



## **EyeMoCamera Manual**

EyeMoCamera Version 1.0.0  
Document Version 1.0.0

© 2006 GestureTek Inc.  
317 Adelaide Street West, Toronto, Ontario, M5V 1P9 Canada  
Web: <http://www.gesturetek.com> Email: [support@gesturetek.com](mailto:support@gesturetek.com)  
Phone: 416.340.9290 Fax: 416.348.9809

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Copyright and Legal Matters.....	1
1.2	Prerequisites.....	1
1.3	Device requirements.....	1
1.4	Overview.....	1
1.5	Sound .....	2
1.6	Suspend.....	2
1.7	Resolution Modes.....	2
1.8	Camera Callback.....	3
<b>2</b>	<b>Methods</b>	<b>5</b>
2.1	IEYEMOCAMERA_AddFrameListener.....	5
2.2	IEYEMOCAMERA_AddNewFrameListener.....	5
2.3	IEYEMOCAMERA_GetDeviceSettings.....	6
2.4	IEYEMOCAMERA_GetFlip.....	7
2.5	IEYEMOCAMERA_GetResolution.....	8
2.6	IEYEMOCAMERA_GetResolutionMode.....	9
2.7	IEYEMOCAMERA_Initialize.....	10
2.8	IEYEMOCAMERA_Pause.....	10
2.9	IEYEMOCAMERA_RemoveFrameListener.....	11
2.10	IEYEMOCAMERA_Resume.....	12
2.11	IEYEMOCAMERA_SetFlip.....	13
2.12	IEYEMOCAMERA_SetResolution.....	13
2.13	IEYEMOCAMERA_SetResolutionMode.....	14
2.14	IEYEMOCAMERA_Start.....	15
2.15	IEYEMOCAMERA_Stop.....	16

---

# 1 Introduction

The EyeMoCamera extension provides an easy-to-use BREW interface to access a mobile device's camera. It simplifies development by handling device-specific initialization and implementation complexities that may occur, such as devices that contain more than one camera, various camera orientations, devices supporting only a subset of common resolutions, and those devices which require unusual initialization steps.

## 1.1 Copyright and Legal Matters

GestureTek and EyeMobile are registered trademarks of GestureTek Inc.

BREW is a registered trademark of QUALCOMM Incorporated.

GestureTek technology and software is protected under United States patent number 5,534,917 (Video image based control system).

The contents of this documentation are the property of GestureTek Inc.

## 1.2 Prerequisites

Familiarity with BREW development in the C programming language.

## 1.3 Device requirements

- Device with BREW version 2.1.3 or above.
- Device camera should be available to BREW (device specifications are available from the BREW Developer extranet).
- Color or grey-scale displays are supported.
- Frame rate should be at least 15 frames per second.
- Minimum supported resolution is 80x60 pixels.
- For best results the camera image should represent the entire field-of-view of the camera, rather than a sub-window of the original view.
- A small amount of memory is required for EyeMoCamera to allocate its internal buffers.
- BREW Simulator (included with BREW SDK version 3 and above) is required to run camera-enabled BREW applications directly from Microsoft Visual Studio.
- Running EyeMoCamera-enabled applications on the BREW Simulator requires a webcam.

## 1.4 Overview

The EyeMoCamera extension allows the handset camera to be controlled in preview (or viewfinder) mode. The capture resolution can be controlled, either by selecting from among a small set of known working resolutions, or the application can request a specific resolution, if desired. The camera can be started, stopped, paused, or resumed as necessary. A listener interface is used to allow the application to receive notifications when a new frame arrives from the camera.

Special effort has been made when designing this API to provide the application developer access to the underlying ICamera interface being used by this extension. This will allow the developer to implement additional functionality to supplement the existing extension interface without significantly increasing the complexity of the API. The application can get a reference to the instance of ICamera used by EyeMoCamera by calling IEYEMOCAMERA\_QueryInterface, with a classID argument of AEECLSID\_CAMERA.

