Namespace Documentation

Package MagicMirrorHelper

Namespaces

- package **Object**

Classes

- class **AppHelper**
- class **HardwareHelper**

Delegate Functions

- delegate void **Email_Success** ()
  Delegate call if Email has been successfully sent.
- delegate void **Email_Failed** ()
  Delegate call if Email failed to send.
- delegate void **Facebook_LoginSuccessful** ()
  Delegate call if Facebook login process succeed.
- delegate void **Facebook_UploadSuccessful** ()
  Delegate call if Facebook upload process succeed.
- delegate void **Facebook_LoginFailed** ()
  Delegate call if Facebook login process failed.
- delegate void **Twitter_LoginSuccessful** ()
  Delegate call if Twitter login process succeed.
- delegate void **Twitter_UploadSuccessful** ()
  Delegate call if Twitter upload process succeed.
- delegate void **Twitter_LoginFailed** ()
  Delegate call if Twitter login process failed.
- delegate void **Camera_OnAdded** ()
  Delegate call when a camera is detected. Eg: when a camera is plugged in, or turned on.
- delegate void **Camera_OnRemoved** ()
  Delegate call when a camera is removed. Eg: when a camera is unplugged, or turned off.
- delegate void **Camera_OnError** (int code)
  Delegate call when error is detected. Examples of errors detected are autofocus failed, device busy, communication error, and etc.
- delegate void **Camera_OnLiveViewUpdated** (System.Drawing.Image img)
  Delegate call when live view image is updated from camera, will be called after StartLiveView()
- delegate void **Camera_OnFileGenerated** (String pathToFile)
  Delegate call when a file is generated from camera, will be called after TakePicture() or StopRecording()
- delegate void **Kinect_OnError** (int code)
  Delegate call when kinect detects any error
- delegate void **Kinect_OnPersonDetected** (Skeleton skeleton)
  Delegate call when kinect found a new skeleton

- delegate void **Kinect_OnPersonTracked** (Skeleton skeleton)
  Delegate call when kinect is tracking a skeleton. When a new skeleton entered the detection range, Kinect_OnPersonDetetced will be called, any futher movement for this skeleton will be pushed via this callback method.
- delegate void **Kinect_OnPersonExited** (Skeleton skeleton)
  Delegate call when a skeleton being tracked has exited the detection range.
- delegate void **Kinect_OnSwipeLeft** ()
  Delegate call if main skeleton being tracked performed a swipe left gesture.
- delegate void **Kinect_OnSwipeRight** ()
  Delegate call if main skeleton being tracked performed a swipe right gesture.
- delegate void **Kinect_OnRaiseLeftHand** ()
  Delegate call if main skeleton being tracked performed a raise left hand gesture.
- delegate void **Kinect_OnRaiseRightHand** ()
  Delegate call if main skeleton being tracked performed a raise right hand gesture.
- delegate void **Kinect_OnHandPushed** ()
  Delegate call if main skeleton being tracked performed a push gesture with any hand.
- delegate void **ZebraPrinter_OnError** (int code)
  Delegate call when printer detects any error

Function Documentation

delegate void MagicMirrorHelper.Camera_OnAdded ()

Delegate call when a camera is detected. Eg: when a camera is plugged in, or turned on.

delegate void MagicMirrorHelper.Camera_OnError (int *code*)

Delegate call when error is detected. Examples of errors detected are autofocus failed, device busy, communication error, and etc.

Parameters:

code	Code of error. Refer to Canon EDSDK document for description

delegate void MagicMirrorHelper.Camera_OnFileGenerated (String *pathToFile*)

Delegate call when a file is generated from camera, will be called after TakePicture() or StopRecording()

Parameters:

pathToFile	Absolute path to generated file

delegate void MagicMirrorHelper.Camera_OnLiveViewUpdated (System.Drawing.Image *img*)

Delegate call when live view image is updated from camera, will be called after StartLiveView()

Parameters:

img	Live view object in Image form

delegate void MagicMirrorHelper.Camera_OnRemoved ()

Delegate call when a camera is removed. Eg: when a camera is unplugged, or turned off.

