

Lambda Vue

2.0.10

Generated by Doxygen 1.8.11



# Contents

- 1 Namespace Index** **1**
  - 1.1 Namespace List . . . . . 1
  
- 2 Class Index** **3**
  - 2.1 Class List . . . . . 3
  
- 3 Namespace Documentation** **5**
  - 3.1 `lambdavue` Namespace Reference . . . . . 5
    - 3.1.1 Detailed Description . . . . . 8
    - 3.1.2 Enumeration Type Documentation . . . . . 8
      - 3.1.2.1 `ComposeMode` . . . . . 8
      - 3.1.2.2 `EventType` . . . . . 9
      - 3.1.2.3 `LicenseStatus` . . . . . 9
      - 3.1.2.4 `OutputFileType` . . . . . 10
      - 3.1.2.5 `ParameterType` . . . . . 10
      - 3.1.2.6 `ProcessMethod` . . . . . 10
      - 3.1.2.7 `RecordType` . . . . . 10
      - 3.1.2.8 `SourceAccess` . . . . . 11
      - 3.1.2.9 `SourceOperation` . . . . . 11
      - 3.1.2.10 `ValueType` . . . . . 12
    - 3.1.3 Function Documentation . . . . . 12
      - 3.1.3.1 `ControlSource(enum SourceAccess type, int value=0)` . . . . . 12
      - 3.1.3.2 `GetActivationCode()` . . . . . 12
      - 3.1.3.3 `GetCameraCount()` . . . . . 12

3.1.3.4	GetComposeMode()	13
3.1.3.5	GetExpirationDate()	13
3.1.3.6	GetFilterParameter()	13
3.1.3.7	GetLast5CharOfActivationCode()	13
3.1.3.8	GetProcessMethod()	13
3.1.3.9	GetSourceName()	13
3.1.3.10	GetVerificationErrorString(enum LicenseErrorCode verification_error_code)	13
3.1.3.11	GetWriterState()	14
3.1.3.12	InitMagEngine(CallBackFunction cb, const char *product=""SDK2"")	14
3.1.3.13	LAMBDAVUE_SDK_VERSION()	14
3.1.3.14	OpenSource(enum SourceOperation operation, const char *source, bool idle=true)	14
3.1.3.15	QuerySource(enum SourceAccess type)	15
3.1.3.16	SetBypassMagnify(bool bypass)	15
3.1.3.17	SetComposeMode(enum ComposeMode mode)	15
3.1.3.18	SetFilterParameter(enum ParameterType type, float value)	15
3.1.3.19	SetOutputFilename(enum OutputFileType type, const char *filename)	15
3.1.3.20	SetProcessMethod(enum ProcessMethod method)	16
3.1.3.21	SetRecordType(enum RecordType type)	16
3.1.3.22	SetROIView(uint16_t x, uint16_t y, uint16_t width, uint16_t height)	16
<b>4</b>	<b>Class Documentation</b>	<b>17</b>
4.1	lambdavue::EventValue Struct Reference	17
4.1.1	Detailed Description	18
4.1.2	Constructor & Destructor Documentation	18
4.1.2.1	EventValue(EventType evt=NONE_EVENT)	18
4.1.2.2	EventValue(EventType evt, bool i)	18
4.1.2.3	EventValue(EventType evt, int i)	18
4.1.2.4	EventValue(EventType evt, size_t i)	18
4.1.2.5	EventValue(EventType evt, int64_t i)	18
4.1.2.6	EventValue(EventType evt, float d)	18
4.1.2.7	EventValue(EventType evt, double d)	18

---

4.1.2.8	EventValue(EventType evt, std::string s) . . . . .	18
4.1.2.9	EventValue(EventType evt, char *data, uint32_t len, uint16_t w, uint16_t h) . . . . .	19
4.2	lambdavue::FilterParameter Struct Reference . . . . .	19
4.2.1	Detailed Description . . . . .	19
4.3	lambdavue::Magnifier Class Reference . . . . .	19
4.3.1	Detailed Description . . . . .	20
4.3.2	Member Function Documentation . . . . .	20
4.3.2.1	MagGetActivationCode() . . . . .	20
4.3.2.2	MagGetComposeMode() . . . . .	21
4.3.2.3	MagGetExpiryTimeT() . . . . .	21
4.3.2.4	MagGetFilterParameter() . . . . .	21
4.3.2.5	MagGetImage(BufferStruct &result) . . . . .	21
4.3.2.6	MagGetLast5CharActivationCode() . . . . .	21
4.3.2.7	MagGetLicenseVerificationResult() . . . . .	22
4.3.2.8	MagGetProcessMethod() . . . . .	22
4.3.2.9	MagHaveFilter() . . . . .	22
4.3.2.10	MagPushImage(BufferStruct &source) . . . . .	22
4.3.2.11	MagSetComposeMode(enum ComposeMode mode) . . . . .	22
4.3.2.12	MagSetFilterParameter(enum ParameterType type, float value) . . . . .	22
4.3.2.13	MagSetProcessMethod(enum ProcessMethod method, float framerate=-1) . . . . .	23
4.3.2.14	MagSetRoi(uint16_t x, uint16_t y, uint16_t width, uint16_t height) . . . . .	23
4.4	lambdavue::MediaWriter Class Reference . . . . .	23
4.4.1	Detailed Description . . . . .	24
4.4.2	Member Function Documentation . . . . .	24
4.4.2.1	MwGetChannelNumber() . . . . .	24
4.4.2.2	MwGetDestFilename() . . . . .	24
4.4.2.3	MwGetDestFullFilename() . . . . .	25
4.4.2.4	MwGetDestPath() . . . . .	25
4.4.2.5	MwGetHeight() . . . . .	25
4.4.2.6	MwGetSampleRate() . . . . .	25

4.4.2.7	MwGetWidth()	25
4.4.2.8	MwSetAV(int width, int height, float framerate, int sample_rate, int channel_number)	25
4.4.2.9	MwSetFileName(std::string path, std::string filename)	26
4.4.2.10	MwSetHeight(int height)	26
4.4.2.11	MwSetWidth(int width)	26
4.4.2.12	MwWriteVideo(lambdavue::buffer image, int64_t timestamp)	26
4.5	lambdavue::SourceManager Class Reference	27
4.5.1	Detailed Description	27
4.5.2	Member Function Documentation	28
4.5.2.1	sm_is_file()	28
4.5.2.2	sm_is_valid()	28
4.5.2.3	SmControl(enum lambdavue::SourceAccess type, int value)	28
4.5.2.4	SmGetCameraCount()	28
4.5.2.5	SmGetFrameCount()	28
4.5.2.6	SmGetFramerate()	29
4.5.2.7	SmGetHeight()	29
4.5.2.8	SmGetImage(BufferStruct &output_frame)	29
4.5.2.9	SmGetSourceName()	29
4.5.2.10	SmGetWidth()	29
4.5.2.11	SmQuery(enum lambdavue::SourceAccess type)	29
4.5.2.12	SmSetSource(enum SourceOperation operation, const char *source)	30
4.6	lambdavue::VerificationResult Struct Reference	30
4.6.1	Detailed Description	30
4.7	lambdavue::WriterManager Class Reference	31
4.7.1	Detailed Description	31
4.7.2	Member Function Documentation	31
4.7.2.1	WmCloseWriter(enum OutputFileType output_file_type)	31
4.7.2.2	WmGetWriterState()	32
4.7.2.3	WmInitWriter(int width, int height, float framerate, int sample_rate, int channel_number)	32
4.7.2.4	WmSetOutputDestination(enum OutputFileType output_file_type, std::string input_full_filename=string(), std::string output_file_destination=string())	32
4.7.2.5	WmSetOutputResolution(enum OutputFileType output_file_type, int width, int height)	32
4.7.2.6	WmSetRecordType(enum RecordType type)	33
4.7.2.7	WmWriteImage(BufferStruct &img, enum RecordType)	33
4.8	lambdavue::WriterState Struct Reference	33
4.8.1	Detailed Description	33

# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

- [lambdavue](#)  
This is the header file where API functions of LambdaVue SDK is contained. This file contains API functions decalaration . . . . . 5





# Chapter 2

## Class Index

### 2.1 Class List

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

<a href="#">lambdavue::EventValue</a>	A struct about LambdaVue event This is the structure of callback event value. Callback event have different type. You have to check the value type before any access or you may get wrong value (e.g. the callback value is string but you access integer) . . . . .	17
<a href="#">lambdavue::FilterParameter</a>	A struct about parameters of algorithm, store all the parameter value . . . . .	19
<a href="#">lambdavue::Magnifier</a>	This class provides some API functions to control the LambdaVue magnifier . . . . .	19
<a href="#">lambdavue::MediaWriter</a>	This class provides some API functions control LambdaVue media writer . . . . .	23
<a href="#">lambdavue::SourceManager</a>	This class provides some API functions to management LambdaVue video source . . . . .	27
<a href="#">lambdavue::VerificationResult</a>	A struct to describe license verification result . . . . .	30
<a href="#">lambdavue::WriterManager</a>	This class provides some API functions to management LambdaVue media writer <a href="#">WriterManager</a> contains two <a href="#">MediaWriter</a> , one for original video and the other for processed video . . . . .	31
<a href="#">lambdavue::WriterState</a>	A struct to describe current video writer status . . . . .	33



## Chapter 3

# Namespace Documentation

### 3.1 lambdavue Namespace Reference

This is the header file where API functions of LambdaVue SDK is contained. This file contains API functions decalaration.

