



# **ActiveGigE SDK**

User Guide

Version 6

**Copyright © 2007-2017 by A&B Software LLC**

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form by any means, electronic, mechanical, by photocopying, recording , or otherwise, without the prior written permission of A&B Software, LLC

Information furnished by A&B Software LLC is believed to be accurate and reliable; however, no responsibility is assumed by A&B Software LLC for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent rights of A&B Software LLC.

Use, duplication, or disclosure by the United States Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer software clause at 48 C.F.R. 252.227-7013, or in subparagraph (c)(2) of the Commercial Computer Software - Registered Rights clause at 48 C.F.R. 52-227-19 as applicable.

ActiveGige is a trademark of A&B Software LLC.

All other brand and product names are trademarks or registered trademarks of their respective companies.

**Sixth Edition**

*October 2017*

*A&B Software LLC  
New London, CT 06320  
USA*

[www.ab-soft.com](http://www.ab-soft.com)  
[support@ab-soft.com](mailto:support@ab-soft.com)

# Table of Contents

<b>Part I Introduction</b>	<b>8</b>
1 License Agreement .....	10
2 System Requirements .....	12
3 Installation .....	13
4 Filter Driver .....	16
5 Registration .....	20
6 Network Setup .....	21
7 IP configuration utility .....	25
8 GcamViewer .....	30
<b>Part II Getting started</b>	<b>34</b>
1 Visual Basic .....	35
2 Visual C++ .....	37
3 VB.NET .....	41
4 Visual C# .....	44
5 Matlab .....	48
6 Python .....	52
7 Using ActiveGigE API at runtime .....	53
8 Working with multiple cameras .....	57
9 Distributing your application .....	59
<b>Part III ActiveX Reference</b>	<b>59</b>
1 Properties .....	60
Acquire .....	66
AcquisitionFrameCount .....	68
AcquisitionFrameRate .....	70
AcquisitionFrameRateAbs .....	71
AcquisitionFrameRateRaw .....	72
AcquisitionMode .....	73
Affinity .....	75
Alpha .....	76
AntiTearing .....	78
BackColor .....	80
BalanceRatio .....	81
BalanceRatioAbs .....	83
BalanceRatioRaw .....	85
BalanceRatioSelector .....	87
BalanceWhiteAuto .....	89
Bayer .....	91
BinningX .....	93

BinningY .....	94
BkgCorrect .....	95
BkgName .....	97
BlackLevel .....	98
BlackLevelAbs .....	100
BlackLevelAuto .....	102
BlackLevelRaw .....	104
BlackLevelSelector .....	106
Camera .....	108
ColorCorrect .....	110
DecimationX .....	111
DecimationY .....	112
Display .....	113
Edge .....	115
ExposureAuto .....	116
ExposureMode .....	118
ExposureTime .....	120
ExposureTimeAbs .....	122
ExposureTimeRaw .....	124
Flip .....	126
Font .....	128
Format .....	129
Gain .....	132
GainAbs .....	134
GainAuto .....	136
GainRaw .....	138
GainSelector .....	140
Gamma .....	142
Heartbeat .....	144
HotPixelCorrect .....	146
HotPixelLevel .....	147
Integrate .....	148
IntegrateWnd .....	149
LensCorrect .....	150
LineFormat .....	152
LineInverter .....	154
LineMode .....	156
LineSelector .....	158
LineSource .....	160
LUTMode .....	162
Magnification .....	163
MonitorSync .....	165
Multicast .....	167
Overlay .....	169
OverlayColor .....	170
OverlayFont .....	171
PacketSize .....	172
Palette .....	174
Privilege .....	176
Rotate .....	178
ScrollBars .....	180
ScrollX .....	182
ScrollY .....	183
SizeX .....	184

---

SizeY .....	185
OffsetX .....	187
OffsetY .....	188
Timeout .....	189
TestImageSelector .....	190
Trigger .....	192
TriggerActivation .....	194
TriggerDelay .....	196
TriggerDelayAbs .....	198
TriggerDelayRaw .....	200
TriggerSelector .....	202
TriggerSource .....	204
UserOutputSelector .....	206
UserOutputValue .....	208
UserSetSelector .....	209
WebStream .....	211
<b>2 Methods .....</b>	<b>213</b>
CloseVideo .....	225
CreateSequence .....	226
CreateVideo .....	228
Draw .....	230
DrawAlphaClear .....	231
DrawAlphaEllipse .....	232
DrawAlphaLine .....	234
DrawAlphaPixel .....	236
DrawAlphaRectangle .....	237
DrawAlphaText .....	239
DrawEllipse .....	241
DrawLine .....	243
DrawPixel .....	244
DrawRectangle .....	245
DrawText .....	247
GetActionAcknowledgeInfo .....	248
GetAcquisitionFrameRateMax .....	250
GetAcquisitionFrameRateMin .....	251
GetAudioLevel .....	252
GetAudioList .....	253
GetAudioSource .....	254
GetBalanceRatioMax .....	255
GetBalanceRatioMin .....	256
GetBarcode .....	257
GetBitsPerChannel .....	259
GetBlackLevelMax .....	261
GetBlackLevelMin .....	262
GetBlockId .....	263
GetBytesPerPixel .....	264
GetCameraIP .....	265
GetCameraIPList .....	266
GetCameraList .....	268
GetCameraMAC .....	270
GetCameraMACList .....	271
GetChunkPointer .....	273
GetChunkSize .....	275
GetCodec .....	277

GetCodecList .....	278
GetCodecProperties .....	279
GetComponentData .....	280
GetComponentLine .....	282
GetDIB .....	284
GetEnumList .....	286
GetExposureTimeMax .....	288
GetExposureTimeMin .....	289
GetFeature .....	290
GetFeature64 .....	292
GetFeatureAccess .....	294
GetFeatureArray .....	296
GetFeatureDependents .....	298
GetFeatureDescription .....	299
GetFeatureList .....	300
GetFeatureIncrement .....	302
GetFeatureMin .....	304
GetFeatureMax .....	306
GetFeatureRepresentation .....	308
GetFeatureString .....	309
GetFeatureTip .....	311
GetFeatureType .....	312
GetFeatureVisibility .....	314
GetFileAccessMode .....	315
GetFileList .....	316
GetFileSize .....	318
GetFileTransferProgress .....	320
GetFormatList .....	322
GetFPSAcquired .....	324
GetFPS .....	325
GetGainMax .....	326
GetGainMin .....	327
GetHeight .....	328
GetHeightMax .....	330
GetHistogram .....	331
GetImageData .....	333
GetImageLine .....	335
GetImagePointer .....	337
GetImageStat .....	339
GetImageWindow .....	341
GetLevels .....	343
GetLUT .....	344
GetOptimalPacketSize .....	345
GetPicture .....	347
GetPixel .....	349
GetRawData .....	351
GetRGBPixel .....	353
GetROI .....	355
GetSequenceFrameCount .....	356
GetSequencePicture .....	357
GetSequencePixel .....	358
GetSequencePointer .....	360
GetSequenceRawData .....	362
GetSequenceTimestamp .....	364

---

GetSequenceWindow .....	365
GetTimestamp .....	367
GetTriggerDelayMax .....	368
GetTriggerDelayMin .....	369
GetVideoFPS .....	370
GetVideoFrameCount .....	371
GetVideoPosition .....	372
GetVideoVolume .....	373
GetWidthMax .....	374
Grab .....	375
IsFeatureAvailable .....	376
IsMasterApplication .....	377
LoadImage .....	378
LoadSettings .....	379
LoadSequence .....	381
OpenVideo .....	382
OverlayClear .....	384
OverlayEllipse .....	385
OverlayLine .....	386
OverlayPixel .....	387
OverlayRectangle .....	388
OverlayText .....	389
PlayVideo .....	390
ReadBlock .....	392
ReadFile .....	394
ReadRegister .....	396
SaveBkg .....	397
SaveImage .....	399
SaveSettings .....	401
SaveSequence .....	403
SendActionCommand .....	405
SetActionConditions .....	408
SetAudioLevel .....	410
SetAudioSource .....	411
SetCodec .....	413
SetCodecProperties .....	415
SetColorMatrix .....	417
SetControlKey .....	419
SetDestinationIP .....	421
SetDeviceKey .....	423
SetImageWindow .....	425
SetFeature .....	427
SetFeature64 .....	429
SetFeatureArray .....	431
SetFeatureString .....	433
SetGains .....	435
SetLensDistortion .....	437
SetLevels .....	439
SetLUT .....	442
SetROI .....	444
SetStreamChannel .....	446
SetVideoFPS .....	448
SetVideoPosition .....	449
SetVideoSync .....	451

SetVideoVolume .....	453
SetWebStreamer .....	454
ShowAudioDlg .....	456
ShowCodecDlg .....	457
ShowCompressionDlg .....	458
ShowProperties .....	460
SoftTrigger .....	462
StartCapture .....	463
StopCapture .....	465
StartSequenceCapture .....	466
StopSequenceCapture .....	468
StartVideoCapture .....	469
StopVideoCapture .....	471
StopVideo .....	472
TriggerVideo .....	473
WriteBlock .....	474
WriteFile .....	476
WriteRegister .....	478
<b>3 Events .....</b>	<b>479</b>
CameraPlugged .....	481
CameraUnplugged .....	482
CaptureCompleted .....	483
EventMessage .....	484
EventDataMessage .....	486
FormatChanged .....	488
FrameAcquired .....	489
FrameAcquiredX .....	490
FrameDropped .....	492
FrameLoaded .....	493
FrameRecorded .....	494
FrameReady .....	495
MouseDown .....	498
MouseDownRight .....	499
MouseMove .....	501
MouseUp .....	502
MouseUpRight .....	503
PlayCompleted .....	504
RawFrameAcquired .....	505
Scroll .....	506
Timeout .....	507
<b>4 Property Pages .....</b>	<b>508</b>
Source .....	509
Format .....	511
Analog .....	513
Inp/Out .....	515
GenICam .....	517
Display .....	519
<b>Part IV DirectShow .....</b>	<b>522</b>
<b>1 Quick Reference Guide .....</b>	<b>523</b>
FilterConfig utility .....	525
Retrieving the Filter .....	526



---

Building the Graph .....	528
Displaying the Preview .....	529
Capturing to AVI .....	530
Getting the Image Data .....	531
Displaying Property Pages .....	533
<b>2 Interfaces .....</b>	<b>534</b>
IAMCameraControl .....	535
IAMVideoProcAmp .....	537
IAMVideoControl .....	539
IActiveGige .....	541
IAMStreamConfig .....	542
IAMVideoCompression .....	544
IAMDroppedFrames .....	546
ISpecifyPropertyPages .....	547
<b>Part V TWAIN .....</b>	<b>548</b>
<b>Part VI Samples .....</b>	<b>550</b>
<b>Part VII Camera list .....</b>	<b>554</b>
<b>Part VIII Troubleshooting .....</b>	<b>555</b>
<b>Index .....</b>	<b>559</b>

---

# 1 Introduction

*ActiveGigE* is an SDK and ActiveX control designed for rapid application development tools, such as Visual Basic, VB.NET, Visual C++, C#, Java, Delphi, Python, Matlab, etc. Provided is GigeViewer application allowing customers to operate multiple cameras and save images in a number of formats. Also included are TWAIN and DirectShow drivers for interfacing to third-party imaging and video capture software. With ActiveGigE your application immediately supports GigE Vision™ 1.x and 2.x compliant cameras.

In general, with *ActiveGigE* you can:

- Create 32-bit applications for the 32- and 64-bit Windows, as well as native 64-bit applications for the 64-bit Windows.
  - Acquire and display live video from one or several GigE Vision™ cameras.
  - Stream video from a single camera to several computers and applications using the Multicast mode.
  - Receive video from a dual-port camera through a LAG (link aggregation) connection.
  - Set a desired video format and triggering mode.
  - Select among several hardware and software trigger sources.
  - Grab 8-, 10-, 12- and 16-bit monochrome images, or 24-, 30-, 32-, 36- and 48-bit color images.
  - Perform automatic color interpolation of a raw video generated by Bayer cameras.
  - Perform real-time decoding of JPEG and H.264 video streams generated by GEV 2.x cameras.
  - Perform real-time demodulation of images generated by double-rate and quad-rate cameras.
  - Select the desired size and position of the scan area.
  - Flip and rotate the live image.
  - Adjust multiple camera features in real time.
  - Activate automatic or one-push control over selected camera features, such as exposure and white balance.
  - Save device settings into a file and reload them on demand.
  - Control non-standard camera features through direct access to camera registers.
  - Receive message events from cameras in real time.
  - Broadcast an action command to multiple devices.
  - Transfer data to and from files hosted on the camera.
  - Choose among several palettes for pseudo-color display.
  - Get an instant access to pixel values and pixel arrays.
  - Retrieve individual color planes from RGB images.
  - Retrieve chunk data appended to each image.
  - Import live video to a PictureBox object.
  - Perform image processing on captured frames and display processed video in real-time.
  - Perform real-time histogram and statistical analysis over a selected color component.
  - Implement real-time background correction over the dark and bright fields.
  - Automatically identify hot-pixels and eliminate them from incoming images.
  - Perform the running average and integration of incoming video frames.
  - Correct barrel and pincushion lens distortion in real time.
  - Apply custom LUTs (lookup tables) to incoming video frames.
  - Apply 3x3 color correction matrix to incoming video frames.
  - Perform manual and automatic Window/Level processing (brightness, contrast, white balance).
  - Save images in RAW, BMP, TIF, DPX and JPEG formats with adjustable compression.
  - Load and display images in BMP, TIF and JPEG formats.
  - Decode 1D and 2D barcodes (UPC-A, UPC-E, EAN-8, Code 128, Code 39, Interleaved 2/5, QR Code, DataMatrix, PDF417).
  - Perform time-lapse capture to an AVI file or series of sequentially-named images.
  - Select a video compression codec for the AVI recording and adjust its settings.
-

- Scroll and zoom the live video with the full screen option.
- Overlay custom graphic and texts on the live video.
- Draw multi-colored graphics and texts with an adjustable transparency over the live video.
- Synchronize video rendering with the monitor refresh rate to eliminate the tearing artifact.
- Interface to third-party imaging applications using the included TWAIN driver.
- Interface to DirectShow-based applications via the included Video Capture Source filter.

With the extended *ActiveGige DVR* version you can:

- Record multiple AVI files with the sound.
- Reserve a space for AVI files to eliminate dropped frames.
- Play back AVI files with an adjustable speed, direction and frame interval.
- Browse through the frames in an AVI file with the full access to recorded pixel values.
- Use a proprietary raw uncompressed codec to record and play back raw Bayer video with no quality degradation.
- Control the recording and play-back volume.
- Record the incoming video into a memory sequence.
- Perform a loop recording.
- Play back the recorded memory sequence with an adjustable speed, direction and frame interval.
- Get an instant access to pixel values and timestamp of each frame in the memory sequence.
- Stream incoming video to your network in H.264 RTSP format and view it on a remote desktop or mobile device.

*ActiveGige* uses multiple threads to support video acquisition, therefore it does not require separate components for thread management.

This document gives a detailed description of *ActiveGige*, its properties and methods; it also explains how to use the *ActiveGige* to perform the most common tasks.

[License agreement](#)

[System requirements](#)

[Installation](#)

[Registration](#)

[Distributing your application](#)

## 1.1 License Agreement

This legal document is an agreement between you, the Licensee, and A&B Software LLC. By installing *ActiveGigE SDK* on your computer, you are agreeing to be bound by the terms of this agreement. If you do not agree to the terms of this agreement, promptly return the unopened package, together with all the other material which comprises the product, respectively delete all *ActiveGigE SDK* related files.

### 1. Subject of agreement

The subject of this agreement is the software *ActiveGigE*, the operating manuals, and all other accompanying material. It will be referred to henceforth as *ActiveGigE SDK*.

### 2. Grant of license

A&B Software LLC grants the Licensee a non-exclusive, non-transferable, personal and worldwide license to use one copy of *ActiveGigE SDK* in the development of an end-user application, as described in section 3 (below). This license is for a single developer/one computer and not for an entire company. If additional programmers wish to use *ActiveGigE SDK*, additional copies must be licensed.

### 3. End user application

An *end user application* is a specific application program that is licensed to a person or firm for business or personal use. The files which are not listed under section 5 must not be included with the end user application. Furthermore, the end user must not be in a position to be able to neither modify the program, nor to create *ActiveGigE SDK* based programs. Likewise, the end user must not be given the *ActiveGigE SDK* serial number.

### 4. Royalties

The *ActiveGigE SDK* is NOT royalty free. The cost and licensing issue breaks down into the cost of the development environment and the cost for each run-time license that must be shipped with any product that embeds *ActiveGigE SDK*.

### 5. Redistributable files

The redistributable components of *ActiveGigE SDK* are those files specifically designated as being distributable in the [distributing your application](#) section of ActiveGigE User's Guide.

### 6. Trial version

A&B Software LLC grants the Licensee a non-exclusive license to test *ActiveGigE SDK* for 21 days on one computer system using the Trial version. The Trial version must be used solely for the trial and evaluation of the Software. The Licensee is not permitted to use the Trial version for any other purpose, including without limitation any use of the Software for productive purposes, hardware testing or in the operation of any Licensee's business.

At the end of the evaluation period the Licensee shall promptly remove all coding and other vestiges of the Trial version from the computer system, and make no further use of it, except to the extent that may be permitted under any subsequent agreements between the Licensee and A&B Software LLC

### 7. Third party software

*ActiveGigE SDK* is distributed with the runtime version of GenICam™ group reference implementation, subject to GenICam™ licensing agreement.

Certain third party software included with the Software is subject to additional terms and conditions imposed by third party licensor(s).

### 8. Copyright

The Software is the property of A&B Software LLC. A&B Software LLC reserves all rights to the publishing, duplication, processing and utilization of *ActiveGigE SDK*. A single copy may be made exclusively for security and archiving purposes. Without the express written permission

---

of *A&B Software LLC* it is forbidden to:

- reverse engineer, emulate, decompile, decrypt, disassemble, or in any way derive source code from *ActiveGige SDK*
- modify, translate, adapt, alter or create derivative work(s) of *ActiveGige SDK*
- copy *ActiveGige SDK*'s accompanying written documentation
- lend, hire out or lease *ActiveGige SDK*.

#### **9. Exclusion of warranties**

A&B Software LLC offers and the Licensee accepts the product 'as is'. A&B Software LLC does not warrant *ActiveGige SDK* will meet the Licensee's requirements, nor will operate uninterrupted, nor error free.

#### **10. Liability**

With the exception of damage caused by willful or gross negligence, neither A&B Software LLC nor its distributors are responsible for any damage whatsoever which is put down to the use of *ActiveGige SDK*. This is valid without exception, including loss of profits, lost working time, lost company information or other financial losses. In any event the liability of A&B Software LLC is limited to the purchase price.

#### **11. Duration of Agreement**

This agreement is valid for an indefinite period of time. The Licensee's rights as a user automatically expire if the conditions of this agreement are in any way violated. In this event all data storage material and all copies of *ActiveGige SDK* are to be destroyed.

## 1.2 System Requirements

### Hardware requirements:

- 2 GHz Pentium 4 or better CPU recommended.
- 512 Mb or more RAM recommended.
- A graphic card supporting 65535 colors or higher.
- One or more Gigabit Ethernet boards or notebook card installed on the system.
- One or more GigE Vision™-compatible cameras connected to the system.

*Note - ActiveGige only supports devices that comply with GigE Vision™ standard, versions 1.x and 2.x. IP-cameras and other "GigE" cameras that have Ethernet physical interface, but are not GigE Vision™-compliant will not be recognized by ActiveGige.*

*Note - While it is possible to acquire images with Ethernet and Fast Ethernet ports, which support 10 MB/s and 100 MB/s respectively, this will only work at very slow frame rates and small resolutions. It is highly recommended that you use a Gigabit Ethernet Network Interface Controller.*

### Software requirements:

- Windows 2000, XP, Vista, Windows 7, Windows 8, Windows 10
  - .NET FrameWork (for .NET compatibility)
-

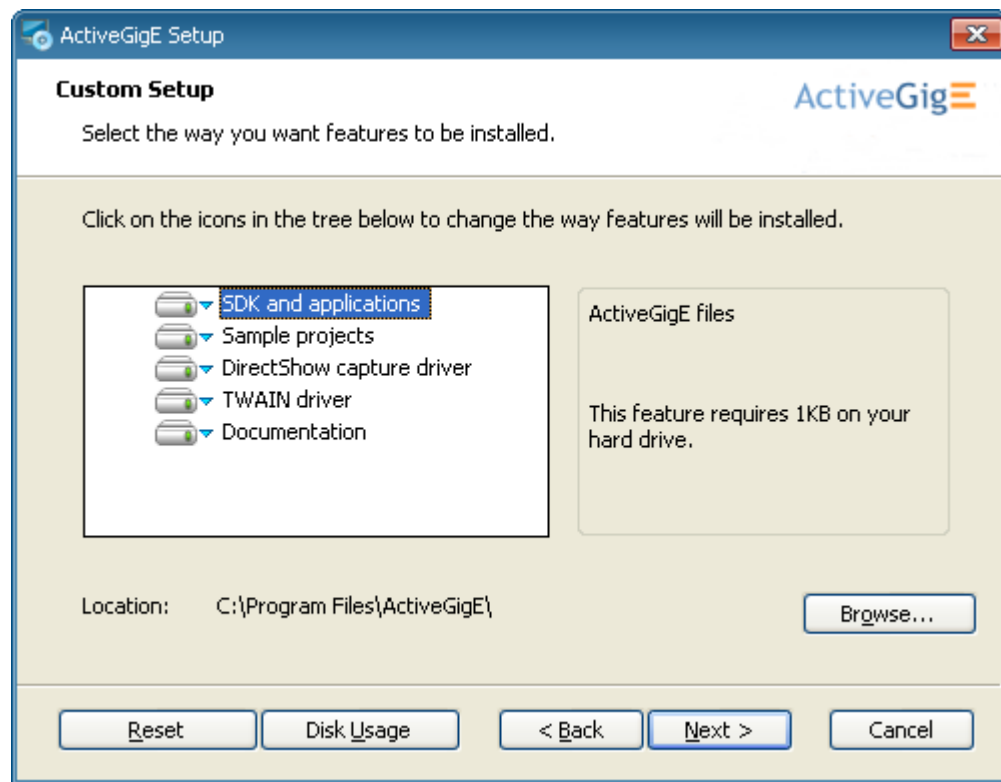
## 1.3 Installation

To install *ActiveGige*, perform the following steps:

1. Save and exit out of all currently running applications.
2. Insert *ActiveGige* CD into the CD-ROM drive. The setup program should start automatically. The **Windows Installer** box with a status bar will appear while setup prepares to start the installation process.
3. After the setup program has verified your system has the appropriate installer files, you are ready to start installing *ActiveGige*. The **Welcome** dialog box will appear. After reading the preliminary information, click **Next**.



4. The **License Agreement** dialog box will appear. To accept the license and continue, click **I Agree**, and then click **Next**. Note that the **Next** button is not available until you click **I Agree**. If you do not accept the license, click **I Do Not Agree**, and setup will terminate.
5. The **Choose Setup Type** dialog box will appear. Select which version of *ActiveGige* you want to install: **Basic** or **DVR**.
6. The **Select Installation Options** dialog box will appear, where you can modify features to be installed. The default location of *ActiveGige* files is *C:\Program Files\ActiveGige*. If you want to change the location, click **Browse...** and enter the path for the desired folder. Click **Next**.