The ICamera interface is allocated during the call to IEYEMOCAMERA\_Start<sup>[15]</sup>, and is released during IEYEMOCAMERA\_Stop<sup>[16]</sup>, unless the application adds an additional reference to the interface. Whenever IEYEMOCAMERA\_Start is called, the EyeMoCamera extension ensures that the ICamera interface is in a known stable state before starting the camera in preview mode. Once IEYEMOCAMERA\_Start has been called, the EyeMoCamera assumes that it has a lock on the ICamera interface until IEYEMOCAMERA\_Stop is called. The application should not attempt to mutate the ICamera instance during this time (stopping the camera, changing the callback function, taking a snapshot, etc.), though ICamera accessor methods are acceptable (ICAMERA\_GetParm and others). When the lock is not in place, the application can freely use any of the ICamera functions.

The EyeMoCamera extension will attempt to work on untested devices. Workarounds of platform-specific bugs are based on the platformID supplied to the extension. The ID of the current device platform may be overridden to allow a new device to use an existing device's settings. All workaround parameters may be accessed individually as well, thus allowing the application developer to tweak the settings for a new device.

## 1.5 Sound

Current BREW applications cannot play sounds while the camera is running. This is due to a limitation of the BREW ICamera interface, not of the EyeMoCamera extension. To integrate sounds with your application, you can call IEYEMOCAMERA\_Pause<sup>[10]</sup> before playing a sound and call IEYEMOCAMERA\_Resume<sup>[12]</sup> afterwards. Alternatively you could play vibrations or frequency tones, which do not require the camera to be paused.

## 1.6 Suspend

When your BREW application is being suspended by the system, it is important to deallocate any resources that may be required by other applications. EyeMoCamera simplifies this process, by maintaining a reference internally to the BREW ICamera interface only during the time between calls to IEYEMOCAMERA\_Start<sup>[15]</sup> and IEYEMOCAMERA\_Stop<sup>[16]</sup>.

When your application receives a suspend event from the BREW platform, the application should call IEYEMOCAEMRA\_Stop to deallocate the internal ICamera interface. This applies regardless, even if the camera is currently in the paused state, since there may not be enough time for the camera to properly resume when processing the suspend event. Once the application receives a resume event from the BREW platform, the application may then call IEYEMOCAMERA\_Start. **Note:** If the camera was paused before it was suspended, in order to restore the paused state, the application would need to call IEYEMOCAMERA\_Pause<sup>[10]</sup> again.

## 1.7 Resolution Modes

Mobile device cameras do not have standard display resolutions. This creates a challenge for the developer, since there is no way to be certain that an application designed using a particular resolution will work on most devices. Attempting to capture using unsupported resolutions may have undesired results. EyeMoCamera seeks to resolve this by providing a set of resolution modes. By specifying a resolution mode, EyeMoCamera will attempt to use a resolution supported by the current device.

The resolution modes provided by the EyeMoCamera API are:

**EYEMO\_RESOLUTION\_CUSTOM** - The developer has set the resolution to a custom mode (using `IEYEMOCAMERA_SetResolution`), even if the resolution requested matches the returned resolution for the current device. EyeMoCamera cannot guarantee that the device properly handles this resolution mode. The possible results are...

- Device fails to initialize
- Returns closest resolution smaller than requested and is padded with black pixels
- Returns closest resolution larger than requested and is cropped to requested resolution
- Stretches closest resolution to match requested
- Device functions properly

**EYEMO\_RESOLUTION\_LOW** - The smallest resolution supported by the device camera without sacrificing field-of-view or image quality. If the camera image is visible on screen, or the size of the onscreen image is limited, use this resolution to decrease the amount of work that any EyeMobile trackers will have to do. This resolution is typically 80 x 60.

**EYEMO\_RESOLUTION\_MED** - A moderate resolution, typically 160 x 120. This resolution is supported by most (but not all) devices, thus avoiding the greatest amount of device-specific behavior.

**EYEMO\_RESOLUTION\_HIGH** - A resolution that is closest to the full screen resolution of the device. This is the highest practical resolution available to display the entire camera image on screen. Not all devices support camera resolutions close to the full size of the display, so there is wide variation in this mode from device to device. For example, on most devices with display widths of 176 pixels, the resolution EyeMoCamera provides is 176 x 176.

