <a href="#">LvipOption</a> . If the <a href="#">LvipOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvipImgAttr</a> for (re)allocated image. |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.41.2.8 LV\_EXTC LV\_DLLIMPORT LvStatus LvipBdVariableGradients ( LvipImgInfo \* pSrcImgInfo, LvipImgInfo \* pDstImgInfo, uint32\_t DstPixelFormat, uint32\_t Options, LvipHLut hLut )

Bayer Decoding: Variable gradients method One of the best known methods for Bayer decoding, but significantly slower than the bilinear interpolation. It is based on evaluation the color gradients in 8 directions around the pixel and selecting the set of best set for the color interpolation.

- Supported input pixel formats: 8-bit mono.
- Supported output pixel formats: 24-bit BGR, 32-bit BGR.
- Can be done in-place: No.

#### Parameters

|                       |                                                                                                                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i>    | Bayer encoded source image info                                                                                                                                                                                 |
| <i>pDstImgInfo</i>    | Destination image info                                                                                                                                                                                          |
| <i>DstPixelFormat</i> | To which <a href="#">LvPixelFormat</a> convert                                                                                                                                                                  |
| <i>Options</i>        | Options - OR-ed combination of <a href="#">LvipOption</a> . If the <a href="#">LvipOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvipImgAttr</a> for (re)allocated image. |
| <i>hLut</i>           | Handle to LUT                                                                                                                                                                                                   |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

## 5.42 Rotation and line manipulation functions

### Functions

- LV\_EXTC LV\_DLLIMPORT LvStatus LvipDeinterlace (LvimgInfo \*pSrcImgInfo, LvimgInfo \*pDstImgInfo, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipRotate90 (LvimgInfo \*pSrcImgInfo, LvimgInfo \*pDstImgInfo, int32\_t ClockWise, uint32\_t Options, LvipHLut hLut)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipMirror (LvimgInfo \*pSrcImgInfo, LvimgInfo \*pDstImgInfo, int32\_t TopBottomMirror, int32\_t LeftRightMirror, uint32\_t Options, LvipHLut hLut)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipRotate90AndMirror (LvimgInfo \*pSrcImgInfo, LvimgInfo \*pDstImgInfo, int32\_t ClockWise, int32\_t TopBottomMirror, int32\_t LeftRightMirror, uint32\_t Options, LvipHLut hLut)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipReverseLines (LvimgInfo \*pSrcImgInfo, LvimgInfo \*pDstImgInfo, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipReverseLinesFast (LvimgInfo \*pSrcImgInfo, LvimgInfo \*pDstImgInfo, void \*pLineBuffer, uint32\_t Options)

### 5.42.1 Detailed Description

### 5.42.2 Function Documentation

#### 5.42.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvipDeinterlace ( LvimgInfo \* pSrcImgInfo, LvimgInfo \* pDstImgInfo, uint32\_t Options )

Deinterlacing. Deinterlaces by averaging the neighbour lines. Deinterlace function reduces the artefacts resulting from capturing a moving object by an interlaced camera.

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: Yes (DstImgInfo can be NULL).

#### Parameters

|                    |                                                                                                                                                                                               |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i> | Source image info                                                                                                                                                                             |
| <i>pDstImgInfo</i> | Destination image info                                                                                                                                                                        |
| <i>Options</i>     | Options - OR-ed combination of <a href="#">LvipOption</a> . If the LvipOption_ReallocateDst is used, then also can contain attributes from <a href="#">LvimgAttr</a> for (re)allocated image. |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.42.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvipMirror ( LvimgInfo \* pSrcImgInfo, LvimgInfo \* pDstImgInfo, int32\_t TopBottomMirror, int32\_t LeftRightMirror, uint32\_t Options, LvipHLut hLut )

Mirrors the image along the horizontal axis (TopBottomMirror) or vertical axis (LeftRightMirror).

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: Yes (DstImgInfo can be NULL).

## Parameters

|                        |                                                                                                                                                                                                              |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i>     | Source image info                                                                                                                                                                                            |
| <i>pDstImgInfo</i>     | Destination image info.                                                                                                                                                                                      |
| <i>TopBottomMirror</i> | <b>1</b> for top-bottom mirror, <b>0</b> if not                                                                                                                                                              |
| <i>LeftRightMirror</i> | <b>1</b> for left-right mirror, <b>0</b> if not                                                                                                                                                              |
| <i>Options</i>         | Options - OR-ed combination of <a href="#">LvipOption</a> . If the <code>LvipOption_ReallocateDst</code> is used, then also can contain attributes from <a href="#">LviplmgAttr</a> for (re)allocated image. |
| <i>hLut</i>            | Handle to LUT                                                                                                                                                                                                |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

## Note

the LUT in this function is not yet implemented.

#### 5.42.2.3 LV\_EXTC LV\_DLLIMPORT LvStatus LvipReverseLines ( LviplmgInfo \* pSrcImgInfo, LviplmgInfo \* pDstImgInfo, uint32\_t Options )

Reversed lines for switching between the top-down and bottom-up formats. Performs the same action as `TopBottom` mirror, but updates also the [LviplmgInfo](#) with a flag indicating the orientation (this has a meaning when switching between top-down and bottom-up formats).

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: Yes (`DstImgInfo` can be NULL).

## Parameters

|                    |                                                                                                                                                                                                              |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i> | Source image info                                                                                                                                                                                            |
| <i>pDstImgInfo</i> | Destination image info                                                                                                                                                                                       |
| <i>Options</i>     | Options - OR-ed combination of <a href="#">LvipOption</a> . If the <code>LvipOption_ReallocateDst</code> is used, then also can contain attributes from <a href="#">LviplmgAttr</a> for (re)allocated image. |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.42.2.4 LV\_EXTC LV\_DLLIMPORT LvStatus LvipReverseLinesFast ( LviplmgInfo \* pSrcImgInfo, LviplmgInfo \* pDstImgInfo, void \* pLineBuffer, uint32\_t Options )

Fastly reverses lines (copying whole lines). The `pDstImgInfo` can be NULL (in-place reversion). In such case a temporary buffer for a line is needed.

The buffer can be supplied in `pLineBuffer` (must have sufficient size to hold the whole line in its pixel format, that means  $\geq \text{ImgInfo.dwLineIncrement}$ ).

If the `pLineBuffer` is NULL, the buffer is temporarily allocated and deallocated, which might require additional CPU time, so for the repeated call of this function it is better to allocate the buffer outside the function and pass it as `pLineBuffer` parameter.

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: Yes (`DstImgInfo` can be NULL).

## Parameters

|                    |                                                                                                                                                                                                                 |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i> | Source image info                                                                                                                                                                                               |
| <i>pDstImgInfo</i> | Destination image info; can be NULL - then in-place reversion will be done (but in this case a temporary buffer for line is needed)                                                                             |
| <i>pLineBuffer</i> | Pointer to temporary line buffer. Can be NULL.                                                                                                                                                                  |
| <i>Options</i>     | Options - OR-ed combination of <a href="#">LvIpOption</a> . If the <a href="#">LvIpOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvIpImgAttr</a> for (re)allocated image. |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.42.2.5 LV\_EXTC LV\_DLLIMPORT LvStatus LvIpRotate90 ( LvIpImgInfo \* pSrcImgInfo, LvIpImgInfo \* pDstImgInfo, int32\_t ClockWise, uint32\_t Options, LvIpHLut hLut )

Rotates the image by 90 degrees clockwise or counterclockwise.

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: No.

## Note

the LUT in this function is not yet implemented.

For 180 degrees rotation use the [LvIpMirror\(\)](#) function and set mirroring along both axes.

## Parameters

|                    |                                                                                                                                                                                                                 |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i> | Source image info                                                                                                                                                                                               |
| <i>pDstImgInfo</i> | Destination image info                                                                                                                                                                                          |
| <i>ClockWise</i>   | <b>1</b> if the image has to be rotated clockwise, <b>0</b> if the image has to be rotated by counterclockwise                                                                                                  |
| <i>Options</i>     | Options - OR-ed combination of <a href="#">LvIpOption</a> . If the <a href="#">LvIpOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvIpImgAttr</a> for (re)allocated image. |
| <i>hLut</i>        | Handle to LUT                                                                                                                                                                                                   |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

5.42.2.6 LV\_EXTC LV\_DLLIMPORT LvStatus LvIpRotate90AndMirror ( LvIpImgInfo \* pSrcImgInfo, LvIpImgInfo \* pDstImgInfo, int32\_t ClockWise, int32\_t TopBottomMirror, int32\_t LeftRightMirror, uint32\_t Options, LvIpHLut hLut )

It does the rotation and mirroring in the same step. If the Options contain [LvIpOption\\_ReallocateDst](#) and the *pDstImgInfo* contains different image width or height or the *pData* is NULL, the *pData* is reallocated and the image parameters are adjusted. The Options in such case can contain also [LvIpImgAttr](#) flags for new image descriptor creation.

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: No.

## Parameters

|                        |                                                                                                                                                                                                              |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i>     | Source image info                                                                                                                                                                                            |
| <i>pDstImgInfo</i>     | Destination image info                                                                                                                                                                                       |
| <i>ClockWise</i>       | <b>1</b> if image has to be rotated by 90 degrees clockwise, otherwise (counterclockwise) <b>0</b>                                                                                                           |
| <i>TopBottomMirror</i> | <b>1</b> if top-bottom mirror has to be used, otherwise <b>0</b>                                                                                                                                             |
| <i>LeftRightMirror</i> | <b>1</b> if left-right mirror has to be used, otherwise <b>0</b>                                                                                                                                             |
| <i>Options</i>         | Options - OR-ed combination of <a href="#">LvipOption</a> . If the <code>LvipOption_ReallocateDst</code> is used, then also can contain attributes from <a href="#">LviplmgAttr</a> for (re)allocated image. |
| <i>hLut</i>            | Handle to LUT                                                                                                                                                                                                |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

## Note

the LUT in this function is not yet implemented.

## 5.43 Pixel format conversion functions

### Functions

- LV\_EXTC LV\_DLLIMPORT LvStatus LvipConvertToPixelFormat (LvplmInfo \*pSrcImInfo, LvplmInfo \*pDstImInfo, uint32\_t DstPixelFormat, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT uint32\_t LvipCanConvertToPixelFormat (uint32\_t dwSrcPixelFormat, uint32\_t dwDstPixelFormat)

### 5.43.1 Detailed Description

### 5.43.2 Function Documentation

5.43.2.1 LV\_EXTC LV\_DLLIMPORT uint32\_t LvipCanConvertToPixelFormat ( uint32\_t dwSrcPixelFormat, uint32\_t dwDstPixelFormat )

Returns 1 if the source pixel format can be converted to destination pixel format by the [LvipConvertToPixelFormat\(\)](#) function.

#### Parameters

|                         |                                           |
|-------------------------|-------------------------------------------|
| <i>dwSrcPixelFormat</i> | Source <a href="#">LvPixelFormat</a>      |
| <i>dwDstPixelFormat</i> | Destination <a href="#">LvPixelFormat</a> |

#### Returns

1 in case of the conversion is available, otherwise 0.

5.43.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvipConvertToPixelFormat ( LvplmInfo \* pSrcImInfo, LvplmInfo \* pDstImInfo, uint32\_t DstPixelFormat, uint32\_t Options )

Converts the image from one pixel format to another one.

- Supported input pixel formats: 8-bit to 16-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: 8-bit mono, 24-bit BGR, 32-bit BGR.
- Can be done in-place: No.

#### Parameters

|                       |                                                                                                                                                                                                |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImInfo</i>     | Source Image Info                                                                                                                                                                              |
| <i>pDstImInfo</i>     | Destination Image Info                                                                                                                                                                         |
| <i>DstPixelFormat</i> | Destination <a href="#">LvPixelFormat</a>                                                                                                                                                      |
| <i>Options</i>        | Options - OR-ed combination of <a href="#">LvipOption</a> . If the LvipOption_ReallocateDst is used, then also can contain attributes from <a href="#">LvipImAttr</a> for (re)allocated image. |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure



## 5.44 Saving/loading functions

### Functions

- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvipSaveToTiff](#) (const char \*pFileName, [LvipImgInfo](#) \*pImgInfo, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvipLoadFromTiff](#) (const char \*pFileName, [LvipImgInfo](#) \*pImgInfo, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvipSaveToBmp](#) (const char \*pFileName, [LvipImgInfo](#) \*pImgInfo, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvipLoadFromBmp](#) (const char \*pFileName, [LvipImgInfo](#) \*pImgInfo, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvipSaveToJpeg](#) (const char \*pFileName, [LvipImgInfo](#) \*pImgInfo, uint32\_t QualityFactor)
- LV\_EXTC LV\_DLLIMPORT [LvStatus](#) [LvipLoadFromJpeg](#) (const char \*pFileName, [LvipImgInfo](#) \*pImgInfo, uint32\_t Options)

#### 5.44.1 Detailed Description

#### 5.44.2 Function Documentation

##### 5.44.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvipLoadFromBmp ( const char \* pFileName, LvipImgInfo \* pImgInfo, uint32\_t Options )

Loads image from BMP file. Formats with less 8 bits per pixel are not supported. The color palette by 8-bit pixel format is expected to be greyscale.