#### Classes

- struct [EventValue](#)  
*A struct about LambdaVue event This is the structure of callback event value. Callback event have different type. You have to check the value type before any access or you may get wrong value (e.g. the callback value is string but you access integer).*
- struct [FilterParameter](#)  
*A struct about parameters of algorithm, store all the parameter value.*
- class [Magnifier](#)  
*This class provides some API functions to control the LambdaVue magnifier.*
- class [MediaWriter](#)  
*This class provides some API functions control LambdaVue media writer.*
- class [SourceManager](#)  
*This class provides some API functions to management LambdaVue video source.*
- struct [VerificationResult](#)  
*A struct to describe license verification result.*
- class [WriterManager](#)  
*This class provides some API functions to management LambdaVue media writer [WriterManager](#) contains two [MediaWriter](#), one for original video and the other for processed video.*
- struct [WriterState](#)  
*A struct to describe current video writer status.*

#### Typedefs

- typedef void(\* [CallBackFunction](#)) (struct [EventValue](#))  
*Callback function.*

## Enumerations

- enum `EventType` {
  - `NONE_EVENT` = -1, `ORIGINAL_IMAGE` = 0, `PROCESSED_IMAGE` = 1, `ROI_RESET` = 2,
  - `FILTER_CHANGED` = 3, `COMPOSE_MODE_CHANGED` = 4, `VALUE_ALPHA` = 5, `VALUE_LOWCUT` = 6,
  - `VALUE_HIGHCUT` = 7, `VALUE_LAMBDA` = 8, `VALUE_LEVEL` = 9, `VALUE_SOURCE_FRAME_RATE` = 10,
  - `VALUE_PROCESS_FRAME_RATE` = 11, `VALUE_FILE_DURATION` = 12, `VALUE_FILE_PROGRESS` = 13,
  - `VALUE_CAMERA_COUNT` = 14,
  - `AUTOFOCUS_AVAILABLE` = 15, `AUTOFOCUS_ENABLED` = 16, `OPEN_CAMERA_SUCCESS` = 17, `OPEN_CAMERA_FAIL` = 18,
  - `OPEN_FILE_SUCCESS` = 19, `OPEN_FILE_FAIL` = 20, `CLOSE_SOURCE_SUCCESS` = 21, `FILE_ARCHIVED` = 22,
  - `END_OF_FILE` = 23, `DEBUG_MESSAGE` = 24, `INFO_MESSAGE` = 25, `WARNING_MESSAGE` = 26,
  - `ERROR_MESSAGE` = 27 }

*An enumerations about LambdaVue Callback Events.*
- enum `ValueType` {
  - `NON_VALUE`, `INTEGER_VALUE`, `FLOAT_VALUE`, `STRING_VALUE`,
  - `FRAME_VALUE` }

*The callback event value type.*
- enum `ProcessMethod` { `NO_METHOD` = -1, `LINEAR_MOTION` = 0, `LINEAR_COLOR` = 1, `RIESZ_MOTION` = 2 }
 

*An enumerations about filter algorithm.*
- enum `SourceOperation` { `CLOSE_SOURCE` = 0, `OPEN_CAMERA_SOURCE` = 1, `OPEN_FILE_SOURCE` = 2 }
 

*An enumerations about video source opening and closing.*
- enum `SourceAccess` {
  - `CAMERA_EXPOSURE` = 0, `CAMERA_AUTO_FOCUS` = 1, `CAMERA_FRAME_RATE` = 2, `CAMERA_BRIGHTNESS` = 3,
  - `CAMERA_CONTRAST` = 4, `CAMERA_SATURATION` = 5, `CAMERA_AUTO_WHITEBALANCE` = 6, `CAMERA_BACKLIGHT_COMPENSATION` = 7,
  - `CAMERA_ABSOLUTE_FOCUS` = 8, `CAMERA_ZOOM_IN` = 9, `CAMERA_ZOOM_OUT` = 10, `CAMERA_ZOOM_RESET` = 11,
  - `CAMERA_PAN_LEFT` = 12, `CAMERA_PAN_RIGHT` = 13, `CAMERA_PAN_RESET` = 14, `CAMERA_TILT_UP` = 15,
  - `CAMERA_TILT_DOWN` = 16, `CAMERA_TILT_RESET` = 17, `CAMERA_PTZ_RESET` = 18, `FILE_SEEK` = 100,
  - `FILE_PROGRESS` = 101, `FILE_DURATION` = 102, `FILE_PLAY` = 103, `FILE_PAUSE` = 104,
  - `FILE_FORWARD` = 105, `FILE_BACKWARD` = 106, `FILE_STOP` = 107, `FILE_NORMAL_PLAYBACK` = 108,
  - `FILE_TURBO_PLAYBACK` = 109, `FILE_LOOP_PLAY` = 110, `FILE_SINGLE_PLAY` = 111, `FILE_RECORD_SEEK` = 112 }

*An enumerations about video source control/query type, only some of them are available.*
- enum `ParameterType` {
  - `ALPHA` = 0, `HIGHCUT` = 1, `LOWCUT` = 2, `LEVEL` = 3,
  - `LEVEL_BOUND` = 4, `LAMBDA` = 5, `LAMBDA_BOUND` = 6, `FRAMERATE` = 7,
  - `CUTOFF_BOUND` = 8 }

*An enumerations about parameters type of filter algorithm.*
- enum `RecordType` { `RECORD_NONE` = 0, `RECORD_ORIGINAL_ONLY` = 1, `RECORD_PROCESSED_ORIGINAL` = 2, `RECORD_BOTH` = 3 }
 

*An enumerations about video record types.*
- enum `ComposeMode` { `COMPOSE_NONE` = 0, `COMPOSE_PAP` = 1, `COMPOSE_PIP_MAIN_PROCESS` = 2, `COMPOSE_PIP_MAIN_ORIGINAL` = 3 }
 

*An enumerations about video compose modes.*
- enum `OutputFileType` { `OUTPUT_BOTH` = 0, `OUTPUT_ORIGINAL` = 1, `OUTPUT_PROCESSED` = 2 }
 

*An enumerations about output video file types.*
- enum `LicenseStatus` { `LICENSE_OFFLINE_WARNING` = -2, `LICENSE_OFFLINE_SUCCESS` = -1, `LICENSE_SUCCESS` = 0, `LICENSE_FAILURE` = 1 }

An enumerations about LambdaVue license verification status.

- enum [LicenseErrorCode](#) {  
**LICENSE\_ERROR\_NONE** = 0, **LICENSE\_ERROR\_MISSING\_LICENSE** = 201, **LICENSE\_ERROR\_INCOMPLETE\_LICENSE** = 202, **LICENSE\_ERROR\_INVALID\_SIGNATURE\_LOCAL** = 203,  
**LICENSE\_ERROR\_INVALID\_SIGNATURE\_REMOTE** = 204, **LICENSE\_ERROR\_MISMATCHED\_MAC** = 205, **LICENSE\_ERROR\_EXPIRED** = 206, **LICENSE\_ERROR\_TIMESTAMP\_MODIFIED** = 207,  
**LICENSE\_ERROR\_INVALID\_ACTIVATION\_CODE** = 208, **LICENSE\_ERROR\_OFFLINE\_FORBIDDEN** = 209, **LICENSE\_ERROR\_MISMATCHED\_PRODUCT\_TYPE** = 210 }

An enumerations to describe LambdaVue license verification error.

## Functions

- struct [VerificationResult](#) [InitMagEngine](#) ([CallbackFunction](#) cb, const char \*product="SDK2")  
Initialize LambdaVue magnification engine.
- time\_t [GetExpirationDate](#) ()  
Get the expired date of your LambdaVue license.
- std::string [GetVerificationErrorString](#) (enum [LicenseErrorCode](#) verification\_error\_code)  
Return string containing an verification error string corresponding to the verification error code.
- std::string [GetLast5CharOfActivationCode](#) ()  
Return string containing last 5 character of users activation code.
- std::string [GetActivationCode](#) ()  
Return string containing users activation code.
- void [DestroyEngine](#) ()  
Release LambdaVue magnification engine.
- void [OpenSource](#) (enum [SourceOperation](#) operation, const char \*source, bool idle=true)  
Open video file or a capturing device as source video Set the video source form a file or webcam if the device is a webcam, use videoX, X is the camera index.
- void [CloseSource](#) (bool idle=true)  
Close the video source.
- void [SetBypassMagnify](#) (bool bypass)  
Setting whether to bypass magnify stage.
- void [SetOutputFilename](#) (enum [OutputFileType](#) type, const char \*filename)  
Setting the filename of output videos.
- void [SetRecordType](#) (enum [RecordType](#) type)  
Reccord type setting.
- struct [WriterState](#) [GetWriterState](#) ()  
Get the current state of video writer.
- int [GetCameraCount](#) ()  
Get the number of connected cameras.
- bool [ControlSource](#) (enum [SourceAccess](#) type, int value=0)  
Contorl video source.
- int64\_t [QuerySource](#) (enum [SourceAccess](#) type)  
Query the ability or status of video source.
- std::string [GetSourceName](#) ()  
Get the name of video source.
- void [SetProcessMethod](#) (enum [ProcessMethod](#) method)  
Filter algorithm setting function Set video filtering method to specified method (linear motion, linear color, Riesz motion)
- enum [ProcessMethod](#) [GetProcessMethod](#) ()  
Get Current filter algorithm.
- void [SetComposeMode](#) (enum [ComposeMode](#) mode)

- Compose mode setting function Set video compose mode to specified mode (none, pap, pip)*
- enum [ComposeMode](#) [GetComposeMode](#) ()  
*Get Current compose mode.*
  - void [SetFilterParameter](#) (enum [ParameterType](#) type, float value)  
*Filter parameter setting.*
  - struct [FilterParameter](#) [GetFilterParameter](#) ()  
*Get the FilterParameter structure The [FilterParameter](#) structure contains the current value of parameters.*
  - void [SetROIView](#) (uint16\_t x, uint16\_t y, uint16\_t width, uint16\_t height)  
*Set the region of interest.*
  - std::string [LAMBDAVUE\\_SDK\\_VERSION](#) ()  
*Get the version of LambdaVue SDK.*

### 3.1.1 Detailed Description

This is the header file where API functions of LambdaVue SDK is contained. This file contains API functions decalaration.