## 1.8 Camera Callback

EyeMoCamera sends camera frames to the application through the use of a callback function. The callback function is invoked every time a new frame arrives from the camera. This allows the application developer to take an arbitrary action in response to a camera frame, allowing for the flexibility necessary to implement games.

The `IEYEMOCAMERA_AddNewFrameListener` function is used to register the callback function with EyeMoCamera. This function encapsulates the callback inside of an `IFrameListener` function for ease of use. Any number of callback functions may be registered in this way, and independently added or removed as necessary.

The camera callback function has the following signature:

```
void
CAM_CALLBACK_F(
    IBitmap * pFrame,
    CAM_STATE_E eState,
    int nElapsedTime,
    void * pvUser
);
```

The camera state can be any of the following:

**CAM\_STATE\_NONE**

The camera has not yet been initialized.

**CAM\_STATE\_STARTED**

The camera has been started but has not yet returned any frames.

**CAM\_STATE\_FIRST\_FRAME**

The camera has returned the first frame. This allows the application to easily defer some operations, such as initializing the extension, until the first frame has arrived.

**CAM\_STATE\_FRAME**

The camera is returning frames normally.

**CAM\_STATE\_STOPPED**

The camera has been stopped by the application.

**CAM\_STATE\_FAIL**

The camera has experienced an error of some kind.

**CAM\_STATE\_INFO**

Reserved for future use. The camera is requesting more information from the application. There is currently no need to handle this state.

Here the `pvUser` parameter is used to pass application-specific data into the callback. For example, a pointer to the structure containing the application state can be passed, so that the callback function can query or modify the state.

## 2 Methods

### 2.1 IEYEMOCAMERA\_AddFrameListener

Adds an IFrameListener object to the camera. IFrameListener objects that have been added to the camera will have their IFRAMELISTENER\_ConsumeFrame methods invoked whenever a new frame arrives from the camera. This is the primary method used by this extension for returning camera information to the application.

If more than one IFrameListener is added to the camera extension, then the callbacks are invoked in the order that they were added. If the same IFrameListener is added multiple times to the camera extension, then the instance gets as many notification per frame as the number of times it was added.

#### Definition

```
int
IEYEMOCAMERA_AddFrameListener(
    IEyeMoCamera * po,
    IFrameListener * pFL
);
```

#### Parameters

po

Pointer to an instance of the extension interface object.

pFL

Pointer to an IFrameListener interface.

#### Returns

EYEMO\_OKAY

The frame listener has been added successfully.

EYEMO\_ERROR\_BAD\_PARAMETER

One of the input pointers is NULL.

EYEMO\_ERROR\_NOT\_INITIALIZED

The extension has not yet been initialized.

EYEMO\_ERROR

An error occurred while adding the frame listener.

#### Notes

The camera extension maintains a reference to the frame listener interface until IEYEMOCAMERA\_RemoveFrameListener<sup>[11]</sup> is called, or the extension itself is released. If the application has a reference to the frame listener, it must be released separately.

### 2.2 IEYEMOCAMERA\_AddNewFrameListener

Creates a new instance of IFrameListener, and adds it to the camera. The new IFrameListener will invoke the user-supplied callback function with the given data as an argument, whenever the IFRAMELISTENER\_ConsumeFrame method is called.

The new IFrameListener is returned to the user through the ppFL argument. If the argument is NULL, then the new IFrameListener is not returned. Otherwise, the new IFrameListener has a positive reference count, and will need to be released by the application.

This function shares semantics with the `IEYEMOCAMERA_AddFrameListener`. In particular, the order in which `IFrameListener` objects are notified by the camera is dependent on the order in which they were added, and not at all on the method used to add them.

**Definition**

```
int
IEYEMOCAMERA_AddNewFrameListener(
    IEyeMoCamera * po,
    CAM_CALLBACK_F * pfCallback,
    void * pvData,
    IFrameListener ** ppFL
);
```

**Parameters**

`po`

Pointer to an instance of the extension interface object.

`callback`

The callback function to be invoked when a new frame arrives from the camera.