##### Parameters

|                  |                                                                                                                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pFileName</i> | File name                                                                                                                                                                                                       |
| <i>pImgInfo</i>  | Image info for the loaded image                                                                                                                                                                                 |
| <i>Options</i>   | Options - OR-ed combination of <a href="#">LvipOption</a> . If the <a href="#">LvipOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvipImgAttr</a> for (re)allocated image. |

##### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

##### 5.44.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvipLoadFromJpeg ( const char \* pFileName, LvipImgInfo \* pImgInfo, uint32\_t Options )

Loads the image from JPEG file.

##### Parameters

|                  |                                                                                                                                                                                                                                                                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pFileName</i> | File name.                                                                                                                                                                                                                                                                                                                                                      |
| <i>pImgInfo</i>  | Image info for the loaded image.                                                                                                                                                                                                                                                                                                                                |
| <i>Options</i>   | Options - OR-ed combination of <a href="#">LvipOption</a> . If the <a href="#">LvipOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvipImgAttr</a> for (re)allocated image. See <a href="#">LvipOption_Jpeg↔ConvertToBgr</a> , <a href="#">LvipOption_JpegReadHeaderOnly</a> and <a href="#">LvipOption_ReallocateDst</a> . |

##### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

**Note**

You can either supply the `plmgInfo` with already allocated buffer or use empty `ImgInfo` and specify the [LvOption\\_ReallocateDst](#) flag. In the first case you can utilize the [LvOption\\_JpegReadHeaderOnly](#) flag to obtain the image attributes.

#### 5.44.2.3 LV\_EXTC LV\_DLLIMPORT LvStatus LvLoadFromTiff ( const char \* *pFileName*, LvplmgInfo \* *plmgInfo*, uint32\_t *Options* )

Loads the image from TIFF file. Is preferred to load the image from TIFF file which had been previously saved by [LvSaveToTiff\(\)](#) function - this library supports only a base TIFF format and there it is not assured that the TIFF image created by another application could be loaded without error.

**Parameters**

|                  |                                                                                                                                                                                                            |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pFileName</i> | File name                                                                                                                                                                                                  |
| <i>plmgInfo</i>  | Image info for the loaded image.                                                                                                                                                                           |
| <i>Options</i>   | Options - OR-ed combination of <a href="#">LvOption</a> . If the <a href="#">LvOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LvplmgAttr</a> for (re)allocated image. |

**Returns**

LVSTATUS\_OK in case of success, or error status in case of failure

**Note**

The `pData` are always reallocated.

#### 5.44.2.4 LV\_EXTC LV\_DLLIMPORT LvStatus LvSaveToBmp ( const char \* *pFileName*, LvplmgInfo \* *plmgInfo*, uint32\_t *Options* )

Saves the image to a BMP file if the pixel format is compatible with BMP. *Compatible with BMP* means that [LvPixelFormat](#) is one of 8-bit mono, 24- or 32-bit BGR.

- Supported pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <i>pFileName</i> | File name                                                 |
| <i>plmgInfo</i>  | Image info of an image to be saved                        |
| <i>Options</i>   | Options - OR-ed combination of <a href="#">LvOption</a> . |

**Returns**

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.44.2.5 LV\_EXTC LV\_DLLIMPORT LvStatus LvSaveToJpeg ( const char \* *pFileName*, LvplmgInfo \* *plmgInfo*, uint32\_t *QualityFactor* )

Saves the image to the JPEG file. In contrast to the BMP format, it enables to store also 9- to 16-bit mono formats.

- Supported pixel formats: 8-bit to 16-bit mono, all RGB and BGR formats. For JPEG the native pixel format is either 8-bit mono or 24-bit BGR. If the image is in different pixel format, it is automatically converted to one of these 2 formats.

## Parameters

|                      |                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <i>pFileName</i>     | File name.                                                                                                                  |
| <i>pImgInfo</i>      | Image info of an image to be saved.                                                                                         |
| <i>QualityFactor</i> | The quality factor in range from 0 to 100. The higher the quality, the lower is the compression. The default quality is 75. |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure.

#### 5.44.2.6 LV\_EXTC LV\_DLLIMPORT LvStatus LvipSaveToTiff ( const char \* *pFileName*, LvipImgInfo \* *pImgInfo*, uint32\_t *Options* )

Saves the image to the TIFF file. In contrast to the BMP format, it enables to store also 9- to 16-bit mono formats. The flag LvipOption\_TiffConvertTo16Bit can be used to force the conversion to 16bit format, which is supported by wider range of applications.

- Supported pixel formats: 8-bit to 16-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.

## Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <i>pFileName</i> | File name                                                   |
| <i>pImgInfo</i>  | Image info of an image to be saved                          |
| <i>Options</i>   | Options - OR-ed combination of <a href="#">LvipOption</a> . |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure.

## 5.45 Overlay functions

## 5.46 RGB color correction and convolution functions

### Functions

- LV\_EXTC LV\_DLLIMPORT LvStatus LvipApplyRgbColorCorrection (LvIpImgInfo \*pSrcImgInfo, LvIpImgInfo \*pDstImgInfo, int32\_t \*piMatrix, uint32\_t Options, LvIpHLut hLut)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipSetSaturationMatrix (uint32\_t SaturationFactor, int32\_t \*piMatrix, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet3x3MatrixSharpening (int32\_t Factor, int32\_t \*piMatrix, uint32\_t Options)
- LV\_EXTC LV\_DLLIMPORT LvStatus LvipApply3x3Convolution (LvIpImgInfo \*pSrcImgInfo, LvIpImgInfo \*pDstImgInfo, int32\_t \*piMatrix, uint32\_t Options, LvIpHLut hLut)

#### 5.46.1 Detailed Description

#### 5.46.2 Function Documentation

##### 5.46.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvipApply3x3Convolution ( LvIpImgInfo \* pSrcImgInfo, LvIpImgInfo \* pDstImgInfo, int32\_t \* piMatrix, uint32\_t Options, LvIpHLut hLut )

Does 3x3 convolution. Applies the 3x3 matrix convolution operation. Typically, if the matrix is set for sharpening, sharpens the image.

#### See also

[LvipSet3x3MatrixSharpening\(\)](#) for creation of the sharpening matrix.

- Supported input pixel formats: 8-bit to 16-bit mono, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: No.

#### Parameters

|                    |                                                                                                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i> | Source image                                                                                                                                                                                    |
| <i>pDstImgInfo</i> | Destination image info                                                                                                                                                                          |
| <i>piMatrix</i>    | Matrix for the convolution operation.                                                                                                                                                           |
| <i>Options</i>     | Options - OR-ed combination of <a href="#">LvIpOption</a> . If the LvIpOption_ReallocateDst is used, then also can contain attributes from <a href="#">LvIpImgAttr</a> for (re)allocated image. |
| <i>hLut</i>        | Handle to LUT                                                                                                                                                                                   |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### Note

the LUT in this function is not yet implemented.

##### 5.46.2.2 LV\_EXTC LV\_DLLIMPORT LvStatus LvipApplyRgbColorCorrection ( LvIpImgInfo \* pSrcImgInfo, LvIpImgInfo \* pDstImgInfo, int32\_t \* piMatrix, uint32\_t Options, LvIpHLut hLut )

RGB color correction. A color correction 3x3 matrix is applied to RGB components of each pixel.

- Supported input pixel formats: 24-bit BGR, 32-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: Yes (DstImgInfo can be NULL).

## Parameters

|                    |                                                                                                                                                                                                                 |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i> | Image which needs to correct colors                                                                                                                                                                             |
| <i>pDstImgInfo</i> | Where to save image with correct colors                                                                                                                                                                         |
| <i>piMatrix</i>    | 3x3 matrix used to correct colors. It could be filled up using <a href="#">LvipSetSaturationMatrix()</a> . The factors in the matrix are expressed as multiplied by 1000, that means 1000 = factor 1.0.         |
| <i>Options</i>     | Options - OR-ed combination of <a href="#">LvipOption</a> . If the <a href="#">LvipOption_ReallocateDst</a> is used, then also can contain attributes from <a href="#">LviplmgAttr</a> for (re)allocated image. |
| <i>hLut</i>        | Handle to LUT (could be NULL)                                                                                                                                                                                   |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

## Note

the LUT in this function is not yet implemented.

#### 5.46.2.3 LV\_EXTC LV\_DLLIMPORT LvStatus LvipSet3x3MatrixSharpening ( int32\_t *Factor*, int32\_t \* *piMatrix*, uint32\_t *Options* )

Sets up sharpening matrix. Fills the matrix with weighted values for 3x3 sharpening. The factor is 0 for no-change matrix, +100 for maximum sharpening, -100 for blurring

## Parameters

|                 |                                                                                                              |
|-----------------|--------------------------------------------------------------------------------------------------------------|
| <i>Factor</i>   | Factor of sharpening                                                                                         |
| <i>piMatrix</i> | 3x3 matrix of int32_t values which will obtain the calculated values                                         |
| <i>Options</i>  | Options: 0 for faster sharpening from 4 neighboring pixels, 1 for full sharpening from 8 neighboring pixels. |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

#### 5.46.2.4 LV\_EXTC LV\_DLLIMPORT LvStatus LvipSetSaturationMatrix ( uint32\_t *SaturationFactor*, int32\_t \* *piMatrix*, uint32\_t *Options* )

Sets up the color saturation 3x3 matrix. The saturation factor is in percents, eg. 100 = 1.0 = unchanged image. The matrix can be used as parameter in the [LvipApplyRgbColorCorrection\(\)](#) function.

## Parameters

|                         |                                                                       |
|-------------------------|-----------------------------------------------------------------------|
| <i>SaturationFactor</i> | the saturation factor in percents                                     |
| <i>piMatrix</i>         | 3x3 matrix of int32_t values which will obtain the calculated factors |
| <i>Options</i>          | Options, reserved for future use, must be 0.                          |

## Returns

LVSTATUS\_OK in case of success, or error status in case of failure

## 5.47 Shading correction functions

### Functions

- LV\_EXTC LV\_DLLIMPORT LvStatus LvipApplyShadingCorrection (LvIpImgInfo \*pSrcImgInfo, LvIpImgInfo \*pDstImgInfo, LvIpImgInfo \*pBlackRefImgInfo, LvIpImgInfo \*pWhiteRefImgInfo, uint32\_t Options, LvIpHLut hLut)

#### 5.47.1 Detailed Description

#### 5.47.2 Function Documentation

5.47.2.1 LV\_EXTC LV\_DLLIMPORT LvStatus LvipApplyShadingCorrection ( LvIpImgInfo \* pSrcImgInfo, LvIpImgInfo \* pDstImgInfo, LvIpImgInfo \* pBlackRefImgInfo, LvIpImgInfo \* pWhiteRefImgInfo, uint32\_t Options, LvIpHLut hLut )

Applies the shading correction. The pBlackRefImgInfo and pWhiteRefImgInfo must be either NULL or must point to a valid image of the same pixel format as the pSrcImgInfo.

- Supported input pixel formats: 8-bit to 16-bit mono, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: Yes (DstImgInfo can be NULL).

#### Parameters

|                         |                                                                                                                                                                                                 |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pSrcImgInfo</i>      | Source image info                                                                                                                                                                               |
| <i>pDstImgInfo</i>      | Destination image info                                                                                                                                                                          |
| <i>pBlackRefImgInfo</i> | Black reference image                                                                                                                                                                           |
| <i>pWhiteRefImgInfo</i> | White reference image                                                                                                                                                                           |
| <i>Options</i>          | Options - OR-ed combination of <a href="#">LvIpOption</a> . If the LvIpOption_ReallocateDst is used, then also can contain attributes from <a href="#">LvIpImgAttr</a> for (re)allocated image. |
| <i>hLut</i>             | Handle to LUT                                                                                                                                                                                   |

#### Returns

LVSTATUS\_OK in case of success, or error status in case of failure

## 5.48 SynView INI file API

### Functions

- LV\_EXTC LVINI\_PUBLIC LvHIniFile [LvIniOpen](#) (const char \*pCommentSeparator)
- LV\_EXTC LVINI\_PUBLIC void [LvIniClose](#) (LvHIniFile hIniFile)
- LV\_EXTC LVINI\_PUBLIC uint32\_t [LvIniLoad](#) (LvHIniFile hIniFile, const char \*pFileName)
- LV\_EXTC LVINI\_PUBLIC uint32\_t [LvIniSave](#) (LvHIniFile hIniFile, const char \*pFileName, uint32\_t CreateBackup)
- LV\_EXTC LVINI\_PUBLIC uint32\_t [LvIniModified](#) (LvHIniFile hIniFile)
- LV\_EXTC LVINI\_PUBLIC uint32\_t [LvIniItemExists](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pName, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC uint32\_t [LvIniSectionExists](#) (LvHIniFile hIniFile, const char \*pSection)
- LV\_EXTC LVINI\_PUBLIC void [LvIniDeleteItem](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pName, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC void [LvIniDeleteSection](#) (LvHIniFile hIniFile, const char \*pSection)
- LV\_EXTC LVINI\_PUBLIC int32\_t [LvIniGetInteger](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pName, int32\_t Default, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC void [LvIniSetInteger](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pName, int32\_t Value, uint32\_t Hexadecimal, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC double [LvIniGetFloat](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pName, double Default, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC void [LvIniSetFloat](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pName, double Value, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC uint32\_t [LvIniGetBool](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pName, uint32\_t Default, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC void [LvIniSetBool](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pName, uint32\_t Value, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC void [LvIniGetString](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pName, const char \*pDefault, char \*pString, uint32\_t Size, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC uint32\_t [LvIniGetStringSize](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pName, const char \*pDefault, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC void [LvIniSetString](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pName, const char \*pValue, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC void [LvIniGetSectionRawLine](#) (LvHIniFile hIniFile, const char \*pSection, char \*pLine, uint32\_t Size, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC uint32\_t [LvIniGetSectionRawLineSize](#) (LvHIniFile hIniFile, const char \*pSection, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC void [LvIniSetSectionRawLine](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pLine, uint32\_t Order)
- LV\_EXTC LVINI\_PUBLIC void [LvIniSetParent](#) (LvHIniFile hIniFile, const char \*pSection, const char \*pName)

### 5.48.1 Detailed Description

The sv.synview.ini is a helper library, enabling to read and write INI files in all supported operating systems.