LmabdaVue.h

Author

Nathan Chen

Copyright

2016 QRI. All rights reserved.

### 3.1.2 Enumeration Type Documentation

#### 3.1.2.1 enum `lambdavue::ComposeMode`

An enumerations about video compose modes.

Enumerator

**COMPOSE\_PAP** Do not compose video

**COMPOSE\_PIP\_MAIN\_PROCESSED** Picture-and-Picture, original video at left side

**COMPOSE\_PIP\_MAIN\_ORIGINAL** Picture-in-Picture, use processed video as main picutre

### 3.1.2.2 enum lambdavue::EventType

An enumerations about LambdaVue Callback Events.

@/definedblock

Enumerator

- ORIGINAL\_IMAGE** None event
- PROCESSED\_IMAGE** Evoked when source image is captured
- ROI\_RESET** Evoked when processed image is done
- FILTER\_CHANGED** Evoked when region of interest had been changed
- COMPOSE\_MODE\_CHANGED** Evoked when filter method had been changed
- VALUE\_ALPHA** Evoked when compose mode had been changed
- VALUE\_LOWCUT** Evoked when magnification value had been changed
- VALUE\_HIGHCUT** Evoked when filter low cutoff had been changed
- VALUE\_LAMBDA** Evoked when filter high cutoff had been changed
- VALUE\_LEVEL** Evoked when spatial frequency cutoff had been changed (used in linear motion only)
- VALUE\_SOURCE\_FRAME\_RATE** Evoked when Gaussian pyramid level had been changed (used in linear color only)
- VALUE\_PROCESS\_FRAME\_RATE** Evoked when source frame rate had been changed
- VALUE\_FILE\_DURATION** Evoked when processing frame rate had been changed
- VALUE\_FILE\_PROGRESS** Evoked when get video file length
- VALUE\_CAMERA\_COUNT** Evoked during file processing, return the timestamp in video file
- AUTOFOCUS\_AVAILABLE** Evoked when camera count had been changed
- AUTOFOCUS\_ENABLED** Evoked after source opened, indicate whether the source can do auto focus
- OPEN\_CAMERA\_SUCCESS** Evoked after source opened, indicate whether auto focus is enabled
- OPEN\_CAMERA\_FAIL** Evoked when webcam opened successful
- OPEN\_FILE\_SUCCESS** Evoked when webcam opened fail
- OPEN\_FILE\_FAIL** Evoked when file opened successful
- CLOSE\_SOURCE\_SUCCESS** Evoked when file opened fail
- FILE\_ARCHIVED** Evoked when source closed successful
- END\_OF\_FILE** Evoked when file archived
- DEBUG\_MESSAGE** Evoked when reach to end of file
- INFO\_MESSAGE** Evoked when debug event occurred
- WARNING\_MESSAGE** Evoked when info event occurred
- ERROR\_MESSAGE** Evoked when warning event occurred

### 3.1.2.3 enum lambdavue::LicenseStatus

An enumerations about LambdaVue license verification status.

Enumerator

- LICENSE\_OFFLINE\_SUCCESS** The LambdaVue license has passed offline verification, but must take an online verification in 7 days
- LICENSE\_SUCCESS** The LambdaVue license has passed offline verification
- LICENSE\_FAILURE** The LambdaVue license has passed online verification

### 3.1.2.4 enum `lambdavue::OutputFileType`

An enumerations about output video file types.

Enumerator

- OUTPUT\_ORIGINAL*** Both original and processed video
- OUTPUT\_PROCESSED*** The original video

### 3.1.2.5 enum `lambdavue::ParameterType`

An enumerations about parameters type of filter algorithm.

Enumerator

- HIGHCUT*** The amplification factor
- LOWCUT*** High cutoff of temporal filter
- LEVEL*** Low cutoff of temporal filter
- LEVEL\_BOUND*** Gaussian pyramid level (used in linear color only)
- LAMBDA*** Maximum value for level
- LAMBDA\_BOUND*** Spatial frequency cutoff (used in linear motion only)
- FRAMERATE*** Maximum value for lambda
- CUTOFF\_BOUND*** Framerate of video source

### 3.1.2.6 enum `lambdavue::ProcessMethod`

An enumerations about filter algorithm.

Enumerator

- LINEAR\_MOTION*** No method selected
- LINEAR\_COLOR*** Eulerian linear motion amplification method
- RIESZ\_MOTION*** Eulerian linear color amplification method

### 3.1.2.7 enum `lambdavue::RecordType`

An enumerations about video record types.

Enumerator

- RECORD\_ORIGINAL\_ONLY*** Do not record any video
- RECORD\_PROCESSED\_ONLY*** Only record original video
- RECORD\_BOTH*** Only record processed video



### 3.1.2.8 enum lambdavue::SourceAccess

An enumerations about video source control/query type, only some of them are available.

Enumerator

**CAMERA\_AUTO\_FOCUS** Camera exposure access  
**CAMERA\_FRAME\_RATE** Camera auto focus access  
**CAMERA\_BRIGHTNESS** Camera frame rate access  
**CAMERA\_CONTRAST** Camera brightness access  
**CAMERA\_SATURATION** Camera contrast access  
**CAMERA\_AUTO\_WHITEBALANCE** Camera saturation access  
**CAMERA\_BACKLIGHT\_COMPENSATION** Camera whitebalance access  
**CAMERA\_ABSOLUTE\_FOCUS** Camera backlight compensation access  
**CAMERA\_ZOOM\_IN** Camera absolute focus access  
**CAMERA\_ZOOM\_OUT** Camera zoom in  
**CAMERA\_ZOOM\_RESET** Camera zoom out  
**CAMERA\_PAN\_LEFT** Camera zoom reset  
**CAMERA\_PAN\_RIGHT** Camera pan left  
**CAMERA\_PAN\_RESET** Camera pan right  
**CAMERA\_TILT\_UP** Camera pan reset  
**CAMERA\_TILT\_DOWN** Camera tilt up  
**CAMERA\_TILT\_RESET** Camera tilt down  
**CAMERA\_PTZ\_RESET** Camera tilt reset  
**FILE\_SEEK** Reset pan, tilt and zoom  
**FILE\_PROGRESS** Video file seek to some timestamp  
**FILE\_DURATION** Video file current progress  
**FILE\_PLAY** Video file length  
**FILE\_PAUSE** Play video file  
**FILE\_FORWARD** Pause video file  
**FILE\_BACKWARD** Forward video file  
**FILE\_STOP** Backward video file  
**FILE\_NORMAL\_PLAYBACK** Stop video file  
**FILE\_TURBO\_PLAYBACK** Video file normal playback  
**FILE\_LOOP\_PLAY** Video file turbo playback  
**FILE\_SINGLE\_PLAY** Loop play video file  
**FILE\_RECORD\_SEEK** Play video file once

### 3.1.2.9 enum lambdavue::SourceOperation

An enumerations about video source opening and closing.

Enumerator

**OPEN\_CAMERA\_SOURCE** Close video source  
**OPEN\_FILE\_SOURCE** Open webcam source

### 3.1.2.10 enum `lambdavue::ValueType`

The callback event value type.

Enumerator

```

INTEGER_VALUE NON_VALUE
FLOAT_VALUE INTEGER_VALUE
STRING_VALUE FLOAT_VALUE
FRAME_VALUE STRING_VALUE
FRAME_VALUE

```

## 3.1.3 Function Documentation

### 3.1.3.1 `bool lambdavue::ControlSource ( enum SourceAccess type, int value = 0 )`

Control video source.

Parameters

<i>type</i>	Specify control type
<i>value</i>	Specify the value

Returns

`bool` Whether the control is successful

### 3.1.3.2 `std::string lambdavue::GetActivationCode ( )`

Return string containing users activation code.

Returns

`std::string` The users activation code

### 3.1.3.3 `int lambdavue::GetCameraCount ( )`

Get the number of connected cameras.