7. The **Ready To Install** dialog box will appear. Click **Install**. *ActiveGigE* will begin installing on your system.
8. Once installation is finished, the **Complete Setup** dialog box will appear. Read the important information in the dialog and then click **Finish** to exit the installer. Note that depending on your operation system you might need to reboot your system at this point. You will be prompted if a reboot is required; if a message appears, follow the on-screen instructions.
9. Refer to the [Filter Driver](#) chapter to continue with the installation.



-

## 1.4 Filter Driver

*ActiveGigE SDK* is provided with *GCAM GigE Vision Filter Driver*.

Installation of the Filter Driver is not necessary for the operation of *ActiveGigE*, but it is recommended to increase the performance of your applications during the video capture, especially if your GigE network adapter does not support Jumbo frames. See [Network Setup](#) for more details. The Filter Driver reduces the CPU load by bypassing the Windows IP Stack and transmitting incoming video packets directly to *ActiveGigE*.

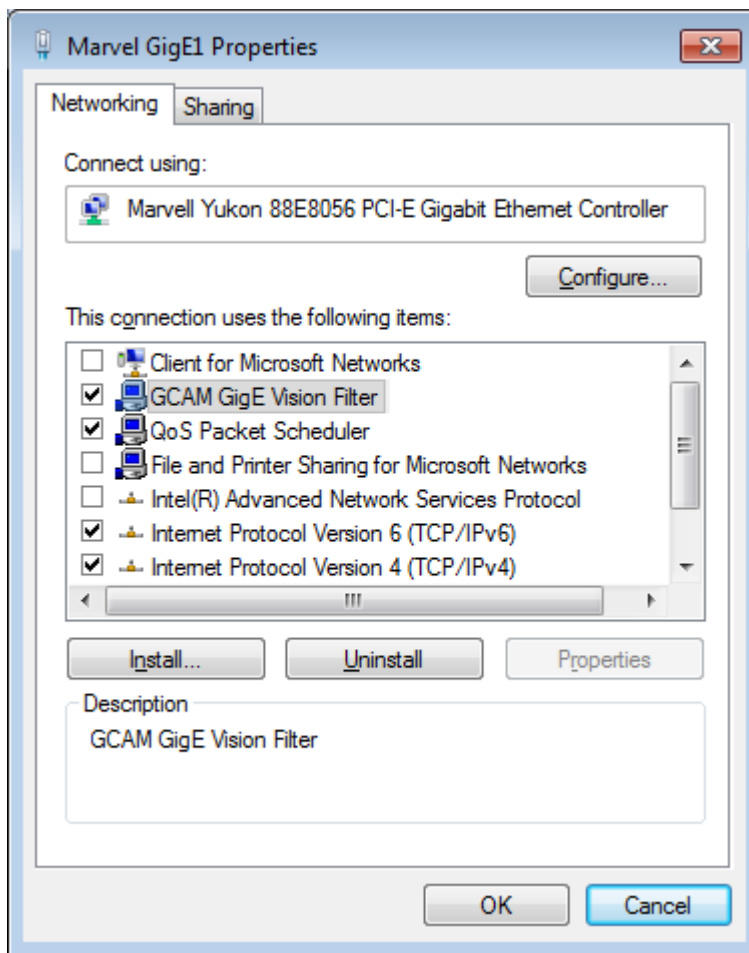
To install the Filter Driver, execute the "Install Filter Driver" option in the *ActiveGigE* start menu or use the Advanced tab of the [IP configuration utility](#). This will automatically install the driver in Windows Vista and Windows 7. If you are installing the Filter Driver in Windows XP, you will see several times a warning dialog such as this:



Ignore the warning by clicking **Continue Anyway** button each time you see the dialog.

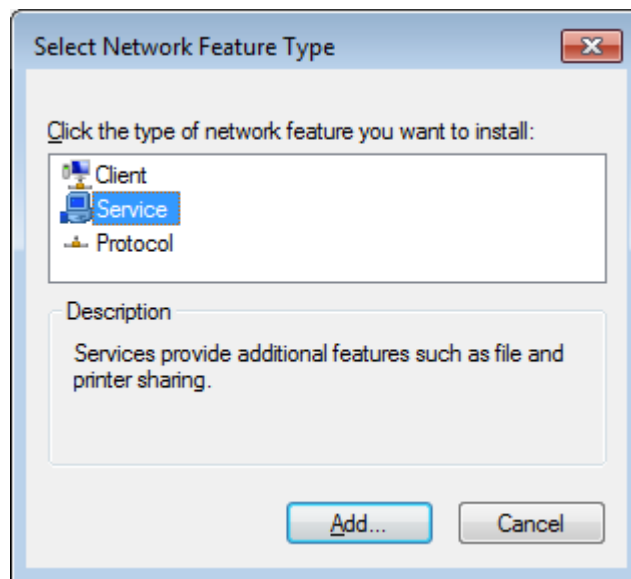
When the installation is finished, make sure that *GCAM GigE Vision Filter* appears in the Properties of your GigE Network adapter with the check mark next to it:

---



If for any reason the filter installation fails, install the driver manually using the following procedure:

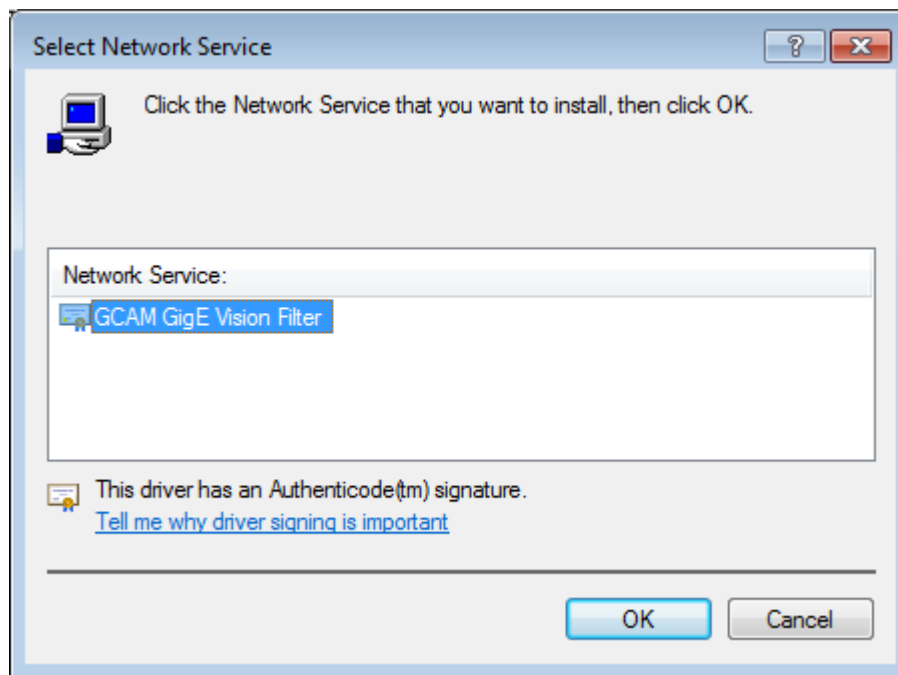
1. Open **Local Area Connection Properties** dialog for your GigE adapter and click the **Install...** button (see image above). The **Select Network Component Type** dialog will appear. Select Service and click **Add...**



2. The **Select Network Service** dialog will appear. Click **Have Disk...** and provide the path to the **Driver** folder where *GcamFilter.inf* is located. A typical location would be:

Windows 2000/XP	C:\Program Files\ActiveGigE\Driver\XP32
Windows Vista, Windows 7/8 32-bit	C:\Program Files\ActiveGigE\Driver\Vista32
Windows XP 64-bit	C:\Program Files\ActiveGigE\Driver\XP64
Windows Vista, Windows 7/8 64-bit, Windows Server 64 bit	C:\Program Files\ActiveGigE\Driver\Vista64

3. In the **Select Network Service** dialog highlight *GCAM GigE Vision Filter* and click **OK**.



If the installation is performed in Windows XP, you will see several warning messages from Microsoft. Ignore them.

4. Confirm the presence of the filter driver in the **Local Area Connection Properties** list.

*Note - If you are upgrading from the earlier version of ActiveGige, reboot your system after the installation to make sure that you are using the latest version of the filter.*

*Note - to temporarily disable the Filter Driver, unselect the corresponding check box in the **Local Area Connection Properties** list of your GigE adapter.*

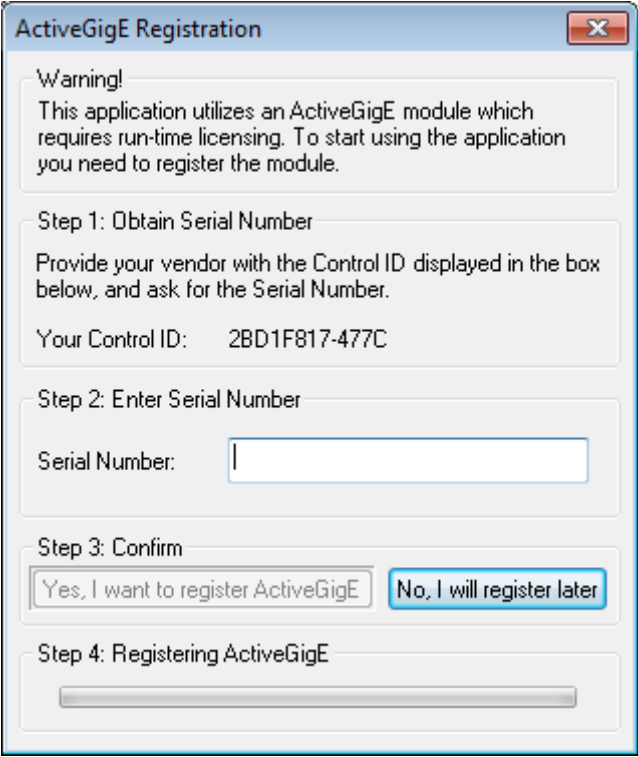
*Note - to completely remove Gcam GigE Vision Filter from your system, use the Uninstall option in the **Local Area Connection Properties** dialog or use the Advanced panel of the IP Configuration utility.*

## 1.5 Registration

The *ActiveGige* setup includes the demonstration license that allows you to use the control in both design and run-time mode for a period of 21 days.

If you wish to continue using the control, you must acquire a design-time license from your distributor. The design-time license is provided in form of a license file *activegige.lic* that must be saved in the ActiveGige folder (typically, C:\Program Files\ActiveGige) replacing the existing file. In addition, if you need to execute ActiveGige based applications outside of your VB development environment, you should perform a run-time registration. This is done through the following procedure.

When you first start an executable file that was created using *ActiveGige* control, the registration dialog box will appear.

The image shows a Windows-style dialog box titled "ActiveGigE Registration" with a close button (X) in the top right corner. The dialog is divided into several sections. The first section, labeled "Warning!", contains the text: "This application utilizes an ActiveGigE module which requires run-time licensing. To start using the application you need to register the module." The second section, "Step 1: Obtain Serial Number", instructs the user to provide their vendor with the Control ID displayed in the box below and ask for the Serial Number. It shows "Your Control ID: 2BD1F817-477C". The third section, "Step 2: Enter Serial Number", features a text input field labeled "Serial Number:". The fourth section, "Step 3: Confirm", contains two buttons: "Yes, I want to register ActiveGigE" and "No, I will register later". The fifth section, "Step 4: Registering ActiveGigE", contains a progress bar.

The dialog will display a Control ID uniquely generated for your system. Based on this ID, your distributor will provide you with a Serial Number that will validate a run-time permission for *ActiveGige*. After the proper Serial Number is entered in the dialog and registration completed, all ActiveGige based application will become unlocked.

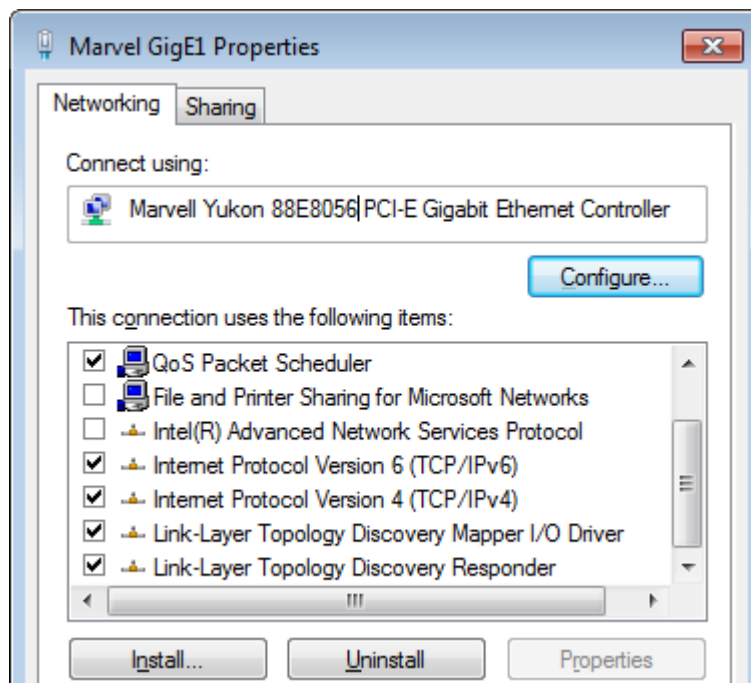
---

## 1.6 Network Setup

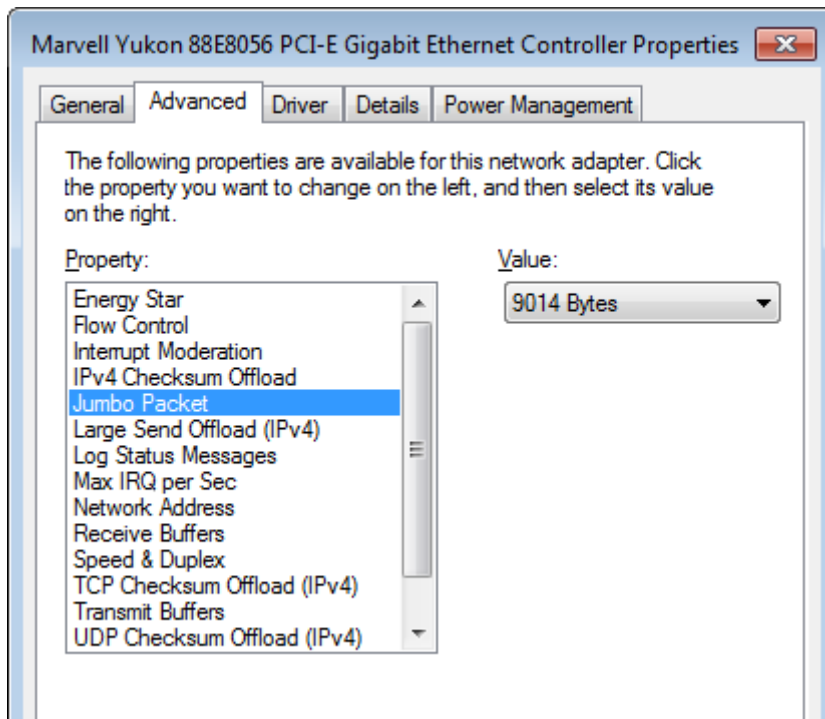
1. Make sure a Gigabit Ethernet card is installed on your system. To achieve the best performance, use a card that supports "Jumbo" frames of at least 9 KB size. You should only use a default Windows driver or the driver provided by the manufacturer of your network card.

*Attention - do not use special drivers (such as "high-performance driver") provided by your camera manufacturer. Those drivers use their own protocols not compatible with ActiveGige and would prevent ActiveGige-based software from recognizing your GigE Vision™ cameras.*

2. From the **Start** menu open **Control Panel**, then click **Network Connections**. Right click the network connection which will be used with your camera. From the context menu select **Properties**. This will display **Local Area Connection Properties** window.



3. Click **Configure** button. The adapter properties window will be displayed. Select the **Advanced** tab. This will bring up the **Property** list which will be different depending on the model and type of the network card. Select **Jumbo Frames** and change the value to the maximum size allowed. If the **Jumbo Frames** option does not appear in the list, your network card might not support it, in which case the performance of the video transmission will be reduced.

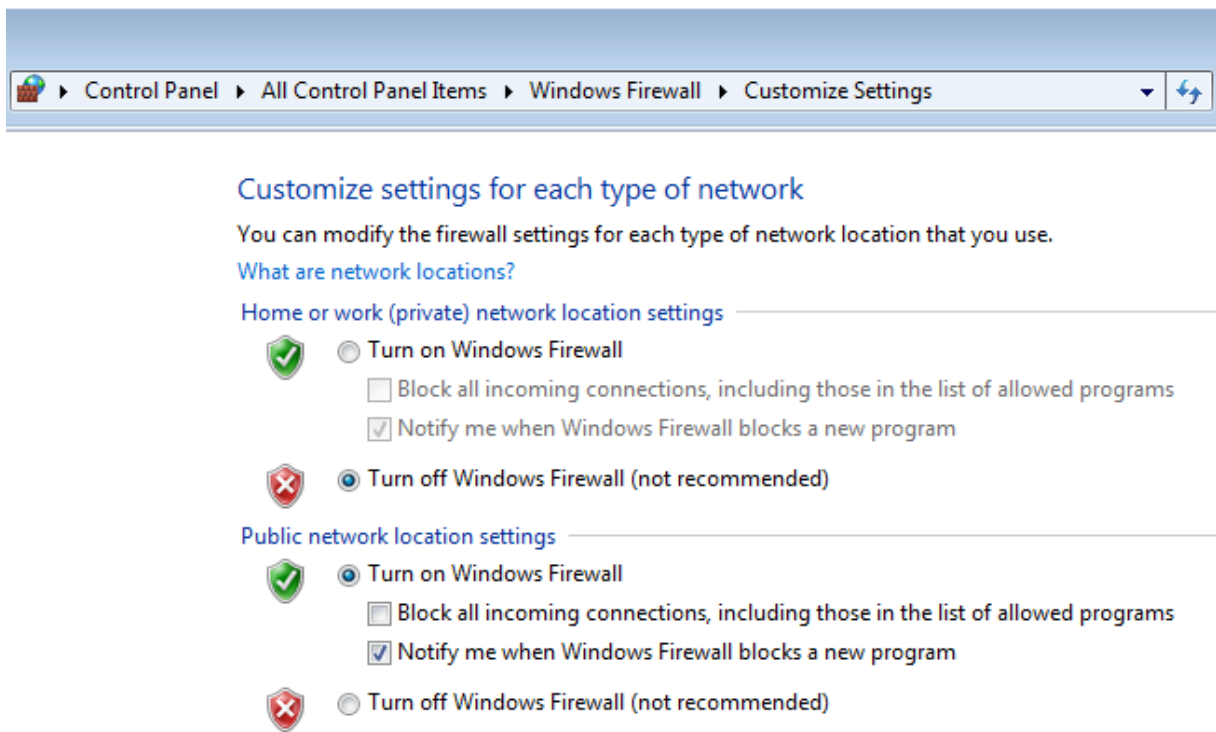


*Note - if your network card does not support jumbo frames, make sure that the packet size of the camera does not exceed 1500, or otherwise the network card will not be able to receive video data. The packet size property of the camera can be modified in the [Format](#) property page.*

4. If the **Receive Descriptors** (or **Receive Buffers**) option is available in the list, change the value to 512. Set **Max IRQ per Second** (if available) to 1000. Set **Interrupt Moderation** (if available) to **On**. Set **Interrupt Moderation Rate** (if available) to **Extreme**. Click **OK**.

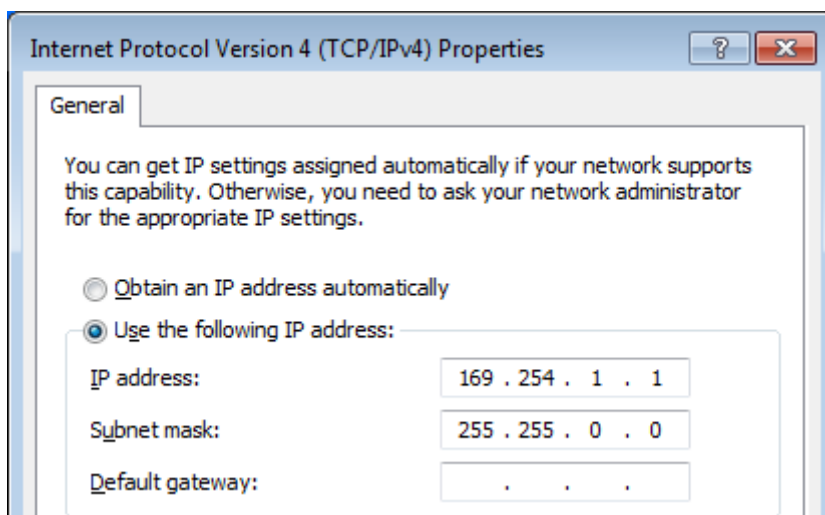
5. Go to **Start Menu ->Control Panel** and select **Windows Firewall -> Turn Windows Firewall on or off**. The **Windows Firewall** window will be displayed. Select **Off** to turn off the Windows firewall. The camera may not communicate with the system if the firewall is active.





*Note - if your camera supports the firewall traversing, ActiveGigE-based application will be able to work with it even when the Windows firewall is enabled.*

6. Reopen **Local Area Connection Properties** by right-clicking the network connection icon in the **Network Connections** window and selecting **Properties**. From the list of items select **Internet Protocol (TCP/IPv4)** and click the **Properties** button. The **Internet Protocol (TCP/IP) Properties** window will be displayed. Select **Use the following IP** address and enter **169.254.1.1** in the **IP address field**. Enter **255.255.0.0** in the **Subnet mask** field. Click **OK** to save your changes.



*Note - the network card IP address can be any 169.254.xxx.yyy value as long as it is unique in your local network. The best way to choose this number is to first select **Obtain IP addresses***

**automatically.** Then once the camera is connected, double click the network connection icon in the **Network Connections** window, select the **Support** tab and view the discovered IP address. Then redo step 6 to configure the network card with that address.

*Note - the configuration procedure above assumes that your camera has the IP address in the **169.254.x.x** class B range, which should be a default factory setting. If the camera IP address has been re-assigned to a different address range, you either reset the camera IP address to the default factory setting or set up a corresponding IP/subnet for your network card.*

7. Back in the **Local Area Connection Properties** window, in the list of items used by the connection, unselect every item except **Internet Protocol (TCP/IPv4)**, then click **OK**. The card is now fully configured for use with GigE Vision™ compliant cameras.

8. Connect your camera to the network card using a CAT-5e Ethernet cable. Power up the camera. Proceed to the [IP Configuration utility](#).