### 5.48.2 Function Documentation

#### 5.48.2.1 LV\_EXTC LVINI\_PUBLIC void LvIniClose ( LvHIniFile *hIniFile* )

Closes the INI file underlying structures and all its parents. It does not write the contents to disk; if you want to save the INI content, use first [LvIniSave\(\)](#).



## Parameters

|                 |                         |
|-----------------|-------------------------|
| <i>hIniFile</i> | Handle to the INI file. |
|-----------------|-------------------------|

5.48.2.2 LV\_EXTC LVINI\_PUBLIC void LvIniDeleteItem ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pName*, uint32\_t *Order* )

Deletes the item.

## Parameters

|                 |                                                                                    |
|-----------------|------------------------------------------------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                                                            |
| <i>pSection</i> | Section in the INI file (without brackets).                                        |
| <i>pName</i>    | Name of the item in the section.                                                   |
| <i>Order</i>    | Can be used to distinguish between multiple items of the same name in one section. |

5.48.2.3 LV\_EXTC LVINI\_PUBLIC void LvIniDeleteSection ( LvHIniFile *hIniFile*, const char \* *pSection* )

Deletes the section.

## Parameters

|                 |                                             |
|-----------------|---------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                     |
| <i>pSection</i> | Section in the INI file (without brackets). |

5.48.2.4 LV\_EXTC LVINI\_PUBLIC uint32\_t LvIniGetBool ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pName*, uint32\_t *Default*, uint32\_t *Order* )

Reads a boolean value.

## Parameters

|                 |                                                                                    |
|-----------------|------------------------------------------------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                                                            |
| <i>pSection</i> | Section in the INI file (without brackets).                                        |
| <i>pName</i>    | Name of the item in the section.                                                   |
| <i>Default</i>  | Default value to be used when the item is not found or is empty.                   |
| <i>Order</i>    | Can be used to distinguish between multiple items of the same name in one section. |

## Returns

Read value (0 or 1).

5.48.2.5 LV\_EXTC LVINI\_PUBLIC double LvIniGetFloat ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pName*, double *Default*, uint32\_t *Order* )

Reads a float value.

## Parameters

|                 |                                             |
|-----------------|---------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                     |
| <i>pSection</i> | Section in the INI file (without brackets). |
| <i>pName</i>    | Name of the item in the section.            |

|                |                                                                                    |
|----------------|------------------------------------------------------------------------------------|
| <i>Default</i> | Default value to be used when the item is not found or is empty.                   |
| <i>Order</i>   | Can be used to distinguish between multiple items of the same name in one section. |

**Returns**

Read value.

5.48.2.6 LV\_EXTC LVINI\_PUBLIC int32\_t LvIniGetInteger ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pName*, int32\_t *Default*, uint32\_t *Order* )

Reads an integer value.

**Parameters**

|                 |                                                                                    |
|-----------------|------------------------------------------------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                                                            |
| <i>pSection</i> | Section in the INI file (without brackets).                                        |
| <i>pName</i>    | Name of the item in the section.                                                   |
| <i>Default</i>  | Default value to be used when the item is not found or is empty.                   |
| <i>Order</i>    | Can be used to distinguish between multiple items of the same name in one section. |

**Returns**

Read value.

5.48.2.7 LV\_EXTC LVINI\_PUBLIC void LvIniGetSectionRawLine ( LvHIniFile *hIniFile*, const char \* *pSection*, char \* *pLine*, uint32\_t *Size*, uint32\_t *Order* )

Gets the raw string in specified section at position, specified by Order. Commented out and empty lines are not counted. If the item is not found, empty string is returned.

**Parameters**

|                 |                                                         |
|-----------------|---------------------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                                 |
| <i>pSection</i> | Section in the INI file (without brackets).             |
| <i>pLine</i>    | Buffer for the line contents.                           |
| <i>Size</i>     | Size of the pLine buffer.                               |
| <i>Order</i>    | Order of the raw line. The first valid line has Order=1 |

5.48.2.8 LV\_EXTC LVINI\_PUBLIC uint32\_t LvIniGetSectionRawLineSize ( LvHIniFile *hIniFile*, const char \* *pSection*, uint32\_t *Order* )

Gets the raw size of buffer needed for the raw line in specified section at position, specified by Order. Commented out and empty lines are not counted. If the item is not found, empty string is returned.

**Parameters**

|                 |                                                          |
|-----------------|----------------------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                                  |
| <i>pSection</i> | Section in the INI file (without brackets).              |
| <i>Order</i>    | Order of the raw line. The first valid line has Order=1. |

**Returns**

Size of the buffer (string length+1).

---

5.48.2.9 LV\_EXTC LVINI\_PUBLIC void LvIniGetString ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pName*, const char \* *pDefault*, char \* *pString*, uint32\_t *Size*, uint32\_t *Order* )

Reads a string value.

## Parameters

|                 |                                                                                    |
|-----------------|------------------------------------------------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                                                            |
| <i>pSection</i> | Section in the INI file (without brackets).                                        |
| <i>pName</i>    | Name of the item in the section.                                                   |
| <i>pDefault</i> | Default value to be used when the item is not found or is empty.                   |
| <i>pString</i>  | The string value is returned in this parameter.                                    |
| <i>Size</i>     | Size of the pString buffer.                                                        |
| <i>Order</i>    | Can be used to distinguish between multiple items of the same name in one section. |

5.48.2.10 LV\_EXTC LVINI\_PUBLIC uint32\_t LvIniGetStringSize ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pName*, const char \* *pDefault*, uint32\_t *Order* )

Returns a size of buffer needed to read the string.

## Parameters

|                 |                                                                                    |
|-----------------|------------------------------------------------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                                                            |
| <i>pSection</i> | Section in the INI file (without brackets).                                        |
| <i>pName</i>    | Name of the item in the section.                                                   |
| <i>pDefault</i> | Default value to be used when the item is not found or is empty.                   |
| <i>Order</i>    | Can be used to distinguish between multiple items of the same name in one section. |

## Returns

Size of the buffer (string length+1).

5.48.2.11 LV\_EXTC LVINI\_PUBLIC uint32\_t LvIniItemExists ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pName*, uint32\_t *Order* )

Checks if the item exists.

## Parameters

|                 |                                                                                    |
|-----------------|------------------------------------------------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                                                            |
| <i>pSection</i> | Section in the INI file (without brackets).                                        |
| <i>pName</i>    | Name of the item in the section.                                                   |
| <i>Order</i>    | Can be used to distinguish between multiple items of the same name in one section. |

## Returns

The return value 1 indicates the existence of the item.

5.48.2.12 LV\_EXTC LVINI\_PUBLIC uint32\_t LvIniLoad ( LvHIniFile *hIniFile*, const char \* *pFileName* )

Loads the ini file contents. All subsequent changes are done in memory, until the ini file is saved by the [LvIniSave\(\)](#) function. If there is a parent already specified (see [LvIniSetParent\(\)](#)), it is opened (recursively) as well.

## Parameters

|                  |                         |
|------------------|-------------------------|
| <i>hIniFile</i>  | Handle to the INI file. |
| <i>pFileName</i> | The INI file name.      |

## Returns

If the file does not exist, or file I/O fails, it returns 0, otherwise 1.

5.48.2.13 LV\_EXTC LVINI\_PUBLIC uint32\_t LvIniModified ( LvHIniFile *hIniFile* )

Returns 1 if the file content was modified.

## Returns

1 if at least one item was modified or deleted or section deleted.

5.48.2.14 LV\_EXTC LVINI\_PUBLIC LvHIniFile LvIniOpen ( const char \* *pCommentSeparator* )

Creates the INI file underlying structures and returns a handle to it. It does not read the INI file contents, this is done by [LvIniLoad\(\)](#).

## Parameters

|                          |                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pCommentSeparator</i> | If specified, all lines where it appears at the beginning of the line are considered to be a comment. It need not be placed at the very first position, but to the left must be only whitespace characters, otherwise the line is not considered to be a comment. If a value is commented out and the new value of the same name is written, it is placed before the commented value (not to the end of the section). |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

5.48.2.15 LV\_EXTC LVINI\_PUBLIC uint32\_t LvIniSave ( LvHIniFile *hIniFile*, const char \* *pFileName*, uint32\_t *CreateBackup* )

Saves the INI file contents to a file.

## Parameters

|                     |                                                                        |
|---------------------|------------------------------------------------------------------------|
| <i>hIniFile</i>     | Handle to the INI file.                                                |
| <i>pFileName</i>    | The INI file name.                                                     |
| <i>CreateBackup</i> | If is set, the original file is preserved with added ".bak" extension. |

## Returns

The return value 1 indicates a success of the file I/O.

5.48.2.16 LV\_EXTC LVINI\_PUBLIC uint32\_t LvIniSectionExists ( LvHIniFile *hIniFile*, const char \* *pSection* )

Checks is the section exists.

## Parameters

|                 |                                             |
|-----------------|---------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                     |
| <i>pSection</i> | Section in the INI file (without brackets). |

## Returns

1 if the section exists.

5.48.2.17 LV\_EXTC LVINI\_PUBLIC void LvIniSetBool ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pName*, uint32\_t *Value*, uint32\_t *Order* )

Writes a boolean value.

## Parameters

|                 |                                                                                    |
|-----------------|------------------------------------------------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                                                            |
| <i>pSection</i> | Section in the INI file (without brackets).                                        |
| <i>pName</i>    | Name of the item in the section.                                                   |
| <i>Value</i>    | Value of the item to be set.                                                       |
| <i>Order</i>    | Can be used to distinguish between multiple items of the same name in one section. |

5.48.2.18 LV\_EXTC LVINI\_PUBLIC void LvIniSetFloat ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pName*, double *Value*, uint32\_t *Order* )

Writes a float value.

## Parameters

|                 |                                                                                    |
|-----------------|------------------------------------------------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                                                            |
| <i>pSection</i> | Section in the INI file (without brackets).                                        |
| <i>pName</i>    | Name of the item in the section.                                                   |
| <i>Value</i>    | Value of the item to be set.                                                       |
| <i>Order</i>    | Can be used to distinguish between multiple items of the same name in one section. |

5.48.2.19 LV\_EXTC LVINI\_PUBLIC void LvIniSetInteger ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pName*, int32\_t *Value*, uint32\_t *Hexadecimal*, uint32\_t *Order* )

Writes an integer value.

## Parameters

|                    |                                                                                    |
|--------------------|------------------------------------------------------------------------------------|
| <i>hIniFile</i>    | Handle to the INI file.                                                            |
| <i>pSection</i>    | Section in the INI file (without brackets).                                        |
| <i>pName</i>       | Name of the item in the section.                                                   |
| <i>Value</i>       | Value of the item to be set.                                                       |
| <i>Hexadecimal</i> | If true, the value is written as hexa with the "0x" prefix.                        |
| <i>Order</i>       | Can be used to distinguish between multiple items of the same name in one section. |

5.48.2.20 LV\_EXTC LVINI\_PUBLIC void LvIniSetParent ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pName* )

Sets the section and name, where the specification of the parent INI file should be read. For example for the following INI the call would be SetParent("Linked", "Parent");

```
[Linked]
Parent=main.ini
```

Use empty strings to disable the parent.

When you specify a parent, you can automatically work with a hierarchy of INI files - when the file is open, it searches for a section and name, specified in the SetParent() function. If found and a valid file name is specified there, it creates a parent class instance and reads to it the contents of the file. Then every item, which is not explicitly specified in own INI file is searched in this parent INI file. The opening is recursive - the parent INI can have its own parent etc., so be sure you do not make a circular reference. The maximum level of recursion is intentionally limited to 10.