Returns

`int` The number of connected cameras

#### 3.1.3.4 enum `ComposeMode` `lambdavue::GetComposeMode ( )`

Get Current compose mode.

##### Returns

enum `ComposeType` The current chosen mode

#### 3.1.3.5 `time_t` `lambdavue::GetExpirationDate ( )`

Get the expired date of your LambdaVue license.

##### Returns

`time_t` The Expired date

#### 3.1.3.6 struct `FilterParameter` `lambdavue::GetFilterParameter ( )`

Get the FilterParameter structure The [FilterParameter](#) structure contains the current value of parameters.

##### Returns

struct [FilterParameter](#) The current filter parameter struct

#### 3.1.3.7 `std::string` `lambdavue::GetLast5CharOfActivationCode ( )`

Return string containing last 5 character of users activation code.

##### Returns

`std::string` The last 5 character of users activation code

#### 3.1.3.8 enum `ProcessMethod` `lambdavue::GetProcessMethod ( )`

Get Current filter algorithm.

##### Returns

enum `ProcessMethod` The current chosen algorithm

#### 3.1.3.9 `std::string` `lambdavue::GetSourceName ( )`

Get the name of video source.

##### Returns

`std::string` Current source name

#### 3.1.3.10 `std::string` `lambdavue::GetVerificationErrorString ( enum LicenseErrorCode verification_error_code )`

Return string containing an verification error string corresponding to the verification error code.

## Parameters

<i>verification_error_code</i>	Error code to describe
--------------------------------	------------------------

## Returns

std::string The verification error string

3.1.3.11 struct `WriterState` `lambdavue::GetWriterState ( )`

Get the current state of video writer.

## Returns

struct [WriterState](#) Current video writer status

3.1.3.12 struct `VerificationResult` `lambdavue::InitMagEngine ( CallbackFunction cb, const char * product = "SDK2" )`

Initialize LambdaVue magnification engine.

## Parameters

<i>cb</i>	Pass your CallBackFuntion as a parameter
-----------	--

## Returns

enum `LicenseStatus` The status of your LambdaVue license

3.1.3.13 std::string `lambdavue::LAMBDAVUE_SDK_VERSION ( )`

Get the version of LambdaVue SDK.

## Returns

std::string The version of LambdaVue SDK

3.1.3.14 void `lambdavue::OpenSource ( enum SourceOperation operation, const char * source, bool idle = true )`

Open video file or a capturing device as source video Set the video source form a file or webcam if the device is a webcam, use videoX, X is the camera index.

## Parameters

<i>operation</i>	Specify the SourceOperation
<i>source</i>	The name of the video source (filename or device name)
<i>idle</i>	Default value is true, set this parameter to false only if you are developing an OSX console program

3.1.3.15 `int64_t` `lambdavue::QuerySource ( enum SourceAccess type )`

Query the ability or status of video source.

## Parameters

<i>type</i>	Specify control type
-------------	----------------------

## Returns

`int64_t` The current value of the query type, if the type is unaccessible then UNAVAILABLE is returned

3.1.3.16 `void` `lambdavue::SetBypassMagnify ( bool bypass )`

Setting whether to bypass magnify stage.

## Parameters

<i>bypass</i>	Specify whether to bypass magnify stage
---------------	---

3.1.3.17 `void` `lambdavue::SetComposeMode ( enum ComposeMode mode )`

Compose mode setting function Set video compose mode to specified mode (none, pap, pip)

## Parameters

<i>mode</i>	The compose mode
-------------	------------------

3.1.3.18 `void` `lambdavue::SetFilterParameter ( enum ParameterType type, float value )`

Filter parameter setting.

## Parameters

<i>type</i>	Specify parameter type
<i>value</i>	Parameter value

3.1.3.19 `void` `lambdavue::SetOutputFilename ( enum OutputFileType type, const char * filename )`

Setting the filename of output videos.

## Parameters

<i>type</i>	Specify the output file (original or processed) type you'd like to named it
<i>filename</i>	Specify the file name

### 3.1.3.20 void lambdavue::SetProcessMethod ( enum ProcessMethod *method* )

Filter algorithm setting function Set video filtering method to specified method (linear motion, linear color, Riesz motion)

#### Parameters

<i>method</i>	The filter algorithm
---------------	----------------------

### 3.1.3.21 void lambdavue::SetRecordType ( enum RecordType *type* )

Reccord type setting.

#### Parameters

<i>type</i>	Specify the record type
-------------	-------------------------

### 3.1.3.22 void lambdavue::SetROIView ( uint16\_t *x*, uint16\_t *y*, uint16\_t *width*, uint16\_t *height* )

Set the region of interest.

#### Parameters

<i>x</i>	The x coordinate of the top-left corner of ROI
<i>y</i>	The y coordinate of the top-left corner of ROI
<i>width</i>	The width of ROI
<i>height</i>	The height of ROI

## Chapter 4

# Class Documentation

### 4.1 `lambdavue::EventValue` Struct Reference

A struct about LambdaVue event This is the structure of callback event value. Callback event have different type. You have to check the value type before any access or you may get wrong value (e.g. the callback value is string but you access integer).

```
#include <core.h>
```

#### Public Member Functions

- [EventValue](#) ([EventType](#) evt=NONE\_EVENT)
- [EventValue](#) ([EventType](#) evt, bool i)
- [EventValue](#) ([EventType](#) evt, int i)
- [EventValue](#) ([EventType](#) evt, size\_t i)
- [EventValue](#) ([EventType](#) evt, int64\_t i)
- [EventValue](#) ([EventType](#) evt, float d)
- [EventValue](#) ([EventType](#) evt, double d)
- [EventValue](#) ([EventType](#) evt, std::string s)
- [EventValue](#) ([EventType](#) evt, char \*data, uint32\_t len, uint16\_t w, uint16\_t h)

#### Public Attributes

- [EventType](#) **event**
- [ValueType](#) **value\_type**
- std::string **s\_value**
- union {
  - double **d\_value**
  - int64\_t **i\_value**
- };
- void \* **data\_buffer**
- uint16\_t **image\_width**
- uint16\_t **image\_height**

### 4.1.1 Detailed Description

A struct about LambdaVue event This is the structure of callback event value. Callback event have different type. You have to check the value type before any access or you may get wrong value (e.g. the callback value is string but you access integer).

event A enum EventType, indicate the type of event value\_type The callback event value type s\_value String value type d\_value Double value type i\_value Integer int64\_t value type data\_buffer Pointer to data buffer which contains image data image\_width Image width image\_height Image height

### 4.1.2 Constructor & Destructor Documentation

4.1.2.1 `lambdavue::EventValue::EventValue ( EventType evt = NONE_EVENT ) [inline]`

Constructor for no value event type

4.1.2.2 `lambdavue::EventValue::EventValue ( EventType evt, bool i ) [inline]`

Constructor for boolean type

4.1.2.3 `lambdavue::EventValue::EventValue ( EventType evt, int i ) [inline]`

Constructor for integer type

4.1.2.4 `lambdavue::EventValue::EventValue ( EventType evt, size_t i ) [inline]`

Constructor for unsigned integer type

4.1.2.5 `lambdavue::EventValue::EventValue ( EventType evt, int64_t i ) [inline]`

Constructor for 64bits integer type

4.1.2.6 `lambdavue::EventValue::EventValue ( EventType evt, float d ) [inline]`

Constructor for floating point type

4.1.2.7 `lambdavue::EventValue::EventValue ( EventType evt, double d ) [inline]`

Constructor for double floating point type

4.1.2.8 `lambdavue::EventValue::EventValue ( EventType evt, std::string s ) [inline]`

Constructor for string type



4.1.2.9 `lambdavue::EventValue::EventValue ( EventType evt, char * data, uint32_t len, uint16_t w, uint16_t h )`  
[inline]

Constructor for image buffer type

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

- `LambdaVue/core/include/core.h`

## 4.2 `lambdavue::FilterParameter` Struct Reference

A struct about parameters of algorithm, store all the parameter value.

```
#include <core.h>
```

### Public Attributes

- `size_t alpha`
- `float high_cutoff`
- `float low_cutoff`
- `float lambda`
- `size_t level`
- `float framerate`
- `float lambda_bound`
- `float cutoff_bound`

### 4.2.1 Detailed Description

A struct about parameters of algorithm, store all the parameter value.