delegate void MagicMirrorHelper.Email_Failed ()

Delegate call if Email failed to send.

delegate void MagicMirrorHelper.Email_Success ()

Delegate call if Email has been successfully sent.

delegate void MagicMirrorHelper.Facebook_LoginFailed ()

Delegate call if Facebook login process failed.

delegate void MagicMirrorHelper.Facebook_LoginSuccessful ()

Delegate call if Facebook login process succeed.

delegate void MagicMirrorHelper.Facebook_UploadSuccessful ()

Delegate call if Facebook upload process succeed.

delegate void MagicMirrorHelper.Kinect_OnError (int *code*)

Delegate call when kinect detects any error

Parameters:

code	Code of error

delegate void MagicMirrorHelper.Kinect_OnHandPushed ()

Delegate call if main skeleton being tracked performed a push gesture with any hand.

delegate void MagicMirrorHelper.Kinect_OnPersonDetected (Skeleton *skeleton*)

Delegate call when kinect found a new skeleton

Parameters:

skeleton	Skeleton object of person detected

delegate void MagicMirrorHelper.Kinect_OnPersonExited (Skeleton *skeleton*)

Delegate call when a skeleton being tracked has exited the detection range.

Parameters:

skeleton	Skeleton object of person being tracked

delegate void MagicMirrorHelper.Kinect_OnPersonTracked (Skeleton *skeleton*)

Delegate call when kinect is tracking a skeleton. When a new skeleton entered the detection range, Kinect_OnPersonDetetced will be called, any futher movement for this skeleton will be pushed via this callback method.

Parameters:

skeleton	Skeleton object of person being tracked

delegate void MagicMirrorHelper.Kinect_OnRaiseLeftHand ()

Delegate call if main skeleton being tracked performed a raise left hand gesture.

delegate void MagicMirrorHelper.Kinect_OnRaiseRightHand ()

Delegate call if main skeleton being tracked performed a raise right hand gesture.

delegate void MagicMirrorHelper.Kinect_OnSwipeLeft ()

Delegate call if main skeleton being tracked performed a swipe left gesture.

delegate void MagicMirrorHelper.Kinect_OnSwipeRight ()

Delegate call if main skeleton being tracked performed a swipe right gesture.

delegate void MagicMirrorHelper.Twitter_LoginFailed ()

Delegate call if Twitter login process failed.

delegate void MagicMirrorHelper.Twitter_LoginSuccessful ()

Delegate call if Twitter login process succeed.

delegate void MagicMirrorHelper.Twitter_UploadSuccessful ()

Delegate call if Twitter upload process succeed.

delegate void MagicMirrorHelper.ZebraPrinter_OnError (int *code*)

Delegate call when printer detects any error

Parameters:

code	Code of error

Class Documentation

MagicMirrorHelper.AppHelper Class Reference

Helper class providing access to :
- Manage User object
- Manage Apps and Products
- Sharing on Facebook, Twitter
- Email

Public Member Functions

- **AppHelper GetInstance** (int companyId, String companyName, int branchId, String branchName, String branchCode, String softwareName)
  *Get an instance of **AppHelper** that helps you manage most of Magic Mirror's basic flow. In most situations you only need to get this instance once throughout the whole application flow.*
- **User GenerateNewUser** ()
  *Generate a new User object, replacing the old user session. Any further call made to **GetCurrentUser()** will return the same instance of User, until this function is called again. You might want to consider logging user data before generate a new one by calling **LogUserData()**.*
- **User GetCurrentUser** ()
  Returns an instance of current User object
- void **LogUserData** ()
  Write the mail details of current user object into a CSV file, so you could have a "report" generated.
- void **InitialiseAppsAndProducts** ()
  Pre-generate list of Apps and Products using default CSV setting.
- void **SetApps** (List< **FrameData** > CustomFrames)
  Use this function to override default list of Apps if you would like to maintain your own list of Apps
- List< **FrameData** > **GetApps** ()
  Get the List of Apps being used
- void **MoveToNextApp** ()
  Increase the selected index of Apps list
- void **MoveToPreviousApp** ()
  Decrease the selected index of Apps list
- **FrameData GetSelectedApp** ()
  Get an instance of currently selected App
- void **SetProducts** (List< **GroupData** > CustomGroups)
  Use this function to override default list of Products if you would like to maintain your own list of Products
- List< **GroupData** > **GetProducts** ()
  Get the List of Products being used
- void **MoveToNextGroupOfProducts** ()
  Increase the selected index of Products list
- void **MoveToPreviousGroupOfProducts** ()
  Decrease the selected index of Products list
- **GroupData GetSelectedGroup** ()
  Get an instance of currently selected Product