## Parameters

|                 |                                             |
|-----------------|---------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                     |
| <i>pSection</i> | Section in the INI file (without brackets). |
| <i>pName</i>    | Name of the item in the section.            |

5.48.2.21 LV\_EXTC LVINI\_PUBLIC void LvIniSetSectionRawLine ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pLine*, uint32\_t *Order* )

Sets the raw string in specified section at position, specified by Order. Commented out and empty lines are not counted. The first valid line has Order=1 If the item with the Order is not found, a new line is created and added at the end of section.

Parameters

|                 |                                             |
|-----------------|---------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                     |
| <i>pSection</i> | Section in the INI file (without brackets). |
| <i>pLine</i>    | The raw line to be set.                     |
| <i>Order</i>    | Order of the raw line.                      |

5.48.2.22 LV\_EXTC LVINI\_PUBLIC void LvIniSetString ( LvHIniFile *hIniFile*, const char \* *pSection*, const char \* *pName*, const char \* *pValue*, uint32\_t *Order* )

Writes a string value.

Parameters

|                 |                                                                                    |
|-----------------|------------------------------------------------------------------------------------|
| <i>hIniFile</i> | Handle to the INI file.                                                            |
| <i>pSection</i> | Section in the INI file (without brackets).                                        |
| <i>pName</i>    | Name of the item in the section.                                                   |
| <i>pValue</i>   | Value of the item to be set.                                                       |
| <i>Order</i>    | Can be used to distinguish between multiple items of the same name in one section. |

## 5.49 LvStatus definitions

### Macros

- `#define LVSTATUS_OK`
- `#define LVSTATUS_LIBRARY_NOT_LOADED`
- `#define LVSTATUS_LIBRARY_NOT_OPEN`
- `#define LVSTATUS_AVIASVER_TOO_MANY_INSTANCES`
- `#define LVSTATUS_DEVICE_TOO_MANY_INSTANCES`
- `#define LVSTATUS_CANNOT_LOAD_GENTL`
- `#define LVSTATUS_DISABLED_BY_CALLBACK`
- `#define LVSTATUS_DISPLAY_LIBRARY_NOT_LOADED`
- `#define LVSTATUS_DISPLAY_CANNOT_DISPLAY`
- `#define LVSTATUS_DISPLAY_NOT_OPEN`
- `#define LVSTATUS_ENUM_ENTRY_INVALID`
- `#define LVSTATUS_EVENT_NOT_POSSIBLE`
- `#define LVSTATUS_EVENT_TOO_MANY_INSTANCES`
- `#define LVSTATUS_FILE_CANNOT_CREATE`
- `#define LVSTATUS_FILE_CANNOT_OPEN`
- `#define LVSTATUS_GENICAM_EXCEPTION`
- `#define LVSTATUS_HANDLE_INVALID`
- `#define LVSTATUS_CHUNK_ADAPTER_NOT_AVAILABLE`
- `#define LVSTATUS_INDEX_OUT_OF_RANGE`
- `#define LVSTATUS_INTERFACE_TOO_MANY_INSTANCES`
- `#define LVSTATUS_INVALID_IN_THIS_MODULE`
- `#define LVSTATUS_ITEM_GROUP_INVALID`
- `#define LVSTATUS_ITEM_INVALID`
- `#define LVSTATUS_NO_CONSTANT_FOR_THIS_ENUMENTRY`
- `#define LVSTATUS_INVALID_ENUMENTRY_ID`
- `#define LVSTATUS_ITEM_NOT_APPLICABLE`
- `#define LVSTATUS_ITEM_NOT_AVAILABLE`
- `#define LVSTATUS_ITEM_NOT_READABLE`
- `#define LVSTATUS_ITEM_NOT_WRITABLE`
- `#define LVSTATUS_NODE_MAP_CANNOT_GET`
- `#define LVSTATUS_NOT_IMPLEMENTED`
- `#define LVSTATUS_PARAMETER_INVALID`
- `#define LVSTATUS_RENDERER_TOO_MANY_INSTANCES`
- `#define LVSTATUS_STREAM_ALREADY_STARTED`
- `#define LVSTATUS_STREAM_ALREADY_STOPPED`
- `#define LVSTATUS_STREAM_TOO_MANY_INSTANCES`
- `#define LVSTATUS_SYSTEM_TOO_MANY_INSTANCES`
- `#define LVSTATUS_DEVICE_NOT_ACCESSIBLE`
- `#define LVSTATUS_DEVICE_NOT_READWRITE`
- `#define LVSTATUS_NOT_SUPPORTED_FOR_THIS_EVENT`
- `#define LVSTATUS_NOT_ENOUGH_BUFFERS`
- `#define LVSTATUS_INSUFFICIENT_BUFFER_SIZE`
- `#define LVSTATUS_INVALID_IP_OR_MAC_ADDRESS_FORMAT`
- `#define LVSTATUS_CANNOT_LOAD_XML`
- `#define LVSTATUS_INSUFFICIENT_STRING_BUFFER_SIZE`
- `#define LVSTATUS_NOT_FOUND`
- `#define LVSTATUS_PARAM_NOT_APPLICABLE`
- `#define LVSTATUS_ENUM_ENTRY_NOT_AVAILABLE`
- `#define LVSTATUS_TIMEOUT`
- `#define LVSTATUS_LUT_NOT_AVAILABLE`



- #define LVSTATUS\_LUT\_UNSUPPORTED\_SIZE
- #define LVSTATUS\_XML\_UNZIP\_ERROR
- #define LVSTATUS\_ACQUISITION\_CANNOT\_BE\_STARTED
- #define LVSTATUS\_ACQUISITION\_CANNOT\_BE\_STOPPED
- #define LVSTATUS\_SETTINGS\_INCOMPATIBLE\_MODEL
- #define LVSTATUS\_SETTINGS\_INCOMPATIBLE\_VERSION
- #define LVSTATUS\_SETTINGS\_INCOMPATIBLE\_ID
- #define LVSTATUS\_BUFFER\_IS\_QUEUED
- #define LVSTATUS\_BUFFER\_NOT\_FILLED
- #define LVSTATUS\_CANNOT\_REOPEN\_LIBRARY
- #define LVSTATUS\_GC\_UNKNOWN
- #define LVSTATUS\_GC\_ERROR
- #define LVSTATUS\_GC\_NOT\_INITIALIZED
- #define LVSTATUS\_GC\_NOT\_IMPLEMENTED
- #define LVSTATUS\_GC\_RESOURCE\_IN\_USE
- #define LVSTATUS\_GC\_ACCESS\_DENIED
- #define LVSTATUS\_GC\_INVALID\_HANDLE
- #define LVSTATUS\_GC\_INVALID\_ID
- #define LVSTATUS\_GC\_NO\_DATA
- #define LVSTATUS\_GC\_INVALID\_PARAMETER
- #define LVSTATUS\_GC\_IO
- #define LVSTATUS\_GC\_TIMEOUT
- #define LVSTATUS\_GC\_ABORT
- #define LVSTATUS\_GC\_INVALID\_BUFFER
- #define LVSTATUS\_GC\_NOT\_AVAILABLE
- #define LVSTATUS\_GC\_INVALID\_ADDRESS
- #define LVSTATUS\_GC\_BUFFER\_TOO\_SMALL
- #define LVSTATUS\_GC\_INVALID\_INDEX
- #define LVSTATUS\_GC\_PARSING\_CHUNK\_DATA
- #define LVSTATUS\_GC\_INVALID\_VALUE
- #define LVSTATUS\_GC\_RESOURCE\_EXHAUSTED
- #define LVSTATUS\_GC\_OUT\_OF\_MEMORY
- #define LVSTATUS\_GC\_BUSY
- #define LVSTATUS\_GC\_CUSTOM\_ID
- #define **LVSTATUS\_GC\_GIGEVERSION\_NOT\_SUPPORTED**
- #define LVSTATUS\_SRCGEN\_TEMPLATE\_NOT\_AVAILABLE
- #define LVSTATUS\_SRCGEN\_SYMBOLIC\_NOT\_AVAILABLE
- #define LVSTATUS\_LICENSE\_NOT\_AVAILABLE
- #define LVSTATUS\_LAST\_ERROR\_NOT\_AVAILABLE
- #define LVSTATUS\_ERROR

## Typedefs

- typedef uint32\_t LvStatus

### 5.49.1 Detailed Description

### 5.49.2 Macro Definition Documentation

#### 5.49.2.1 #define LVSTATUS\_ACQUISITION\_CANNOT\_BE\_STARTED

The AcquisitionStart command cannot be executed. This is probably because the acquisition is already running, or the conditions for starting the acquisition are not satisfied.

#### 5.49.2.2 #define LVSTATUS\_ACQUISITION\_CANNOT\_BE\_STOPPED

The AcquisitionStop command cannot be executed. This is probably because the acquisition already had been stopped, or the conditions for stopping the acquisition are not satisfied.

#### 5.49.2.3 #define LVSTATUS\_AVISAVER\_TOO\_MANY\_INSTANCES

Too many concurrent instances of the AviSaver class.

#### 5.49.2.4 #define LVSTATUS\_BUFFER\_IS\_QUEUED

The operation on the buffer is not possible, the buffer is queued for acquisition.

#### 5.49.2.5 #define LVSTATUS\_BUFFER\_NOT\_FILLED

The buffer was not yet filled with image data.

#### 5.49.2.6 #define LVSTATUS\_CANNOT\_LOAD\_GENTL

The GenTL library cannot be loaded. Check its name specification and compatibility.

#### 5.49.2.7 #define LVSTATUS\_CANNOT\_LOAD\_XML

The XML file with feature definitions could not be loaded. This may indicate a misconfiguration of the SynView system - reinstallation recommended.

#### 5.49.2.8 #define LVSTATUS\_CANNOT\_REOPEN\_LIBRARY

The SynView library cannot be reopened after it was once closed.

#### 5.49.2.9 #define LVSTATUS\_CHUNK\_ADAPTER\_NOT\_AVAILABLE

The chunk data cannot be parsed, because the ChunkAdapter could not be obtained.

#### 5.49.2.10 #define LVSTATUS\_DEVICE\_NOT\_ACCESSIBLE

The device is not accessible. It is either used by other application or, if it is a GigE device, it can be on a different network or there can be an IP address conflict.

#### 5.49.2.11 #define LVSTATUS\_DEVICE\_NOT\_READWRITE

The device (camera) does not permit a read-write access. The application attempted to open the device for controllable or exclusive access, but it is probably currently used by another application.

#### 5.49.2.12 #define LVSTATUS\_DEVICE\_TOO\_MANY\_INSTANCES

Too many concurrent instances of the Device class.

**5.49.2.13 #define LVSTATUS\_DISABLED\_BY\_CALLBACK**

This function is disabled, because the callback function was registered.

**5.49.2.14 #define LVSTATUS\_DISPLAY\_CANNOT\_DISPLAY**

The buffer contents cannot be displayed, either because its pixel format cannot be converted to a displayable one, or the display options are not supported by the operating system.

**5.49.2.15 #define LVSTATUS\_DISPLAY\_LIBRARY\_NOT\_LOADED**

The lv.display library is not loaded. Failure to load this library may be caused by the operating system environment.

**5.49.2.16 #define LVSTATUS\_DISPLAY\_NOT\_OPEN**

The display is not open - an invalid window handle (and/or in Linux display pointer) was set.

**5.49.2.17 #define LVSTATUS\_ENUM\_ENTRY\_INVALID**

The requested enum entry name or value does not exist.

**5.49.2.18 #define LVSTATUS\_ENUM\_ENTRY\_NOT\_AVAILABLE**

The requested enumeration entry is not available.

**5.49.2.19 #define LVSTATUS\_ERROR**

Undefined error. More info in the sv.synview.log file.

**5.49.2.20 #define LVSTATUS\_EVENT\_NOT\_POSSIBLE**

The requested functionality cannot be used for this Event type.

**5.49.2.21 #define LVSTATUS\_EVENT\_TOO\_MANY\_INSTANCES**

Too many concurrent instances of the Event class.

**5.49.2.22 #define LVSTATUS\_FILE\_CANNOT\_CREATE**

Cannot create file. Might be caused by a wrong path specification, insufficient rights or protected file existence.

**5.49.2.23 #define LVSTATUS\_FILE\_CANNOT\_OPEN**

Cannot open file. Might be caused by a wrong path specification, insufficient rights or file nonexistence.

**5.49.2.24 #define LVSTATUS\_GC\_ABORT**

The GenTL library returned the GC\_ERR\_ABORT error. More info in the sv.synview.log file.

#### 5.49.2.25 `#define LVSTATUS_GC_ACCESS_DENIED`

The GenTL library returned the `GC_ERR_ACCESS_DENIED` error. More info in the `sv.synview.log` file.

#### 5.49.2.26 `#define LVSTATUS_GC_BUFFER_TOO_SMALL`

The GenTL library returned the `GC_ERR_BUFFER_TOO_SMALL` error. More info in the `sv.synview.log` file.

#### 5.49.2.27 `#define LVSTATUS_GC_BUSY`

The GenTL library returned the `GC_ERR_BUSY` error. More info in the `sv.synview.log` file.

#### 5.49.2.28 `#define LVSTATUS_GC_CUSTOM_ID`

The GenTL library returned the `GC_ERR_CUSTOM_ID` error. More info in the `sv.synview.log` file.