`alpha` The amplification factor `high_cutoff` High cutoff of temporal filter `low_cutoff` Low cutoff of temporal filter `lambda` Spatial frequency cutoff `level` Gaussian pyramid level `framerate` Framerate of video source `lambda_bound` Maximum value for `lambda` `cutoff_bound` Maximum value for temporal filter cutoff

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

- `LambdaVue/core/include/core.h`

## 4.3 `lambdavue::Magnifier` Class Reference

This class provides some API functions to control the `LambdaVue` magnifier.

```
#include <magnifier.h>
```

## Public Member Functions

- [Magnifier](#) (const char \*product)  
*Constructor of magnifier.*
- [~Magnifier](#) ()  
*Destructor of magnifier.*
- void [MagDeleteFilter](#) ()  
*Delete the filter of LambdaVue magnifier.*
- bool [MagHaveFilter](#) ()  
*Return whether the LambdaVue magnifier has any filter.*
- void [MagSetProcessMethod](#) (enum [ProcessMethod](#) method, float framerate=-1)  
*LambdaVue magnifier filter algorithm setting function.*
- enum [ProcessMethod](#) [MagGetProcessMethod](#) ()  
*Get Current filter algorithm of LambdaVue magnifier.*
- void [MagSetComposeMode](#) (enum [ComposeMode](#) mode)  
*LambdaVue magnifier compose mode setting function.*
- enum [ComposeMode](#) [MagGetComposeMode](#) ()  
*Get Current compose mode of LambdaVue magnifier.*
- void [MagSetFilterParameter](#) (enum [ParameterType](#) type, float value)  
*LambdaVue magnifier filter parameter setting.*
- struct [FilterParameter](#) [MagGetFilterParameter](#) ()  
*Get the FilterParameter structure of LambdaVue magnifier The [FilterParameter](#) structure contains the current value of parameters.*
- void [MagPushImage](#) (BufferStruct &source)  
*Push an image into LambdaVue magnifier.*
- void [MagGetImage](#) (BufferStruct &result)  
*Return the processed image from LambdaVue magnifier.*
- void [MagSetRoi](#) (uint16\_t x, uint16\_t y, uint16\_t width, uint16\_t height)  
*Set the region of interest of LambdaVue magnifier.*
- struct [VerificationResult](#) [MagGetLicenseVerificationResult](#) ()  
*Return license verification result.*
- time\_t [MagGetExpiryTimeT](#) ()  
*Get the expired date of your LambdaVue license.*
- std::string [MagGetLast5CharActivationCode](#) ()  
*Get last 5 character of your LambdaVue activation code.*
- std::string [MagGetActivationCode](#) ()  
*Get your LambdaVue activation code.*

### 4.3.1 Detailed Description

This class provides some API functions to control the LambdaVue magnifier.

### 4.3.2 Member Function Documentation

#### 4.3.2.1 std::string lambdavue::Magnifier::MagGetActivationCode ( )

Get your LambdaVue activation code.

#### Returns

std::string The activation code

#### 4.3.2.2 `enum ComposeMode` `lambdavue::Magnifier::MagGetComposeMode ( )`

Get Current compose mode of LambdaVue magnifier.

##### Returns

`enum ComposeMode` The current compose mode of LambdaVue magnifier

#### 4.3.2.3 `time_t` `lambdavue::Magnifier::MagGetExpiryTimeT ( )`

Get the expired date of your LambdaVue license.

##### Returns

`time_t` The Expired date

#### 4.3.2.4 `struct FilterParameter` `lambdavue::Magnifier::MagGetFilterParameter ( )`

Get the FilterParameter structure of LambdaVue magnifier The [FilterParameter](#) structure contains the current value of parameters.

##### Returns

`struct FilterParameter` The current filter parameter struct

#### 4.3.2.5 `void` `lambdavue::Magnifier::MagGetImage ( BufferStruct & result )`

Return the processed image from LambdaVue magnifier.

##### Returns

`BufferStruct` The BufferStruct of processed image

#### 4.3.2.6 `std::string` `lambdavue::Magnifier::MagGetLast5CharActivationCode ( )`

Get last 5 character of your LambdaVue activation code.

##### Returns

`std::string` The last 5 character of activation code

#### 4.3.2.7 struct `VerificationResult` `lambdavue::Magnifier::MagGetLicenseVerificationResult ( )`

Return license verification result.

##### Returns

struct [VerificationResult](#) The verification result struct of your LambdaVue license

#### 4.3.2.8 enum `ProcessMethod` `lambdavue::Magnifier::MagGetProcessMethod ( )`

Get Current filter algorithm of LambdaVue magnifier.

##### Returns

enum `ProcessMethod` The current chosen algorithm of LambdaVue magnifier

#### 4.3.2.9 bool `lambdavue::Magnifier::MagHaveFilter ( )`

Return whether the LambdaVue magnifier has any filter.

##### Returns

bool Whether the magnifier has any filter

#### 4.3.2.10 void `lambdavue::Magnifier::MagPushImage ( BufferStruct & source )`

Push an image into LambdaVue magnifier.

##### Parameters

<i>source</i>	The image BufferStruct
---------------	------------------------

#### 4.3.2.11 void `lambdavue::Magnifier::MagSetComposeMode ( enum ComposeMode mode )`

LambdaVue magnifier compose mode setting function.

##### Parameters

<i>mode</i>	The compose mode
-------------	------------------

#### 4.3.2.12 void `lambdavue::Magnifier::MagSetFilterParameter ( enum ParameterType type, float value )`

LambdaVue magnifier filter parameter setting.

## Parameters

<i>type</i>	Specify parameter type
<i>value</i>	Parameter value

4.3.2.13 `void lambdavue::Magnifier::MagSetProcessMethod ( enum ProcessMethod method, float framerate = -1 )`

LambdaVue magnifier filter algorithm setting function.

## Parameters

<i>method</i>	The filter algorithm
<i>framerate</i>	The framerate of video source

4.3.2.14 `void lambdavue::Magnifier::MagSetRoi ( uint16_t x, uint16_t y, uint16_t width, uint16_t height )`

Set the region of interest of LambdaVue magnifier.

## Parameters

<i>x</i>	The x coordinate of the top-left corner of ROI
<i>y</i>	The y coordinate of the top-left corner of ROI
<i>width</i>	The width of ROI
<i>height</i>	The height of ROI

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

- `/Users/nathanchen/Desktop/LambdaVue/LambdaVueSDK/LambdaVue/magnifier/include/magnifier.h`

## 4.4 `lambdavue::MediaWriter` Class Reference

This class provides some API functions control LambdaVue media writer.

```
#include <writer.hpp>
```

### Public Member Functions

- [MediaWriter](#) ()  
*Constructor of media writer.*
- [~MediaWriter](#) ()  
*Destructor of media writer.*
- void [MwOpenNewFile](#) ()  
*Prepare a new output video file.*
- void [MwClose](#) ()

- Close the output video and delete the encoder.*

  - void [MwSetAV](#) (int width, int height, float framerate, int sample\_rate, int channel\_number)
    - Set the basic information of audio (not yet supported) and video for media writer.*
  - void [MwSetFileName](#) (std::string path, std::string filename)
    - Set the filename of output video.*
  - void [MwWriteVideo](#) (lambdavue::buffer image, int64\_t timestamp)
    - Encode an image buffer and write to video file.*
  - void [MwSetWidth](#) (int width)
    - Set the width of the output video.*
  - int [MwGetWidth](#) ()
    - Return the width of the output video.*
  - void [MwSetHeight](#) (int height)
    - Set the height of the output video.*
  - int [MwGetHeight](#) ()
    - Return the height of the output video.*
  - int [MwGetSampleRate](#) ()
    - Return the audio sample rate of the output video.*
  - int [MwGetChannelNumber](#) ()
    - Return the number of channels of the output video.*
  - std::string [MwGetDestPath](#) ()
    - Return the destination path of the output video.*
  - std::string [MwGetDestFilename](#) ()
    - Return the filename of the output video.*
  - std::string [MwGetDestFullFilename](#) ()
    - Return the full filename (path and filename) of the output video.*

#### 4.4.1 Detailed Description

This class provides some API functions control LambdaVue media writer.

#### 4.4.2 Member Function Documentation

##### 4.4.2.1 int lambdavue::MediaWriter::MwGetChannelNumber ( ) [inline]

Return the number of channels of the output video.

##### Returns

int The number of channels of the output video

##### 4.4.2.2 std::string lambdavue::MediaWriter::MwGetDestFilename ( ) [inline]

Return the filename of the output video.

##### Returns

std::string The filename of output video

4.4.2.3 `std::string lambdavue::MediaWriter::MwGetDestFullFilename ( ) [inline]`

Return the full filename (path and filename) of the output video.

**Returns**

`std::string` The full filename (path and filename) of output video

4.4.2.4 `std::string lambdavue::MediaWriter::MwGetDestPath ( ) [inline]`

Return the destination path of the output video.

**Returns**

`std::string` The destination path of output video

4.4.2.5 `int lambdavue::MediaWriter::MwGetHeight ( ) [inline]`

Return the height of the output video.

**Returns**

`int` The height of the output video

4.4.2.6 `int lambdavue::MediaWriter::MwGetSampleRate ( ) [inline]`

Return the audio sample rate of the output video.

**Returns**

`int` The audio sample rate of the output video

4.4.2.7 `int lambdavue::MediaWriter::MwGetWidth ( ) [inline]`

Return the width of the output video.

**Returns**

`int` The width of the output video

4.4.2.8 `void lambdavue::MediaWriter::MwSetAV ( int width, int height, float framerate, int sample_rate, int channel_number )`

Set the basic information of audio (not yet supported) and video for media writer.

## Parameters

<i>width</i>	Width of output video
<i>height</i>	Height of output video
<i>framerate</i>	Framerate of output video
<i>sample_rate</i>	Audio sample rate of output video (not yet supported)
<i>channel_number</i>	Channel number of output video (not yet supported)

4.4.2.9 void `lambdavue::MediaWriter::MwSetFileName ( std::string path, std::string filename )`

Set the filename of output video.