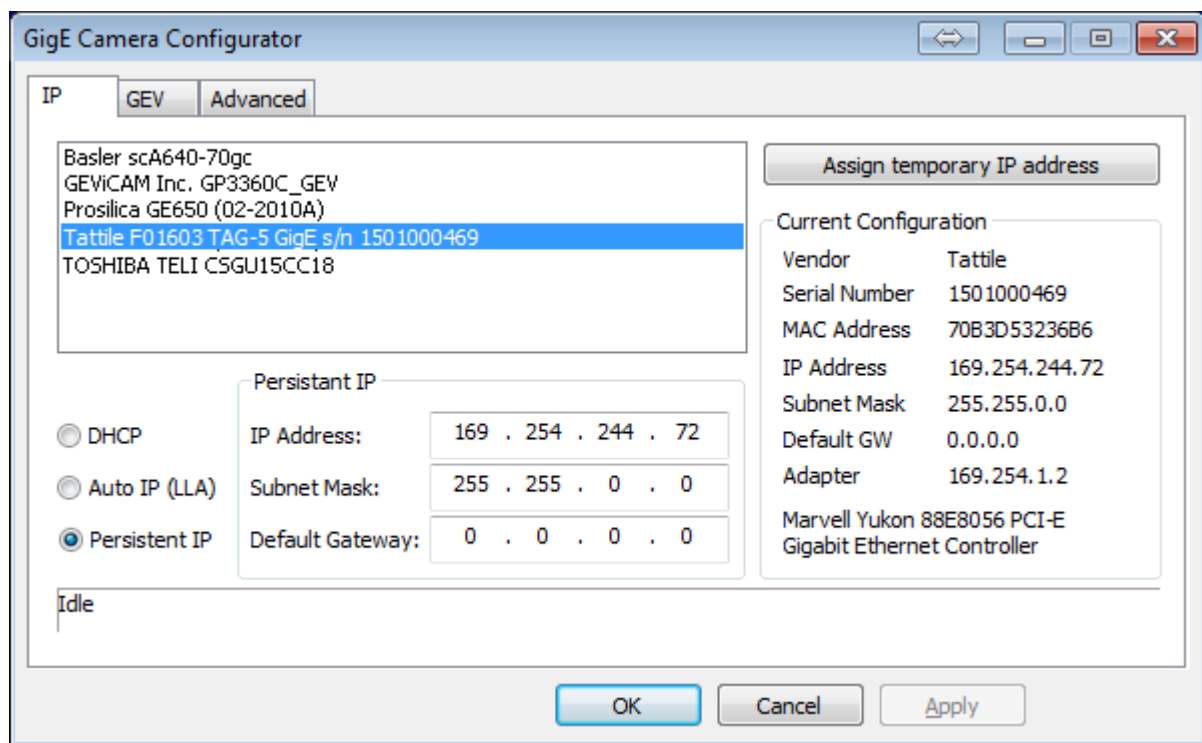
---

## 1.7 IP configuration utility

Provided with *ActiveGige* is the camera IP configuration utility. Normally, a camera's IP address has a factory setting in the 169.254.x.x class B range. In this case you should follow the [Network Setup](#) instructions to properly configure your network adapter. However, there might be when the camera IP needs to be changed, for example when [multiple cameras](#) are connected to several network cards. You can also have a situation when your camera becomes "invisible" due to the IP address mismatch between the camera and network card. The IP configuration utility can locate the camera on the network and modify its IP address for the proper operation.

To start the IP configuration utility run "gevconfig.exe" from *ActiveGige* working folder. You can also run it through the Windows Start Menu: Start -> All Programs -> ActiveGige -> IP Configurator. The following user interface will be displayed:

### IP Tab



The upper window displays the list of all GigE Vision cameras currently found in the network. If there are more than one cameras, select the one you want to reconfigure. The current camera configuration information will appear on the right. If the IP of the selected camera doesn't match the TCP/IP settings of your network card, the IP configurator will ask you to assign the camera a temporary (forced) IP address with parameters matching the subnet mask and IP address of your network card.

Once the current IP address of the camera is set to a proper value, the IP configurator can proceed with changing a permanent IP address of the camera. The following options are available:

**DHCP** - the camera will reset its IP address automatically using the Dynamic Host Configuration Protocol. The IP address will be assigned in the 169.254.x.x class B range.

**LLA** - the camera will reset its IP address automatically using the Link Local Address (Auto IP). The IP address will be assigned in the 169.254.x.x class B range.

*Persistent IP* - the IP address is assigned manually by entering the desired parameters in the test boxes. Assigning a persistent IP address to the camera will provide the fastest time for establishing connection between the camera and network card. When using this option make sure that the IP Address and Subnet Mask of the camera match the corresponding parameters of the network card.

After selecting the desired option, click the *Change IP configuration* button. The camera should reset itself and the updated IP information should appear in the *Current Configuration* pane on the right. Note that some cameras require resetting the power in order for the new IP configuration to take effect.

*Note* - Before assigning the camera a *Persistent IP* it is recommended to try it as a temporary (forced) IP address. The camera will revert to the previous IP once the power is reset.

## GigE Tab

The screenshot shows the 'GigE Camera Configurator' window with the 'Advanced' tab selected. The window is divided into three main sections: Broadcast Channel, Control Channel, and Stream Channel. Each section contains several configuration parameters with spin buttons and checkboxes.

Broadcast Channel	Control Channel	Stream Channel
Period, ms: 1500	Heartbeat, ms (0 - Default): 0	Resends: 3
Timeout, ms: 200	Timeout, ms: 200	Timeout, ms: 50
Attempts: 3	Attempts: 5	Lookback: 20
<input type="checkbox"/> Use multiple sockets		Multicast Address: 239 . 0 . 0 . 1

At the bottom of the window are three buttons: OK, Cancel, and Apply.

The left pane displays the settings of the **Broadcast channel** which is used for monitoring and discovering a presence of GigE Vision cameras on the network. *ActiveGigE* periodically sends the discovery request throughout the network and collects the information about all cameras that respond to this message in time. The following options are available:

*Period, ms* - the interval in milliseconds at which *ActiveGigE* sends discovery requests looking for GigE Vision cameras on the network. If this value is set to a large number (such as 1000000), the discovery request will be sent only once upon the start of an application. This is a recommended scenario when the application is not required to monitor the presence of cameras on the fly. The default value of the broadcast period is 1500 ms.

*Timeout, ms* - the interval in milliseconds during which *ActiveGigE* waits for responses from GigE Vision cameras after each discovery broadcast. Decreasing this time will make your application to initialize quicker, but it may prevent the discovery of some cameras especially in case of multiple cameras on the network. A recommended time for a single-camera setup is 200-300 ms, for multiple cameras - 600-1000 ms.

*Attempts* - the number of discovery requests used by *ActiveGige* to determine that a camera was removed from the network. When a new camera responds to the discovery message, *ActiveGige* immediately adds it to the [list](#) of connected cameras. However, it contains a safety mechanism which prevents it from immediately removing a camera from the list in case there was no response, because it may be caused by an accidental lost of a packet in the network. This parameter defines the number of discovery attempts that will be made by *ActiveGige* before it will consider the camera disconnected.

*Use multiple sockets* - selecting this option will make *ActiveGige* to use multiple sockets for sending discovery requests in case when multiple network cards are installed on the system. This can be used as a remedy against an erroneous IP address mismatch due to a non-standard implementation of the discovery protocol in some cameras. In general, it is recommended to keep this option unchecked to reduce the network load.

The central pane displays the settings for the **Control channel** which is used to obtain and modify camera parameters. The following options are available:

*Heartbeat, ms* - the interval at which *ActiveGige* sends the heartbeat signal to the camera. A GigE Vision camera uses the heartbeat signal as an indication that the application remains in a control of the camera. Once the application closes and the heartbeat sequence stops coming to the camera, it frees itself to another process. In some situations such as a trigger acquisition you may want to set the heartbeat interval to a large value to decrease the network traffic. If this value is set to 0 (default), the heartbeat period will be equal to 1/4 of the internal camera timeout setting which can be changed by modifying the [Heartbeat](#) property.

*Timeout, ms* - the interval in milliseconds during which *ActiveGige* waits for a response from the camera after sending a command. For example, when you move the Exposure slider in the [Property Pages](#), the application sends a series of commands to a corresponding camera register, and each command must be acknowledged by the camera before the application can send the next command, in order to maintain the synchronization between the application and camera. The default value for this parameter is 200 ms, but some cameras have a slower response and may require using a larger timeout.

*Attempts* - the number of times *ActiveGige* will try to send the same command to the camera. If the camera fails to respond in time (timeout) on all of the attempts, *ActiveGige* will raise an error. The default value for this parameter is 5.

The right pane displays the settings for the **Stream channel** which is used to transmit video from cameras to *ActiveGige*.

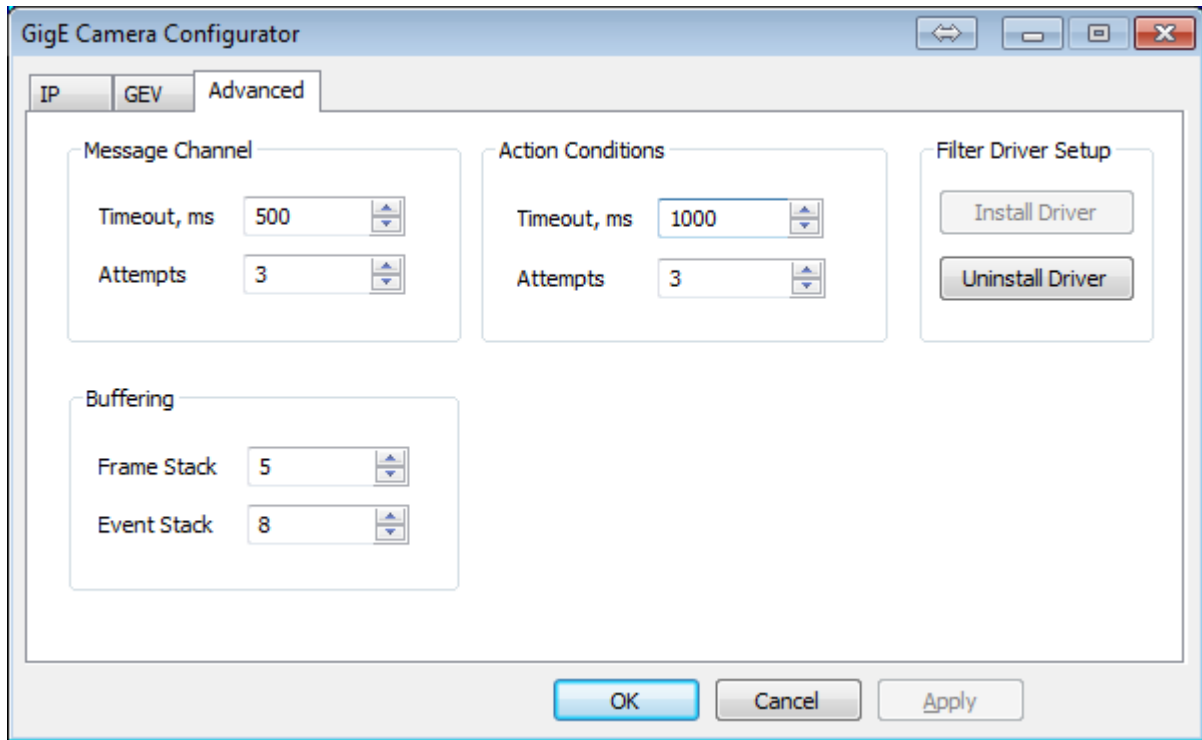
*Resends* - the number of attempts that *ActiveGige* will make to request a missing video packet. GigE Vision cameras use the User Datagram Protocol (UDP) to send the video through the network. UDP guarantees that the video is delivered with no delays, but it is inherently unreliable in terms of the packet loss. If *ActiveGige* detects a missing packet, it will send a resend request to the camera asking it to retransmit the missing packet. You may want to disable this feature on slow systems to prevent the network socket from further clogging. The default value for this parameter is 3.

*Timeout, ms* - the interval in milliseconds during which *ActiveGige* will await for a video packet from the camera after sending a resend request.

*Lookback* - size of the lookback window in packets. When packets arrive out of order, *ActiveGige* looks back to determine if all the packets previous to this point have arrived. If not, a resend is issued. If the value is too low, unnecessary resends will be issued. If the value is too high and a packet is truly missing, a resend can be issued too late when the required packet is no longer in the camera buffer. This setting is especially relevant for dual-port cameras operating via a virtual LAG (link aggregation) connection.

*Multicast Address* - the destination IP address used for the Multicast streaming. Per UDP requirements, the range of multicast addresses for local networks must be 239.0.0.0 - 239.255.255.255. The default value for this parameter is 239.0.0.1

### Advanced Tab



The top left pane displays the settings for the **Message channel** which is used to transmit message events from cameras to *ActiveGigE*.

*Timeout* - the transmission timeout for the message channel in milliseconds. Assigns the amount of time for the devices to wait for acknowledgement after a message is sent.

*Retry Count* - the number of retransmission attempts allowed for a message when it times out.

The middle pane displays the settings for the **Action conditions** which are used for Action commands.

*Timeout* - the interval in milliseconds during which *ActiveGigE* waits for an acknowledge response from one or several cameras after sending an Action command.

*Attempts* - the number of times *ActiveGigE* will try to send an Action command to the network. If devices failed to respond in time (timeout) on all of the attempts, *ActiveGigE* will raise an error. For more information refer to the description of the *WaitForAck* parameter of the [SendActionCommand](#) method.

The bottom left pane displays the settings used for **Buffering** of image frames and message events. The following options are available:

*Frame stack* - the size of the image frame buffer. In order to process all incoming frames without

missing some of them, an *ActiveGigE*-based application should be able to complete processing of a frame before the next frame arrives from the camera. In some cases an intermittent overload of the system may cause the application to pause and extend the processing of the current frame beyond the frame interval. The existence of the frame stack allows *ActiveGigE* to buffer incoming frames and delay their arrival to the application's processing queue without skipping them. The larger the frame stack is, the higher the allowance for intermittent overloads will be. A drawback of selecting a large size for the stack is a high memory consumption. The default size of this parameter is 5.

*Event stack* - the size of the event buffer. Used for processing of message events coming from the camera. The existence of the event stack allows *ActiveGigE* to buffer incoming messages and delay their arrival to the application's processing queue without skipping them. The default size of the event stack is 8.

The right pane provides options for installing and uninstalling the [Filter Driver](#). Installation of the Filter Driver is not necessary for the operation of *ActiveGigE*, but it is recommended to increase the performance of your applications during the video capture, especially if your GigE network adapter does not support Jumbo frames.

*Install* - click this button to install GCAM GigE Vision Filter driver. Note that the driver will be installed on all network adapters found in the system. If you do not want the filter driver to be active on a certain network adapter, you should disable it in the Properties of the network adapter.

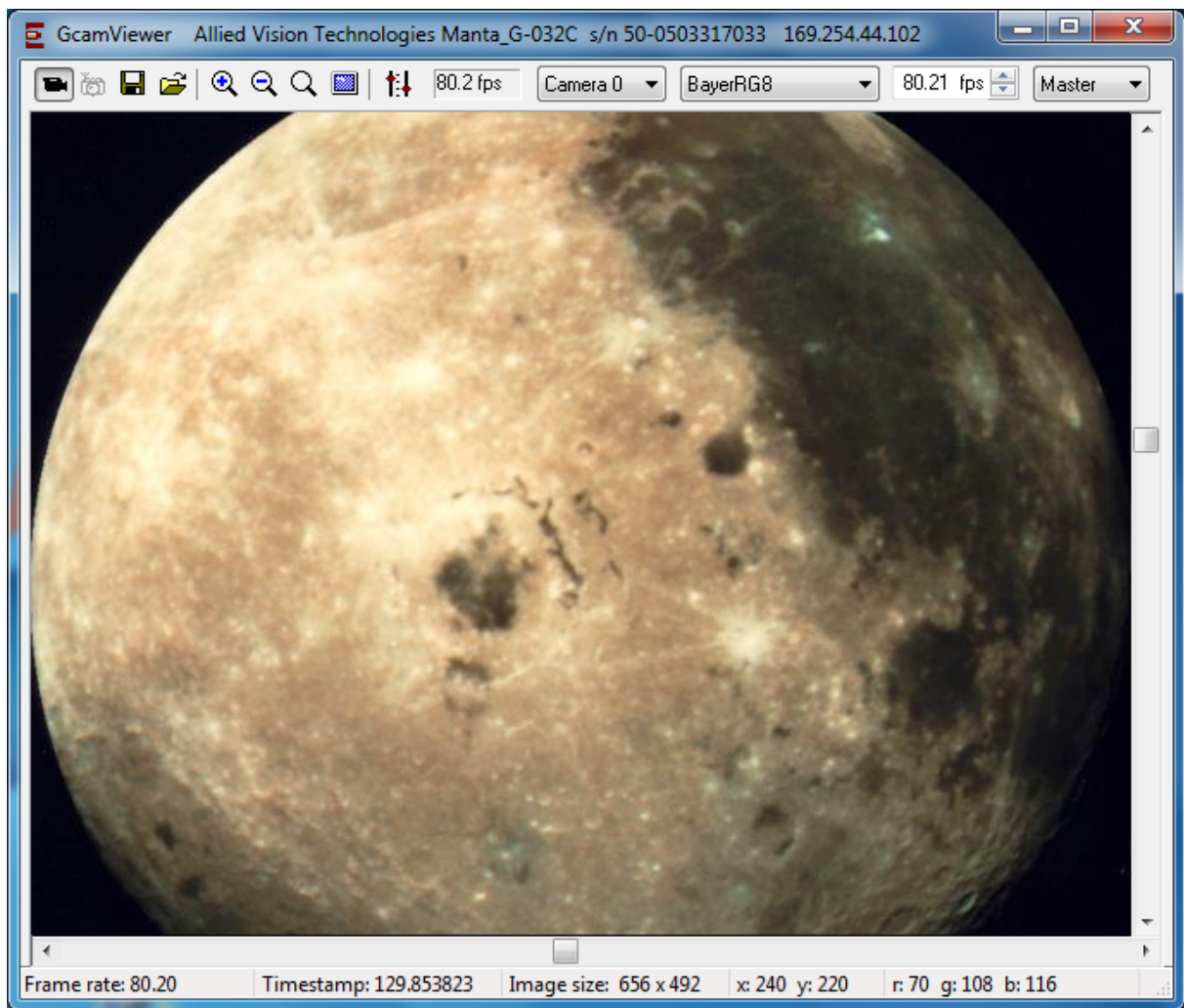
*Uninstall* - click this button to remove GCAM GigE Vision Filter from the system.



## 1.8 GcamViewer

GcamViewer is a universal plug-and-play application for GigE Vision™ compliant cameras which is intended for camera and network performance evaluation. It can display live video from multiple cameras, provide full control over camera parameters, analyze incoming video frames and save them to image files. GcamViewer is available only as part of *ActiveGigE* SDK and not as separate distribution.

To start GcamViewer run "gcamviewer.exe" from *ActiveGigE* working folder. You can also run it through the Windows Start Menu: Start -> All Programs -> ActiveGigE -> GcamViewer or by double-clicking on the GcamViewer icon at the desktop. The following user interface will be displayed:



When GcamViewer is launched for the first time, it will attempt to connect to the first camera in the GigE Vision camera list, set the optimal packet size and start acquiring the live video.

### Toolbar

The most often used commands to operate the viewer are provided through the toolbar at the upper part of the application window:



**Live Video**

Switches the camera into the continuous acquisition mode and initiates live preview.

**One Shot**

Initiates an acquisition of one frame.

**Save Image**

Lets you save the current frame buffer in an image file. When you click this button, the **Save As** dialog box will appear where you can select the file name and one of the image file formats: BMP, TIF and JPEG.

**Open Image**

Lets you open the static image from a file and display the image. When you click this button, the **Open** dialog box will appear where you can select the file name and one of the image file formats: BMP, TIF and JPEG.

**Zoom in**

Click this button to increase the magnification of the image.

**Zoom out**

Click this button to decrease the magnification of the image.

**Fit to screen**

Click this button to change the magnification factor of the image so that it fits to the image window.

**Full screen**

Click this button to enter the full screen mode. To exit the full screen mode, press the Esc key or double-click on the image.

**Settings**

Click this button to display *ActiveGigE* [Property Pages](#).

In addition to buttons, the toolbar also contains the following controls:

**Display rate**

Show the current display rate of the camera in frames per second. The display rate can be slower than the actual camera fps and may depend on the network throughput and image processing intensity.

**Select Camera**

Use this option to switch between several connected cameras.

*Select Format*

Use this option to select the desired pixel format from the list of available formats.

*Select Frame Rate*

Use this option to set the desired frame rate for the camera.

*Privilege*

Select the desired level of Privilege.

**Status Bar**

The Status Bar appears along the lower edge of GcamViewer window and contains the following information fields in the order from left to right:

*FPS*

Displays the actual frame rate of the camera.

*Timestamp*

Displays the timestamp associated with the current video frame.

*Image Size*

Displays the horizontal and vertical size of the video frames.

*Cursor Position*

Displays the X and Y coordinates of the mouse pointer. The coordinates are expressed in pixels relative to the top left corner of the image frame.

*Pixel Value*

Displays the value of the pixel identified by the mouse pointer. For a monochrome image, each pixel value will be represented by a single number, while for an RGB image by a triad of numbers.

**Menu**

Right-clicking the video window of GcamViewer will display the context menu with the following commands:

*Acquire Video*

Switches the camera into the continuous acquisition mode and initiates live preview. Equivalent to pressing the **Live** button.

*Store camera parameters*

Selecting this option will make GcamViewer store the main camera parameters upon its exit and restore them upon the next start. The parameters of each camera are stored independently in the system registry.

*Set optimal packet size*

Attempts to set the optimal packet size for the camera using the auto-negotiation functionality. If the camera does not support the packet size negotiation, the packet size will be set to 1500.

*Settings*

Displays ActiveGigE [Property Pages](#).

*Affinity*

Lets you select the number of logical CPUs that will be used for image processing. By default ActiveGigE splits the processing tasks among all logical CPUs (cores) available on the system.

---

You may want to reduce the amount of utilized cores for benchmarking or to free CPU resources for other tasks.

#### *File Transfer*

Opens the File Transfer dialog allowing you to download a data file from the camera or upload it to the camera. The File Transfer dialog will list the names of all files in the camera that can be accessed via the file transfer. When a specific file name is selected, the "Download from camera" or/and "Upload to camera" buttons will become available depending on the access mode of the file. Clicking one of these buttons will let initiate the transfer of data between a file in your system and file in the camera. The File Transfer option is typically used for updating the firmware of the camera and it is only available if the camera supports the file access functionality.

#### *Action Command*

Opens the Action Command dialog allowing you to trigger an action in one or several GigE Vision devices by firing an Action message. This is typically used to simultaneously trigger a frame acquisition in several cameras or to reset the timestamp counter. The Action Co

download a data file from the camera or upload it to the camera. The File Transfer dialog will list the names of all files in the camera that can be accessed via the file transfer. When a specific file name is selected, the "Download from camera" or/and "Upload to camera"

#### *Save Image*

Lets you save the current frame buffer in an image file.

#### *Open Image*

Lets you open the static image from a file.

#### *Zoom in*

Increases the magnification of the image.

#### *Zoom out*

Decreases the magnification of the image.

#### *Fit to screen*

Changes the magnification factor of the image so that it fits to the image window.

## **Operation**

When GcamViewer connects to a camera unit for the first time, it will attempt to negotiate and set the optimal packet size. The resulting packet size will be saved in the system registry and used on the next run of GcamViewer with the same camera. If you want other parameters of the camera to be memorized, activate the *Store camera parameters* option from the context menu.

You can run several instances of GcamViewer at the same time. Typically each of them will operate a separate camera. If several instances of GcamViewer are connected to the same camera, the first instance will receive the Master access while all other instances will receive the Monitor access. Instances with the monitor access can only read the parameters of the camera, but not modify them. By default they will also not be able to receive the video and will display the message "Camera is in use by another process". You can however make several instances of GcamViewer display the video from the same camera as long as the camera supports the Multicast mode. In order to do it, run the first instance of the viewer, click the *Settings* button and check the *Multicast* option. Then run additional instances of GcamViewer. They will open in the Monitor mode and display the live video.