#### 5.49.2.29 `#define LVSTATUS_GC_ERROR`

The GenTL library returned the `GC_ERR_ERROR` error. More info in the `sv.synview.log` file.

#### 5.49.2.30 `#define LVSTATUS_GC_INVALID_ADDRESS`

The GenTL library returned the `GC_ERR_INVALID_ADDRESS` error. More info in the `sv.synview.log` file.

#### 5.49.2.31 `#define LVSTATUS_GC_INVALID_BUFFER`

The GenTL library returned the `GC_ERR_INVALID_BUFFER` error. More info in the `sv.synview.log` file.

#### 5.49.2.32 `#define LVSTATUS_GC_INVALID_HANDLE`

The GenTL library returned the `GC_ERR_INVALID_HANDLE` error. More info in the `sv.synview.log` file.

#### 5.49.2.33 `#define LVSTATUS_GC_INVALID_ID`

The GenTL library returned the `GC_ERR_INVALID_ID` error. More info in the `sv.synview.log` file.

#### 5.49.2.34 `#define LVSTATUS_GC_INVALID_INDEX`

The GenTL library returned the `GC_ERR_INVALID_INDEX` error. More info in the `sv.synview.log` file.

#### 5.49.2.35 `#define LVSTATUS_GC_INVALID_PARAMETER`

The GenTL library returned the `GC_ERR_INVALID_PARAMETER` error. More info in the `sv.synview.log` file.

#### 5.49.2.36 `#define LVSTATUS_GC_INVALID_VALUE`

The GenTL library returned the `GC_ERR_INVALID_VALUE` error. More info in the `sv.synview.log` file.

**5.49.2.37 #define LVSTATUS\_GC\_IO**

The GenTL library returned the GC\_ERR\_IO error. More info in the sv.synview.log file.

**5.49.2.38 #define LVSTATUS\_GC\_NO\_DATA**

The GenTL library returned the GC\_ERR\_NO\_DATA error. More info in the sv.synview.log file.

**5.49.2.39 #define LVSTATUS\_GC\_NOT\_AVAILABLE**

The GenTL library returned the GC\_ERR\_NOT\_AVAILABLE error. More info in the sv.synview.log file.

**5.49.2.40 #define LVSTATUS\_GC\_NOT\_IMPLEMENTED**

The GenTL library returned the GC\_ERR\_NOT\_IMPLEMENTED error. More info in the sv.synview.log file.

**5.49.2.41 #define LVSTATUS\_GC\_NOT\_INITIALIZED**

The GenTL library returned the GC\_ERR\_NOT\_INITIALIZED error. More info in the sv.synview.log file.

**5.49.2.42 #define LVSTATUS\_GC\_OUT\_OF\_MEMORY**

The GenTL library returned the GC\_ERR\_OUT\_OF\_MEMORY error. More info in the sv.synview.log file.

**5.49.2.43 #define LVSTATUS\_GC\_PARSING\_CHUNK\_DATA**

The GenTL library returned the GC\_ERR\_PARSING\_CHUNK\_DATA error. More info in the sv.synview.log file.

**5.49.2.44 #define LVSTATUS\_GC\_RESOURCE\_EXHAUSTED**

The GenTL library returned the GC\_ERR\_RESOURCE\_EXHAUSTED error. More info in the sv.synview.log file.

**5.49.2.45 #define LVSTATUS\_GC\_RESOURCE\_IN\_USE**

The GenTL library returned the GC\_ERR\_RESOURCE\_IN\_USE error. More info in the sv.synview.log file.

**5.49.2.46 #define LVSTATUS\_GC\_TIMEOUT**

The GenTL library returned the GC\_ERR\_TIMEOUT error. More info in the sv.synview.log file.

**5.49.2.47 #define LVSTATUS\_GC\_UNKNOWN**

The GenTL library returned the GC\_ERR\_UNKNOWN error. More info in the sv.synview.log file.

**5.49.2.48 #define LVSTATUS\_GENICAM\_EXCEPTION**

An exception occurred when using the GenICam GenApi library. More info about the exception nature is recorded to the sv.synview.log file.

**5.49.2.49 #define LVSTATUS\_HANDLE\_INVALID**

An invalid handle was passed to a SynView API function.

**5.49.2.50 #define LVSTATUS\_INDEX\_OUT\_OF\_RANGE**

The specified index is out of range.

**5.49.2.51 #define LVSTATUS\_INSUFFICIENT\_BUFFER\_SIZE**

Some of the allocated buffers are of smaller size, than is the payload size needed for the acquisition. See also [LvStream\\_LvCalcPayloadSize](#).

**5.49.2.52 #define LVSTATUS\_INSUFFICIENT\_STRING\_BUFFER\_SIZE**

The string buffer passed to the function does not have the size big enough to hold the returned string.

**5.49.2.53 #define LVSTATUS\_INTERFACE\_TOO\_MANY\_INSTANCES**

Too many concurrent instances of the Interface class.

**5.49.2.54 #define LVSTATUS\_INVALID\_ENUMENTRY\_ID**

The specified enum entry ID is not a valid SynView constant. Use the enum entry string identifiers to handle enum entries which do not have a SynView constant defined.

**5.49.2.55 #define LVSTATUS\_INVALID\_IN\_THIS\_MODULE**

This function cannot be used in this module.

**5.49.2.56 #define LVSTATUS\_INVALID\_IP\_OR\_MAC\_ADDRESS\_FORMAT**

The IP or MAC Address used in the [LvSetString\(\)](#) has wrong format. The proper format is N.N.N.N for IP address and XX:XX:XX:XX:XX:XX for MAC address, where N is decadic number between 0 and 255, XX is 2 digit hexadecimal number.

**5.49.2.57 #define LVSTATUS\_ITEM\_GROUP\_INVALID**

Invalid FtrGroup specified.

**5.49.2.58 #define LVSTATUS\_ITEM\_INVALID**

Invalid Item ID specified.

**5.49.2.59 #define LVSTATUS\_ITEM\_NOT\_APPLICABLE**

This function is not applicable to this item.

**5.49.2.60 #define LVSTATUS\_ITEM\_NOT\_AVAILABLE**

This function requires availability of specific item, but it is not available. More info in the sv.synview.log file.

**5.49.2.61 #define LVSTATUS\_ITEM\_NOT\_READABLE**

The item is not readable.

**5.49.2.62 #define LVSTATUS\_ITEM\_NOT\_WRITABLE**

The item is not writable.

**5.49.2.63 #define LVSTATUS\_LAST\_ERROR\_NOT\_AVAILABLE**

The last error status could not be recorded. This is most probably caused by too many threads used by the application.

**5.49.2.64 #define LVSTATUS\_LIBRARY\_NOT\_LOADED**

The base library is not loaded. Failure to load the library may be caused by missing DLLs, check the SynView installation.

**5.49.2.65 #define LVSTATUS\_LIBRARY\_NOT\_OPEN**

The SynView library was not open by the [LvOpenLibrary\(\)](#) function.

**5.49.2.66 #define LVSTATUS\_LICENSE\_NOT\_AVAILABLE**

License error. License not available

**5.49.2.67 #define LVSTATUS\_LUT\_NOT\_AVAILABLE**

The lookup table is not available for the current pixel format.

**5.49.2.68 #define LVSTATUS\_LUT\_UNSUPPORTED\_SIZE**

The lookup table has unsupported size.

**5.49.2.69 #define LVSTATUS\_NO\_CONSTANT\_FOR\_THIS\_ENUMENTRY**

SynView constant does not exist for this enum entry. Use the enum entry string identifier to handle it.

**5.49.2.70 #define LVSTATUS\_NODE\_MAP\_CANNOT\_GET**

Cannot obtain a feature node map for the device. More info in the sv.synview.log file.

**5.49.2.71 #define LVSTATUS\_NOT\_ENOUGH\_BUFFERS**

The number of allocated buffers is smaller than required minimum number of buffers. See also the [LvStream\\_↔StreamAnnounceBufferMinimum](#).

**5.49.2.72 #define LVSTATUS\_NOT\_FOUND**

The Interface or Device was not found according to the search criteria.

**5.49.2.73 #define LVSTATUS\_NOT\_IMPLEMENTED**

The functionality is not implemented for the requested parameters.

**5.49.2.74 #define LVSTATUS\_NOT\_SUPPORTED\_FOR\_THIS\_EVENT**

The requested function is not supported by this event type.

**5.49.2.75 #define LVSTATUS\_OK**

No error.

**5.49.2.76 #define LVSTATUS\_PARAM\_NOT\_APPLICABLE**

A parameter passed to the function is not applicable at this function. For example in the [LvSystemFindInterface\(\)](#) the constant `LvFindBy_UserID` is not applicable, because this constant can be used only for devices.

**5.49.2.77 #define LVSTATUS\_PARAMETER\_INVALID**

Invalid parameter passed to a SynView API function (for example an invalid pointer).

**5.49.2.78 #define LVSTATUS\_RENDERER\_TOO\_MANY\_INSTANCES**

Too many concurrent instances of the `Renderer` class.

**5.49.2.79 #define LVSTATUS\_SETTINGS\_INCOMPATIBLE\_ID**

The file with settings were saved with an ID. The ID specified when loading is different.

**5.49.2.80 #define LVSTATUS\_SETTINGS\_INCOMPATIBLE\_MODEL**

The file with settings was created by different remote device vendor/model. This may cause its incompatibility with the current remote device.

**5.49.2.81 #define LVSTATUS\_SETTINGS\_INCOMPATIBLE\_VERSION**

The file with settings was created by different remote device firmware version. This may cause its incompatibility with the current remote device.

**5.49.2.82 #define LVSTATUS\_SRCGEN\_SYMBOLIC\_NOT\_AVAILABLE**

The symbolic for requested item is not available.

**5.49.2.83 #define LVSTATUS\_SRCGEN\_TEMPLATE\_NOT\_AVAILABLE**

The template for requested item is not available.



**5.49.2.84 #define LVSTATUS\_STREAM\_ALREADY\_STARTED**

The stream was already started.

**5.49.2.85 #define LVSTATUS\_STREAM\_ALREADY\_STOPPED**

The stream was already stopped.

**5.49.2.86 #define LVSTATUS\_STREAM\_TOO\_MANY\_INSTANCES**

Too many concurrent instances of the Stream class.

**5.49.2.87 #define LVSTATUS\_SYSTEM\_TOO\_MANY\_INSTANCES**

Too many concurrent instances of the System class.

**5.49.2.88 #define LVSTATUS\_TIMEOUT**

The function has returned because a timeout has expired.

**5.49.2.89 #define LVSTATUS\_XML\_UNZIP\_ERROR**

The XML file with camera remote features could not be extracted from the ZIP file.

**5.49.3 Typedef Documentation****5.49.3.1 typedef uint32\_t LvStatus**

General typedef for the error status. Status values are available as defines prefixed with LVSTATUS\_xxx. Value of 0 (LVSTATUS\_OK) indicates no error. Most SynView functions are returning the status value to indicate the success of the function call. See also the [LvGetErrorMessage\(\)](#) function.

## 5.50 SynView Image Processing Library LvStatus definitions

### Macros

- `#define LVSTATUS_LVIP_INVALID_POINTER`
- `#define LVSTATUS_LVIP_INVALID_SRC_POINTER`
- `#define LVSTATUS_LVIP_INVALID_DST_POINTER`
- `#define LVSTATUS_LVIP_INVALID_PIXEL_FORMAT`
- `#define LVSTATUS_LVIP_IMAGEINFO_NOT_INITIALIZED`
- `#define LVSTATUS_LVIP_MEMORY_ALLOC_FAILED`
- `#define LVSTATUS_LVIP_UNSUPPORTED_BMP_HEADER`
- `#define LVSTATUS_LVIP_BMP_INCOMPATIBLE_PIXEL_FORMAT`
- `#define LVSTATUS_LVIP_BMP_INCOMPATIBLE_LINE_INCREMENT`
- `#define LVSTATUS_LVIP_IMAGEINFO_NOT_EQUAL`
- `#define LVSTATUS_LVIP_UNSUPPORTED`
- `#define LVSTATUS_LVIP_UNSUPPORTED_SRC_PIXEL_FORMAT`
- `#define LVSTATUS_LVIP_UNSUPPORTED_DST_PIXEL_FORMAT`
- `#define LVSTATUS_LVIP_UNSUPPORTED_COLOR_PLANES`
- `#define LVSTATUS_LVIP_UNSUPPORTED_REVERSION`
- `#define LVSTATUS_LVIP_LINEINCREMENT_TOO_BIG`
- `#define LVSTATUS_LVIP_DST_IMG_INFO_INCOMPATIBLE`
- `#define LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_SIZE`
- `#define LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_SIZE_ROTATED`
- `#define LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_PIXEL_FORMAT`
- `#define LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_FLAGS`
- `#define LVSTATUS_LVIP_DST_RECT_OUTSIDE_SRC`
- `#define LVSTATUS_LVIP_SRC_IMAGEINFO_NO_DATA`
- `#define LVSTATUS_LVIP_DST_IMAGEINFO_NO_DATA`
- `#define LVSTATUS_LVIP_NOT_DISPLAYABLE_FORMAT`
- `#define LVSTATUS_LVIP_INVALID_LUT_HANDLE`
- `#define LVSTATUS_LVIP_INVALID_LUT_TYPE`
- `#define LVSTATUS_LVIP_INCOMPATIBLE_REF_PIXEL_FORMAT`
- `#define LVSTATUS_LVIP_INCOMPATIBLE_REF_FLAGS`
- `#define LVSTATUS_LVIP_CANNOT_OPEN_READ_FILE`
- `#define LVSTATUS_LVIP_CANNOT_CREATE_WRITE_FILE`
- `#define LVSTATUS_LVIP_TIFF_CONTENTS_INVALID`
- `#define LVSTATUS_LVIP_BMP_CONTENTS_INVALID`
- `#define LVSTATUS_LVIP_NOT_BAYER_PIXEL_FORMAT`
- `#define LVSTATUS_LVIP_JPEG_SAVE_FAILED`
- `#define LVSTATUS_LVIP_JPEG_LOAD_FAILED`

### 5.50.1 Detailed Description

### 5.50.2 Macro Definition Documentation

#### 5.50.2.1 `#define LVSTATUS_LVIP_BMP_CONTENTS_INVALID`

The contents of the BMP file is in the invalid or in the unsupported form. You are trying to read a BMP file which has different (or invalid) format than which is supported by this library.