- void **Facebook_LoadWebpage** (WebBrowser webBrowser)
  Web browser supplied will be navigated to Facebook login page. You might want to listen to WebBrowser.Navigated event to make sure it has successfully landed on Facebook Login page.
- void **Facebook_SignIn** (String UserName, String Password, List< String > imagesToUpload, **Facebook_LoginSuccessful Facebook_LoginSuccessful**, **Facebook_LoginFailed Facebook_LoginFailed**, **Facebook_UploadSuccessful Facebook_UploadSuccessful**)
  *Sign in to Facebook using given username and password. Please make sure this is called only if **Facebook_LoadWebpage()** has been called and the WebBrowser is landed on Facebook Login page.*
- void **Twitter_LoadWebpage** (WebBrowser webBrowser)
  Web browser supplied will be navigated to Twitter login page. You might want to listen to WebBrowser.Navigated event to make sure it has successfully landed on Twitter Login page.
- void **Twitter_SignIn** (String UserName, String Password, List< String > imagesToUpload, **Twitter_LoginSuccessful Twitter_LoginSuccessful**, **Twitter_LoginFailed Twitter_LoginFailed**, **Twitter_UploadSuccessful Twitter_UploadSuccessful**)
  *Sign in to Twitter using given username and password. Please make sure this is called only if **Twitter_LoadWebpage()** has been called and the WebBrowser is landed on Twitter Login page.*
- void **Email_Send** (String Email, List< String > imagesToSend, **Email_Success Email_Success**, **Email_Failed Email_Failed**)
  *Sends an email to given email address. If imagesToSend is null, anything in **GetCurrentUser()**.ProcessedMediaFiles will be uploaded. If imagesToSend is not null, anything in this list will be uploaded.*

Detailed Description

Helper class providing access to :

- Manage User object
- Manage Apps and Products
- Sharing on Facebook, Twitter
- Email

Member Function Documentation

void MagicMirrorHelper.AppHelper.Email_Send (String *Email*, List< String > *imagesToSend*, Email_Success *Email_Success*, Email_Failed *Email_Failed*)

Sends an email to given email address. If imagesToSend is null, anything in **GetCurrentUser()**.ProcessedMediaFiles will be uploaded. If imagesToSend is not null, anything in this list will be uploaded.

Parameters:

Email	Receiver's email address
imagesToSend	List of absolute path to images to be sent
Email_Success	Delegate callback if mail successfully sent
Email_Failed	Delegate call back if mail sending failed

void MagicMirrorHelper.AppHelper.Facebook_LoadWebpage (WebBrowser *webBrowser*)

Web browser supplied will be navigated to Facebook login page. You might want to listen to WebBrowser.Navigated event to make sure it has successfully landed on Facebook Login page.

Parameters:

webBrowser	An instance of WebBrowser being displayed on WPF page

void MagicMirrorHelper.AppHelper.Facebook_SignIn (String *UserName*, String *Password*, List< String > *imagesToUpload*, Facebook_LoginSuccessful *Facebook_LoginSuccessful*, Facebook_LoginFailed *Facebook_LoginFailed*, Facebook_UploadSuccessful *Facebook_UploadSuccessful*)

Sign in to Facebook using given username and password. Please make sure this is called only if **Facebook_LoadWebpage()** has been called and the WebBrowser is landed on Facebook Login page.

Upon successful sign in : If imagesToUpload is null, anything in **GetCurrentUser()**.ProcessedMediaFiles will be uploaded. If imagesToUpload is not null, anything in this list will be uploaded.