To switch the Master access from on instance of the viewer to another one, set the Privilege first instance to the Switchover access, and then do the same in another instance. This will cause the first instance to switch to the monitor mode, while the second instance will receive the master level control.

If the camera does not support the Switchover access, you will have to manually put the first instance in the Monitor mode and then switch another instance to the Master mode. Note that several instances of GcamViewer can operate on different computers connected to the same camera through a network switch.

If the camera starts generating event messages, GcamViewer will display the Event Messages window which will provide information about incoming messages in real time.

You are now ready to move to the next chapter and create your own GigE Vision application!

## 2 Getting started

This chapter describes how to create a simple application with *ActiveGigE* control using various development platforms.

[Visual Basic](#)

[Visual C++](#)

[VB.NET](#)

[Visual C#](#)

---

## 2.1 Visual Basic

This chapter shows you how to get started with *ActiveGige* control in VB 6.0. With just a few mouse clicks and one line of code, you will be able to display a live video image in your Visual Basic program and report a value of a selected pixel in real time.

### Creating the Project

Assuming that you have already run the *ActiveGige* installation program and started Visual Basic, the next step is to create a project. Begin by selecting the *New Project* command from the file menu, and select *Standard EXE* as your project type. Then use the *Project / Components...* command and select *ActiveGige Control* from the list. You will see *ActiveGige* icon appear at the bottom of the toolbox:

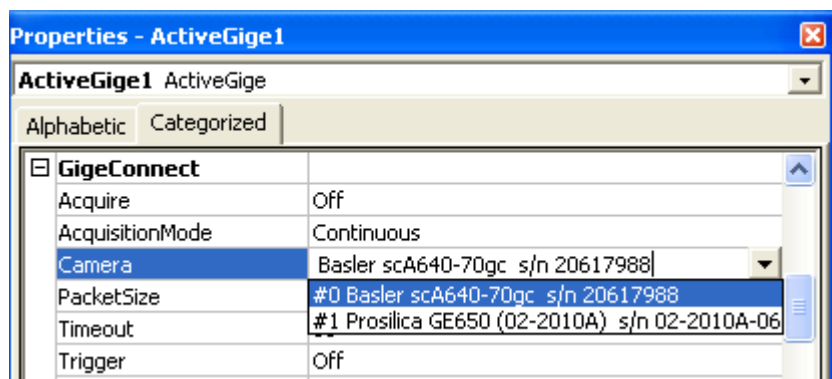
### Creating the Control

Click the *ActiveGige* icon in the *Toolbox* and draw a rectangular area on the form. A rectangle with the text "ActiveGige Control" will appear on the form, and the *Project* window on the right will display *ActiveGige*'s properties.



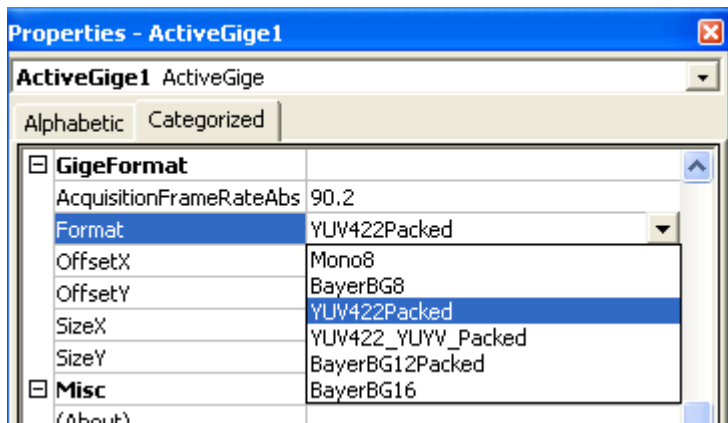
### Selecting the Camera

In the properties window, click the *Camera* property. The list box will display all GigE Vision™ cameras connected to your system. Select the one you intend to use:



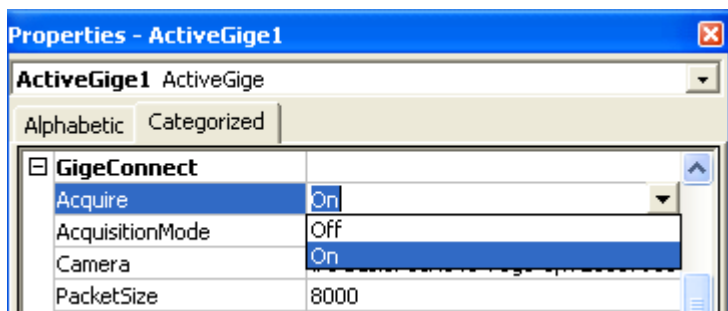
### Selecting the Pixel Format

In the properties window, click the *Format* property. The list box will display all pixel formats provided by the selected camera. Choose the one you intend to use:



### Activating continuous acquisition

In the properties window, click the *Acquire* property and set it to "On". The live video should appear on the form in the *ActiveGige* control's window.



### Adding a label

Click the label icon on the toolbox and draw a small rectangular area on the form outside of the *ActiveGige* window. A label *Label 1* will appear on the form.

### Adding the FrameAcquired event

Double-click in the control window. The code window will appear with the empty *FrameAcquired* subroutine in it. It is now time to enter the single line of code mentioned earlier:

```
Private Sub ActiveGige1_FrameAcquired()  
    'the following line needs to be added to the code  
    Label1.Caption = ActiveGige1.GetPixel (64, 32)  
End Sub
```

### Running the application

You are now ready to hit F5 and watch your program display a live video and report a real-time pixel value in the coordinates  $x = 64$ ,  $y = 32$ .

## 2.2 Visual C++

This chapter shows you how to get started with *ActiveGige* control in Visual C++. With just a few mouse clicks and a few lines of code, you will be able to display a live video image in your C++ program and report a value of a pixel pointed by a mouse cursor in real time.

### Creating the Project

VC++ 6.0

In the development environment select the *New* command from the file menu, and then select *Projects/MFC AppWizard.exe*. In the *Project name* field on the right type the name of your application, for instance *MyActiveGige* and click *OK*. When *MFC AppWizard* appears, select *Dialog based* radio button and click *Finish*. The project will be created, and the program dialog *MyActiveGige* will be displayed for editing.

VC++ 2005-2015

In the .NET development environment select *New -> Project*. The *New Project* Dialog box will appear. Select *Visual C++ projects* on the left and *MFC Application* on the right. In the *Name* field below type the name of your application, for instance *MyActiveGige* and click *OK*. When *MFC Application Wizard* appears, click *Application Type*, select *Dialog based* radio button and click *Finish*. The project will be created, and the program dialog *MyActiveGige* will be displayed for editing.

### Creating the Control

Right click in the dialog and select *Insert ActiveX control* from a shortcut menu. From the list of controls select *ActiveGige class* and press *OK*. A white rectangle will appear on the dialog.

### Generating the class for the Control

VC++ 6.0

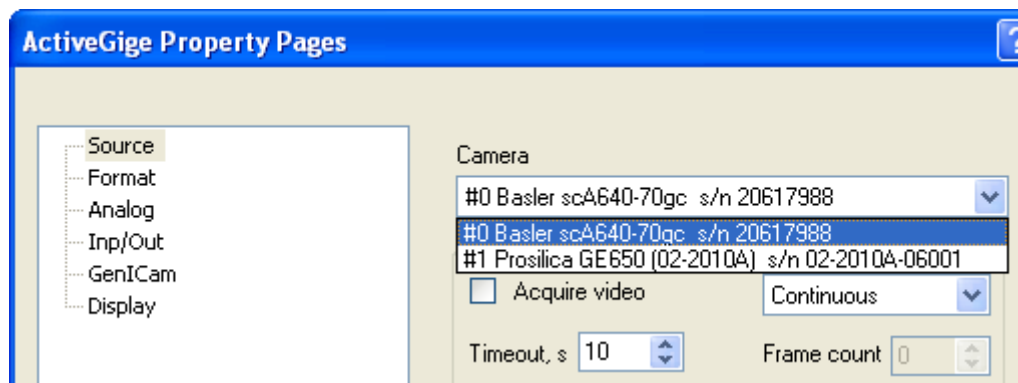
Right click in the dialog and select *Class Wizard* from the shortcut menu. When *MFC Class Wizard* appears, click the *Member Variables* tab. In the *Control ID* window click *IDC\_ACTIVEGIGE1* and then click *Add Variable*. Confirm generating the new *CActiveGige* class. When the *Add Member Variable* dialog appears, enter the name for the *ActiveGige* object, for instance *m\_ActiveGige*. Click *OK* to close the Wizard.

VC++ 2005-2015

Right click on the *ActiveGige* control in the program dialog and select *Add Variable* from the shortcut menu. *Add Member Variable Wizard* will appear. In the *Variable name* field enter the name for the *ActiveGige* object, for instance *m\_ActiveGige*, and click *finish*. The Wizard will generate a wrapper class for *ActiveGige* control and add a corresponding member variable to the main dialog class.

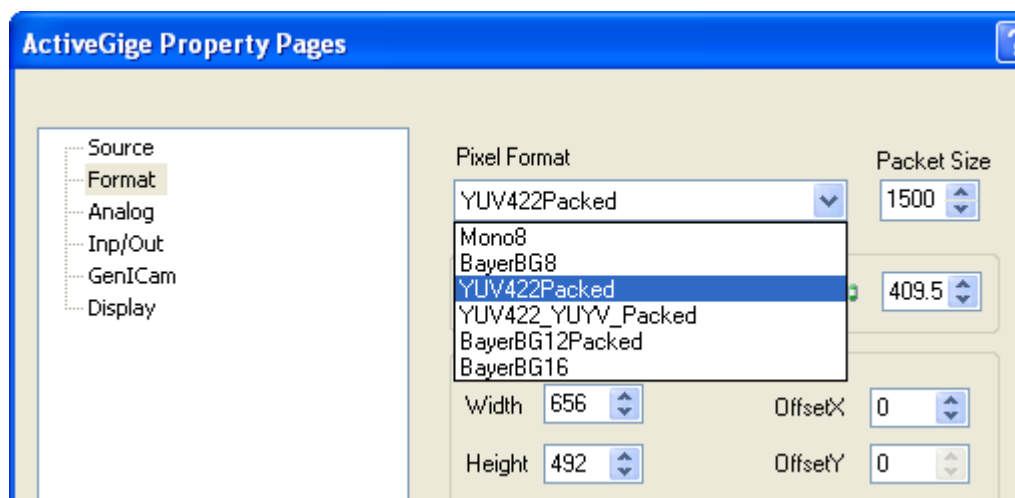
### Selecting the Camera

Right click on the *ActiveGige* control in the program dialog and select *Properties* from the shortcut menu. (In .NET select *Property pages* from the *View* menu). This will display the *ActiveGige Class Properties* tab dialog. Click the *Source* tab. The *Camera* list box will contain the names of *GigE Vision™* cameras connected to your system. Select the one you intend to use.



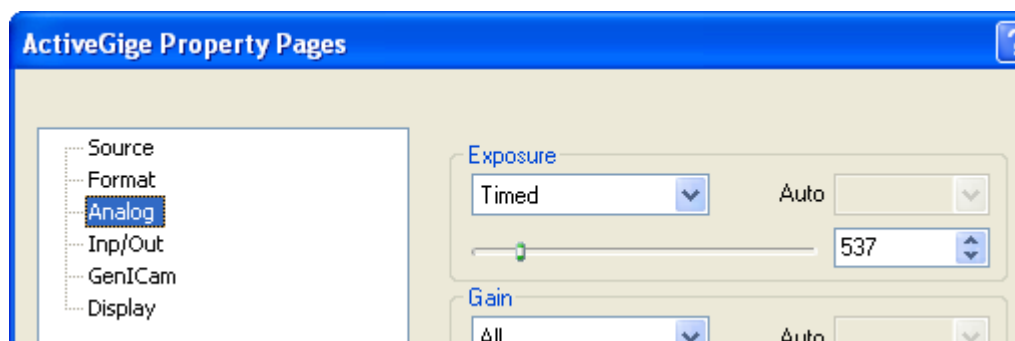
### Selecting the Video Format

Switch to the *Format* tab. Click the arrow next to the *Format* box. The list box will display all the pixel formats available for the chosen camera. Select the one you intend to use. Then select the desired image size in the *Width* and *Height* boxes.



### Adjusting the Analog settings

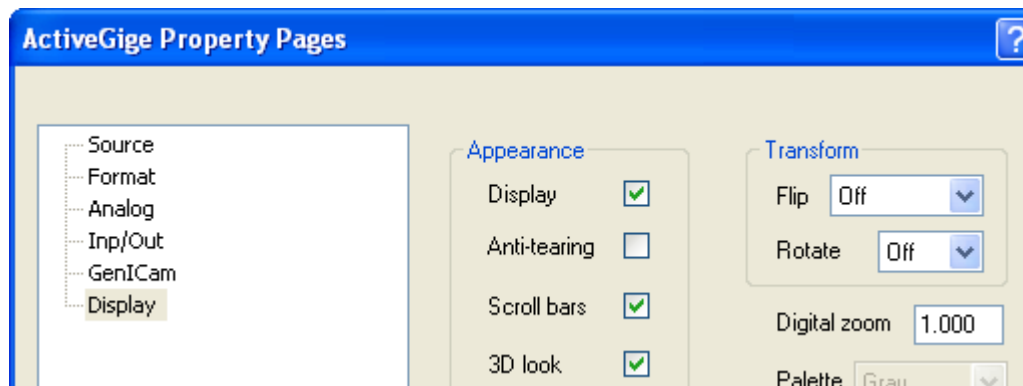
Switch to the *Analog* tab. Using the slider controls and list boxes adjust the exposure, gain, black level and white balance settings. Note that the availability of the controls depends on the selected camera.





## Modifying the Control's appearance

Switch to the *Source* tab. Select *3D look* and *Scroll bars* check boxes:



## Adding the Start button

In the *Controls* toolbox click on the *Button* icon and then draw a rectangular area on the program dialog. A button "Button1" will appear. Right click on the button and select *Properties* from the shortcut menu. In the *Caption* field type "Start" and double click on the button. The *Add Member Function* dialog box will appear. After clicking *OK* the source code window will be displayed with the new member function *OnButton1* added. Insert one line to the function body:

VC++ 6.0

```
void CMyActiveGigeDlg::OnButton1()
{
    // TODO: Add your control notification handler code here
    m_ActiveGige.SetAcquire(TRUE);
}
```

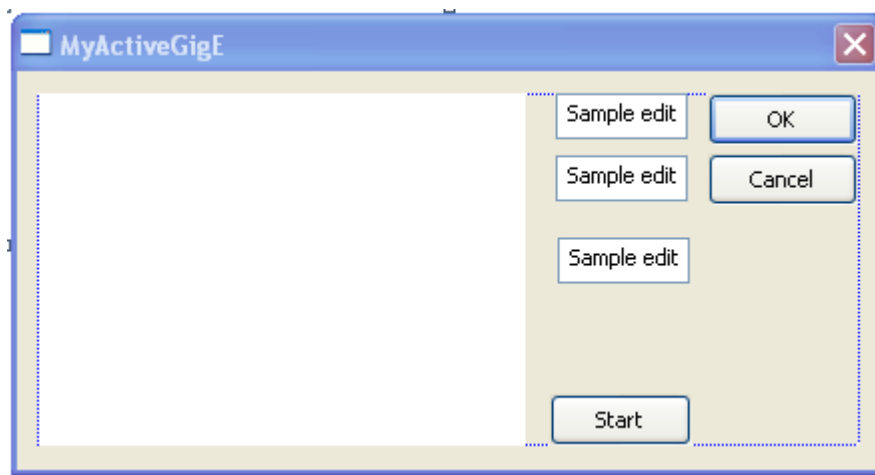
VC++ 2005, 2008, 2010

```
void CMyActiveGigeDlg::OnBnClickedButton1()
{
    // TODO: Add your control notification handler code here
    m_ActiveGige.put_Acquire(TRUE);
}
```

This will activate continuous acquisition when the button is clicked.

## Adding text boxes

In the *Controls* toolbox click on the *Edit Box* icon and then draw a small rectangular area on the program dialog. Repeat this two more times. Your final design of the program dialog will be similar to this:



### Adding the MouseMove event

Right click on the ActiveGige control in the program dialog and select *Events...* from the shortcut menu. Highlight the *MouseMove* event and click *Add and Edit*. The new event handler *OnMouseMoveActivegige1* will be added to the source code. Add the following lines to the body of the function:

```
void CMyActiveGigeDlg::OnMouseMoveActivegige1(short x, short y)
{
    // TODO: Add your control notification handler code here
    SetDlgItemInt(IDC_EDIT1,x);
    SetDlgItemInt(IDC_EDIT2,y);
    SetDlgItemInt(IDC_EDIT3,m_ActiveGige.GetPixel(x,y));
}
```

### Running the application

Close the main application form. From the *Build* menu of the development environment select *Execute MyActiveGige.exe*. This will build and run the application. The application dialog will appear on the screen with a black image window and empty text boxes. Click the *Start* button to activate the continuous video acquisition and move the mouse cursor over the image window. You are now able to watch and scroll the live image and analyze pixel coordinates and values - all in real time!

*Note - make sure to close the form containing ActiveGige control before running your application, or otherwise the IDE will maintain the exclusive control over the camera and will not allow your application to display the video.*

## 2.3 VB.NET

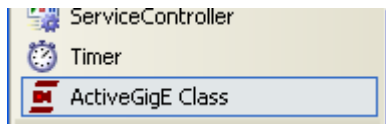
This chapter shows you how to get started with *ActiveGige* control in VB.NET. With just a few mouse clicks and a few lines of code, you will be able to display a live video image in your VB.NET program, access the array of pixel values and display them in a table in real time.

### Creating the Project

In the .NET development environment select *New -> Project*. The *New Project* Dialog box will appear. Select *Visual Basic projects* on the left and *Windows Application* on the right. In the *Name* field below type the name of your application, for instance *MyActiveGige* and click *OK*. The project will be created, and the main application form will be displayed for editing.

### Creating the Control

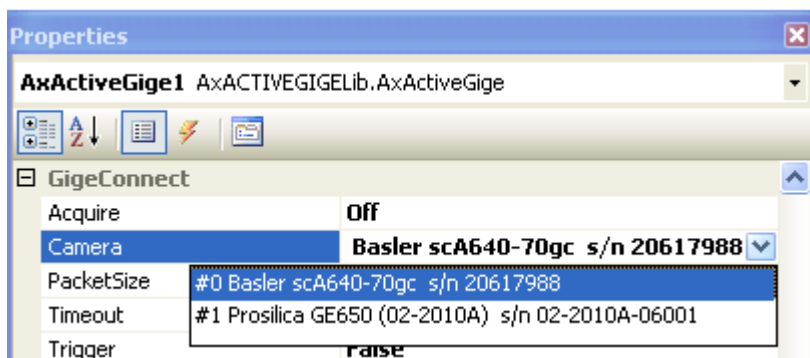
In the *Toolbox* select *Components*. From the *Tools* menu select *Choose Items... -> COM Components* and then select *ActiveGige Class* from the list. You will see *ActiveGige* icon appear at the bottom of the toolbox.



Click the *ActiveGige* icon in the *Toolbox* and draw a rectangular area on the form. A rectangle with the text "ActiveGige Control" will appear on the form, and the *Properties* window on the right will display *ActiveGige's* properties.

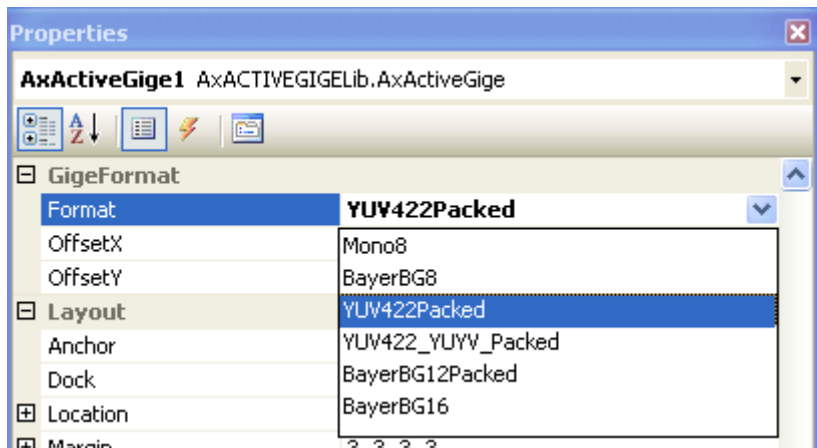
### Selecting the Camera

In the properties window, click the *Camera* property. The list box will display all GigE Vision™ compatible cameras connected to your system. Select the one you intend to use:



### Selecting the Pixel Format

In the *Properties* window, click the *Format* property. The list box will display all pixel formats available for the chosen camera. Select the one you intend to use:



### Adding the Start and Stop buttons

In the *Toolbox* select *Windows Form*, click on the *Button* icon and then draw a rectangular area on the program dialog. A button "Button1" will appear. Go to the *Property* window and change the *Text* property to "Start". Double click on the button. A new function *Button1\_Click* will be added to the *Form1.vb* source code. Insert one line to the function body:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
    AxActiveGige1.Acquire = True
End Sub
```

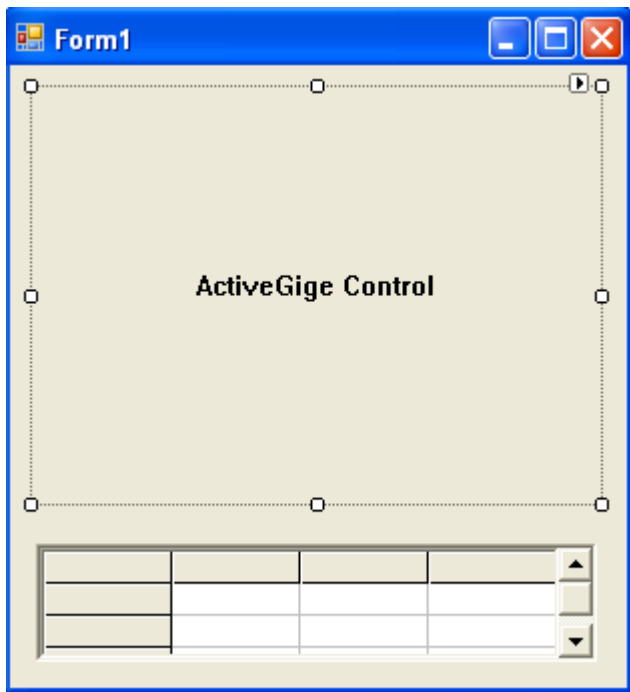
Repeat the same procedure for the *Stop* button adding the following line to the *Button2\_Click* function:

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
    AxActiveGige1.Acquire = False
End Sub
```

### Adding the Grid

From the *Tools* menu select *Customize Toolbox* and then select *Microsoft Flex Grid Control* from the list. A *FlexGrid* icon will appear at the bottom of the toolbox. Click the icon and draw a rectangular area on the form. Go to the *Property* window and set both *Cols* and *Rows* property to 5. You will see a corresponding number of rows and columns added to the grid on the main form.