#### 5.50.2.2 `#define LVSTATUS_LVIP_BMP_INCOMPATIBLE_LINE_INCREMENT`

Color format has incompatible line increment to BMP possibilities. The BMP has to have line increment aligned to 4 bytes.

### 5.50.2.3 #define LVSTATUS\_LVIP\_BMP\_INCOMPATIBLE\_PIXEL\_FORMAT

Source pixel format is incompatible to BITMAP pixel format possibilities. BITMAP could be created only using one of these pixel formats:

```
LVIP_PIXEL_FORMAT_MONO8
LVIP_PIXEL_FORMAT_RGB555_PACKED
LVIP_PIXEL_FORMAT_RGB565_PACKED
LVIP_PIXEL_FORMAT_RGB8_PACKED
LVIP_PIXEL_FORMAT_RGBA8_PACKED
```

If you have the image in other pixel format, you should convert the image by the `LVIP_ConvertToPixelFormat()` function.

### 5.50.2.4 #define LVSTATUS\_LVIP\_CANNOT\_CREATE\_WRITE\_FILE

A system error occurred when trying to create or write to the file. Possible reasons could be an invalid path and/or file name, without rights to be created/written.

### 5.50.2.5 #define LVSTATUS\_LVIP\_CANNOT\_OPEN\_READ\_FILE

A system error occurred when trying to open or read the file. Possible reasons could be a wrong path and/or file name, the file currently locked or without rights to be opened/read or the file does not exist yet.

### 5.50.2.6 #define LVSTATUS\_LVIP\_DST\_IMAGEINFO\_NO\_DATA

The destination image info structure has no data. The destination image info has no image data - it means that the destination image info does not point to any valid image data and it is not permitted to allocate the buffer automatically (the `LvipOption_ReallocateDst` flag was not specified).

### 5.50.2.7 #define LVSTATUS\_LVIP\_DST\_IMG\_INFO\_INCOMPATIBLE

Destination image info is incompatible to expected output format. You are trying to call a function and the supplied destination image info has other than expected contents.

Possible reason is that you don't specify the `LvipOption_ReallocateDst` flag, so if the function has different output format than specified in the destination image info, the image info could not be reallocated and this error happens.

Note that some functions may require different destination parameters than expected, the size of the destination image in the `LVIP_CopyArea()` function depends on whether the desired rectangle lies fully in the image or not.

### 5.50.2.8 #define LVSTATUS\_LVIP\_DST\_RECT\_OUTSIDE\_SRC

The specified rectangle is outside of source image data. It means that you are trying to copy an area which not exist in the source image. To create a destination image, the rectangle must at least partially overlap the source image.

### 5.50.2.9 #define LVSTATUS\_LVIP\_IMAGEINFO\_NOT\_EQUAL

Image info not equal.

### 5.50.2.10 #define LVSTATUS\_LVIP\_IMAGEINFO\_NOT\_INITIALIZED

The `LvipImgInfo` parameter is not initialized. The supplied `LvipImgInfo` parameter has invalid contents. Either initialize it with the `LVIP_InitImgInfo()` function or setup in your code all the members to appropriate values.

#### 5.50.2.11 #define LVSTATUS\_LVIP\_INCOMPATIBLE\_REF\_FLAGS

The reference image has incompatible flags. When using the reference image info (in the `LVIP_ApplyShadingCorrection()` function), it must have compatible flags as the source image. For example: [LvimgAttr\\_BottomUp](#) flags must be same.

#### 5.50.2.12 #define LVSTATUS\_LVIP\_INCOMPATIBLE\_REF\_PIXEL\_FORMAT

The reference image is in incompatible pixel format. When using the reference image info (in the `LVIP_ApplyShadingCorrection()` function), it must be in the same pixel format as the source image.

#### 5.50.2.13 #define LVSTATUS\_LVIP\_INCOMPATIBLE\_SRC\_AND\_DST\_FLAGS

Source and destination image info have incompatible flags. Some functions need to have both of source and destination image info flags compatible. This applies for example for the bitmap orientation (top-down versus bottom-up).

#### 5.50.2.14 #define LVSTATUS\_LVIP\_INCOMPATIBLE\_SRC\_AND\_DST\_PIXEL\_FORMAT

The source and destination image info have different pixel format. You are trying to call some function which expects that both of source and destination image info has the same pixel format. But the destination image info has different pixel format set and it is not permitted to change this (by the [LvipOption\\_ReallocateDst](#) flag).

#### 5.50.2.15 #define LVSTATUS\_LVIP\_INCOMPATIBLE\_SRC\_AND\_DST\_SIZE

Destination image info size is different from the source image info size. It means that the function expects that the source and destination image info size is the same. Because it is not, and the destination image info cannot be changed using the [LvipOption\\_ReallocateDst](#) flag, this error happens.

#### 5.50.2.16 #define LVSTATUS\_LVIP\_INCOMPATIBLE\_SRC\_AND\_DST\_SIZE\_ROTATED

Rotated destination image info size is different to the expected size. It means that the function expects that the source width of image info has to be the same to the height of the destination image info (and vice versa) - and it is not - and additionally destination image info cannot be changed using the [LvipOption\\_ReallocateDst](#) flag.

#### 5.50.2.17 #define LVSTATUS\_LVIP\_INVALID\_DST\_POINTER

Pointer to the destination data image info or its data is invalid.

#### 5.50.2.18 #define LVSTATUS\_LVIP\_INVALID\_LUT\_HANDLE

Invalid LUT handle passed as a parameter of the function. The LUT is in incompatible format which couldn't be used in this function.

General this happens when trying to use some Bayer-decoding function and a non-Bayer LUT is used or the LUT has been created for the different pixel format.

#### 5.50.2.19 #define LVSTATUS\_LVIP\_INVALID\_LUT\_TYPE

Invalid LUT type passed as a parameter of the function. The LUT is in incompatible format which couldn't be used in this function.

General this happens when trying to use some Bayer-decoding function and a non-Bayer LUT is used or the LUT has been created for the different pixel format.

## 5.50.2.20 #define LVSTATUS\_LVIP\_INVALID\_PIXEL\_FORMAT

The source or destination image info has invalid or unexpected pixel format or the dwPixelIncrement value in the of [LvIpImgInfo](#) structure. Another reason when this error code could be retrieved is when using the [LVIP\\_BmpInfoToImgInfo\(\)](#) function with BITMAPINFOHEADER containing in its *biBitCount* member a different value than is supported. Supported values in BITMAPINFOHEADER are:

```
8          LVIP_PIXEL_FORMAT_MONO8
16         LVIP_PIXEL_FORMAT_RGB555_PACKED or LVIP_PIXEL_FORMAT_RGB565_PACKED
           (depends on biCompression member of the BITMAPINFOHEADER)
24         LVIP_PIXEL_FORMAT_RGB8_PACKED
32         LVIP_PIXEL_FORMAT_RGBA8_PACKED
```

## 5.50.2.21 #define LVSTATUS\_LVIP\_INVALID\_POINTER

Invalid pointer. One of the pointers used by the function is NULL or invalid.

## 5.50.2.22 #define LVSTATUS\_LVIP\_INVALID\_SRC\_POINTER

Pointer to the source data image info or its data is invalid.

## 5.50.2.23 #define LVSTATUS\_LVIP\_JPEG\_LOAD\_FAILED

Loading from JPEG failed. More info in the LOG file.

## 5.50.2.24 #define LVSTATUS\_LVIP\_JPEG\_SAVE\_FAILED

Saving to JPEG failed. More info in the LOG file.

## 5.50.2.25 #define LVSTATUS\_LVIP\_LINEINCREMENT\_TOO\_BIG

Image line increment is too big. This some functions support only limited line increment. This currently applies only to [LVIP\\_Deinterlace\(\)](#) function, which has a limit of 2048 \* 32bit RGB image - it means that line increment could be more than 8192 bytes.

## 5.50.2.26 #define LVSTATUS\_LVIP\_MEMORY\_ALLOC\_FAILED

Memory allocation failed. This error code happens when the operating system does not allow to allocate any new memory to this library.

When this error happens, there is a critical insufficiency of memory; it might indicate a huge memory leak, typically caused by not deallocating used images when processing in a loop. Another cause could be big image dimensions resulting in an attempt to allocate a huge memory amount.

Typically this error code could be retrieved from [LVIP\\_AllocateImageData\(\)](#) or [LVIP\\_SaveToTiff\(\)](#) functions.

## 5.50.2.27 #define LVSTATUS\_LVIP\_NOT\_BAYER\_PIXEL\_FORMAT

The PixelFormat is not BayerArray

## 5.50.2.28 #define LVSTATUS\_LVIP\_NOT\_DISPLAYABLE\_FORMAT

The image isn't in the displayable format. If there is a need to display an image, there is a need to have this image in a displayable format (in Windows one of the BMP pixel formats and the line increment aligned to 4 bytes). The image does not have such format and the automatic conversion to a displayable format was not enabled (using the third parameter of the [LVIP\\_DisplayImage\(\)](#) function and optionally the [LvIpOption\\_ReallocateDst](#) flag).

#### 5.50.2.29 `#define LVSTATUS_LVIP_SRC_IMAGEINFO_NO_DATA`

The source image info structure has no data. The source image info has no image data - it means that the source image info does not point to any valid image data.

Use `LVIP_AllocateImageData()` function to allocate the buffer for the image or point the `pData` member(s) to a valid image.

#### 5.50.2.30 `#define LVSTATUS_LVIP_TIFF_CONTENTS_INVALID`

The contents of the TIFF file is in the invalid or in the unsupported form. You are probably trying to read a TIFF file, which was not created by this library (see the `LVIP_SaveToTiff()` function)

#### 5.50.2.31 `#define LVSTATUS_LVIP_UNSUPPORTED`

The requested function or format is not supported. If you are not sure which functionality is not supported, see New Electronic Technology Log Messages Receiver application for details.

#### 5.50.2.32 `#define LVSTATUS_LVIP_UNSUPPORTED_BMP_HEADER`

BMP header is unsupported. Some members of `BITMAPINFOHEADER` (part of `BITMAPINFO`) have unexpected or unsupported values. Check if the `BITMAPINFO` has correct data and that is correctly filled up. Members *biSize* and *biCompression* have to be correctly filled up. The *biCompression* member has to be filled up with `BI_RGB` or `BI_BITFIELDS` values.

#### 5.50.2.33 `#define LVSTATUS_LVIP_UNSUPPORTED_COLOR_PLANES`

The image uses color planes and the called function doesn't support it.

#### 5.50.2.34 `#define LVSTATUS_LVIP_UNSUPPORTED_DST_PIXEL_FORMAT`

Unsupported pixel format of the destination. See documentation of function which returns this error code for supported destination pixel formats.

#### 5.50.2.35 `#define LVSTATUS_LVIP_UNSUPPORTED_REVERSION`

The image uses reversion and the called function doesn't support it.

#### 5.50.2.36 `#define LVSTATUS_LVIP_UNSUPPORTED_SRC_PIXEL_FORMAT`

Unsupported pixel format of the source. See documentation of the function which returns this error code for supported source pixel formats.