## Parameters

<i>path</i>	Destination path of output video
<i>filename</i>	Filename of output video

4.4.2.10 void `lambdavue::MediaWriter::MwSetHeight ( int height )`

Set the height of the output video.

## Returns

int The height of the output video

4.4.2.11 void `lambdavue::MediaWriter::MwSetWidth ( int width )`

Set the width of the output video.

## Returns

int The width of the output video

4.4.2.12 void `lambdavue::MediaWriter::MwWriteVideo ( lambdavue::buffer image, int64_t timestamp )`

Encode an image buffer and write to video file.

## Parameters

<i>img</i>	Image buffer
<i>timestamp</i>	The timestamp of the image buffer

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



- `/Users/nathanchen/Desktop/LambdaVue/LambdaVueSDK/LambdaVue/writer/include/writer.hpp`

## 4.5 `lambdavue::SourceManager` Class Reference

This class provides some API functions to management LambdaVue video source.

```
#include <source.h>
```

### Public Member Functions

- [SourceManager](#) ()  
*Constructor of source manager.*
- [~SourceManager](#) ()  
*Destructor of source manager.*
- `bool` [SmSetSource](#) (enum [SourceOperation](#) operation, `const char *source`)  
*Video source setting or closing.*
- `void` [SmGetImage](#) (`BufferStruct &output_frame`)  
*Get the decoded frame image from LambdaVue source manager.*
- `std::string` [SmGetSourceName](#) ()  
*Get the name of video source.*
- `size_t` [SmGetFrameCount](#) ()  
*Return the amount of frame in video source.*
- `size_t` [SmGetWidth](#) ()  
*Return the image width of video source.*
- `size_t` [SmGetHeight](#) ()  
*Return the image height of video source.*
- `float` [SmGetFramerate](#) ()  
*Return the framerate of video source.*
- `int` [SmGetCameraCount](#) ()  
*Return the amount of connected camera.*
- `bool` [sm\\_is\\_valid](#) ()  
*Return whether the source manager is valid.*
- `bool` [sm\\_is\\_file](#) ()  
*Return whether the current video source is file.*
- `int64_t` [SmQuery](#) (enum [lambdavue::SourceAccess](#) type)  
*Query the ability or status of video source.*
- `bool` [SmControl](#) (enum [lambdavue::SourceAccess](#) type, `int` value)  
*Contorl video source.*

### 4.5.1 Detailed Description

This class provides some API functions to management LambdaVue video source.

## 4.5.2 Member Function Documentation

### 4.5.2.1 `bool lambdavue::SourceManager::sm_is_file ( )`

Return whether the current video source is file.

#### Returns

`bool` True if the source is a file, false otherwise

### 4.5.2.2 `bool lambdavue::SourceManager::sm_is_valid ( )`

Return whether the source manager is valid.

#### Returns

`bool` Whether the source manager is valid

### 4.5.2.3 `bool lambdavue::SourceManager::SmControl ( enum lambdavue::SourceAccess type, int value )`

Control video source.

#### Parameters

<i>type</i>	Specify control type
<i>value</i>	Specify the value

#### Returns

`bool` Whether the control is successful

### 4.5.2.4 `int lambdavue::SourceManager::SmGetCameraCount ( )`

Return the amount of connected camera.

#### Returns

`int` The total number of connected camera

### 4.5.2.5 `size_t lambdavue::SourceManager::SmGetFrameCount ( )`

Return the amount of frame in video source.

#### Returns

`size_t` The total number of frame in video source

#### 4.5.2.6 `float lambdavue::SourceManager::SmGetFramerate ( )`

Return the framerate of video source.

##### Returns

`float` The framerate of video source

#### 4.5.2.7 `size_t lambdavue::SourceManager::SmGetHeight ( )`

Return the image height of video source.

##### Returns

`size_t` The image height of video source

#### 4.5.2.8 `void lambdavue::SourceManager::SmGetImage ( BufferStruct & output_frame )`

Get the decoded frame image from LambdaVue source manager.

##### Returns

`output_frame` The BufferStruct to store the frame image

#### 4.5.2.9 `std::string lambdavue::SourceManager::SmGetSourceName ( )`

Get the name of video source.

##### Returns

`std::string` Current source name

#### 4.5.2.10 `size_t lambdavue::SourceManager::SmGetWidth ( )`

Return the image width of video source.

##### Returns

`size_t` The image width of video source

#### 4.5.2.11 `int64_t lambdavue::SourceManager::SmQuery ( enum lambdavue::SourceAccess type )`

Query the ability or status of video source.

## Parameters

<i>type</i>	Specify control type
-------------	----------------------

## Returns

int64\_t The current value of the query type, if the type is inaccessible then UNAVAILABLE is returned

4.5.2.12 `bool lambdavue::SourceManager::SmSetSource ( enum SourceOperation operation, const char * source )`

Video source setting or closing.

## Parameters

<i>operation</i>	Specify the SourceOperation
<i>source</i>	The name of the video source (filename or device name)

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

- /Users/nathanchen/Desktop/LambdaVue/LambdaVueSDK/LambdaVue/source/include/source.h

## 4.6 lambdavue::VerificationResult Struct Reference

A struct to describe license verification result.

```
#include <core.h>
```

### Public Attributes

- enum [LicenseStatus](#) **status**
- enum [LicenseErrorCode](#) **error\_code**

#### 4.6.1 Detailed Description

A struct to describe license verification result.

status License verification status error\_code License verification error code

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

- LambdaVue/core/include/core.h

## 4.7 lambdavue::WriterManager Class Reference

This class provides some API functions to management LambdaVue media writer [WriterManager](#) contains two [MediaWriter](#), one for original video and the other for processed video.

```
#include <writer.hpp>
```

### Public Member Functions

- [WriterManager](#) ()  
*Constructor of writer manager.*
- [~WriterManager](#) ()  
*Destructor of writer manager.*
- void [WmSetRecordType](#) (enum [RecordType](#) type)  
*Reccord type setting.*
- void [WmInitWriter](#) (int width, int height, float framerate, int sample\_rate, int channel\_number)  
*Initial MediaWriter.*
- void [WmSetOutputResolution](#) (enum [OutputFileType](#) output\_file\_type, int width, int height)  
*Set resolution of output video.*
- void [WmWriteImage](#) (BufferStruct &img, enum [RecordType](#))  
*Encode an image buffer and write to video file.*
- void [WmCloseWriter](#) (enum [OutputFileType](#) output\_file\_type)  
*Close original and processed MediaWriter.*
- void [WmSetOutputDestination](#) (enum [OutputFileType](#) output\_file\_type, std::string input\_full\_filename=string(), std::string output\_file\_destination=string())  
*Set the destination of output videos.*
- struct [WriterState](#) [WmGetWriterState](#) ()  
*Return the current WriterManager status.*

#### 4.7.1 Detailed Description

This class provides some API functions to management LambdaVue media writer [WriterManager](#) contains two [MediaWriter](#), one for original video and the other for processed video.

#### 4.7.2 Member Function Documentation

##### 4.7.2.1 void lambdavue::WriterManager::WmCloseWriter ( enum OutputFileType output\_file\_type )

Close original and processed [MediaWriter](#).

##### Parameters

<i>output_file_type</i>	Indicate which type of output video
-------------------------	-------------------------------------

#### 4.7.2.2 struct `WriterState` `lambdavue::WriterManager::WmGetWriterState ( )`

Return the current `WriterManager` status.

##### Returns

struct `WriterState` The current `WriterManager` status

#### 4.7.2.3 void `lambdavue::WriterManager::WmInitWriter ( int width, int height, float framerate, int sample_rate, int channel_number )`

Initial `MediaWriter`.

##### Parameters

<i>width</i>	Width of output video
<i>height</i>	Height of output video
<i>framerate</i>	Framerate of output video
<i>sample_rate</i>	Audio sample rate of output video (not yet supported)
<i>channel_number</i>	Channel number of output video (not yet supported)

#### 4.7.2.4 void `lambdavue::WriterManager::WmSetOutputDestination ( enum OutputFileType output_file_type, std::string input_full_filename = string(), std::string output_file_destination = string() )`

Set the destination of output videos.

##### Parameters

<i>output_file_type</i>	Indicate which type of output video
<i>input_full_filename</i>	The filename of input video source
<i>output_file_destination</i>	The destination of output video

#### 4.7.2.5 void `lambdavue::WriterManager::WmSetOutputResolution ( enum OutputFileType output_file_type, int width, int height )`

Set resolution of output video.

##### Parameters

<i>output_file_type</i>	Indicate which type of output video
<i>width</i>	Width of output video
<i>height</i>	Height of output video

4.7.2.6 void lambdavue::WriterManager::WmSetRecordType ( enum RecordType *type* )

Record type setting.

Parameters

<i>type</i>	Specify the record type
-------------	-------------------------

4.7.2.7 void lambdavue::WriterManager::WmWriteImage ( BufferStruct & *img*, enum RecordType )

Encode an image buffer and write to video file.

Parameters

<i>img</i>	Image BufferStruct
<i>RecordType</i>	Record type

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

- /Users/nathanchen/Desktop/LambdaVue/LambdaVueSDK/LambdaVue/writer/include/writer.hpp

## 4.8 lambdavue::WriterState Struct Reference

A struct to describe current video writer status.

```
#include <core.h>
```

### Public Attributes

- [RecordType](#) **record\_type**
- int **processed\_width**
- int **processed\_height**
- std::string **processed\_dest\_filename**
- int **original\_width**
- int **original\_height**
- std::string **origianl\_dest\_filename**

### 4.8.1 Detailed Description

A struct to describe current video writer status.