Parameters:

UserName	the username
Password	the password
imagesToUpload	if this value is not null, this list of images will be uploaded upon sign in succeed
Facebook_LoginSuccessful	Delegate callback when login succeed
Facebook_LoginFailed	Delegate callback when login failed
Facebook_UploadSuccessful	Delegate callback when upload succeed

User MagicMirrorHelper.AppHelper.GenerateNewUser ()

Generate a new User object, replacing the old user session. Any further call made to **GetCurrentUser()** will return the same instance of User, until this function is called again. You might want to consider logging user data before generate a new one by calling **LogUserData()**.

Returns:

Generated User object

List<FrameData> MagicMirrorHelper.AppHelper.GetApps ()

Get the List of Apps being used

Returns:

The instance of Apps list currently being used

User MagicMirrorHelper.AppHelper.GetCurrentUser ()

Returns an instance of current User object

Returns:

Instance of current User object

AppHelper MagicMirrorHelper.AppHelper.GetInstance (int *companyId*, String *companyName*, int *branchId*, String *branchName*, String *branchCode*, String *softwareName*)

Get an instance of **AppHelper** that helps you manage most of Magic Mirror's basic flow. In most situations you only need to get this instance once throughout the whole application flow.

For all parameters below please request from Magic Mirror's team.

Parameters:

companyId	companyId assigned
companyName	companyName assigned
branchId	branchId assigned
branchName	branchName assigned
branchCode	branchCode assigned
softwareName	softwareName assigned

Returns:

An instance of **AppHelper**

List<GroupData> MagicMirrorHelper.AppHelper.GetProducts ()

Get the List of Products being used

Returns:

The instance of Products list currently being used

FrameData MagicMirrorHelper.AppHelper.GetSelectedApp ()

Get an instance of currently selected App

Returns:

An instance of currently selected App

GroupData MagicMirrorHelper.AppHelper.GetSelectedGroup ()

Get an instance of currently selected Product

Returns:

An instance of currently selected Product

void MagicMirrorHelper.AppHelper.InitialiseAppsAndProducts ()

Pre-generate list of Apps and Products using default CSV setting.

void MagicMirrorHelper.AppHelper.LogUserData ()

Write the mail details of current user object into a CSV file, so you could have a "report" generated.

void MagicMirrorHelper.AppHelper.MoveToNextApp ()

Increase the selected index of Apps list

void MagicMirrorHelper.AppHelper.MoveToNextGroupOfProducts ()

Increase the selected index of Products list

void MagicMirrorHelper.AppHelper.MoveToPreviousApp ()

Decrease the selected index of Apps list

void MagicMirrorHelper.AppHelper.MoveToPreviousGroupOfProducts ()

Decrease the selected index of Products list

void MagicMirrorHelper.AppHelper.SetApps (List< FrameData > *CustomFrames*)

Use this function to override default list of Apps if you would like to maintain your own list of Apps

Parameters:

CustomFrames	Your custom list of Apps

void MagicMirrorHelper.AppHelper.SetProducts (List< GroupData > *CustomGroups*)

Use this function to override default list of Products if you would like to maintain your own list of Products

Parameters:

CustomGroups	Your custom list of Products

void MagicMirrorHelper.AppHelper.Twitter_LoadWebpage (WebBrowser *webBrowser*)

Web browser supplied will be navigated to Twitter login page. You might want to listen to WebBrowser.Navigated event to make sure it has successfully landed on Twitter Login page.

Parameters:

webBrowser	An instance of WebBrowser being displayed on WPF page
Twitter_LoginSuccessful	
Twitter_LoginFailed	
Twitter_UploadSuccessful	

void MagicMirrorHelper.AppHelper.Twitter_SignIn (String *UserName*, String *Password*, List< String > *imagesToUpload*, Twitter_LoginSuccessful *Twitter_LoginSuccessful*, Twitter_LoginFailed *Twitter_LoginFailed*, Twitter_UploadSuccessful *Twitter_UploadSuccessful*)

Sign in to Twitter using given username and password. Please make sure this is called only if **Twitter_LoadWebpage()** has been called and the WebBrowser is landed on Twitter Login page.