Your final design of the main form will be similar to this:



### Adding the FrameAcquired event

Double-click in the ActiveGige control window. The code window will appear with an empty `AxActiveGige1_FrameAcquired` subroutine in it. It is now time to enter the code that will retrieve an image data array and display pixel values in the grid cells:

```
Private Sub AxActiveGige1_FrameAcquired(ByVal sender As System.Object, ByVal e As
    AxACTIVEGIGELib._IActiveGigeEvents_FrameAcquiredEvent) Handles
    AxActiveGige1.FrameAcquired
        Dim A As Array
        Dim X As Integer
        Dim Y As Integer
        A = AxActiveGige1.GetImageData
        For y = 1 To 4
            For x = 1 To 4
                AxMSFlexGrid1.set_TextMatrix(Y, X, A(X, Y))
            Next
        Next
    End Sub
```

### Running the application

Close the design window. From the *Debug* menu of the development environment select *Start* or press F5. This will build and run the application. The application dialog will appear on the screen with a black image window and empty text boxes. Click the *Start* button to activate the continuous video acquisition. The live image will appear in the control window and the real-time pixel values will be displayed in the table.

*Note - make sure to close the form containing ActiveGige control before running your application, or otherwise the IDE will maintain the exclusive control over the camera and will not allow your application to display the video.*

## 2.4 Visual C#

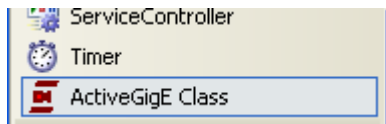
This chapter shows you how to get started with *ActiveGige* control in Visual C#. With just a few mouse clicks and a few lines of code, you will be able to display a live video image in your C# program and report a value of a pixel pointed by a mouse cursor in real time.

### Creating the Project

In the .NET development environment select *New -> Project*. The *New Project* Dialog box will appear. Select *Visual C# projects* on the left and *Windows Application* on the right. In the *Name* field below type the name of your application, for instance *MyActiveGige* and click *OK*. The project will be created, and the main application form will be displayed for editing.

### Creating the Control

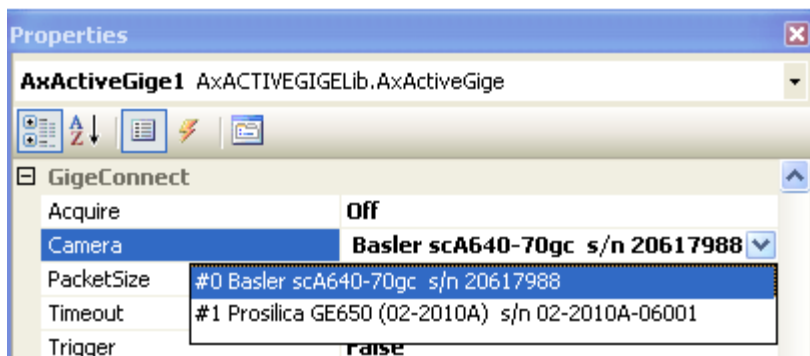
In the *Toolbox* select *Components*. From the *Tools* menu select *Choose Items...-> COM Components* and then select *ActiveGige Class* from the list. You will see *ActiveGige* icon appear at the bottom of the toolbox.



Click the *ActiveGige* icon in the *Toolbox* and draw a rectangular area on the form. A rectangle with the text "ActiveGige Control" will appear on the form, and the *Properties* window on the right will display *ActiveGige's* properties.

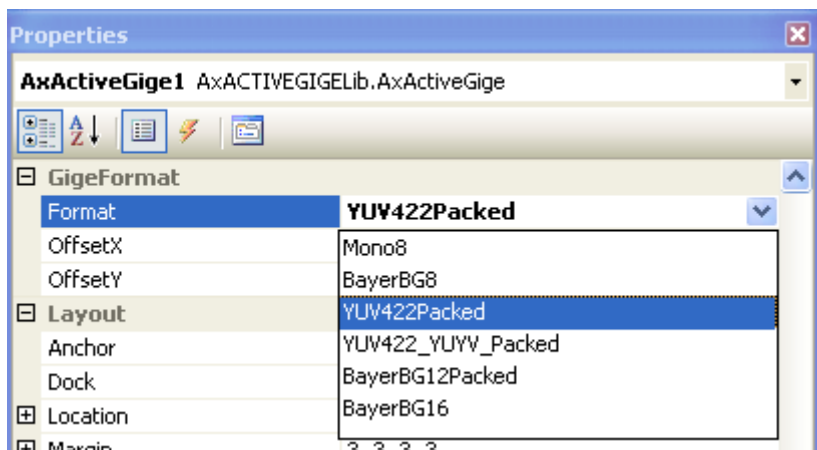
### Selecting the Camera

In the properties window, click the *Camera* property. The list box will display all GigE Vision™ compatible cameras connected to your system. Select the one you intend to use:



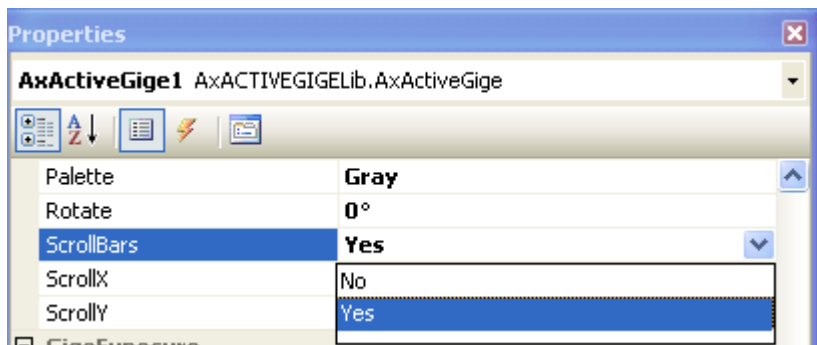
### Selecting the Pixel Format

In the *Properties* window, click the *Format* property. The list box will display all pixel formats available for the chosen camera. Select the one you intend to use:



### Modifying the Control's appearance

In the Properties window set the *Edge* and *ScrollBars* fields to "Yes"



### Adding the Start button

In the *Toolbox* select *Windows Form*, click on the *Button* icon and then draw a rectangular area on the program dialog. A button "Button1" will appear. Go to the *Property* window and change the *Text* property to "Start". Double click on the button. A new member function *button1\_Click* will be added to the *Form1* source window. Insert one line to the function body:

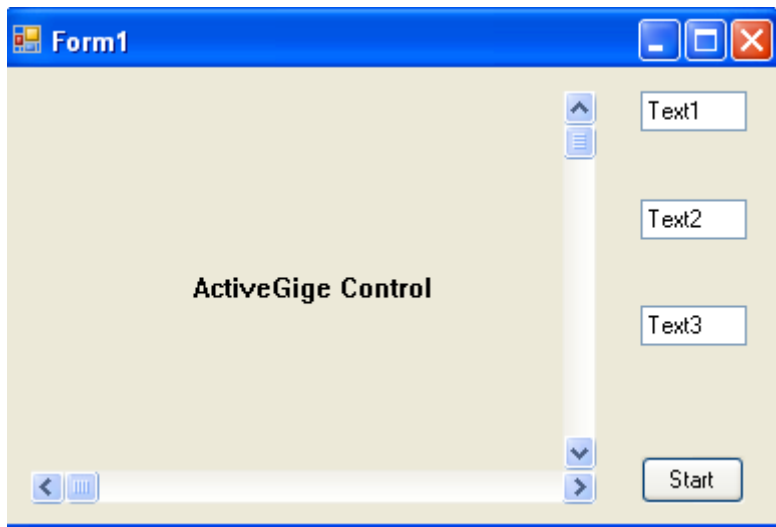
```
private void button1_Click(object sender, System.EventArgs e)
{
    axActiveGige1.Acquire=true;
}
```

This will activate continuous acquisition when the button is clicked.

This will activate continuous acquisition when the button is clicked.

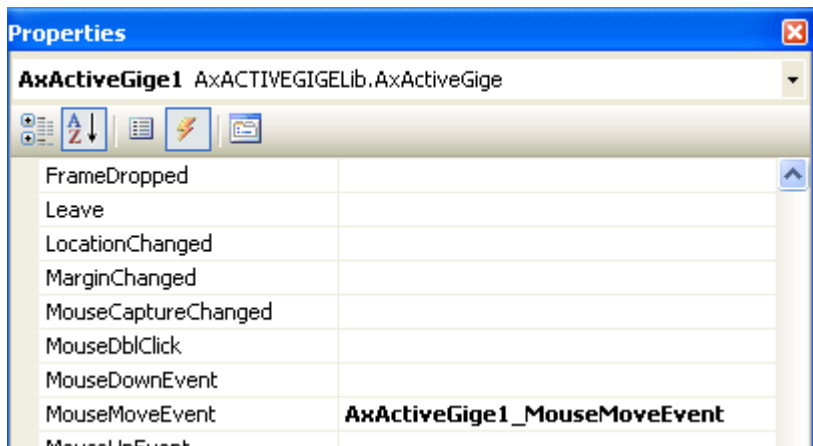
### Adding text boxes

In the *Toolbox* click on the *TextBox* icon and then draw a small rectangular area on the program dialog. Repeat this two more times. Your final design of the main form will be similar to this:



### Adding the MouseMove event

Click on the ActiveGigE control on the main form. In the *Property* window click the *Event* button. In the list of events double-click the *MouseMoveEvent*.



The new event handler `axActiveGigE1_MouseMoveEvent` will be added to the source code. Add the following lines to the body of the function:

```
private void axActiveGigE1_MouseMoveEvent(object sender,
AxACTIVEGIGELib._IActiveGigEvents_MouseMoveEvent e)
{
    textBox1.Text=Convert.ToString(e.x);
    textBox2.Text=Convert.ToString(e.y);
    textBox3.Text=Convert.ToString(axActiveGigE1.GetPixel(e.x,e.y));
}
```

### Running the application

Close the design window. From the *Debug* menu of the development environment select *Start* or press F5. This will build and run the application. The application dialog will appear on the screen with a



---

black image window and empty text boxes. Click the *Start* button to activate the continuous video acquisition and move the mouse cursor over the image window. You are now able to watch and scroll the live image and analyze pixel coordinates and values - all in real time!

*Note - make sure to close the design window with the form containing ActiveGige control before running your application, or otherwise the IDE will maintain the exclusive control over the camera and will not allow your application to display the video.*

---

## 2.5 Matlab

This chapter shows you how to get started with *ActiveGigE* control in Matlab without any need for Mathwork's Image Acquisition Toolbox. With just a few mouse clicks and a few lines of code, you will be able to display a live video image in your Matlab program and report a value of a selected pixel in real time.

### Creating the Project

In the Matlab environment type the following command in the Command Window prompt:

```
>>GUIDE
```

The *GUIDE Quick Start* dialog box will appear. Select *Create New GUI*, then select Blank GUI and click *OK*. A new GUI window will open with a component palette on the left.



### Creating the Control

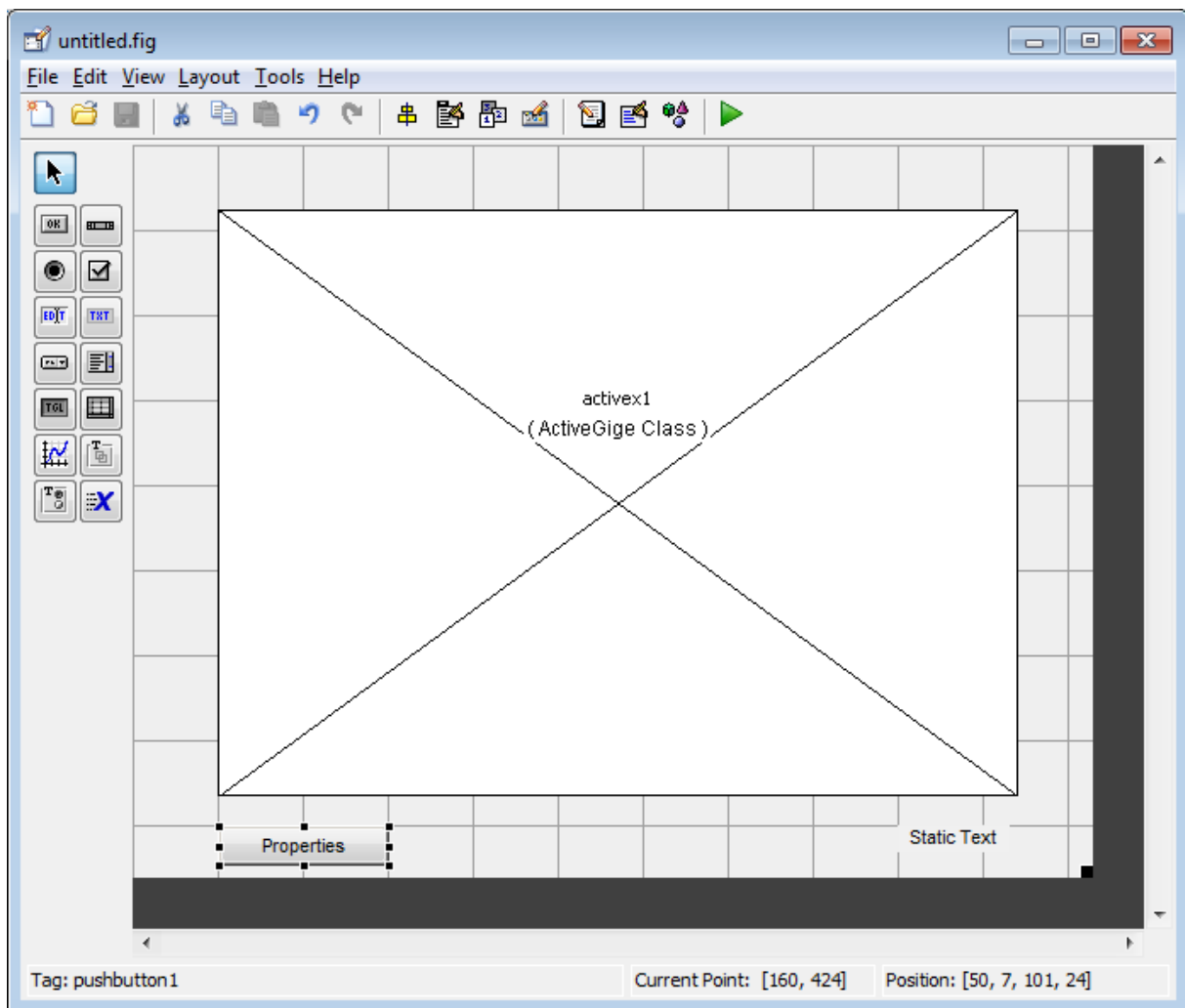
Click the ActiveX Control button (the last button in the palette with a blue "X" sign) and draw a rectangular area in the gridded UI window. The *Select an ActiveX Control* dialog will appear. Select *ActiveGigE Class* in the list and click *Create*. A rectangle with the text "activex1 (ActiveGigE Class)" will appear in the UI window.

### Adding a Text box and Push button

Click the Static Text icon (TXT) in the component palette and draw a small rectangular area in the UI window outside of the *activex1* window. A *Static Text* box will appear in the GUI. Double-click the box and rename the Tag property from "text1" to "pixel".

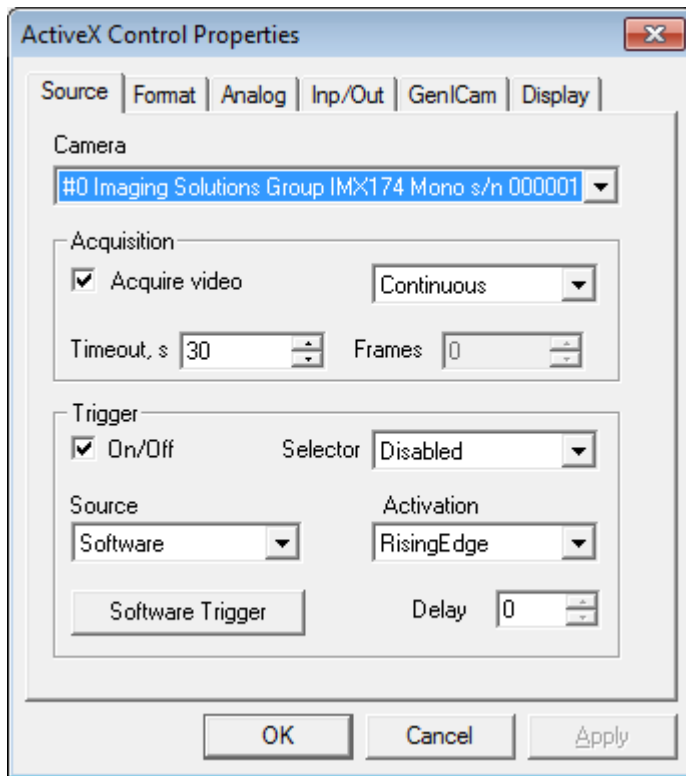
Click the Push Button icon (ON) in the component palette and draw a small rectangular area in the UI window outside of the *activex1* window. A *Push Button* will appear in the GUI. Double-click the box and rename the String property to "Properties".

---



### Selecting the Camera and Pixel Format

Right-click in the *activex1* rectangle and select *ActiveX Property Editor*. The *ActiveX Control Properties* dialog box will appear. In the *Camera* box select the camera you want to use. Then switch to the *Format* panel of the dialog and select the desired pixel format in the *Format* box.



### Adding the FrameAcquired event

Right-click in the *activex1* rectangle and move the cursor to *View Callbacks*. The list of ActiveGigE events will appear. Select *FrameAcquired*. The Program Editor will open in the Matlab environment containing a blank program with *activex1\_FrameAcquired* function added. It is now time to add a few lines of code to the program.

### Adding ActiveGigE calls

Add the following lines to *\_OpeningFcn* function. This will add scroll bars to ActiveGigE window, start a continuous acquisition and initiate live video display:

```
handles.activex1.ScrollBars=true;
handles.activex1.Acquire=true;
```

Add the following line to *pushbutton1\_Callback* function. This will display ActiveGigE property pages upon clicking the Property button:

```
handles.activex1.ShowProperties();
```

Add the following line to *activex1\_FrameAcquired* function. This will retrieve the value of pixel in coordinates *x=32, y=64* and display it in the text box upon each frame acquisition:

```
set(handles.pixel, 'String', handles.activex1.GetPixel(32,64));
```

### Running the application

You are now ready to hit the Run button in the Matlab environment and watch your program display a live video, allow you to adjust all camera settings and report a real-time pixel value in the coordinates *x*

---

= 32, y = 64. Make sure to close your GUI editor before running the application, or otherwise the GUI editor will block the application from accessing the camera.

*If you need to create a Matlab application with no GUI, refer to [Using ActiveGigE API at runtime](#).*

## 2.6 Python

Please refer to [Using ActiveGigE API at runtime](#) chapter.

---

## 2.7 Using ActiveGigE API at runtime

Most applications would use *ActiveGigE* by embedding one or several ActiveX objects into an application form at design time, as described in the previous topics. Sometimes however you may want to use *ActiveGigE* SDK dynamically at runtime. This can be useful if you want a user of your application to decide if he wants to use GigE Vision cameras or if you are creating a dynamic link library that has no GUI. *ActiveGigE* easily allows you to do it via its COM programming interface.

### C/C++

1. Copy *ActiveGigE\_i.c* and *ActiveGigE.h* files into your project folder and include them into your project.

2. Add the following defines to a module which will be using *ActiveGigE* functions:

```
#include <comdef.h>
#include <atlbase.h>
#include <atlconv.h>
#include "ActiveGigE.h"
```

3. Initialize COM library and instantiate an *ActiveGigE* object:

```
IActiveGigE *pActiveGigE;
HRESULT hr = CoInitialize(0);
if (hr==S_OK)
{
    hr = CoCreateInstance(CLSID_ActiveGigE, NULL, CLSCTX_INPROC_SERVER, IID_IActiveGigE,
        (void**) &pActiveGigE);
}
```

4. Start using *ActiveGigE* properties and methods, for example:

```
hr = pActiveGigE->put_Camera(0); // selecting camera #0
hr = pActiveGigE->Grab();         // grabbing a frame
hr = pActiveGigE->SaveImage( OLESTR("frame1.jpg") ); //saving a frame as jpg file
```

Refer to *GcamConsole* and *GcamWin* sample applications for more details.

### VB6

1. Add the *ActiveGigE* component to the Toolbox as described in [Getting started in VB](#).

2. Declare a global *ActiveGigE*-type object variable in the beginning of your code:

```
Dim WithEvents AG As ActiveGigE
```

3. Instantiate an *ActiveGigE* object and assign it to the variable:

```
Set AG = New ActiveGigE
```

4. Start using *ActiveGigE* properties and methods, for example:

CameraList=AG.GetCameraList	'retrieving the list of names of connected cameras
AG.Camera=0	'selecting camera #0
AG.SetFeature "GainRaw", 150	'setting Gain value
AG.Acquire=True	'initiating the acquisition

5. To add an ActiveGige event handler to your code, select the AG variable from the Object combo box on top of your code window and then select a desired event from the Procedure box on the right, for example FrameAcquired. The following fragment will be added to your code:

```
Private Sub AG_FrameAcquired()
End Sub
```

Refer to *GcamByRef* sample application for more details.

6. When you are done with using the ActiveGige object, destroy it with the following command:

```
Set AG = Nothing
```

## VB.NET

1. 1. Right click on your application in the Solution Explorer, select Add Reference...->COM, then select *ActiveGige* Control from the list.

2. Declare a global *ActiveGige*-type object variable in the beginning of your code:

```
Dim WithEvents AG As ACTIVEGIGELib.ActiveGige
```

3. Instantiate an *ActiveGige* object and assign it to the variable:

```
AG = New ACTIVEGIGELib.ActiveGige
```

4. Start using *ActiveGige* properties and methods, for example:

```
Dim CamLst As Object
Dim i As Integer
CamLst = AG.GetCameraList           'retrieving camera list
CamNumber = UBound(CamLst)         'number of connected cameras found
For i = 0 To CamNumber              'filling out combo box with the names of cameras
    ComboBox1.Items.Add(CamLst(i))
Next
.....
AG.Camera = 0
AG.Acquire = True
```

5. To add an ActiveGige event handler to your code, select the AG variable from the Class Name box on top of your code window and then select a desired event from the Method Name box on the right, for example FrameAcquired. The following fragment will be added to your code:

```
Private Sub AG_FrameAcquired() Handles AG.FrameAcquired
End Sub
```

6. When you are done using the ActiveGige object, destroy it with the following command:

```
AG = Nothing
```

## C#

1. Right click on your application in the Solution Explorer, select Add Reference.../COM, then select *ActiveGige* Control from the list.



2. Declare a global *ActiveGige*-type object variable:

```
ACTIVEGIGELib.ActiveGige AG;
```

3. Instantiate an *ActiveGige* object and assign it to the variable:

```
AG = new ACTIVEGIGELib.ActiveGige();
```

4. Start using *ActiveGige* properties and methods, for example:

```
AG.Camera = 0
AG.Acquire=true;           //starting automatic acquisition
AG.ShowProperties (true, 0); //displaying built-in property pages
```

Refer to *GcamDynamic* sample application for more details.

## MATLAB

1. Add the following global variable to your program:

```
global AG
```

2. Instantiate a *ActiveGigE* object and assing it to the variable:

```
AG=actxcontrol('ActiveGigE.ActiveGigE.1');
```

3. Start using *ActiveGigE* properties and methods, for example:

```
AG.Camera = 0
AG.Grab();           %acquiring a frame
pix=AG.GetPixel(32,64); %retreiving a pixel value at specific coordinates
```

Refer to *SingleCam.m* and *MultiCam.m* sample applications for more details.