**record\_type** Current output video record type **processed\_width** Width of processed video **processed\_height** Height of processed video **processed\_dest\_filename** File name of the processed video **original\_width** Width of original video **original\_height** Height of original video **origianl\_dest\_filename** File name of the origianl video

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

- LambdaVue/core/include/core.h





# Index

- AUTOFOCUS\_AVAILABLE
  - [lambda](#), 9
- AUTOFOCUS\_ENABLED
  - [lambda](#), 9
- CAMERA\_ABSOLUTE\_FOCUS
  - [lambda](#), 11
- CAMERA\_AUTO\_FOCUS
  - [lambda](#), 11
- CAMERA\_AUTO\_WHITEBALANCE
  - [lambda](#), 11
- CAMERA\_BACKLIGHT\_COMPENSATION
  - [lambda](#), 11
- CAMERA\_BRIGHTNESS
  - [lambda](#), 11
- CAMERA\_CONTRAST
  - [lambda](#), 11
- CAMERA\_FRAME\_RATE
  - [lambda](#), 11
- CAMERA\_PAN\_LEFT
  - [lambda](#), 11
- CAMERA\_PAN\_RESET
  - [lambda](#), 11
- CAMERA\_PAN\_RIGHT
  - [lambda](#), 11
- CAMERA\_PTZ\_RESET
  - [lambda](#), 11
- CAMERA\_SATURATION
  - [lambda](#), 11
- CAMERA\_TILT\_DOWN
  - [lambda](#), 11
- CAMERA\_TILT\_RESET
  - [lambda](#), 11
- CAMERA\_TILT\_UP
  - [lambda](#), 11
- CAMERA\_ZOOM\_IN
  - [lambda](#), 11
- CAMERA\_ZOOM\_OUT
  - [lambda](#), 11
- CAMERA\_ZOOM\_RESET
  - [lambda](#), 11
- CLOSE\_SOURCE\_SUCCESS
  - [lambda](#), 9
- COMPOSE\_MODE\_CHANGED
  - [lambda](#), 9
- COMPOSE\_PAP
  - [lambda](#), 8
- COMPOSE\_PIP\_MAIN\_ORIGINAL
  - [lambda](#), 8
- COMPOSE\_PIP\_MAIN\_PROCESSED
  - [lambda](#), 8
- CUTOFF\_BOUND
  - [lambda](#), 10
- ComposeMode
  - [lambda](#), 8
- ControlSource
  - [lambda](#), 12
- DEBUG\_MESSAGE
  - [lambda](#), 9
- END\_OF\_FILE
  - [lambda](#), 9
- ERROR\_MESSAGE
  - [lambda](#), 9
- EventType
  - [lambda](#), 8
- EventValue
  - [lambda::EventValue](#), 18
- FILE\_ARCHIVED
  - [lambda](#), 9
- FILE\_BACKWARD
  - [lambda](#), 11
- FILE\_DURATION
  - [lambda](#), 11
- FILE\_FORWARD
  - [lambda](#), 11
- FILE\_LOOP\_PLAY
  - [lambda](#), 11
- FILE\_NORMAL\_PLAYBACK
  - [lambda](#), 11
- FILE\_PAUSE
  - [lambda](#), 11
- FILE\_PLAY
  - [lambda](#), 11
- FILE\_PROGRESS
  - [lambda](#), 11
- FILE\_RECORD\_SEEK
  - [lambda](#), 11
- FILE\_SEEK
  - [lambda](#), 11
- FILE\_SINGLE\_PLAY
  - [lambda](#), 11
- FILE\_STOP
  - [lambda](#), 11
- FILE\_TURBO\_PLAYBACK
  - [lambda](#), 11
- FILTER\_CHANGED
  - [lambda](#), 9

- FLOAT\_VALUE
  - lambdavue, 12
- FRAME\_VALUE
  - lambdavue, 12
- FRAMERATE
  - lambdavue, 10
- GetActivationCode
  - lambdavue, 12
- GetCameraCount
  - lambdavue, 12
- GetComposeMode
  - lambdavue, 12
- GetExpirationDate
  - lambdavue, 13
- GetFilterParameter
  - lambdavue, 13
- GetLast5CharOfActivationCode
  - lambdavue, 13
- GetProcessMethod
  - lambdavue, 13
- GetSourceName
  - lambdavue, 13
- GetVerificationErrorString
  - lambdavue, 13
- GetWriterState
  - lambdavue, 14
- HIGHCUT
  - lambdavue, 10
- INFO\_MESSAGE
  - lambdavue, 9
- INTEGER\_VALUE
  - lambdavue, 12
- InitMagEngine
  - lambdavue, 14
- LAMBDA\_BOUND
  - lambdavue, 10
- LAMBDVUE\_SDK\_VERSION
  - lambdavue, 14
- LAMBDA
  - lambdavue, 10
- LEVEL\_BOUND
  - lambdavue, 10
- LEVEL
  - lambdavue, 10
- LICENSE\_FAILURE
  - lambdavue, 9
- LICENSE\_OFFLINE\_SUCCESS
  - lambdavue, 9
- LICENSE\_SUCCESS
  - lambdavue, 9
- LINEAR\_COLOR
  - lambdavue, 10
- LINEAR\_MOTION
  - lambdavue, 10
- LOWCUT
  - lambdavue, 10
- lambdavue, 5
  - AUTOFOCUS\_AVAILABLE, 9
  - AUTOFOCUS\_ENABLED, 9
  - CAMERA\_ABSOLUTE\_FOCUS, 11
  - CAMERA\_AUTO\_FOCUS, 11
  - CAMERA\_AUTO\_WHITEBALANCE, 11
  - CAMERA\_BACKLIGHT\_COMPENSATION, 11
  - CAMERA\_BRIGHTNESS, 11
  - CAMERA\_CONTRAST, 11
  - CAMERA\_FRAME\_RATE, 11
  - CAMERA\_PAN\_LEFT, 11
  - CAMERA\_PAN\_RESET, 11
  - CAMERA\_PAN\_RIGHT, 11
  - CAMERA\_PTZ\_RESET, 11
  - CAMERA\_SATURATION, 11
  - CAMERA\_TILT\_DOWN, 11
  - CAMERA\_TILT\_RESET, 11
  - CAMERA\_TILT\_UP, 11
  - CAMERA\_ZOOM\_IN, 11
  - CAMERA\_ZOOM\_OUT, 11
  - CAMERA\_ZOOM\_RESET, 11
  - CLOSE\_SOURCE\_SUCCESS, 9
  - COMPOSE\_MODE\_CHANGED, 9
  - COMPOSE\_PAP, 8
  - COMPOSE\_PIP\_MAIN\_ORIGINAL, 8
  - COMPOSE\_PIP\_MAIN\_PROCESSED, 8
  - CUTOFF\_BOUND, 10
  - ComposeMode, 8
  - ControlSource, 12
  - DEBUG\_MESSAGE, 9
  - END\_OF\_FILE, 9
  - ERROR\_MESSAGE, 9
  - EventType, 8
  - FILE\_ARCHIVED, 9
  - FILE\_BACKWARD, 11
  - FILE\_DURATION, 11
  - FILE\_FORWARD, 11
  - FILE\_LOOP\_PLAY, 11
  - FILE\_NORMAL\_PLAYBACK, 11
  - FILE\_PAUSE, 11
  - FILE\_PLAY, 11
  - FILE\_PROGRESS, 11
  - FILE\_RECORD\_SEEK, 11
  - FILE\_SEEK, 11
  - FILE\_SINGLE\_PLAY, 11
  - FILE\_STOP, 11
  - FILE\_TURBO\_PLAYBACK, 11
  - FILTER\_CHANGED, 9
  - FLOAT\_VALUE, 12
  - FRAME\_VALUE, 12
  - FRAMERATE, 10
  - GetActivationCode, 12
  - GetCameraCount, 12
  - GetComposeMode, 12
  - GetExpirationDate, 13
  - GetFilterParameter, 13
  - GetLast5CharOfActivationCode, 13