Upon successful sign in : If imagesToUpload is null, anything in **GetCurrentUser()**.ProcessedMediaFiles will be uploaded. If imagesToUpload is not null, anything in this list will be uploaded.

Parameters:

UserName	the username
Password	the password
imagesToUpload	if this value is not null, this list of images will be uploaded upon sign in succeed
Facebook_LoginSuccessful	Delegate callback when login succeed
Facebook_LoginFailed	Delegate callback when login failed
Facebook_Upload Successful	Delegate callback when upload succeed

---

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

- AppHelper.cs

MagicMirrorHelper.HardwareHelper Class Reference

Helper class providing access to :
- Camera
- Kinect
- Zebra printer

Public Member Functions

- void **InitialiseCamera** (Camera_OnAdded OnCameraAdded, Camera_OnRemoved OnCameraRemoved, Camera_OnError OnCameraError, Camera_OnLiveViewUpdated OnLiveViewUpdated, Camera_OnFileGenerated OnFileGenerated)
  Initialize the camera, please call this during the startup of project, and only call this function once. Creates a session instance with camera attached, if there is any, which enables the use of most camera functions.
- bool **IsCameraAvailable** ()
  Check if Windows is able to detect any camera attached
- void **StartLiveView** ()
  *Send a command to camera which starts the live view. If there is any OnLiveViewUpdated delegate assigned, expect callback to happen right after this. Do remember to call **StopLiveView()** when live view is no longer needed.*
- void **StopLiveView** ()
  Send a command to camera which stops the live view. Callbacks to OnLiveViewUpdated will stop after this.
- void **SetImageDirectory** (String path)
  Set the directory where images/videos taken will be saved at.
- void **TakePicture** ()
  Send a command to camera which starts taking a picture. Expect a file to be generated and OnFileGenerated called afterwards, if there is no error.
- void **StartRecording** ()
  *Send a command to camera which starts recording. Make a call to **StopRecording()** when you wish to stops recording.*
- void **StopRecording** ()
  *Stops recording. Call this only after a call to **StartRecording()** has been made. Expect a file to be generated and OnFileGenerated called afterwards, if there is no error.*
- void **InitialiseKinect** (Kinect_OnError Kinect_OnError, Kinect_OnPersonDetected Kinect_OnPersonDetected, Kinect_OnPersonTracked Kinect_OnPersonTracked, Kinect_OnPersonExited Kinect_OnPersonExited, Kinect_OnSwipeLeft Kinect_OnSwipeLeft, Kinect_OnSwipeRight Kinect_OnSwipeRight, Kinect_OnRaiseLeftHand Kinect_OnRaiseLeftHand, Kinect_OnRaiseRightHand Kinect_OnRaiseRightHand, Kinect_OnHandPushed Kinect_OnHandPushed)
  Initialize the kinect device, please call this during the startup of project, and only call this function once. Creates a session instance with kinect attached, if there is any. You may use "null" if you do not wish to assign a delegate function for any callback method below.
- bool **IsKinectAvailable** ()
  Check if Windows is able to detect any Kinect device.
- void **InitialiseZebraPrinter** (ZebraPrinter_OnError ZebraPrinter_OnError)
  Initialize the printer device, please call this during the startup of project, and only call this function once. Creates a session instance with printer attached, if there is any.

- bool **IsZebraPrinterAvailable** ()
  Check if Windows is able to detect any Zebra Printer.
- void **Print** (Receipt Receipt)
  Sends a print command to zebra printer