## PYTHON

1. Add the following lines to the beginning of your script:

```
import win32api
import win32com.client
```

2. Instantiate a *ActiveGigE* object and assing it to a variable:

```
AG=win32com.client.Dispatch("ActiveGigE.ActiveGigE")
```

3. Start using *ActiveGigE* properties and methods, for example:

```
cameras=AG.GetCameraList()
for i in cameras :
    print i           %printing names of connected cameras

AG.Camera = 0
AG.Grab()           %acquiring a frame
pix=AG.GetPixel(32,64) %retreiving a pixel value at specific coordinates
```

Refer to *ActiveGigE.py* and *ActiveGigELive.py* sample applications for more details.

---

## 2.8 Working with multiple cameras

Before creating your multiple camera application, you should configure your local network for multiple camera setup.

The easiest way to operate multiple GigE Vision cameras is to use a single network adapter and Gigabit Ethernet switch. You can also operate multiple cameras through multiple network adapters installed on your system. In this case, each network card and corresponding camera must be assigned a subnet different from the subnet of another network card/camera.

For the two-camera connection through a pair of network adapters follow the procedure below.

a) Make sure two Gigabit Ethernet cards are installed on your system.

b) Refer to steps 1-7 of [Network Setup](#) and configure each network card with the following addresses:

Network card 1: IP address: 169.254.100.1	Subnet mask: 255.255.255.0
Network card 2: IP address: 169.254.200.1	Subnet mask: 255.255.255.0

c) Connecting both cameras to corresponding adapters, one by one, and using the manufacturer provided IP configuration utility assign the following address to the camera:

Camera 1: IP address: 169.254.100.2	Subnet mask: 255.255.255.0
Camera 2: IP address: 169.254.200.2	Subnet mask: 255.255.255.0

Your system is now configured for working with two cameras.

Programmatically, interfacing to multiple cameras is as easy as dropping a few *ActiveGige* objects on the surface of your application.

1. Start by creating a new project. Depending on the development environment you are using, refer to one of the following chapters for more details:

[Visual Basic](#)

[Visual C++](#)

[VB.NET](#)

[Visual C#](#)

2. Configure each *ActiveGige* object for a different camera by clicking a corresponding *ActiveGige* window and modifying the *Camera* property. *Do not configure different ActiveGige objects for the same camera. Multiple instances of ActiveGige cannot acquire video from the same camera.* For the same reason, if you run several *ActiveGige*-based applications, make sure that each of them connects to a different camera.

3. Add *FrameAcquired* event handlers per each *ActiveGige* object following the procedure for your language environment. Due to the multithreading nature of *ActiveGige*, they will not interfere with each other.

4. Add the acquisition command for each *ActiveGige* object to initiate the video streaming. In VB.NET you might have the following lines in your code:

```
AxActiveGige1.Acquire = True  
AxActiveGige2.Acquire = True  
AxActiveGige3.Acquire = True
```

5. If you want to process the situation when a camera is being plugged or unplugged, add [CameraPlugged](#) and [CameraUnplugged](#) events to your code.

6. For more information refer to the code of the *MultiGcam* sample.

The same general rules apply if you use DirectShow. You can run several instances of the *Video Capture Source Filter* in your system or in your application provided each instance is configured for a different camera. If you execute several copies of a DirectShow-based video capture application such as Microsoft's Amcap, each of them will attempt to automatically configure itself for a different camera and memorize corresponding camera settings in the system registry upon exiting. As an alternative, you can use the [FilterConfig](#) utility to register several GigE Vision DirectShow devices in the system, each one associated with a specific camera. For more information on using DirectShow refer to [DirectShow Reference Guide](#).

---

## 2.9 Distributing your application

If you create an application using *ActiveGige SDK*, your setup should include *ActiveGige redistributable package* available from A&B Software upon request. There is no need to provide a user of your application with the license file *activegige.lic*. In addition, you must ensure that the following files exist on the end-user's system:

mfc42.dll, v.6.0	MFCDLL shared library
atl.dll	Active template library
msvcrt.dll, msvcrt40.dll	C run-time libraries
oleaut32.dll	OLE property frame
regsvr32.exe	control registration utility

These files are part of the Windows operating system and they are typically located in the Windows system directory. Please refer to Microsoft's redistribution policy if you need to redistribute them. In addition [VC++ 2005 Redistributable Package](#) must be installed on the end-user's system.

When an *ActiveGige* based application is executed for the first time on an end-user's system, it will display the registration dialog (see [Installation](#)). You should instruct the user to provide you or your distributor with a Control ID displayed in the dialog. After the run-time license of the user is validated, the distributor will issue a unique serial number which will unlock the control on the user's machine.

## 3 ActiveX Reference

This chapter provides a complete list of properties, methods and events of the *ActiveGige* object.

[Properties](#)

[Methods](#)

[Events](#)

[Property Pages](#)

## 3.1 Properties

*ActiveGige* properties are divided into several categories and can be modified in a Property Window of your development environment, or in *ActiveGige* property pages.

### *GigE Source Category*

<a href="#">Camera</a>	Returns or sets the index of the currently selected camera
<a href="#">Privilege</a>	Selects or returns the level of the camera control Privilege
<a href="#">Acquire</a>	Enables/disables the continuous acquisition mode
<a href="#">Multicast</a>	Enables/disables the multicast mode
<a href="#">Heartbeat</a>	Returns or sets the heartbeat timeout for the camera, in milliseconds
<a href="#">Timeout</a>	Returns or sets the frame acquisition timeout in seconds
<a href="#">AcquisitionMode</a>	Returns or sets the acquisition mode of the currently selected camera
<a href="#">AcquisitionFrameCount</a>	Returns or sets the number of frames for the multi-acquisition mode
<a href="#">TestImageSelector</a>	Returns or sets the mode of generating internal test images
<a href="#">TriggerSelector</a>	Returns or sets the configuration of a trigger signal
<a href="#">Trigger</a>	Enables/disables the currently selected trigger
<a href="#">TriggerSource</a>	Returns or sets the signal source for the selected trigger
<a href="#">TriggerActivation</a>	Returns or sets the activation mode for the selected trigger
<a href="#">TriggerDelay</a>	Returns or sets the value of the trigger delay for the selected trigger

**GigE Format Category**

<a href="#">Format</a>	Returns or sets the currently selected pixel format
<a href="#">PacketSize</a>	Returns or sets the size of the image data packets in bytes
<a href="#">AcquisitionFrameRate</a>	Returns or sets the camera's frame rate
<a href="#">SizeX</a>	Returns or sets the width of the partial scan window
<a href="#">SizeY</a>	Returns or sets the height of the partial scan window
<a href="#">OffsetX</a>	Returns or sets the horizontal offset of the partial scan window
<a href="#">OffsetY</a>	Returns or sets the vertical offset of the partial scan window
<a href="#">BinningX</a>	Returns or sets the number of horizontal photo-cells to be combined together
<a href="#">BinningY</a>	Returns or sets the number of vertical photo-cells to be combined together
<a href="#">DecimationX</a>	Returns or sets the horizontal subsampling of the image
<a href="#">DecimationY</a>	Returns or sets the vertical subsampling of the image

***GigE Analog Category***

<a href="#"><u>ExposureMode</u></a>	Returns or sets the operation mode of the exposure
<a href="#"><u>ExposureTime</u></a>	Returns or sets the camera's exposure time
<a href="#"><u>ExposureAuto</u></a>	Returns or sets the automatic exposure mode
<a href="#"><u>GainSelector</u></a>	Returns or sets the channel to be configured by the gain adjustments
<a href="#"><u>Gain</u></a>	Returns or sets the camera's gain
<a href="#"><u>GainAuto</u></a>	Returns or sets the automatic gain mode
<a href="#"><u>BlackLevelSelector</u></a>	Returns or sets the channel to be configured by the black level adjustments
<a href="#"><u>BlackLevel</u></a>	Returns or sets the camera's black level
<a href="#"><u>BlackLevelAuto</u></a>	Returns or sets the automatic black level mode
<a href="#"><u>BalanceRatioSelector</u></a>	Returns or sets the channel to be configured by the balance ratio adjustment
<a href="#"><u>BalanceRatio</u></a>	Returns or sets the balance ratio of the selected color channel
<a href="#"><u>BalanceWhiteAuto</u></a>	Returns or sets the automatic white balance mode
<a href="#"><u>Gamma</u></a>	Returns or sets the camera's gamma value

---



***GigE Input/Output Category***

<a href="#"><u>LineSelector</u></a>	Returns or sets the physical I/O line to configure
<a href="#"><u>LineMode</u></a>	Returns or sets the input or output mode of the selected line
<a href="#"><u>LineSource</u></a>	Returns or sets the signal source for the selected line
<a href="#"><u>LineFormat</u></a>	Returns or sets the electrical format for the selected line
<a href="#"><u>LineInverter</u></a>	Returns or sets the inverted state for the selected line
<a href="#"><u>UserOutputSelector</u></a>	Selects the bit of the User Output register to be set
<a href="#"><u>UserOutputValue</u></a>	Returns or sets the value of the selected bit of the User Output register
<a href="#"><u>UserSetSelector</u></a>	Returns or assigns the user set to load, save or configure

***GigE Processing Category***

<a href="#"><u>Affinity</u></a>	Returns or sets the number of logical CPUs used in image processing algorithms
<a href="#"><u>Flip</u></a>	Flips the image horizontally or/and vertically
<a href="#"><u>Rotate</u></a>	Rotates the image at the specified angle
<a href="#"><u>Bayer</u></a>	Enables/disable the Bayer conversion and selects the conversion method
<a href="#"><u>BkgName</u></a>	Returns or sets the name prefix for background data
<a href="#"><u>BkgCorrect</u></a>	Returns or sets the background correction mode
<a href="#"><u>ColorCorrect</u></a>	Enables/disables the color correction mode
<a href="#"><u>HotPixelCorrect</u></a>	Enables/disables the hot pixel correction mode
<a href="#"><u>HotPixelLevel</u></a>	Returns or sets the threshold level to be used in the hot pixel correction mode
<a href="#"><u>Integrate</u></a>	Enables/disables the frame integration operation and selects the integration mode
<a href="#"><u>IntegrateWnd</u></a>	Returns or sets the number of frames to be used in the Integrate operation
<a href="#"><u>LUTMode</u></a>	Enables/disables the lookup talbe operation on the video frames
<a href="#"><u>LensCorrect</u></a>	Enables/disables the lens distortion correction

---

**GigE Display Category**

<a href="#">BackColor</a>	Returns or sets the background color of the control window
<a href="#">Display</a>	Enables/disables automatic live display in the control window
<a href="#">AntiTearing</a>	Enables/disables the anti-tearing feature for the live video display
<a href="#">MonitorSync</a>	Enables/disables the monitor synchronization mode
<a href="#">Edge</a>	Enables/disables the 3D look of the control window
<a href="#">ScrollBars</a>	Enables/disables the scroll bars in the control window
<a href="#">ScrollX</a>	Returns or sets the current horizontal scroll position for the video window
<a href="#">ScrollY</a>	Returns or sets the current vertical scroll position for the video window
<a href="#">Palette</a>	Returns or sets the number of the current live video palette
<a href="#">Magnification</a>	Returns or sets the magnification factor for the live video display
<a href="#">Font</a>	Returns or sets the font for drawing text in the image
<a href="#">Overlay</a>	Enables/disables the overlay
<a href="#">OverlayColor</a>	Returns or sets the color of the overlay graphics
<a href="#">OverlayFont</a>	Returns or sets the font for drawing text in the overlay
<a href="#">Alpha</a>	Shows/hides the alpha plane over the live video

### 3.1.1 Acquire

#### Description

Enables/disables the continuous acquisition mode. If the acquisition mode is enabled and the [Display](#) property is turned on, the live video will be displayed in the control window.

#### Syntax

[VB]

```
objActiveGige.Acquire [= Value]
```

[C/C++]

```
HRESULT get_Acquire ( bool *pValue );  
HRESULT put_Acquire( bool Value );
```

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pValue* [out,retval]

Pointer to the Boolean that is TRUE if the continuous acquisition is enable, or FALSE otherwise

*Value* [in]

Set to TRUE to enable the continuous acquisition, or set to FALSE otherwise

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example activates the continuous acquisition mode upon clicking the Start button:

```
Private Sub cmdStart_Click()  
On Error GoTo err_cmdStart  
    ActiveGige.Acquire=True  
    Exit Sub  
err_cmdStart:  
    MsgBox Err.Description  
End Sub
```

#### Remarks

If this property is set to TRUE, *ActiveGige* will continuously acquire the video into the internal image memory. The [FrameAcquired](#) event will be generated upon completing each frame. To access the content of the image memory, use [GetPixel](#), [GetLine](#), [GetImageWindow](#) and [GetImageData](#) methods.

Note that this property works in both the design and run-time modes. If your development environment

---

---

does not close the design view prior to going to the run-time mode, a conflict will occur between two *ActiveGige* objects trying to acquire the video from the same camera. In order to prevent this, it is recommended to keep the **Acquire** property set to FALSE in the design mode and turn it on by an explicit command in the program code (see the example above).

Use [AcquisitionMode](#) to set up the desired acquisition mode prior to setting **Acquire** to TRUE.

---

### 3.1.2 AcquisitionFrameCount

#### Description

Returns or sets the number of frames to be acquired in MultiFrame [AcquisitionMode](#)

#### Syntax

[VB]

```
objActiveGige.AcquisitionFrameCount [= Value]
```

[C/C++]

```
HRESULT get_AcquisitionFrameCount( long *pValue );  
HRESULT put_AcquisitionFrameCount( long Value );
```

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pValue* [out,retval]

Pointer to the current frame count value

*Value* [in]

The frame count value to be set

#### Return Values

S\_OK

Success

E\_NOINTERFACE

The feature is not available for the selected camera

E\_INVALIDARG

The value is out of range

E\_FAIL

Failure to set the feature value

#### Example

The following VB example activates the acquisition of 100 frames:

```
ActiveGige1.AcquisitionFrameCount=100  
ActiveGige1.AcquisitionMode = "MultiFrame"  
ActiveGige1.Acquire=TRUE
```

#### Remarks

This property is available only if the currently selected camera supports the *AcquisitionFrameCount* feature per GenICam standard.

---



### 3.1.3 AcquisitionFrameRate

#### Description

Returns or sets the absolute value of the camera frame rate in Hz or raw units.

#### Syntax

[VB]  
`objActiveGige.AcquisitionFrameRate [= Value]`

[C/C++]  
`HRESULT get_AcquisitionFrameRate( float *pValue );`  
`HRESULT put_AcquisitionFrameRate( float Value );`

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current frame rate value  
*Value* [in]  
The frame rate to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example sets the frame rate to 30:

```
ActiveGigel.AcquisitionFrameRate = 30
```

#### Remarks

The feature controls the rate at which frames are captured when the frame trigger is disabled. The valid range of the frame rate values can be obtained by the [GetAcquisitionFrameRateMin](#) and [GetAcquisitionFrameRateMax](#) methods.

Note that the property is available only if the currently selected camera supports one of the following GenICam features: *AcquisitionFrameRate*, *AcquisitionFrameRateAbs*, *AcquisitionFrameRateRaw*.

---



### 3.1.4 AcquisitionFrameRateAbs

#### Description

Returns or sets the absolute value of the camera frame rate in Hz. This property is deprecated and replaced by [AcquisitionFrameRate](#).

#### Syntax

[VB]

```
objActiveGige.AcquisitionFrameRateAbs [= Value]
```

[C/C++]

```
HRESULT get_AcquisitionFrameRateAbs( float *pValue );  
HRESULT put_AcquisitionFrameRateAbs( float Value );
```

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current frame rate value  
*Value* [in]  
The frame rate to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example sets the frame rate to 30:

```
ActiveGigel.AcquisitionFrameRateAbs = 30
```

#### Remarks

The feature controls the rate at which frames are captured when the frame trigger is disabled. The valid range of the frame rate values can be obtained by the [GetFeatureMin](#) and [GetFeatureMax](#) methods.

Note that the property is available only if the currently selected camera supports the *AcquisitionFrameRate* or *AcquisitionFrameRateAbs* feature per GenICam standard.

### 3.1.5 AcquisitionFrameRateRaw

#### Description

Returns or sets the relative value of the camera frame rate, typically in KHz. This property is deprecated and replaced by [AcquisitionFrameRate](#).

#### Syntax

[VB]  
`objActiveGige.AcquisitionFrameRateRaw [= Value]`

[C/C++]  
`HRESULT get_AcquisitionFrameRateRaw( long *pValue );`  
`HRESULT put_AcquisitionFrameRateRaw( long Value );`

#### Data Type [VB]

Integer

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current frame rate value  
*Value* [in]  
The frame rate to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example sets the frame rate to 30.3:

```
ActiveGigel.AcquisitionFrameRateRaw = 30300
```

#### Remarks

The feature controls the rate at which frames are captured when the frame trigger is disabled. The valid range of the frame rate values can be obtained by the [GetFeatureMin](#) and [GetFeatureMax](#) methods.

Note that the property is available only if the currently selected camera supports the *AcquisitionFrameRateRaw* feature.

---

### 3.1.6 AcquisitionMode

#### Description

Returns or sets the acquisition mode of the camera which defines how many frames will be captured when [Acquire](#) is set to true. Can be one of the following values:

*"SingleFrame"*  
One frame is captured  
*"MultiFrame"*  
The number of frames to capture is specified by [AcquisitionFrameCount](#)  
*"Continuous"*  
Frames are captured continuously until Acquire is set to FALSE

#### Syntax

[VB]  
`objActiveGige.AcquisitionMode [= Value]`

[C/C++]  
`HRESULT get_AcquisitionMode( bstr *pValue );`  
`HRESULT put_AcquisitionMode( bstr Value );`

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]  
Pointer to the string specifying the current acquisition mode  
*Value* [in]  
The acquisition mode to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is not part of the enumerated set  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example activates the acquisition of 100 frames:

```
ActiveGigel.AcquisitionFrameCount=100
ActiveGigel.AcquisitionMode = "MultiFrame"
ActiveGigel.Acquire=TRUE
```

**Remarks**

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

The desired acquisition mode must be selected before setting [Acquire](#) to TRUE.

---

### 3.1.7 Affinity

#### Description

Returns or sets the number of logical CPUs (cores) used in image processing algorithms.

#### Syntax

[VB]  
`objActiveGige.Affinity [= Value]`

[C/C++]  
`HRESULT get_Affinity ( short *pAffinity );`  
`HRESULT put_Affinity ( short Affinity );`

#### Data Type [VB]

Integer

#### Parameters [C/C++]

*pAffinity*[out,retval]  
Pointer to the currently selected Affinity.  
*Affinity* [in]  
The value of affinity to set. If 0, all available logical CPUs will be used.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB code instructs *ActiveGigE* to use 4 logical CPUs:

```
ActiveGig1.Affinity = 4
```

#### Remarks

This property lets you select the number of logical CPUs that will be used for image processing, such as [Bayer Conversion](#). By default *ActiveGigE* splits the processing tasks among all logical CPUs (cores) available on the system. You may want to reduce the amount of utilized cores for benchmarking or to free CPU resources for other tasks.

The default value of this property is zero which makes *ActiveGigE* to use to all available logical CPUs.

### 3.1.8 Alpha

#### Description

Shows/hides the alpha plane over the live video.

#### Syntax

[VB]  
`objActiveGige.Alpha [= Value]`

[C/C++]  
`HRESULT get_Alpha ( bool *pAlpha );`  
`HRESULT put_Alpha( bool Alpha );`

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pAlpha* [out,retval]  
Pointer to the Boolean that is TRUE if the alpha plane is enable, or FALSE otherwise  
*Alpha* [in]  
Set to TRUE to enable the alpha plane, or set to FALSE to disable it.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

The following VB example moves a transparent ellipse over the live video creating an animation effect:

```
Dim x As Integer
Dim y As Integer
Dim sy As Integer
Dim sx As Integer

Private Sub Form_Load()
ActiveGigel.Acquire=True
ActiveGigel.Alpha=True
End Sub

Private Sub ActiveGigel_FrameAcquired()
ActiveGigel.DrawAlphaEllipse x, 10, 400 + x, 200, 0, 0, 0, 0, 0
x = x + 1
y = y + 1
If x = sx Then
x = 0
End If
If y = sy Then
y = 0
End If
```

---

```
ActiveGigel.DrawAlphaEllipse x, 10, 400 + x, 200, 0, 255, 255, 0, 30  
End Sub
```

### Remarks

The **Alpha** feature lets you display multi-colored custom graphics and text over the live video image. Unlike the [Overlay](#) which has a solid color, objects in the alpha plane can be assigned different colors and opacity.

Setting this property to true causes ActiveGige to show the alpha plane. Disabling this property hides the alpha plane. Note that hiding the alpha plane does not erase graphics and text from it. To clear the alpha plane, use [DrawAlphaClear](#). For more information see [DrawAlphaPixel](#), [DrawAlphaLine](#), [DrawAlphaRectangle](#), [DrawAlphaEllipse](#), [DrawAlphaText](#).

Note that using the alpha plane may increase the CPU load of your application.

### 3.1.9 AntiTearing

#### Description

Enables/disables the anti-tearing feature for the live video display in the control window.

#### Syntax

[VB]  
`objActiveGige.AntiTearing [= Value]`

[C/C++]  
`HRESULT get_AntiTearing ( bool *pValue );`  
`HRESULT put_AntiTearing( bool Value );`

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the Boolean that is TRUE if the anti-tearing is enable, or FALSE otherwise  
*Value* [in]  
Set to TRUE to enable the anti-tearing, or set to FALSE to disable it

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example enables the anti-tearing feature:

```
ActiveGigel.AntiTearing = True  
MsgBox ActiveGigel.AntiTearing
```

#### Remarks

Tearing is a display artifact caused by the difference between the frame rate of the camera and the refresh rate of the monitor. Tearing can be noticed on the live video as horizontal splits between the upper and bottom parts of frames with highly dynamic contest. If **AntiTearing** is set to TRUE, the display update will be synchronized with the vertical blank period of the monitor thus eliminating tearing artifacts.

When **AntiTearing** is enable, the effective frame rate will be limited by the monitor's refresh rate. You can also experience a performance drop resulting in an additional CPU load.

*Note - some advanced graphic controllers may have the anti-tearing functionality implemented in the*

---



---

*hardware. If your system have such a graphic card, do not enable the **AntiTearing** property.*

---

### 3.1.10 BackColor

#### Description

Returns or sets the background color of the control window.

#### Syntax

[VB]

```
objActiveGige.BackColor [= Color]
```

[C/C++]

```
HRESULT get_BackColor(OLE_COLOR& pColor);  
HRESULT put_BackColor(OLE_COLOR Color);
```

#### Data Type [VB]

RGB color

#### Parameters [C/C++]

*pColor* [out,retval]

Pointer to the current background color

*Color* [in]

The background color to be set

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example sets the background color of the control to light gray:

```
ActiveGige1.BackColor = RGB (192, 192, 192)
```

#### Remarks

When the control displays a live video, its background is only visible if the size of the video display in the horizontal or vertical dimension is less than the size of the control window.

**BackColor** is an ambient property, therefore its default value is defined by the color of the form on which the control resides. Changing the background color of the container application will cause an equivalent change in the **BackColor** property of *ActiveGige*.

---

### 3.1.11 BalanceRatio

#### Description

Returns or sets the ratio (amplification factor) of the selected color component in absolute or raw units. Used for white balancing.