GetProcessMethod, 13  
GetSourceName, 13  
GetVerificationErrorString, 13  
GetWriterState, 14  
HIGHCUT, 10  
INFO\_MESSAGE, 9  
INTEGER\_VALUE, 12  
InitMagEngine, 14  
LAMBDA\_BOUND, 10  
LAMBDAVUE\_SDK\_VERSION, 14  
LAMBDA, 10  
LEVEL\_BOUND, 10  
LEVEL, 10  
LICENSE\_FAILURE, 9  
LICENSE\_OFFLINE\_SUCCESS, 9  
LICENSE\_SUCCESS, 9  
LINEAR\_COLOR, 10  
LINEAR\_MOTION, 10  
LOWCUT, 10  
LicenseStatus, 9  
OPEN\_CAMERA\_FAIL, 9  
OPEN\_CAMERA\_SOURCE, 11  
OPEN\_CAMERA\_SUCCESS, 9  
OPEN\_FILE\_FAIL, 9  
OPEN\_FILE\_SOURCE, 11  
OPEN\_FILE\_SUCCESS, 9  
ORIGINAL\_IMAGE, 9  
OUTPUT\_ORIGINAL, 10  
OUTPUT\_PROCESSED, 10  
OpenSource, 14  
OutputFileType, 9  
PROCESSED\_IMAGE, 9  
ParameterType, 10  
ProcessMethod, 10  
QuerySource, 15  
RECORD\_BOTH, 10  
RECORD\_ORIGINAL\_ONLY, 10  
RECORD\_PROCESSED\_ONLY, 10  
RIESZ\_MOTION, 10  
ROI\_RESET, 9  
RecordType, 10  
STRING\_VALUE, 12  
SetBypassMagnify, 15  
SetComposeMode, 15  
SetFilterParameter, 15  
SetOutputFilename, 15  
SetProcessMethod, 16  
SetROIView, 16  
SetRecordType, 16  
SourceAccess, 10  
SourceOperation, 11  
VALUE\_ALPHA, 9  
VALUE\_CAMERA\_COUNT, 9  
VALUE\_FILE\_DURATION, 9  
VALUE\_FILE\_PROGRESS, 9  
VALUE\_HIGHCUT, 9  
VALUE\_LAMBDA, 9  
VALUE\_LEVEL, 9  
VALUE\_LOWCUT, 9  
VALUE\_PROCESS\_FRAME\_RATE, 9  
VALUE\_SOURCE\_FRAME\_RATE, 9  
ValueType, 11  
WARNING\_MESSAGE, 9  
lambdavue::EventValue, 17  
    EventValue, 18  
lambdavue::FilterParameter, 19  
lambdavue::Magnifier, 19  
    MagGetActivationCode, 20  
    MagGetComposeMode, 20  
    MagGetExpiryTimeT, 21  
    MagGetFilterParameter, 21  
    MagGetImage, 21  
    MagGetLast5CharActivationCode, 21  
    MagGetLicenseVerificationResult, 21  
    MagGetProcessMethod, 22  
    MagHaveFilter, 22  
    MagPushImage, 22  
    MagSetComposeMode, 22  
    MagSetFilterParameter, 22  
    MagSetProcessMethod, 23  
    MagSetRoi, 23  
lambdavue::MediaWriter, 23  
    MwGetChannelNumber, 24  
    MwGetDestFilename, 24  
    MwGetDestFullFilename, 24  
    MwGetDestPath, 25  
    MwGetHeight, 25  
    MwGetSampleRate, 25  
    MwGetWidth, 25  
    MwSetAV, 25  
    MwSetFileName, 26  
    MwSetHeight, 26  
    MwSetWidth, 26  
    MwWriteVideo, 26  
lambdavue::SourceManager, 27  
    sm\_is\_file, 28  
    sm\_is\_valid, 28  
    SmControl, 28  
    SmGetCameraCount, 28  
    SmGetFrameCount, 28  
    SmGetFramerate, 28  
    SmGetHeight, 29  
    SmGetImage, 29  
    SmGetSourceName, 29  
    SmGetWidth, 29  
    SmQuery, 29  
    SmSetSource, 30  
lambdavue::VerificationResult, 30  
lambdavue::WriterManager, 31  
    WmCloseWriter, 31  
    WmGetWriterState, 31  
    WmInitWriter, 32  
    WmSetOutputDestination, 32  
    WmSetOutputResolution, 32  
    WmSetRecordType, 32  
    WmWriteImage, 33

- lambdavue::WriterState, 33
- LicenseStatus
  - lambdavue, 9
- MagGetActivationCode
  - lambdavue::Magnifier, 20
- MagGetComposeMode
  - lambdavue::Magnifier, 20
- MagGetExpiryTimeT
  - lambdavue::Magnifier, 21
- MagGetFilterParameter
  - lambdavue::Magnifier, 21
- MagGetImage
  - lambdavue::Magnifier, 21
- MagGetLast5CharActivationCode
  - lambdavue::Magnifier, 21
- MagGetLicenseVerificationResult
  - lambdavue::Magnifier, 21
- MagGetProcessMethod
  - lambdavue::Magnifier, 22
- MagHaveFilter
  - lambdavue::Magnifier, 22
- MagPushImage
  - lambdavue::Magnifier, 22
- MagSetComposeMode
  - lambdavue::Magnifier, 22
- MagSetFilterParameter
  - lambdavue::Magnifier, 22
- MagSetProcessMethod
  - lambdavue::Magnifier, 23
- MagSetRoi
  - lambdavue::Magnifier, 23
- MwGetChannelNumber
  - lambdavue::MediaWriter, 24
- MwGetDestFilename
  - lambdavue::MediaWriter, 24
- MwGetDestFullFilename
  - lambdavue::MediaWriter, 24
- MwGetDestPath
  - lambdavue::MediaWriter, 25
- MwGetHeight
  - lambdavue::MediaWriter, 25
- MwGetSampleRate
  - lambdavue::MediaWriter, 25
- MwGetWidth
  - lambdavue::MediaWriter, 25
- MwSetAV
  - lambdavue::MediaWriter, 25
- MwSetFileName
  - lambdavue::MediaWriter, 26
- MwSetHeight
  - lambdavue::MediaWriter, 26
- MwSetWidth
  - lambdavue::MediaWriter, 26
- MwWriteVideo
  - lambdavue::MediaWriter, 26
- OPEN\_CAMERA\_FAIL
  - lambdavue, 9
- OPEN\_CAMERA\_SOURCE
  - lambdavue, 11
- OPEN\_CAMERA\_SUCCESS
  - lambdavue, 9
- OPEN\_FILE\_FAIL
  - lambdavue, 9
- OPEN\_FILE\_SOURCE
  - lambdavue, 11
- OPEN\_FILE\_SUCCESS
  - lambdavue, 9
- ORIGINAL\_IMAGE
  - lambdavue, 9
- OUTPUT\_ORIGINAL
  - lambdavue, 10
- OUTPUT\_PROCESSED
  - lambdavue, 10
- OpenSource
  - lambdavue, 14
- OutputFileType
  - lambdavue, 9
- PROCESSED\_IMAGE
  - lambdavue, 9
- ParameterType
  - lambdavue, 10
- ProcessMethod
  - lambdavue, 10
- QuerySource
  - lambdavue, 15
- RECORD\_BOTH
  - lambdavue, 10
- RECORD\_ORIGINAL\_ONLY
  - lambdavue, 10
- RECORD\_PROCESSED\_ONLY
  - lambdavue, 10
- RIESZ\_MOTION
  - lambdavue, 10
- ROI\_RESET
  - lambdavue, 9
- RecordType
  - lambdavue, 10
- STRING\_VALUE
  - lambdavue, 12
- SetBypassMagnify
  - lambdavue, 15
- SetComposeMode
  - lambdavue, 15
- SetFilterParameter
  - lambdavue, 15
- SetOutputFilename
  - lambdavue, 15
- SetProcessMethod
  - lambdavue, 16
- SetROIView
  - lambdavue, 16
- SetRecordType

- lambdavue, 16
- sm\_is\_file
  - lambdavue::SourceManager, 28
- sm\_is\_valid
  - lambdavue::SourceManager, 28
- SmControl
  - lambdavue::SourceManager, 28
- SmGetCameraCount
  - lambdavue::SourceManager, 28
- SmGetFrameCount
  - lambdavue::SourceManager, 28
- SmGetFramerate
  - lambdavue::SourceManager, 28
- SmGetHeight
  - lambdavue::SourceManager, 29
- SmGetImage
  - lambdavue::SourceManager, 29
- SmGetSourceName
  - lambdavue::SourceManager, 29
- SmGetWidth
  - lambdavue::SourceManager, 29
- SmQuery
  - lambdavue::SourceManager, 29
- SmSetSource
  - lambdavue::SourceManager, 30
- SourceAccess
  - lambdavue, 10
- SourceOperation
  - lambdavue, 11
  
- VALUE\_ALPHA
  - lambdavue, 9
- VALUE\_CAMERA\_COUNT
  - lambdavue, 9
- VALUE\_FILE\_DURATION
  - lambdavue, 9
- VALUE\_FILE\_PROGRESS
  - lambdavue, 9
- VALUE\_HIGHCUT
  - lambdavue, 9
- VALUE\_LAMBDA
  - lambdavue, 9
- VALUE\_LEVEL
  - lambdavue, 9
- VALUE\_LOWCUT
  - lambdavue, 9
- VALUE\_PROCESS\_FRAME\_RATE
  - lambdavue, 9
- VALUE\_SOURCE\_FRAME\_RATE
  - lambdavue, 9
- ValueType
  - lambdavue, 11
  
- WARNING\_MESSAGE
  - lambdavue, 9
- WmCloseWriter
  - lambdavue::WriterManager, 31
- WmGetWriterState
  - lambdavue::WriterManager, 31
  
- WmInitWriter
  - lambdavue::WriterManager, 32
- WmSetOutputDestination
  - lambdavue::WriterManager, 32
- WmSetOutputResolution
  - lambdavue::WriterManager, 32
- WmSetRecordType
  - lambdavue::WriterManager, 32
- WmWriteImage
  - lambdavue::WriterManager, 33