`pvData`

User data that will be passed to the callback function. This parameter can be used to pass a pointer to your application's main structure so that the callback can change the application's state.

`ppFL` (*Output*)

The location in which to store the new `IFrameListener` instance. Can be NULL.

**Returns**

`EYEMO_OKAY`

The frame listener has been added successfully.

`EYEMO_ERROR_BAD_PARAMETER`

One of the input pointers is NULL.

`EYEMO_ERROR_NOT_INITIALIZED`

The extension has not yet been initialized.

`EYEMO_ERROR`

An error occurred while adding the frame listener.

**Notes**

The camera extension maintains a reference to the frame listener interface until `IEYEMOCAMERA_RemoveFrameListener`[\(11\)](#) is called, or the extension itself is released. If the application has a reference to the frame listener, it must be released separately.

## 2.3 IEYEMOCAMERA\_GetDeviceSettings

Gets the current values of the settings that work around device-specific camera issues.

This call provides access to all of the parameters used by the camera extension to differentiate between devices. An advanced user can use this data to implement extra functionality, such as working at a specific resolution, at the expense of platform independence.

This is the only EyeMoCamera method that can be called before `IEYEMOCAMERA_Initialize` [10]. The interface pointer returned by this call can be passed back to the camera as the second parameter in `IEYEMOCAMERA_Initialize` [10].

**Definition**

```
int
IEYEMOCAMERA_GetDeviceSettings(
    IEyeMoCamera * po,
    IDeviceSettings ** ppDS
);
```

**Parameters**

`po`

Pointer to an instance of the extension interface object.

`ppDS` (*Output*)

A pointer to the memory location in which to store the device settings pointer. This pointer cannot be NULL.

**Returns**

`EYEMO_OKAY`

The device settings structure has been successfully returned.

`EYEMO_ERROR_BAD_PARAMETER`

One of the input pointers is NULL.

`EYEMO_ERROR_NO_MEMORY`

There was an error allocating memory for the structure.

`EYEMO_ERROR`

An unexpected error has occurred.

**Notes**

This function increments the reference count of the device settings interface. The application must release the interface when it is no longer needed.

## 2.4 IEYEMOCAMERA\_GetFlip

Gets the value of the current image orientation.

The device camera is not in the same orientation on all platforms. The extension is therefore responsible for ensuring that the incoming image stream has the orientation that the user expects.

**Definition**

```
int
IEYEMOCAMERA_GetFlip(
    IEyeMoCamera * po,
    int * pnFlip
);
```

**Parameters**

`po`

Pointer to an instance of the extension interface object.

`pnFlip` (*Output*)

The location in which to store the current value of the flip direction. This pointer cannot be NULL.

**Returns**

EYEMO\_OKAY

The flip direction has been successfully returned.

EYEMO\_ERROR\_BAD\_PARAMETER

One of the input pointers is NULL.

EYEMO\_ERROR\_NOT\_INITIALIZED

The extension has not yet been initialized.

EYEMO\_ERROR

An unexpected error has occurred.

**Notes**

The value of the pnFlip parameter will be one of the following:

- EYEMO\_FLIP\_NONE
- EYEMO\_FLIP\_VERTICAL

## 2.5 IEYEMOCAMERA\_GetResolution

Gets the value of the current capture resolution.

The resolution is set during the last call to IEYEMOCAMERA\_SetResolution<sup>[13]</sup>. If that method has not yet been called, then a default value is used corresponding to the currently selected platformID.

For more information regarding the handling of display resolution, see Resolution Modes<sup>[2]</sup>.

**Definition**

```
int
IEYEMOCAMERA_GetResolution(
    IEyeMoCamera * po,
    int * pnWidth,
    int * pnHeight
);
```

**Parameters**

po

Pointer to an instance of the extension interface object.

pnWidth (*Output*)

The location in which to store the current value of the frame width. This pointer cannot be NULL.

pnHeight (*Output*)

The location in which to store the current value of the frame height. This pointer cannot be NULL.

**Returns**

EYEMO\_OKAY

The resolution was successfully returned.