#### Syntax

[VB]  
`objActiveGige.BalanceRatio [= Value]`

[C/C++]  
`HRESULT get_BalanceRatio( float *pValue );`  
`HRESULT put_BalanceRatio( float Value );`

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the white balance ratio  
*Value* [in]  
The white balance ratio to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example adjusts white balance factors for different color channels:

```
ActiveGigel.BalanceRatioSelector="Red"  
ActiveGigel.BalanceRatio = 1.2  
ActiveGigel.BalanceRatioSelector="Green"  
ActiveGigel.BalanceRatio = 1.4  
ActiveGigel.BalanceRatioSelector="Blue"  
ActiveGigel.BalanceRatio = 1.
```

#### Remarks

This property changes the ratio of the selected color component relative to the base level of the

component. The valid range of the ratio values can be obtained by the [GetBalanceRatioMin](#) and [GetBalanceRatioMax](#) methods. Before using **BalanceRatio**, select a corresponding channel with [BalanceRatioSelector](#).

Note that the property is available only if the currently selected camera supports one of the following GenICam features: *BalanceRatio*, *BalanceRatioAbs*, *BalanceRatioRaw*.

---

### 3.1.12 BalanceRatioAbs

#### Description

Returns or sets the ratio (amplification factor) of the selected color component in absolute units. Used for white balancing. This property is deprecated and replaced by [BalanceRatio](#).

#### Syntax

[VB]  
`objActiveGige.BalanceRatioAbs [= Value]`

[C/C++]  
`HRESULT get_BalanceRatioAbs( float *pValue );`  
`HRESULT put_BalanceRatioAbs( float Value );`

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the white balance ratio  
*Value* [in]  
The white balance ratio to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example adjusts white balance factors for different color channels:

```
ActiveGigel.BalanceRatioSelector="Red"  
ActiveGigel.BalanceRatioAbs = 1.2  
ActiveGigel.BalanceRatioSelector="Green"  
ActiveGigel.BalanceRatioAbs = 1.4  
ActiveGigel.BalanceRatioSelector="Blue"  
ActiveGigel.BalanceRatioAbs = 1.
```

#### Remarks

This property changes the ratio of the selected color component relative to the base level of the

component. The valid range of the ratio values can be obtained by the [GetFeatureMin](#) and [GetFeatureMax](#) methods. Before using **BalanceRatioAbs**, select a corresponding channel with [BalanceRatioSelector](#).

Note that the property is available only if the currently selected camera supports the *BalanceRatio* or *BalanceRatioAbs* feature per GenICam standard.

---

### 3.1.13 BalanceRatioRaw

#### Description

Returns or sets the ratio (amplification factor) of the selected color component in relative units. Used for white balancing. This property is deprecated and replaced by [BalanceRatio](#).

#### Syntax

[VB]  
`objActiveGige.BalanceRatioRaw [= Value]`

[C/C++]  
`HRESULT get_BalanceRatioRaw( long *pValue );`  
`HRESULT put_BalanceRatioRaw( long Value );`

#### Data Type [VB]

Integer

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the white balance ratio  
*Value* [in]  
The white balance ratio to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example adjusts white balance factors for different color channels:

```
ActiveGigel.BalanceRatioSelector="Red"  
ActiveGigel.BalanceRatioRaw = 12  
ActiveGigel.BalanceRatioSelector="Green"  
ActiveGigel.BalanceRatioRaw = 14  
ActiveGigel.BalanceRatioSelector="Blue"  
ActiveGigel.BalanceRatioRaw = -10
```

#### Remarks

This property changes the ratio of the selected color component relative to the base level of the component. The valid range of the ratio values can be obtained by the [GetFeatureMin](#) and

[GetFeatureMax](#) methods. Before using **BalanceRatioRaw**, select a corresponding channel with [BalanceRatioSelector](#).

Note that the property is available only if the currently selected camera supports the *BalanceRatioRaw* feature.

---



### 3.1.14 BalanceRatioSelector

#### Description

Returns or sets the color channel to be controlled by [BalanceRatioAbs](#). Used for white balancing and can be one of the following values:

"Red"  
Gain adjustments will be applied to the red channel.  
"Green"  
Gain adjustments will be applied to the green channel.  
"Blue"  
Gain adjustments will be applied to the blue channel.  
"Y"  
Gain adjustments will be applied to Y channel.  
"U"  
Gain adjustments will be applied to U channel.  
"V"  
Gain adjustments will be applied to V channel.

#### Syntax

[VB]  
`objActiveGige.BalanceRatioSelector [= Value]`

[C/C++]  
`HRESULT get_BalanceRatioSelector( string *pValue );`  
`HRESULT put_BalanceRatioSelector( string Value );`

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]  
Pointer to the string specifying the current color channel for white balancing  
*Value* [in]  
The color channel to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is not part of the enumerated set  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the use of a combo box to switch between different white balance channels:

The following VB example adjusts white balance factors for different color channels:

```
ActiveGigE1.BalanceRatioSelector="Red"  
ActiveGigE1.BalanceRatioAbs = 1.2  
ActiveGigE1.BalanceRatioSelector="Green"  
ActiveGigE1.BalanceRatioAbs = 1.4  
ActiveGigE1.BalanceRatioSelector="Blue"  
ActiveGigE1.BalanceRatioAbs = 1.
```

### Remarks

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *BalanceRatioSelector* feature per GenICam standard.

---

### 3.1.15 BalanceWhiteAuto

#### Description

Returns or sets the automatic white balance control (AWB) mode. Can be one of the following values:

*"Off"*

White balance is manually controlled using [BalanceRatioSelector](#) and [BalanceRatioAbs](#).

*"Once"*

The camera sets the optimal balance ratios for each color channel and returns to the "Off" state

*"Continuous"*

The camera constantly performs white balancing

#### Syntax

[VB]  
`objActiveGige.GainAuto [= Value]`

[C/C++]  
`HRESULT get_GainAuto( string *pValue );`  
`HRESULT put_GainAuto( string Value );`

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]  
Pointer to the string specifying the white balance mode  
*Value* [in]  
The white balance mode to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is not part of the enumerated set  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the use of a button to perform a "one push" white balancing.

```
Private Sub Command1_Click()  
ActiveGige1.BalanceWhiteAuto = "Once"  
End Sub
```

**Remarks**

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *BalanceWhiteAuto* feature per GenICam standard.

---

### 3.1.16 Bayer

#### Description

Enables/disables the Bayer conversion mode and selects the Bayer conversion method.

#### Syntax

[VB]  
`objActiveGige.Bayer [= Value]`

[C/C++]  
`HRESULT get_Bayer ( short *pBayer );`  
`HRESULT put_Bayer( short Bayer );`

#### Data Type [VB]

Integer

#### Parameters [C/C++]

*pBayer* [out,retval]

Pointer to the ordinal number of the currently selected Bayer filter, zero if Bayer conversion is disabled.

*Bayer* [in]

Set to zero to disable Bayer conversion, or set to values from 1 to 3 to select one of the following Bayer filters:

- 1 - Nearest Neighbour filter. Missing pixels are substituted with adjacent pixels of the same color.
- 2 - Bilinear filter. Calculates the values of missing pixels by performing bilinear interpolation of the adjacent pixels.
- 3 - High Quality Linear filter. Calculates the values of missing pixels based on the Malvar, He and Cutler algorithm.
- 4 - Chrominance filter. Interpolates the values of missing pixels based on chrominance gradients.

#### Return Values

`S_OK`  
Success  
`E_FAIL`  
Failure.

#### Example

This VB example activates the bilinear Bayer filter:

```
ActiveGigel.Bayer = 1
```

#### Remarks

Bayer images are usually generated by a single-chip CCD camera, which has a color filter mosaic array (CFA) installed in front of the sensor. The most frequently used Bayer pattern has the following layout:

```
G B G B R
R G R G R
G B G B G
R G R G R
```

Each pixel value in a Bayer image corresponds to the intensity of the pixel behind the corresponding color filter. The conversion from such a grayscale image to RGB image is typically done on the camera itself, however some cameras (known as Bayer cameras) output a raw Bayer image unchanged. The Bayer transforms such an image into RGB image by performing real-time Bayer color reconstruction (demosaicing). The layout of the Bayer array is defined by the current [Format](#).

Note that the higher the ordinal number of the selected Bayer filter is, the higher the quality of the resulting image is and the higher the CPU load is. If the application speed is critical, consider using the Nearest Neighbour filter.

---

### 3.1.17 BinningX

#### Description

Returns or sets the number of horizontal photo-sensitive cells that must be combined together.

#### Syntax

[VB]  
`objActiveGige.BinningX [= Value]`

[C/C++]  
`HRESULT get_BinningX( long *pBinningX );`  
`HRESULT put_BinningX( long BinningX );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pBinningX* [out,retval]  
Pointer to the currently set horizontal binning  
*BinningX* [in]  
The horizontal binning to be set

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
Invalid property value.

#### Example

This VB example sets the horizontal binning to 2:

```
ActiveGigel.BinningX = 2  
MsgBox ActiveGigel.BinningX
```

#### Remarks

This property has a net effect of increasing the intensity (or signal to noise ratio) of the pixel value and reducing the horizontal size of the image. A value of 1 indicates that no horizontal binning is performed by the camera. Note that the property is available only if the currently selected camera supports the *BinningHorizontal* feature per GenICam standard. Any change in the **BinningX** property will cause [SizeX](#), and [OffsetX](#) to change accordingly.

### 3.1.18 BinningY

#### Description

Returns or sets the number of vertical photo-sensitive cells that must be combined together.

#### Syntax

[VB]  
`objActiveGige.BinningY [= Value]`

[C/C++]  
`HRESULT get_BinningY( long *pBinningY );`  
`HRESULT put_BinningY( long BinningY );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pBinningY* [out,retval]  
Pointer to the currently set vertical binning  
*BinningY* [in]  
The vertical binning to be set

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
Invalid property value.

#### Example

This VB example sets the vertical binning to 2:

```
ActiveGigel.BinningY = 2  
MsgBox ActiveGigel.BinningY
```

#### Remarks

This property has a net effect of increasing the intensity (or signal to noise ratio) of the pixel value and reducing the vertical size of the image. A value of 1 indicates that no vertical binning is performed by the camera. Note that the property is available only if the currently selected camera supports the *BinningVertical* feature per GenICam standard. Any change in the **BinningY** property will cause [SizeY](#), and [OffsetY](#) to change accordingly.

---



### 3.1.19 BkgCorrect

#### Description

Returns or sets the mode for the background correction. The values from 0 to 2 correspond to the following modes:

0 - *None*

No background correction is performed.

1 - *Dark*

The dark field (offset) background correction is applied to each acquired frame.

2 - *Flat*

The flat field (gain) background correction is applied to each acquired frame.

#### Syntax

[VB]

```
objActiveGige.BkgCorrect [ = Value ]
```

[C/C++]

```
HRESULT get_BkgCorrect( short *pCorrect );  
HRESULT put_BkgCorrect( short Correct );
```

#### Data Types [VB]

Integer

#### Parameters [C/C++]

*pCorrect* [out,retval]

Pointer to the current background correction mode

*Correct* [in]

Background correction mode to be set

#### Return Values

S\_OK

Success

#### Example

The following VB example sets the dark field correction mode:

```
ActiveGige1.BkgCorrect=1
```

#### Remarks

The dark field (offset) correction is used to compensate for the pixel-to-pixel difference in the dark current of the camera sensor. The correction is performed by subtracting background pixel values from corresponding pixel values of the original image. The dark field correction requires the dark field background image for the current video mode to have been stored on the hard drive.

The flat field (gain) correction is used to compensate for the variations in pixel-to-pixel sensitivity as

well as non-uniformity of the illumination. The correction algorithm is based on the following formula:

$$C_{x,y} = \frac{(I_{x,y} - B_{x,y})(W_{max} - B_{x,y})}{(W_{x,y} - B_{x,y})}$$

where

$I_{x,y}$  is a pixel value of the original image  
 $B_{x,y}$  is a pixel value of the dark field image  
 $W_{x,y}$  is a pixel value of the bright field image  
 $W_{max}$  is a maximum value of the bright field image  
 $C_{x,y}$  is a new pixel value of the corrected image

The flat field correction requires the bright field background image for the current video mode to have been stored on the hard drive. If the dark field image for the currently selected video mode does not exist on the hard drive, the value of 0 will be used in place of  $B$ .

If corresponding background images are not found on the hard drive, no correction will be performed. See [SaveBkg](#) for more details on preparing and saving background data.

---

### 3.1.20 BkgName

#### Description

Returns or sets the name prefix for the background data files.

#### Syntax

[VB]

```
objActiveGige.BkgName =[ Value ]
```

[C/C++]

```
HRESULT get_BkgName( bstr *pName );  
HRESULT put_BkgName( bstr Name );
```

#### Data Types [VB]

String

#### Parameters [C/C++]

*pName* [out, retval]

Pointer to the string containing the name prefix of the background data files.

*Name* [in]

The background name prefix to be set.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example assigns the name for background files and stores the current image as a dark field:

```
ActiveGige1.BkgName="MyBackground"  
ActiveGige1.SaveBkg (1)
```

#### Remarks

Background data are stored as raw image files in the Bkgnd subfolder of ActiveGige program directory (typically C:\Program Files\ActiveGige\Bkgnd). The full name of a background file is composed of the **BkgName** prefix and substrings indicating the camera index, video mode and background type. E.g., if the background name is assigned as in the example above and the camera operates in the video mode #3, the background image file will be stored under the name "MyBackground\_cam0\_mode3\_dark.raw"

### 3.1.21 BlackLevel

#### Description

Returns or sets the absolute or raw value of the camera's black level.

#### Syntax

[VB]  
`objActiveGige.BlackLevel [= Value]`

[C/C++]  
`HRESULT get_BlackLevel( float *pValue );`  
`HRESULT put_BlackLevel( float Value );`

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current black level value  
*Value* [in]  
The black level value to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the use of a scroll control for real-time adjustment of the black level value.

```
Private Sub Form_Load()  
HScroll1.Value = ActiveGige.BlackLevel  
HScroll1.Min = ActiveGige.GetBlackLevel  
HScroll1.Max = ActiveGige.GetBlackLevel  
ActiveGige.Acquire=True  
End Sub  
  
Private Sub HScroll1_Scroll()  
ActiveGige.BlackLevel = HScroll1.Value  
End Sub
```

---

**Remarks**

This property changes the black level of the video by adding a constant amount of luminance to each pixel. The valid property range can be retrieved by the [GetBlackLevelMin](#) and [GetBlackLevelMax](#) methods.

Note that the property is available only if the currently selected camera supports one of the following GenICam features: *BlackLevel*, *BlackLevelAbs*, *BlackLevelRaw*.

### 3.1.22 BlackLevelAbs

#### Description

Returns or sets the absolute value of the camera's black level. This property is deprecated and replaced by [BlackLevel](#).

#### Syntax

[VB]  
`objActiveGige.BlackLevelAbs [= Value]`

[C/C++]  
`HRESULT get_BlackLevelAbs( float *pValue );`  
`HRESULT put_BlackLevelAbs( float Value );`

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current black level value  
*Value* [in]  
The black level value to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the use of a scroll control for real-time adjustment of the gain value.

```
Private Sub Form_Load()  
HScroll1.Value = ActiveGige.BlackLevelAbs  
HScroll1.Min = ActiveGige.GetFeatureMin ("BlackLevelAbs")  
HScroll1.Max = ActiveGige.GetFeatureMax ("BlackLevelAbs")  
ActiveGige.Acquire=True  
End Sub  
  
Private Sub HScroll1_Scroll()  
ActiveGige.BlackLevelAbs = HScroll1.Value  
End Sub
```

---

**Remarks**

This property changes the black level of the video by adding a constant amount of luminance to each pixel. The valid property range can be retrieved by the [GetFeatureMin](#) and [GetFeatureMax](#) methods.

Note that the property is available only if the currently selected camera supports the *BlackLevel* or *BlackLevelAbs* feature per GenICam standard.

### 3.1.23 BlackLevelAuto

#### Description

Returns or sets the automatic black level control mode. Can be one of the following values:

"Off"

Black level is manually controlled using [BlackLevelRaw](#) or [BlackLevelAbs](#)

"Once"

The camera sets the optimal black level and returns to the "Off" state

"Continuous"

The camera constantly adjusts the black level

#### Syntax

[VB]

```
objActiveGige.BlackLevelControl [= Value]
```

[C/C++]

```
HRESULT get_BlackLevelAuto( bstr *pValue );  
HRESULT put_BlackLevelAuto( bstr Value );
```

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the string specifying the black level control mode

*Value* [in]

The black level control mode to be set

#### Return Values

S\_OK

Success

E\_NOINTERFACE

The feature is not available for the selected camera

E\_INVALIDARG

The value is not part of the enumerated set

E\_FAIL

Failure to set the feature value

#### Example

The following VB example demonstrates the use of a check box to switch between the manual and automatic black level control.

```
Private Sub Check1_Click()  
If Check1.Value = 1 Then  
ActiveGig1.BlackLevelAuto = "On"  
Else  
ActiveGig1.BlackLevelAuto = "Off"
```

---



```
End If  
End Sub
```

**Remarks**

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *BlackLevelAuto* feature per GenICam standard.

### 3.1.24 BlackLevelRaw

#### Description

Returns or sets the row value of the camera's black level. This property is deprecated and replaced by [BlackLevel](#).

#### Syntax

[VB]  
`objActiveGige.BlackLevelRaw [= Value]`

[C/C++]  
`HRESULT get_BlackLevelRaw( long *pValue );`  
`HRESULT put_BlackLevelRaw( long Value );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current black level value  
*Value* [in]  
The black level value to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the use of a scroll control for real-time adjustment of the camera's black level.

```
Private Sub Form_Load()  
ActiveGige1.Acquire=True  
HScroll11.Value = ActiveGige1.BlackLevelRaw  
HScroll11.Min = ActiveGige1.GetFeatureMin( "BlackLevelRaw" )  
HScroll11.Max = ActiveGige1.GetFeatureMax( "BlackLevelRaw" )  
End Sub  
  
Private Sub HScroll11_Scroll()  
ActiveGige1.BlackLevelRaw = HScroll11.Value  
End Sub
```

---

**Remarks**

This property changes the black level of the video by adding a constant amount of luminance to each pixel. The valid property range can be retrieved by the [GetFeatureMin](#) and [GetFeatureMax](#) methods.

Note that the property is available only if the currently selected camera supports the *BlackLevelRaw* feature per GenICam standard.

### 3.1.25 BlackLevelSelector

#### Description

Returns or sets the channel to be controlled by black level adjustments. Can be one of the following values:

- "All"  
Black level adjustments will be applied to all channels.
- "Red"  
Black level adjustments will be applied to the red channel.
- "Green"  
Black level adjustments will be applied to the green channel.
- "Blue"  
Black level adjustments will be applied to the blue channel.
- "Y"  
Black level adjustments will be applied to Y channel.
- "U"  
Black level adjustments will be applied to U channel.
- "V"  
Black level adjustments will be applied to V channel.

#### Syntax

[VB]

```
objActiveGige.BlackLevelSelector [= Value]
```

[C/C++]

```
HRESULT get_BlackLevelSelector( string *pValue );  
HRESULT put_BlackLevelSelector( string Value );
```

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the string specifying the black level channel setting

*Value* [in]

The black level channel to be set

#### Return Values

S\_OK

Success

E\_NOINTERFACE

The feature is not available for the selected camera

E\_INVALIDARG

The value is not part of the enumerated set

E\_FAIL

Failure to set the feature value

---

### Example

The following VB example demonstrates the use of a combo box to switch between different black level channels:

```
Private Sub Form_Load()  
Lst = ActiveGigel.GetEnumList("BlackLevelSelector")  
For i = 0 To UBound(Lst)  
    Combol.AddItem (Lst(i))  
Next  
Combol.ListIndex = ActiveGigel.BlackLevelSelector  
End Sub  
  
Private Sub Combol_Click()  
ActiveGigel.BlackLevelSelector = Combol.Text  
End Sub
```

### Remarks

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *BlackLevelSelector* feature per GenICam standard.

### 3.1.26 Camera

#### Description

Returns or sets the index of the currently selected camera. If no GigE Vision™ camera is connected to the system, the property will return -1.

#### Syntax

[VB]  
`objActiveGige.Camera [= Value]`

[C/C++]  
`HRESULT get_Camera( long *pCamera );`  
`HRESULT put_Camera( long Camera );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pCamera* [out, retval]  
Pointer to the camera's index.  
*Camera* [in]  
The index of the camera to be selected

#### Return Values

S\_OK  
Success  
E\_FAIL  
The selected camera doesn't comply with GigE Vision™ specifications  
E\_INVALIDARG  
The camera's index is out of range.

#### Example

This VB example initializes a combo box with camera names and uses it to switch between the cameras:

```
Private Sub Form_Load()  
    CamLst = ActiveGigel.GetCameraList  
    For i = 0 To UBound(CamLst)  
        Combol.AddItem (CamLst(i))  
    Next  
    Combol.ListIndex = 0  
    ActiveGigel.Acquire = True  
End Sub  
  
Private Sub Combol_Click()  
    ActiveGigel.Camera = Combol.ListIndex
```

---

End Sub

This VB example shows how to change camera assignment when using multiple *ActiveGigE* objects:

```
'Initial camera assignment
ActiveGig1.Camera=0
ActiveGig2.Camera=1
ActiveGig3.Camera=2
....
'Disconnect all cameras, then change camera assignment
ActiveGig1.Camera=-1
ActiveGig2.Camera=-1
ActiveGig3.Camera=-1
ActiveGig1.Camera=2
ActiveGig2.Camera=1
ActiveGig3.Camera=0
```

### Remarks

The value of the property is a zero-based index into the list of GigE Vision™ cameras discovered on the network. The list of camera names can be retrieved with [GetCameraList](#). If you use the property window of your development environment or *ActiveGigE*'s property pages, the **Camera** property field will be presented as a list box containing the indexes and names of all the connected cameras.

To disconnect the currently selected camera from *ActiveGigE* object, use -1 as the camera index. If your application uses multiple *ActiveGigE* objects connected to different cameras and you need to change the camera assignment, disconnect a camera from an object before assigning the camera to another object, or otherwise the new assignment will be denied.

### 3.1.27 ColorCorrect

#### Description

Enables/disables the color correction mode. Used in combination with [SetColorMatrix](#).

#### Syntax

[VB]

```
objActiveGige.ColorCorrect [= Value]
```

[C/C++]

```
HRESULT get_ColorCorrect ( bool *pCorrect );  
HRESULT put_ColorCorrect( bool Correct );
```

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pCorrect* [out,retval]

Pointer to the Boolean that is TRUE if the color correction is enabled, or FALSE otherwise

*Correct* [in]

Set to TRUE to enable the color correction, or set to FALSE to disable it

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example defines the color correction matrix and turns on the color correction:

```
ActiveGige1.SetColorMatrix 0.85, 0.1, 0.05, 0.05, 0.98, -0.03, 0.13, -0.15,  
1.02  
ActiveGige1.ColorCorrect=True
```

#### Remarks

Color correction is used to eliminate the overlap in the color channels caused by the fact that the light intended to be detected only by pixels of a certain color is partially seen by pixels of other colors. For more information see [SetColorMatrix](#).

---



### 3.1.28 DecimationX

#### Description

Returns or sets the horizontal subsampling of the image.

#### Syntax

[VB]

```
objActiveGige.DecimationX [= Value]
```

[C/C++]

```
HRESULT get_DecimationX( long *pDecimationX );  
HRESULT put_DecimationX( long DecimationX );
```

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pDecimationX* [out,retval]

Pointer to the currently set horizontal binning

*DecimationX* [in]

The horizontal decimation to be set

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid property value.