Properties

- string **Camera_WhiteBalance** `[set]`
  Setter for camera WhiteBalance property
- string **Camera_WhiteBalanceShift** `[set]`
  Setter for camera WhiteBalanceShift property
- string **Camera_StyleSharpness** `[set]`
  Setter for camera StyleSharpness property
- string **Camera_StyleContrast** `[set]`
  Setter for camera StyleContrast property
- string **Camera_StyleSaturation** `[set]`
  Setter for camera StyleSaturation property
- string **Camera_StyleColorTone** `[set]`
  Setter for camera StyleColorTone property
- string **Camera_StyleFilterEffect** `[set]`
  Setter for camera StyleFilterEffect property
- string **Camera_StyleToningEffect** `[set]`
  Setter for camera StyleToningEffect property
- string **Camera_AEMode** `[set]`
  Setter for camera AEMode property
- string **Camera_ISOSpeedLiveView** `[set]`
  Setter for camera ISOSpeedLiveView property
- string **Camera_AvLiveView** `[set]`
  Setter for camera AvLiveView property
- string **Camera_TvLiveView** `[set]`
  Setter for camera TvLiveView property
- string **Camera_ISOSpeedCapture** `[set]`
  Setter for camera ISOSpeedCapture property
- string **Camera_AvCapture** `[set]`
  Setter for camera AvCapture property
- string **Camera_TvCapture** `[set]`
  Setter for camera TvCapture property
- string **Camera_ExposureCompensation** `[set]`
  Setter for camera ExposureCompensation property

Detailed Description

Helper class providing access to :

- Camera

- Kinect
- Zebra printer

Member Function Documentation

void MagicMirrorHelper.HardwareHelper.InitialiseCamera (Camera_OnAdded *OnCameraAdded*, Camera_OnRemoved *OnCameraRemoved*, Camera_OnError *OnCameraError*, Camera_OnLiveViewUpdated *OnLiveViewUpdated*, Camera_OnFileGenerated *OnFileGenerated*)

Initialize the camera, please call this during the startup of project, and only call this function once. Creates a session instance with camera attached, if there is any, which enables the use of most camera functions.

Parameters:

OnCameraAdded	Delegate method when a camera is detected
OnCameraRemoved	Delegate method when a camera is removed
OnCameraError	Delegate method when camera throws an error
OnLiveViewUpdated	Delegate method when live view image is pushed from camera
OnFileGenerated	Delegate method when a file is generated by taking picture, or recording

void MagicMirrorHelper.HardwareHelper.InitialiseKinect (Kinect_OnError *Kinect_OnError*, Kinect_OnPersonDetected *Kinect_OnPersonDetected*, Kinect_OnPersonTracked *Kinect_OnPersonTracked*, Kinect_OnPersonExited *Kinect_OnPersonExited*, Kinect_OnSwipeLeft *Kinect_OnSwipeLeft*, Kinect_OnSwipeRight *Kinect_OnSwipeRight*, Kinect_OnRaiseLeftHand *Kinect_OnRaiseLeftHand*, Kinect_OnRaiseRightHand *Kinect_OnRaiseRightHand*, Kinect_OnHandPushed *Kinect_OnHandPushed*)

Initialize the kinect device, please call this during the startup of project, and only call this function once. Creates a session instance with kinect attached, if there is any. You may use "null" if you do not wish to assign a delegate function for any callback method below.

Parameters:

Kinect_OnError	Delegate method when kinect detects an error
Kinect_OnPersonDetected	Delegate method when kinect detects a new skeleton entering detection range
Kinect_OnPersonTracked	Delegate method when kinect is tracking the same person entered detection range
Kinect_OnPersonExited	Delegate method when kinect detects a tracking skeleton exited its detection range
Kinect_OnSwipeLeft	Delegate method when kinect detects a swipe left gesture
Kinect_OnSwipeRight	Delegate method when kinect detects a swipe right gesture
Kinect_OnRaiseLeftHand	Delegate method when kinect detects a raise left hand gesture
Kinect_OnRaiseRi	Delegate method when kinect detects a raise right hand gesture

ghtHand	
Kinect_OnHandPushed	Delegate method when kinect detects a push gesture by any hand

void MagicMirrorHelper.HardwareHelper.InitialiseZebraPrinter (ZebraPrinter_OnError *ZebraPrinter_OnError*)

Initialize the printer device, please call this during the startup of project, and only call this function once. Creates a session instance with printer attached, if there is any.