EYEMO\_ERROR\_BAD\_PARAMETER

One of the input pointers is NULL.

EYEMO\_ERROR\_NOT\_INITIALIZED

The extension has not yet been initialized.

EYEMO\_ERROR

An unexpected error has occurred.

#### Notes

None

## 2.6 IEYEMOCAMERA\_GetResolutionMode

Gets the value of the current capture resolution mode.

The resolution is set during the last call to `IEYEMOCAMERA_SetResolutionMode`<sup>[14]</sup>. If that method has not yet been called, then a default value is used corresponding to the currently selected platformID. Alternatively, if the `IEYEMOCAMERA_SetResolution`<sup>[13]</sup> is called, this method will write `IEYEMO_RESOLUTION_CUSTOM` into `peMode`.

For more information regarding resolution modes, see [Resolution Modes](#)<sup>[2]</sup>.

#### Definition

```
int
IEYEMOCAMERA_GetResolutionMode(
    IEyeMoCamera * po,
    int * peMode
);
```

#### Parameters

`po`

Pointer to an instance of the extension interface object.

`peMode` (*Output*)

The location in which to store the current value of the resolution mode. This pointer cannot be NULL.

#### Returns

EYEMO\_OKAY

The resolution was successfully returned.

EYEMO\_ERROR\_BAD\_PARAMETER

One of the input pointers is NULL.

EYEMO\_ERROR\_NOT\_INITIALIZED

The extension has not yet been initialized.

EYEMO\_ERROR

An unexpected error has occurred.

#### Notes

None

## 2.7 IEYEMOCAMERA\_Initialize

Initializes the instance of the camera extension. This method allocates necessary buffers and resources for use by the extension, which are released only once the interface has been completely released.

This function must be the first function called on any instance of the extension interface pointer, with the exception of `IEYEMOCAMERA_GetDeviceSettings`. If any other method is called before `IEYEMOCAMERA_Initialize`, it will return `IEYEMO_ERROR_NOT_INITIALIZED`.

The second parameter allows the developer to change the default camera behavior on the current platform. The settings for the current platform are available through the `IEYEMOCAMERA_GetDeviceSettings` method, but the settings cannot be changed once the camera has been initialized. **Note:** This usage is for advanced users only. In most cases, passing `NULL` as the second argument will result in the desired behavior.

### Definition

```
int  
IEYEMOCAMERA_Initialize(  
    IEyeMoCamera * po,  
    IDeviceSettings * psSettings  
);
```

### Parameters

`po`

Pointer to an instance of the extension interface object. This object must be allocated separately using `ISHELL_CreateInstance` before calling this function. Use the constant `AEECLSID_EYEMOCAMERA` as the `CLSID` value for the instance.

`psSettings`

A pointer to the settings structure to be used by the camera. If `NULL` is passed, the extension will use the platform defaults.

### Returns

`EYEMO_OKAY`

No problems were encountered during initialization.

`EYEMO_ERROR_BAD_PARAMETER`

One of the input pointers is `NULL`.

`EYEMO_ERROR_NO_MEMORY`

Insufficient memory to allocate the necessary resources.

`EYEMO_ERROR`

One of the components could not be initialized.

`EYEMO_WARNING_ALREADY`

The extension was already initialized.

### Notes

None

## 2.8 IEYEMOCAMERA\_Pause

Temporarily halts camera operation, preventing the flow of new frames to the registered listeners.

To resume normal operation, call `IEYEMOCAMERA_Resume` <sup>12</sup>.

**Definition**

```
int  
IEYEMOCAMERA_Pause(  
    IEyeMoCamera * po  
);
```

**Parameters**

`po`  
Pointer to an instance of the extension interface object.

**Returns**

`EYEMO_OKAY`

The camera is successfully paused.

`EYEMO_WARNING_ALREADY`

The camera is already in the paused state.

`EYEMO_ERROR_BAD_PARAMETER`

One of the input pointers is NULL.

`EYEMO_ERROR_NOT_INITIALIZED`

The camera has not yet been initialized.

`EYEMO_ERROR_BAD_STATE`

The camera is not currently active.