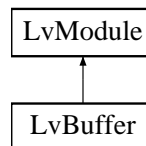
# Chapter 6

## Class Documentation

### 6.1 LvBuffer Class Reference

```
#include <sv.synview.class.h>
```

Inheritance diagram for LvBuffer:



#### Public Member Functions

- [LvStatus AttachProcessBuffer](#) (void \*pDataPointer, size\_t DataSize)
- [LvStatus Queue](#) ()
- [LvStatus ParseChunkData](#) (bool UpdateLayout=false)
- [LvStatus SaveImageToBmpFile](#) (const char \*pFileName)
- [LvStatus SaveImageToJpgFile](#) (const char \*pFileName, uint32\_t Quality)
- [LvStatus SaveImageToTifFile](#) (const char \*pFileName, uint32\_t Options=0)
- [LvStatus GetImgInfo](#) (LvImgInfo &ImgInfo, uint32\_t Options=0)
- [LvStatus GetLastPaintRect](#) (int32\_t \*pX, int32\_t \*pY, int32\_t \*pWidth, int32\_t \*pHeight)
- [LvStatus UniCalculateWhiteBalance](#) ()
- [LvHBuffer GetHandle](#) ()
- void \* [GetUserPtr](#) ()

#### Static Public Member Functions

- static [LvStatus Open](#) (LvStream \*pStream, void \*pDataPointer, size\_t DataSize, void \*pUserPointer, uint32\_t Options, LvBuffer \*&pBuffer)
- static [LvStatus Close](#) (LvBuffer \*&pBuffer)

#### Additional Inherited Members

##### 6.1.1 Detailed Description

The [LvBuffer](#) class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class

instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

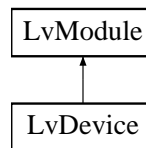
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

## 6.2 LvDevice Class Reference

```
#include <sv.synview.class.h>
```

Inheritance diagram for LvDevice:



### Public Member Functions

- [LvStatus GetNumberOfStreams](#) (uint32\_t \*pNumberOfStreams)
- [LvStatus GetStreamId](#) (uint32\_t Index, char \*pStreamId, size\_t Size)
- [LvStatus GetStreamIdSize](#) (uint32\_t Index, size\_t \*pSize)
- [LvStatus GetStreamId](#) (uint32\_t Index, std::string &sStreamId)
- [LvStatus AcquisitionStart](#) (uint32\_t Options=0)
- [LvStatus AcquisitionStop](#) (uint32\_t Options=0)
- [LvStatus AcquisitionAbort](#) (uint32\_t Options=0)
- [LvStatus AcquisitionArm](#) (uint32\_t Options=0)
- [LvStatus SaveSettings](#) (const char \*pId, const char \*pFileName, uint32\_t Options)
- [LvStatus LoadSettings](#) (const char \*pId, const char \*pFileName, uint32\_t Options)
- [LvStatus LoadBatch](#) (const char \*pFileName)
- [LvStatus UniSetLut](#) (LvLUTSelector Selector, void \*pLUT, size\_t Size, uint32\_t Options=0)
- [LvStatus UniGetLut](#) (LvLUTSelector Selector, void \*pLUT, size\_t Size, uint32\_t Options=0)
- [LvStatus FwGetFilePattern](#) (uint32\_t Which, char \*pFilePattern, size\_t Size)
- [LvStatus FwLoad](#) (uint32\_t Which, const char \*pFilePath)
- [LvStatus FwGetLoadStatus](#) (uint32\_t Which, uint32\_t \*pCurrentByteCount, bool \*plsLoading)
- [LvStatus OpenStream](#) (const char \*pStreamId, LvStream \*pStream)
- [LvStatus CloseStream](#) (LvStream \*pStream)
- [LvStatus OpenEvent](#) (LvEventType EventType, LvEvent \*pEvent)
- [LvStatus CloseEvent](#) (LvEvent \*pEvent)
- [LvHDevice GetHandle](#) ()

### Static Public Member Functions

- static [LvStatus Open](#) (LvInterface \*pInterface, const char \*pDeviceId, LvDevice \*pDevice, LvDeviceAccess Access=LvDeviceAccess\_Exclusive)
- static [LvStatus Close](#) (LvDevice \*pDevice)



## Additional Inherited Members

### 6.2.1 Detailed Description

The [LvDevice](#) class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

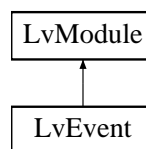
The documentation for this class was generated from the following file:

- `include/sv.synview.class.h`

## 6.3 LvEvent Class Reference

```
#include <sv.synview.class.h>
```

Inheritance diagram for LvEvent:



### Public Member Functions

- [LvStatus Kill](#) ()
- [LvStatus Flush](#) ()
- [LvStatus WaitAndGetData](#) (void \*pBuffer, size\_t \*pSize, uint32\_t Timeout=0xFFFFFFFF)
- [LvStatus WaitAndGetNewBuffer](#) (LvBuffer \*&pBuffer, uint32\_t Timeout=0xFFFFFFFF)
- [LvStatus GetDataInfo](#) (void \*pInBuffer, size\_t InSize, LvEventDataInfo Info, void \*pBuffer, size\_t \*pSize, LvEventDataInfo \*pType=NULL, int32\_t Param=0)
- [LvStatus PutData](#) (void \*pBuffer, size\_t Size)
- [LvStatus SetCallback](#) (LvEventCallbackFunc pFunction, void \*pUserParam)
- [LvStatus SetCallbackNewBuffer](#) (LvEventCallbackNewBufferFunc pFunction, void \*pUserParam)
- [LvStatus StartThread](#) ()
- [LvStatus StopThread](#) ()
- bool [CallbackMustExit](#) ()
- [LvHEvent GetHandle](#) ()

### Static Public Member Functions

- static [LvStatus Open](#) (LvSystem \*pSystem, LvEventType EventType, LvEvent \*&pEvent)
- static [LvStatus Open](#) (LvDevice \*pDevice, LvEventType EventType, LvEvent \*&pEvent)
- static [LvStatus Open](#) (LvStream \*pStream, LvEventType EventType, LvEvent \*&pEvent)
- static [LvStatus Close](#) (LvEvent \*&pEvent)

## Additional Inherited Members

### 6.3.1 Detailed Description

The [LvEvent](#) class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

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

- include/sv.synview.class.h

## 6.4 LvException Class Reference

```
#include <sv.synview.class.h>
```

### Public Member Functions

- **LvException** (const char \*pMessage, [LvStatus](#) Number) throw ()
- **LvException** (const [LvException](#) &e) throw ()
- const char \* **Message** () throw ()
- **LvStatus Number** () throw ()

### 6.4.1 Detailed Description

Undefine LV\_USE\_STDLIB in case you do not want to use the standard template library. If LV\_USE\_STDLIB is defined, the functions returning strings are available also overloaded having a std::string& parameter for returning the string.

Define LV\_USE\_STDEXCEPTION in case you want to use the exception class from the standard library instead of [LvException](#).

Call `LvLibrary::SetThrowErrorEnable(true)` in case you want to use the C++ exceptions of the [LvException](#) type to be thrown when the function returns a status not equal to LVSTATUS\_OK. Then you can use the error handling in the form shown in the example below:

```
try
{
    m_pDevice->AcquisitionStart();
    // ... and more SynView API calls, without checking the return value
}
catch (LvException e)
{
    DisplayErrorMsg(e.Message(), e.Number());
    return;
}
```

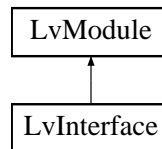
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

## 6.5 LvInterface Class Reference

```
#include <sv.synview.class.h>
```

Inheritance diagram for LvInterface:



## Public Member Functions

- [LvStatus UpdateDeviceList](#) (uint32\_t Timeout=0xFFFFFFFF)
- [LvStatus GetNumberOfDevices](#) (uint32\_t \*pDevices)
- [LvStatus GetDeviceId](#) (uint32\_t Index, char \*pDeviceId, size\_t Size)
- [LvStatus GetDeviceIdSize](#) (uint32\_t Index, size\_t \*pSize)
- [LvStatus GetDeviceId](#) (uint32\_t Index, std::string &sDeviceId)
- [LvStatus FindDevice](#) (LvFindBy FindBy, const char \*pFindStr, char \*pDeviceId, size\_t Size)
- [LvStatus FindDevice](#) (LvFindBy FindBy, const char \*pFindStr, std::string &sDeviceId)
- [LvHInterface GetHandle](#) ()
- [LvStatus OpenDevice](#) (const char \*pDeviceId, [LvDevice](#) \*&pDevice, [LvDeviceAccess](#) Access=LvDevice↔ Access\_Exclusive)
- [LvStatus CloseDevice](#) ([LvDevice](#) \*&pDevice)

## Static Public Member Functions

- static [LvStatus Open](#) ([LvSystem](#) \*pSystem, const char \*pInterfaceld, [LvInterface](#) \*&pInterface)
- static [LvStatus Close](#) ([LvInterface](#) \*&pInterface)

## Additional Inherited Members

### 6.5.1 Detailed Description

The [LvInterface](#) class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

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

- include/sv.synview.class.h

## 6.6 LvIplmgInfo Struct Reference

```
#include <sv.synview.defs.h>
```

### Public Attributes

- uint32\_t [StructSize](#)
- uint32\_t [Width](#)
- uint32\_t [Height](#)
- uint32\_t [PixelFormat](#)
- uint32\_t [Attributes](#)
- uint32\_t [BytesPerPixel](#)

- `uint32_t` [LinePitch](#)
- `uint8_t*` [pData](#)
- `uint8_t*` [pDataR](#)
- `uint8_t*` [pDataG](#)
- `uint8_t*` [pDataB](#)

### 6.6.1 Detailed Description

Image Info structure. Each image handled by the library must be described by the [LvipImgInfo](#) structure. Although you can set the Image Info members directly, it is highly recommended to use the [LvipInitImgInfo\(\)](#) function for the structure initialization.

### 6.6.2 Member Data Documentation

#### 6.6.2.1 `uint32_t` [LvipImgInfo::Attributes](#)

Image attributes. OR-ed definitions from [LvipImgAttr](#) definitions.

#### 6.6.2.2 `uint32_t` [LvipImgInfo::BytesPerPixel](#)

Size of one pixel in bytes.

#### 6.6.2.3 `uint32_t` [LvipImgInfo::Height](#)

Height of the image in pixels.

#### 6.6.2.4 `uint32_t` [LvipImgInfo::LinePitch](#)

Size of one line in bytes.

Example:

```
8-bit mono image: LineIncrement = Width;
24-bit RGB image: LineIncrement = Width * 3;
```

However, when the [LvipImgAttr\\_DWordAligned](#) attribute is used, the line increment must be rounded up to whole double-words, so the calculation would then look like this:

```
8-bit mono image: LineIncrement = (Width+3)/4 * 4;
24-bit RGB image: LineIncrement = ((Width*3)+3)/4 * 4;
```

#### 6.6.2.5 `uint8_t*` [LvipImgInfo::pData](#)

Pointer to image data. If color planes are not used, this member points to the data of the image. Use [LvipAllocateImageData\(\)](#) to allocate the buffer for the image. If you set the pointer to an existing image, which is not owned by this [LvipImgInfo](#), use the [LvipImgAttr\\_NotDataOwner](#) attribute.

#### 6.6.2.6 `uint8_t*` [LvipImgInfo::pDataB](#)

If color planes are used, this member points to the Blue plane data of the image. Use [LvipAllocateImageData\(\)](#) to allocate the buffer for the image. If you set the pointer to an existing image, which is not owned by this [LvipImgInfo](#), use the [LvipImgAttr\\_NotDataOwner](#) attribute.

### 6.6.2.7 uint8\_t\* LvIplmgInfo::pDataG

If color planes are used, this member points to the Green plane data of the image. Use [LvIpAllocateImageData\(\)](#) to allocate the buffer for the image. If you set the pointer to an existing image, which is not owned by this [LvIplmgInfo](#), use the [LvIplmgAttr\\_NotDataOwner](#) attribute.

### 6.6.2.8 uint8\_t\* LvIplmgInfo::pDataR

If color planes are used, this member points to the Red plane data of the image. Use [LvIpAllocateImageData\(\)](#) to allocate the buffer for the image. If you set the pointer to an existing image, which is not owned by this [LvIplmgInfo](#), use the [LvIplmgAttr\\_NotDataOwner](#) attribute.

### 6.6.2.9 uint32\_t LvIplmgInfo::PixelFormat

Pixel format of the image which is saved in this structure. One of the [LvPixelFormat](#). In case of color planes, the pixel format applies to one plane, so use only the MONO formats for the planes. For example for 3x8-bit RGB use the [LvPixelFormat\\_Mono8](#) format.

### 6.6.2.10 uint32\_t LvIplmgInfo::StructSize

Size of image info structure. Should be set to the `sizeof(LvIplmgInfo)`. This member may be used in the future versions for the compatibility check.

### 6.6.2.11 uint32\_t LvIplmgInfo::Width

Width of the image in pixels.