Parameters:

ZebraPrinter_OnError	Delegate method when printer detects any error

bool MagicMirrorHelper.HardwareHelper.IsCameraAvailable ()

Check if Windows is able to detect any camera attached

Returns:

True if camera is detected, false otherwise

bool MagicMirrorHelper.HardwareHelper.IsKinectAvailable ()

Check if Windows is able to detect any Kinect device.

Returns:

True if Kinect device is detected, false otherwise.

bool MagicMirrorHelper.HardwareHelper.IsZebraPrinterAvailable ()

Check if Windows is able to detect any Zebra Printer.

Returns:

True if zebra printer is detected, false otherwise.

void MagicMirrorHelper.HardwareHelper.Print (Receipt *Receipt*)

Sends a print command to zebra printer

Parameters:

Receipt	

void MagicMirrorHelper.HardwareHelper.SetImageDirectory (String *path*)

Set the directory where images/videos taken will be saved at.

Parameters:

path	Absolute path to a folder, please make sure the folder is generated

void MagicMirrorHelper.HardwareHelper.StartLiveView ()

Send a command to camera which starts the live view. If there is any OnLiveViewUpdated delegate assigned, expect callback to happen right after this. Do remember to call **StopLiveView()** when live view is no longer needed.

void MagicMirrorHelper.HardwareHelper.StartRecording ()

Send a command to camera which starts recording. Make a call to **StopRecording()** when you wish to stops recording.

void MagicMirrorHelper.HardwareHelper.StopLiveView ()

Send a command to camera which stops the live view. Callbacks to OnLiveViewUpdated will stop after this.

void MagicMirrorHelper.HardwareHelper.StopRecording ()

Stops recording. Call this only after a call to **StartRecording()** has been made. Expect a file to be generated and OnFileGenerated called afterwards, if there is no error.

void MagicMirrorHelper.HardwareHelper.TakePicture ()

Send a command to camera which starts taking a picture. Expect a file to be generated and OnFileGenerated called afterwards, if there is no error.

Property Documentation

string MagicMirrorHelper.HardwareHelper.Camera_AEMode `[set]`

Setter for camera AEMode property

string MagicMirrorHelper.HardwareHelper.Camera_AvCapture `[set]`

Setter for camera AvCapture property

string MagicMirrorHelper.HardwareHelper.Camera_AvLiveView `[set]`

Setter for camera AvLiveView property

string MagicMirrorHelper.HardwareHelper.Camera_ExposureCompensation `[set]`

Setter for camera ExposureCompensation property

string MagicMirrorHelper.HardwareHelper.Camera_ISOSpeedCapture `[set]`

Setter for camera ISOSpeedCapture property

string MagicMirrorHelper.HardwareHelper.Camera_ISOSpeedLiveView `[set]`

Setter for camera ISOSpeedLiveView property

string MagicMirrorHelper.HardwareHelper.Camera_StyleColorTone `[set]`

Setter for camera StyleColorTone property

string MagicMirrorHelper.HardwareHelper.Camera_StyleContrast `[set]`

Setter for camera StyleContrast property

string MagicMirrorHelper.HardwareHelper.Camera_StyleFilterEffect `[set]`

Setter for camera StyleFilterEffect property

string MagicMirrorHelper.HardwareHelper.Camera_StyleSaturation `[set]`

Setter for camera StyleSaturation property

string MagicMirrorHelper.HardwareHelper.Camera_StyleSharpness `[set]`

Setter for camera StyleSharpness property

string MagicMirrorHelper.HardwareHelper.Camera_StyleToningEffect `[set]`

Setter for camera StyleToningEffect property

string MagicMirrorHelper.HardwareHelper.Camera_TvCapture `[set]`

Setter for camera TvCapture property

string MagicMirrorHelper.HardwareHelper.Camera_TvLiveView `[set]`

Setter for camera TvLiveView property

string MagicMirrorHelper.HardwareHelper.Camera_WhiteBalance `[set]`

Setter for camera WhiteBalance property

string MagicMirrorHelper.HardwareHelper.Camera_WhiteBalanceShift `[set]`

Setter for camera WhiteBalanceShift property

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

- HardwareHelper.cs