`EYEMO_ERROR`

An error occurred while attempting to pause the camera.

**Notes**

None

## 2.9 IEYEMOCAMERA\_RemoveFrameListener

Removes the frame listener.

**Definition**

```
int  
IEYEMOCAMERA_RemoveFrameListener(  
    IEyeMoCamera * po,  
    IFrameListener * pListener  
);
```

**Parameters**

`po`  
Pointer to an instance of the extension interface object.

`pListener`  
Pointer to the frame listener that needs to be removed..

**Returns**

EYEMO\_OKAY

The frame listener was removed successfully.

EYEMO\_ERROR\_BAD\_PARAMETER

One of the input pointers is NULL.

EYEMO\_ERROR\_NOT\_INITIALIZED

The extension has not yet been initialized.

EYEMO\_ERROR

An error occurred while removing the frame listener.

**Notes**See IEYEMOCAMERA\_AddFrameListener<sup>[5]</sup>

## 2.10 IEYEMOCAMERA\_Resume

Resumes camera operation, thus sending frames again to the list of frame callbacks.

This function restores the camera to normal operation after a call to IEYEMOCAMERA\_Pause<sup>[10]</sup>.**Definition**

```
int
IEYEMOCAMERA_Resume(
    IEyeMoCamera * po
);
```

**Parameters**

po

Pointer to an instance of the extension interface object.

**Returns**

EYEMO\_OKAY

The camera has successfully resumed normal operation.

EYEMO\_WARNING\_ALREADY

Camera was already running. It was not in a paused state.

EYEMO\_ERROR\_BAD\_PARAMETER

One of the input pointers is NULL.

EYEMO\_ERROR\_NOT\_INITIALIZED

The extension has not yet been initialized.

EYEMO\_ERROR\_BAD\_STATE

The camera has not yet been started.

**Notes**

On some platforms, pausing the camera operation for too short a period before resuming can cause the camera to be in an inconsistent state. On these platforms, the extension enforces a minimum pause duration. If resume is called before the duration expires, then a timer will be created to resume camera operation only after the duration is completed.

## 2.11 IEYEMOCAMERA\_SetFlip

Sets the current flip direction value. This new value will override any previous calls to this method, and will remain active until this method is called again with a new value.

**Definition**

```
int
IEYEMOCAMERA_SetFlip(
    IEyeMoCamera * po,
    int nFlip
);
```

**Parameters**

po

Pointer to an instance of the extension interface object.

nFlip

The new value for the flip direction.

**Returns**

EYEMO\_OKAY

The value was successfully set.

EYEMO\_ERROR\_BAD\_PARAMETER

One of the input pointers is NULL.

EYEMO\_ERROR\_NOT\_INITIALIZED

The extension has not yet been initialized.

EYEMO\_ERROR

An unexpected error has occurred.

**Notes**

This parameter helps to work around the fact that device cameras do not have the same orientation on all platforms.

The value of the nFlip parameter will be one of the following:

- EYEMO\_FLIP\_NONE
- EYEMO\_FLIP\_VERTICAL

## 2.12 IEYEMOCAMERA\_SetResolution

Sets the capture resolution value. This new value will override any previous calls to this method, and will remain active until either this method is called again with a new value, or the method IEYEMOCAMERA\_SetResolutionMode<sup>[14]</sup> is called.

**Definition**

```
int  
IEYEMOCAMERA_SetResolution(  
    IEyeMoCamera * po,  
    int nHeight,  
    int nWidth  
);
```

**Parameters**

po  
 Pointer to an instance of the extension interface object.

nHeight  
 The new value for the height.

nWidth  
 The new value for the width.

**Returns**

EYEMO\_OKAY  
 The values were successfully set.

EYEMO\_ERROR\_BAD\_PARAMETER  
 One of the input pointers is NULL.

EYEMO\_ERROR\_NOT\_INITIALIZED  
 The extension has not yet been initialized.

EYEMO\_ERROR  
 An unexpected error has occurred.

**Notes**

By calling this function, the resolution mode is automatically set to EYEMO\_RESOLUTION\_CUSTOM. For more information regarding the handling of display resolution, see Resolution Modes [\[ 2 \]](#).