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

- `include/sv.synview.defs.h`

## 6.7 LvLibrary Class Reference

```
#include <sv.synview.class.h>
```

### Static Public Member Functions

- static `uint32_t` [GetVersion](#) ()
- static `LvStatus` [OpenLibrary](#) ()
- static `LvStatus` [CloseLibrary](#) ()
- static void [GetErrorMessage](#) (`LvStatus` Error, char \*pMessage, size\_t Size)
- static std::string [GetErrorMessage](#) (`LvStatus` Error)
- static void [GetLastErrorMessage](#) (char \*pMessage, size\_t Size)
- static std::string [GetLastErrorMessage](#) ()
- static void [Log](#) (const char \*pLogMessage)
- static `LvStatus` [GetLibInfo](#) (`LvEnum` Info, int32\_t \*pInfo, int32\_t Param=0)
- static `LvStatus` [GetLibInfoStr](#) (`LvEnum` Info, char \*pInfoStr, size\_t Size, int32\_t Param=0)
- static `LvStatus` [GetLibInfoStrSize](#) (`LvEnum` Info, size\_t \*pSize, int32\_t Param=0)
- static `LvStatus` [GetLibInfoStr](#) (`LvEnum` Info, std::string &sInfo, int32\_t Param=0)
- static `LvStatus` [UpdateSystemList](#) ()
- static `LvStatus` [GetNumberOfSystems](#) (uint32\_t \*pNumberOfSystems)
- static `LvStatus` [GetSystemId](#) (uint32\_t Index, char \*pSystemId, size\_t Size)

- static [LvStatus GetSystemIdSize](#) (uint32\_t Index, size\_t \*pSize)
- static [LvStatus GetSystemId](#) (uint32\_t Index, std::string &sSystemId)
- static void [SetThrowErrorEnable](#) (bool bEnable)

### 6.7.1 Detailed Description

The [LvLibrary](#) class has all its members static and it is not possible to create the instance of this class. You can consider its methods as global functions.

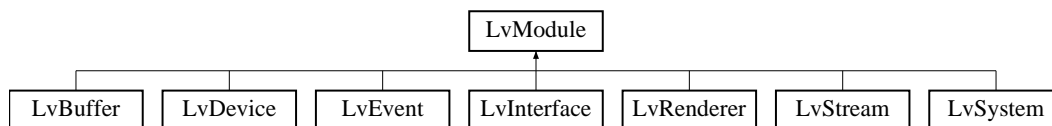
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

## 6.8 LvModule Class Reference

```
#include <sv.synview.class.h>
```

Inheritance diagram for LvModule:



### Public Member Functions

- [LvStatus GetNumFeatures](#) (LvFtrGroup FtrGroup, uint32\_t \*pNumFeatures)
- [LvStatus GetFeatureAt](#) (LvFtrGroup FtrGroup, uint32\_t Index, [LvFeature](#) \*pFeature, uint32\_t \*pLevel=NULL)
- [LvStatus GetFeatureByName](#) (LvFtrGroup FtrGroup, const char \*pName, [LvFeature](#) \*pFeature)
- bool [IsImplemented](#) ([LvFeature](#) Feature)
- bool [IsImplementedByName](#) ([LvEnum](#) FeatureGroup, const char \*pName)
- bool [IsAvailable](#) ([LvFeature](#) Feature)
- bool [IsAvailableByName](#) ([LvEnum](#) FeatureGroup, const char \*pName)
- bool [IsReadable](#) ([LvFeature](#) Feature)
- bool [IsWritable](#) ([LvFeature](#) Feature)
- bool [IsAvailableEnumEntry](#) ([LvFeature](#) Feature, [LvEnum](#) EnumEntry)
- bool [IsImplementedEnumEntry](#) ([LvFeature](#) Feature, [LvEnum](#) EnumEntry)
- [LvStatus GetType](#) ([LvFeature](#) Feature, [LvFtrType](#) \*pFtrType, [LvFtrGui](#) \*pFtrGui=NULL, [LvFtrGroup](#) \*pFtrGroup=NULL)
- [LvStatus GetBool](#) ([LvFeature](#) Feature, bool \*pValue)
- [LvStatus SetBool](#) ([LvFeature](#) Feature, bool Value)
- [LvStatus GetInt32](#) ([LvFeature](#) Feature, int32\_t \*pValue)
- [LvStatus SetInt32](#) ([LvFeature](#) Feature, int32\_t Value)
- [LvStatus GetInt32Range](#) ([LvFeature](#) Feature, int32\_t \*pMinValue, int32\_t \*pMaxValue, int32\_t \*pIncrement)
- [LvStatus GetInt64](#) ([LvFeature](#) Feature, int64\_t \*pValue)
- [LvStatus SetInt64](#) ([LvFeature](#) Feature, int64\_t Value)
- [LvStatus GetInt64Range](#) ([LvFeature](#) Feature, int64\_t \*pMinValue, int64\_t \*pMaxValue, int64\_t \*pIncrement)
- [LvStatus GetInt](#) ([LvFeature](#) Feature, int64\_t \*pValue)
- [LvStatus SetInt](#) ([LvFeature](#) Feature, int64\_t Value)
- [LvStatus GetIntRange](#) ([LvFeature](#) Feature, int64\_t \*pMinValue, int64\_t \*pMaxValue, int64\_t \*pIncrement)
- [LvStatus GetFloat](#) ([LvFeature](#) Feature, double \*pValue)
- [LvStatus SetFloat](#) ([LvFeature](#) Feature, double Value)

- [LvStatus GetFloatRange](#) ([LvFeature](#) Feature, double \*pMinValue, double \*pMaxValue, double \*pIncrement=NULL)
- [LvStatus GetString](#) ([LvFeature](#) Feature, char \*pValue, size\_t Size)
- [LvStatus GetStringSize](#) ([LvFeature](#) Feature, size\_t \*pSize)
- [LvStatus GetString](#) ([LvFeature](#) Feature, std::string &sValue)
- [LvStatus SetString](#) ([LvFeature](#) Feature, const char \*pValue)
- [LvStatus GetBuffer](#) ([LvFeature](#) Feature, void \*pBuffer, size\_t Size)
- [LvStatus GetBufferSize](#) ([LvFeature](#) Feature, size\_t \*pSize)
- [LvStatus SetBuffer](#) ([LvFeature](#) Feature, void \*pBuffer, size\_t Size)
- [LvStatus GetPtr](#) ([LvFeature](#) Feature, void \*\*ppValue)
- [LvStatus SetPtr](#) ([LvFeature](#) Feature, void \*pValue)
- [LvStatus GetEnum](#) ([LvFeature](#) Feature, [LvEnum](#) \*pValue)
- [LvStatus SetEnum](#) ([LvFeature](#) Feature, [LvEnum](#) Value)
- [LvStatus GetEnumStr](#) ([LvFeature](#) Feature, char \*pSymbolicName, size\_t Size)
- [LvStatus GetEnumStr](#) ([LvFeature](#) Feature, std::string &sSymbolicName)
- [LvStatus SetEnumStr](#) ([LvFeature](#) Feature, const char \*pSymbolicName)
- [LvStatus GetEnumValByStr](#) ([LvFeature](#) Feature, const char \*pSymbolicName, [LvEnum](#) \*pValue, [LvFtrAccess](#) \*pFtrAccess=NULL)
- [LvStatus GetEnumStrByVal](#) ([LvFeature](#) Feature, [LvEnum](#) Value, char \*pSymbolicName, size\_t SymbolicNameSize, [LvFtrAccess](#) \*pFtrAccess=NULL)
- [LvStatus GetEnumStrByVal](#) ([LvFeature](#) Feature, [LvEnum](#) Value, std::string &sSymbolicName, [LvFtrAccess](#) \*pFtrAccess=NULL)
- [LvStatus CmdExecute](#) ([LvFeature](#) Feature, uint32\_t Timeout=0)
- [LvStatus CmdIsDone](#) ([LvFeature](#) Feature, bool \*pIsDone)
- [LvStatus GetAccess](#) ([LvFeature](#) Feature, [LvFtrAccess](#) \*pFtrAccess)
- [LvStatus GetVisibility](#) ([LvFeature](#) Feature, [LvFtrVisibility](#) \*pFtrVisibility)
- [LvStatus GetInfo](#) ([LvFeature](#) Feature, [LvFtrInfo](#) FtrInfo, int32\_t \*pInfo, int32\_t Param=0)
- [LvStatus GetInfoStr](#) ([LvFeature](#) Feature, [LvFtrInfo](#) FtrInfo, char \*pInfoStr, size\_t Size, int32\_t Param=0)
- [LvStatus GetInfoStrSize](#) ([LvFeature](#) Feature, [LvFtrInfo](#) FtrInfo, size\_t \*pSize, int32\_t Param=0)
- [LvStatus GetInfoStr](#) ([LvFeature](#) Feature, [LvFtrInfo](#) FtrInfo, std::string &sInfoStr, int32\_t Param=0)
- [LvStatus RegisterFeatureCallback](#) ([LvFeature](#) Feature, [LvFeatureCallbackFunc](#) pFunction, void \*pUserParam=NULL, void \*pFeatureParam=NULL)
- [LvStatus StartPollingThread](#) (uint32\_t PollingTime=1000, bool PollChildren=false)
- [LvStatus StopPollingThread](#) ()
- [LvStatus Poll](#) ()

## Protected Attributes

- [LvHModule m\\_hModule](#)

### 6.8.1 Detailed Description

The base class for all modules. It provides methods for manipulating the features, if the module provides any. This class cannot be instantiated, it only serves as a base class.

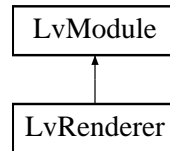
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

## 6.9 LvRenderer Class Reference

```
#include <sv.synview.class.h>
```

Inheritance diagram for LvRenderer:



### Public Member Functions

- [LvStatus SetWindow](#) (void \*pDisplay, int64\_t hWindow)
- [LvStatus DisplayImage](#) (LvBuffer \*pBuffer, uint32\_t RenderFlags=0)
- [LvStatus Repaint](#) (uint32\_t RenderFlags=0)
- [LvHRenderer GetHandle](#) ()

### Static Public Member Functions

- static [LvStatus Open](#) (LvStream \*pStream, LvRenderer \*&pRenderer)
- static [LvStatus Close](#) (LvRenderer \*&pRenderer)

### Additional Inherited Members

#### 6.9.1 Detailed Description

The [LvRenderer](#) class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

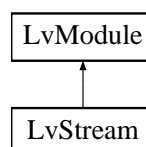
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

## 6.10 LvStream Class Reference

```
#include <sv.synview.class.h>
```

Inheritance diagram for LvStream:





## Public Member Functions

- [LvStatus GetBufferAt](#) (uint32\_t BufferIndex, [LvBuffer](#) \*&pBuffer)
- [LvStatus FlushQueue](#) ([LvQueueOperation](#) Operation)
- [LvStatus Start](#) (uint32\_t StartFlags=0x00000000, uint32\_t ImagesToAcquire=0xFFFFFFFF)
- [LvStatus Stop](#) (uint32\_t StopFlags=0x00000000)
- [LvHStream GetHandle](#) ()
- [LvStatus OpenBuffer](#) (void \*pDataPointer, size\_t DataSize, void \*pUserPointer, uint32\_t Options, [LvBuffer](#) \*&pBuffer)
- [LvStatus CloseBuffer](#) ([LvBuffer](#) \*&pBuffer)
- [LvStatus OpenEvent](#) ([LvEventType](#) EventType, [LvEvent](#) \*&pEvent)
- [LvStatus CloseEvent](#) ([LvEvent](#) \*&pEvent)
- [LvStatus OpenRenderer](#) ([LvRenderer](#) \*&pRenderer)
- [LvStatus CloseRenderer](#) ([LvRenderer](#) \*&pRenderer)

## Static Public Member Functions

- static [LvStatus Open](#) ([LvDevice](#) \*pDevice, const char \*pStreamId, [LvStream](#) \*&pStream)
- static [LvStatus Close](#) ([LvStream](#) \*&pStream)

## Additional Inherited Members

### 6.10.1 Detailed Description

The [LvStream](#) class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

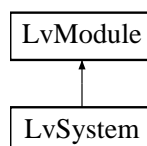
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

## 6.11 LvSystem Class Reference

```
#include <sv.synview.class.h>
```

Inheritance diagram for LvSystem:



## Public Member Functions

- [LvStatus UpdateInterfaceList](#) (uint32\_t Timeout=0xFFFFFFFF)
- [LvStatus GetNumberOfInterfaces](#) (uint32\_t \*pNumberOfInterfaces)
- [LvStatus GetInterfaceId](#) (uint32\_t Index, char \*pInterfaceId, size\_t Size)
- [LvStatus GetInterfaceIdSize](#) (uint32\_t Index, size\_t \*pSize)
- [LvStatus GetInterfaceId](#) (uint32\_t Index, std::string &sInterfaceId)

- [LvStatus FindInterface](#) ([LvFindBy](#) FindBy, const char \*pFindStr, char \*pInterfaceld, size\_t Size)
- [LvStatus FindInterface](#) ([LvFindBy](#) FindBy, const char \*pFindStr, std::string &sInterfaceld)
- [LvHSystem GetHandle](#) ()
- [LvStatus OpenInterface](#) (const char \*pInterfaceld, [LvInterface](#) \*&pInterface)
- [LvStatus CloseInterface](#) ([LvInterface](#) \*&pInterface)
- [LvStatus OpenEvent](#) ([LvEventType](#) EventType, [LvEvent](#) \*&pEvent)
- [LvStatus CloseEvent](#) ([LvEvent](#) \*&pEvent)

### Static Public Member Functions

- static [LvStatus Open](#) (const char \*pSystemId, [LvSystem](#) \*&pSystem)
- static [LvStatus Close](#) ([LvSystem](#) \*&pSystem)

### Additional Inherited Members

#### 6.11.1 Detailed Description

The [LvSystem](#) class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

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

- include/sv.synview.class.h