## 2.13 IEYEMOCAMERA\_SetResolutionMode

Sets the capture resolution mode value. This new value will override any previous calls to this method, and will remain active until either this method is called again with a new value, or the method IEYEMOCAMERA\_SetResolution [\[ 13 \]](#) is called.

**Definition**

```
int  
IEYEMOCAMERA_SetResolutionMode(  
    IEyeMoCamera * po,  
    int eMode  
);
```

**Parameters**

po  
 Pointer to an instance of the extension interface object.

eMode  
 The new value for the resolution mode.

**Returns**

EYEMO\_OKAY

The values were successfully set.

EYEMO\_ERROR\_BAD\_PARAMETER

One of the input pointers is NULL.

EYEMO\_ERROR\_NOT\_INITIALIZED

The extension has not yet been initialized.

EYEMO\_ERROR

An unexpected error has occurred.

**Notes**For more information regarding the handling of display resolution, see Resolution Modes [\[2\]](#).

## 2.14 IEYEMOCAMERA\_Start

Starts the camera and begins passing frames to the registered IFrameListeners. The camera remains active until IEYEMOCAMERA\_Stop [\[16\]](#) is called.

If you need to temporarily halt camera operation, use IEYEMOCAMERA\_Pause [\[10\]](#).

**Definition**

```
int
IEYEMOCAMERA_Start(
    IEyeMoCamera * po
);
```

**Parameters**

po

Pointer to an instance of the extension interface object.

**Returns**

EYEMO\_OKAY

The camera was successfully started.

EYEMO\_WARNING\_ALREADY

The camera has already been started.

EYEMO\_ERROR\_BAD\_PARAMETER

One of the input pointers is NULL.

EYEMO\_ERROR\_NOT\_INITIALIZED

The extension has not yet been initialized.

EYEMO\_ERROR

An unexpected error has occurred.

**Notes**

The camera attempts to capture frames at the resolution specified during the latest call to IEYEMOCAMERA\_SetResolution [\[13\]](#). If no such call has been made, the extension will select a resolution based on the default value given by the current platformID. The capture resolution

can be queried with `IEYEMOCAMERA_GetResolution`<sup>[8]</sup>, or by examining the dimensions of the `IBitmap` returned by the registered callback functions.

On some platforms, the camera may not be started immediately after this call returns. This is due to a bug in the `ICamera` implementation on those platforms. As for these devices, starting the camera will be delayed until it will be successful, but there is no need to call this method again. As far as the application developer is concerned, there will be a slightly increased delay before frames are sent to the callback functions, when the camera starts the first time.

The first frame returned has the special state of `CAM_STATE_FIRST_FRAME`, which is useful for deferring initialization until the actual dimensions of a captured frame are known.

This method should be called when your application is being resumed after having been suspended by the BREW platform.

## 2.15 IEYEMOCAMERA\_Stop

Stops camera operation. A final notification will be sent to the list of frame callbacks indicating that the camera has stopped.

The camera may either be in the running state (after `IEYEMOCAMERA_Start`<sup>[15]</sup>), or in the paused state (after `IEYEMOCAMERA_Pause`<sup>[10]</sup>) when this method is called.

### Definition

```
int
IEYEMOCAMERA_Stop(
    IEyeMoCamera * po
);
```

### Parameters

`po`

Pointer to an instance of the extension interface object.

### Returns

`EYEMO_OKAY`

The camera has successfully been stopped.

`EYEMO_WARNING_ALREADY`

The camera has already been stopped and is not currently running.

`EYEMO_ERROR_BAD_PARAMETER`

One of the input pointers is NULL.

`EYEMO_ERROR_NOT_INITIALIZED`

The extension has not yet been initialized.

`EYEMO_ERROR`

An unexpected error has occurred.

### Notes

On some platforms, running for too short a period may cause the camera to be in an inconsistent state. On these platforms, the extension enforces a minimum run duration. If stop is called before the duration expires, then a timer will be created to stop the camera only after the duration is completed.

This function should be called when your application is being suspended by the BREW platform.