#### Example

This VB example sets the horizontal decimation to 2:

```
ActiveGigel.DecimationX = 2  
MsgBox ActiveGigel.DecimationX
```

#### Remarks

Depending on the camera, the subsampling can be implemented by pixel dropping or by pixel averaging/dropping. This has a net effect of reducing the horizontal size of the image by the specified decimation factor. A value of 1 indicates that no horizontal decimation is performed by the camera. Note that the property is available only if the currently selected camera supports the *DecimationHorizontal* feature per GenICam standard. Any change in the **DecimationX** property will cause [SizeX](#), and [OffsetX](#) to change accordingly.

### 3.1.29 DecimationY

#### Description

Returns or sets the vertical subsampling of the image.

#### Syntax

[VB]  
`objActiveGige.DecimationY [= Value]`

[C/C++]  
`HRESULT get_DecimationY( long *pDecimationY );`  
`HRESULT put_DecimationY( long DecimationY );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pDecimationY* [out,retval]  
Pointer to the currently set vertical binning  
*DecimationY* [in]  
The vertical decimation to be set

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
Invalid property value.

#### Example

This VB example sets the vertical decimation to 2:

```
ActiveGigel.DecimationY = 2  
MsgBox ActiveGigel.DecimationY
```

#### Remarks

Depending on the camera, the subsampling can be implemented by pixel dropping or by pixel averaging/dropping. This has a net effect of reducing the vertical size of the image by the specified decimation factor. A value of 1 indicates that no vertical decimation is performed by the camera. Note that the property is available only if the currently selected camera supports the *Decimationvertical* feature per GenICam standard. Any change in the **DecimationY** property will cause [SizeY](#), and [OffsetY](#) to change accordingly.

---

### 3.1.30 Display

#### Description

Enables/disables live display in the control window. When this property is enabled, each frame will be automatically displayed upon acquisition.

#### Syntax

[VB]

```
objActiveGige.Display [= Value]
```

[C/C++]

```
HRESULT get_Display ( bool *pDisplay );  
HRESULT put_Display( bool Display );
```

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pDisplay* [out,retval]

Pointer to the Boolean that is TRUE if the live display is enable, or FALSE otherwise

*Display* [in]

Set to TRUE to enable the live display, or set to FALSE to disable it

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example disables the live display and uses the [FrameAcquired](#) event to invert pixel value in the bottom left corner of the current frame and display the processed frame in real time.

```
Private Sub Form_Load()  
ActiveGigel.Display = False  
ActiveGigel.Acquire = True  
End Sub  
  
Private Sub ActiveGigel_FrameAcquired()  
a = ActiveGigel.GetImageData  
For x = 0 To 200  
For y = 0 To 200  
a(x, y) = 255 - a(x, y)  
Next  
Next  
ActiveGigel.Draw  
End Sub
```

**Remarks**

When you create a new instance of the control, this property is enabled by default. You should disable the live display when you want to perform real time image processing and display the processed image in the control window. After the processing is done, the current frame should be displayed by calling the [Draw](#) method.

When using [DirectShow Video Capture Filter](#), this property controls the internal conversion to RGB24 format which is required by image data access and image analysis methods, such as [GetImageData](#), [GetComponentData](#), [GetImageLine](#), [GetImageWindow](#), [GetHistogram](#), [GetImageStat](#) and others. Setting **Display** to FALSE will disable the conversion and provide original video formats on the output pin.

Note that disabling the live display does not turn off the image conversion performed on each frame. In order to disable the image conversion pipeline and receive only raw image data, set the [Magnification](#) property to -2.

---

### 3.1.31 Edge

#### Description

Enables/disables the 3D look (sunken edge) of the control window.

#### Syntax

[VB]

```
objActiveGige.Edge [= Value]
```

[C/C++]

```
HRESULT get_Edge ( bool *pEdge );  
HRESULT put_Edge( bool Edge );
```

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pEdge* [out,retval]

Pointer to the Boolean that is TRUE if the 3D look is enable, or FALSE otherwise

*Edge* [in]

Set to TRUE to enable the 3D look, or set to FALSE to disable it

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example enables the 3D look on the control window:

```
ActiveGigel.Edge = True  
MsgBox ActiveGigel.Edge
```

#### Remarks

If this property is set to TRUE, the sunken edge will appear around the border of the *ActiveGige* control window.

### 3.1.32 ExposureAuto

#### Description

Returns or sets the automatic exposure (AE) mode. Can be one of the following values:

"Off"

Exposure is manually controlled using [ExposureTimeRaw](#) or [ExposureTimeAbs](#)

"Once"

The camera sets the optimal exposure level and returns to the "Off" state

"Continuous"

The camera constantly adjusts the exposure level

#### Syntax

[VB]

```
objActiveGige.ExposureAuto [= Value]
```

[C/C++]

```
HRESULT get_ExposureAuto( bstr *pValue );
HRESULT put_ExposureAuto( bstr Value );
```

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the string specifying the exposure control mode

*Value* [in]

The exposure control mode to be set

#### Return Values

S\_OK

Success

E\_NOINTERFACE

The feature is not available for the selected camera

E\_INVALIDARG

The value is not part of the enumerated set

E\_FAIL

Failure to set the feature value

#### Example

The following VB example demonstrates the use of a check box to switch between the manual and automatic exposure control.

```
Private Sub Check1_Click()  
If Check1.Value = 1 Then  
ActiveGige1.ExposureAuto = "On"  
Else  
ActiveGige1.ExposureAuto = "Off"
```

---

```
End If  
End Sub
```

**Remarks**

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *ExposureAuto* feature per GenICam standard.

### 3.1.33 ExposureMode

#### Description

Returns or sets the operation mode of the exposure (shutter). Can be one of the following values:

"Off"

Disables the exposure and lets the shutter open

"Timed"

Exposure time is set using [ExposureTimeRaw](#), [ExposureTimeAbs](#) or [ExposureAuto](#)

"TriggerWidth"

The camera uses the width of the current Frame or Line trigger signal pulse to control the exposure time.

"TriggerControlled"

The camera uses one or more trigger signals to control the exposure time independently from the Frame or Line triggers. See [TriggerSelector](#) for more details

#### Syntax

[VB]

```
objActiveGige.ExposureMode [= Value]
```

[C/C++]

```
HRESULT get_ExposureMode( bstr *pValue );
```

```
HRESULT put_ExposureMode( bstr Value );
```

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the string specifying the exposure operational mode

*Value* [in]

The exposure operational mode to be set

#### Return Values

S\_OK

Success

E\_NOINTERFACE

The feature is not available for the selected camera

E\_INVALIDARG

The value is not part of the enumerated set

E\_FAIL

Failure to set the feature value

#### Example

The following VB example demonstrates the use of a combo box to switch between exposure operational modes.

```
Private Sub Form_Load()  
Lst = ActiveGigel.GetEnumList("ExposureMode")
```



```
For i = 0 To UBound(Lst)
    Combol.AddItem (Lst(i))
Next
Combol.ListIndex = ActiveGigel.ExposureMode
End Sub

Private Sub Combol_Click()
    ActiveGigel.ExposureMode = Combol.Text
End Sub
```

**Remarks**

The property is available only if the currently selected camera supports the *ExposureMode* feature per GenICam standard.

### 3.1.34 ExposureTime

#### Description

Returns or sets the value of the camera exposure time in microseconds or raw units.

#### Syntax

[VB]  
`objActiveGige.ExposureTime [= Value]`

[C/C++]  
`HRESULT get_ExposureTime( float *pValue );`  
`HRESULT put_ExposureTime( float Value );`

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current exposure value  
*Value* [in]  
The exposure value to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the use of a scroll control for real-time adjustment of the exposure time.

```
Private Sub Form_Load()  
HScroll1.Value = ActiveGig1.ExposureTime  
HScroll1.Min = ActiveGig1.GetExposureMin  
HScroll1.Max = ActiveGig1.GetExposureMax  
ActiveGig1.Acquire=True  
End Sub  
  
Private Sub HScroll1_Scroll()  
ActiveGig1.ExposureTime = HScroll1.Value  
End Sub
```

---

**Remarks**

This property changes the integration time of the camera's sensor to a sub-value of the frame period. The valid range of the exposure values can be obtained by the [GetExposureTimeMin](#) and [GetExposureTimeMax](#) methods.

Note that the property is available only if the currently selected camera supports one of the following GenICam features: *ExposureTime*, *ExposureTimeAbs*, *ExposureTimeRaw*.

### 3.1.35 ExposureTimeAbs

#### Description

Returns or sets the value of the camera exposure time in microseconds. This property is deprecated and replaced by [ExposureTime](#).

#### Syntax

[VB]  
`objActiveGige.ExposureTimeAbs [= Value]`

[C/C++]  
`HRESULT get_ExposureTimeAbs( float *pValue );`  
`HRESULT put_ExposureTimeAbs( float Value );`

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current exposure value  
*Value* [in]  
The exposure value to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the use of a scroll control for real-time adjustment of the exposure time.

```
Private Sub Form_Load()  
HScroll1.Value = ActiveGige1.ExposureTimeAbs  
HScroll1.Min = ActiveGige1.GetFeatureMin ("ExposureTimeAbs")  
HScroll1.Max = ActiveGige1.GetFeatureMax ("ExposureTimeAbs")  
ActiveGige1.Acquire=True  
End Sub  
  
Private Sub HScroll1_Scroll()  
ActiveGige1.ExposureTimeAbs = HScroll1.Value
```

---

---

End Sub

### Remarks

This property changes the integration time of the camera's sensor to a sub-value of the frame period. The valid range of the exposure values can be obtained by the [GetFeatureMin](#) and [GetFeatureMax](#) methods.

Note that the property is available only if the currently selected camera supports the *ExposureTime* or *ExposureTimeAbs* feature per GenICam standard.

---

### 3.1.36 ExposureTimeRaw

#### Description

Returns or sets the value of the camera exposure in relative units. This property is deprecated and replaced by [ExposureTime](#).

#### Syntax

[VB]  
`objActiveGige.ExposureTimeRaw [= Value]`

[C/C++]  
`HRESULT get_ExposureTimeRaw( long *pValue );`  
`HRESULT put_ExposureTimeRaw( long Value );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current exposure value  
*Value* [in]  
The exposure value to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the use of a scroll control for real-time adjustment of the exposure value.

```
Private Sub Form_Load()  
HScroll1.Value = ActiveGige.ExposureTimeRaw  
HScroll1.Min = ActiveGige.GetFeatureMin ("ExposureTimeRaw")  
HScroll1.Max = ActiveGige.GetFeatureMax ("ExposureTimeRaw")  
ActiveGige.Acquire=True  
End Sub  
  
Private Sub HScroll1_Scroll()  
ActiveGige.ExposureTimeRaw = HScroll1.Value  
End Sub
```

---

**Remarks**

This property changes the integration time of the camera's sensor to a sub-value of the frame period. Depending on the camera model, the raw exposure value can be directly or inversely related to the integration time absolute value. The valid range of the exposure values can be obtained by the [GetFeatureMin](#) and [GetFeatureMax](#) methods.

Note that the property is available only if the currently selected camera supports the *ExposureTimeRaw* feature per GenICam standard.

### 3.1.37 Flip

#### Description

Returns or sets the horizontal and vertical flipping of the image. The values from 0 to 3 correspond to the following flipping conditions:

- 0 - *None*  
No image flipping is performed.
- 1 - *Horizontal*  
The image is flipped horizontally.
- 2 - *Vertical*  
The image is flipped vertically.
- 3 - *Both*  
The image is flipped horizontally and vertically.

#### Syntax

[VB]  
`objActiveGige.Flip [= Value]`

[C/C++]  
`HRESULT get_Flip( long *pFlip );`  
`HRESULT put_Flip( long Flip );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

- pFlip* [out,retval]  
Pointer to the ordinal number of the currently selected flipping condition.
- Palette* [in]  
The flipping condition to be set.

#### Return Values

- S\_OK  
Success
- E\_FAIL  
Failure.
- E\_INVALIDARG  
Invalid property value.

#### Example

This VB example demonstrates the use of a combo box for flipping the live video:

```
Private Sub Form_Load()  
ActiveGigel.Acquire = True  
Combo1.AddItem ("None")  
Combo1.AddItem ("Horizontal")
```

---



```
Combol.AddItem ("Vertical")
Combol.AddItem ("Diagonal")
Combol.ListIndex = 0
End Sub

Private Sub Combol_Click()
ActiveGigel.Flip = Combol.ListIndex
End Sub
```

### Remarks

Flipping affects the way the video is displayed in the control window as well as actual order of pixels in the image frame. If you apply one of the flipping options, the data returned by [GetImageData](#) and other data access methods will be flipped as well.

Note that the flip operation has precedence over [Rotate](#). If both **Flip** and **Rotate** properties are non-zero, the frame will be flipped first and the resulting image rotated.

### 3.1.38 Font

#### Description

Returns or sets the font for [DrawText](#).

#### Syntax

[VB]  
`objActiveGige.Font [= Font]`

[C/C++]  
`HRESULT get_Font(IFontDisp* *pFont);`  
`HRESULT put_Font(IFontDisp* Font);`

#### Data Type [VB]

StdFont

#### Parameters [C/C++]

*pFont* [out,retval]  
Pointer to the *IFontDisp* interface object corresponding to the current overlay font  
*Font* [in]  
*IFontDisp* interface object corresponding to the overlay font to be set

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

The following VB example inserts two string of text in the live video:

```
Private Sub ActiveGige1_FrameAcquired()  
Dim Font1 As New StdFont  
Font1.Name = "Arial"  
Font1.Size = 30  
Font1.Bold = True  
ActiveGige1.Font = Font1  
ActiveGige1.DrawText 10, 100, "ActiveGige", 255, 0, 0  
Font2.Name = "Arial"  
Font2.Size = 20  
Font2.Italic = True  
ActiveGige1.Font = Font2  
ActiveGige1.DrawText 10, 200, "The most powerful GigE Vision SDK", 0, 0,  
255  
End Sub
```

#### Remarks

Also see [DrawText](#).

---

### 3.1.39 Format

#### Description

Returns or sets the index of the currently selected pixel format.

#### Syntax

[VB]  
`objActiveGige.Format [= Value]`

[C/C++]  
`HRESULT get_Format( long *pFormat );`  
`HRESULT put_Format( long Format );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pFormat* [out,retval]  
Pointer to the format's index  
*Format* [in]  
The index of the format to be selected

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
The index is out of range

#### Example

This VB example initializes a combo box with the descriptions of available pixel formats and uses it to select a specific format:

```
Private Sub Form_Load()  
FormatLst = ActiveGigel.GetFormatList  
For i = 0 To UBound(FormatLst)  
Combol.AddItem (FormatLst(i))  
Next  
Combol.ListIndex = 0  
ActiveGigel.Acquire = True  
End Sub  
  
Private Sub Combol_Click()  
ActiveGigel.Format = Combol.ListIndex  
End Sub
```

## Remarks

The value of the property is a zero-based index into the list of pixel formats supported by the currently selected [Camera](#). The list of formats can be retrieved with [GetFormatList](#). If you use the property window of your development environment or *ActiveGigE*'s property pages, the **Format** property field will be presented as a list box containing all supported video formats. The format list is built in the ascending order by browsing through all pixel formats supported by the current camera.

*ActiveGigE* supports the following GigE Vision™ pixel formats:

Pixel Format	Description	Bits per pixel
Mono8	8-bit monochrome unsigned	8
Mono8s (=Mono8Signed)	8-bit monochrome signed	8
Mono10	10-bit monochrome unpacked	16
Mono10Packed	10-bit monochrome packed	12
Mono12	12-bit monochrome unpacked	16
Mono12Packed	12-bit monochrome packed	12
Mono14	14-bit monochrome unpacked	14
Mono16	16-bit monochrome	16
Bayer**8	8-bit raw Bayer	8
Bayer**10	10-bit raw Bayer unpacked	16
Bayer**12	12-bit raw Bayer unpacked	16
Bayer**10Packed	10-bit raw Bayer packed	12
Bayer**12Packed	12-bit raw Bayer packed	12
RGB8 (=RGB8Packed)	24-bit RGB color	24
BGR8 (=BGR8Packed)	24-bit BGR color	24
RGBa8 (=RGBA8Packed)	24-bit RGB color with alpha channel	32
BGRa8 (=BGRA8Packed)	24-bit BGR color with alpha channel	32
RGB10Packed (=RGB10Packed)	30-bit RGB color	48
BGR10Packed (=BGR10Packed)	30-bit BGR color	48
RGB12Packed (=RGB12Packed)	36-bit RGB color	48
BGR12Packed (=BGR12Packed)	36-bit BGR color	48
BGR10V1Packed	30-bit BGR color packed	32
BGR12V1Packed	36-bit BGR color packed	36
YUV411_8_UYVY (=YUV411Packed)	12-bit YUV color	12
YUV422_8_UYVY (=YUV422Packed)	16-bit YUV color	16
YUV8_YUV (=YUV444Packed)	24-bit YUV color (YUV 4:4:4)	24
RGB8Planar	24-bit RGB in form of three 8-bit planes	24
RGB10Planar	30-bit BGR in form of three 16-bit planes	48
RGB12Planar	36-bit BGR in form of three 16-bit planes	48
RGB16Planar	48-bit BGR in form of three 16-bit planes	48
Coord3D_ABC16	48-bit 3D-cloud (16-bit XYZ coordinates)	48
Coord3D_ABC32f	96-bit 3D-cloud (floating 32-bit XYZ coord)	96

---

*Note - \*\* in the Bayer format names stands for GR, RG, GB or BG type of [Bayer](#) layout.*

*Note - alternative format names in parenthesis represent older notations from GEV 1.x specifications.*

If your camera supports JPEG or H.264 compression per GEV 2.0 standard, you can select a corresponding compression mode by changing the *ImageCompressionMode* GenICam feature. Compression quality settings are typically available on such cameras via *ImageCompressionQuality* and *ImageCompressionBitrate* features. If JPEG or H.264 compression mode is activated on the camera, **Format** will be automatically reset to Mono8 or RGB8, and *ActiveGigE* will decode incoming compressed frames on the fly.

Depending on your camera, some manufacturer-specific formats can also be supported by *ActiveGige*.

For more information refer to "*GigE Vision Video Streaming and Device Control Over Ethernet Standard*" published by the Automated Imaging Association.

---

### 3.1.40 Gain

#### Description

Returns or sets the value of the camera gain in absolute or raw units.

#### Syntax

[VB]  
`objActiveGige.Gain [= Value]`

[C/C++]  
`HRESULT get_Gain( float *pValue );`  
`HRESULT put_Gain( float Value );`

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current exposure value  
*Value* [in]  
The exposure value to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the use of a scroll control for real-time adjustment of the gain value.

```
Private Sub Form_Load()  
HScroll1.Value = ActiveGige1.Gain  
HScroll1.Min = ActiveGige1.GetGain  
HScroll1.Max = ActiveGige1.GetGain  
ActiveGige1.Acquire=True  
End Sub  
  
Private Sub HScroll1_Scroll()  
ActiveGige1.Gain = HScroll1.Value  
End Sub
```

---

**Remarks**

This property changes the camera's video signal amplification. The valid range of the gain values can be obtained by the [GetGainMin](#) and [GetGainMax](#) methods.

Note that the property is available only if the currently selected camera supports one of the following GenICam features: *Gain*, *GainAbs*, *GainRaw*.

### 3.1.41 GainAbs

#### Description

Returns or sets the value of the camera gain in absolute units. This property is deprecated and replaced by [Gain](#).

#### Syntax

[VB]  
`objActiveGige.GainAbs [= Value]`

[C/C++]  
`HRESULT get_GainAbs( float *pValue );`  
`HRESULT put_GainAbs( float Value );`

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current exposure value  
*Value* [in]  
The exposure value to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the use of a scroll control for real-time adjustment of the gain value.

```
Private Sub Form_Load()  
HScroll1.Value = ActiveGige.GainAbs  
HScroll1.Min = ActiveGige.GetFeatureMin ("GainAbs")  
HScroll1.Max = ActiveGige.GetFeatureMax ("GainAbs")  
ActiveGige.Acquire=True  
End Sub  
  
Private Sub HScroll1_Scroll()  
ActiveGige.GainAbs = HScroll1.Value  
End Sub
```

---



**Remarks**

This property changes the camera's video signal amplification. The valid range of the gain values can be obtained by the [GetFeatureMin](#) and [GetFeatureMax](#) methods.

Note that the property is available only if the currently selected camera supports the *Gain* or *GainAbs* feature per GenICam standard.

### 3.1.42 GainAuto

#### Description

Returns or sets the automatic gain control (AGC) mode. Can be one of the following values:

"Off"

Gain is manually controlled using [GainRaw](#) or [GainAbs](#)

"Once"

The camera sets the optimal gain level and returns to the "Off" state

"Continuous"

The camera constantly adjusts the gain level

#### Syntax

[VB]

```
objActiveGige.GainAuto [= Value]
```

[C/C++]

```
HRESULT get_GainAuto( string *pValue );
HRESULT put_GainAuto( string Value );
```

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the string specifying the gain control mode

*Value* [in]

The gain control mode to be set

#### Return Values

S\_OK

Success

E\_NOINTERFACE

The feature is not available for the selected camera

E\_INVALIDARG

The value is not part of the enumerated set

E\_FAIL

Failure to set the feature value

#### Example

The following VB example demonstrates the use of a check box to switch between the manual and automatic gain control.

```
Private Sub Check1_Click()  
If Check1.Value = 1 Then  
ActiveGige1.GainAuto = "On"  
Else  
ActiveGige1.GainAuto = "Off"
```

```
End If  
End Sub
```

**Remarks**

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *GainAuto* feature per GenICam standard.

### 3.1.43 GainRaw

#### Description

Returns or sets the row value of the gain. This property is deprecated and replaced by [Gain](#).

#### Syntax

[VB]  
`objActiveGige.GainRaw [= Value]`

[C/C++]  
`HRESULT get_GainRaw( long *pValue );`  
`HRESULT put_GainRaw( long Value );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current gain value  
*Value* [in]  
The gain value to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the use of a scroll control for real-time adjustment of the camera's gain.

```
Private Sub Form_Load()  
ActiveGigel.Acquire=True  
HScroll1.Value = ActiveGigel.GainRaw  
HScroll1.Min = ActiveGigel.GetFeatureMin("GainRaw")  
HScroll1.Max = ActiveGigel.GetFeatureMax("GainRaw")  
End Sub  
  
Private Sub HScroll1_Scroll()  
ActiveGigel.GainRaw = HScroll1.Value  
End Sub
```

---

**Remarks**

This property changes the camera's video signal amplification. The valid property range can be retrieved by the [GetFeatureMin](#) and [GetFeatureMax](#) methods.

Note that the property is available only if the currently selected camera supports the *GainRaw* feature per GenICam standard.

### 3.1.44 GainSelector

#### Description

Returns or sets the channel to be controlled by gain adjustments. Can be one of the following values:

- "All"  
Gain adjustments will be applied to all channels.
- "Red"  
Gain adjustments will be applied to the red channel.
- "Green"  
Gain adjustments will be applied to the green channel.
- "Blue"  
Gain adjustments will be applied to the blue channel.
- "Y"  
Gain adjustments will be applied to Y channel.
- "U"  
Gain adjustments will be applied to U channel.
- "V"  
Gain adjustments will be applied to V channel.

#### Syntax

[VB]  
`objActiveGige.GainSelector [= Value]`

[C/C++]  
`HRESULT get_GainSelector( string *pValue );`  
`HRESULT put_GainSelector( string Value );`

#### Data Type [VB]

String

#### Parameters [C/C++]

- pValue* [out, retval]  
Pointer to the string specifying the gain selector setting
- Value* [in]  
The gain selection to be set

#### Return Values

- S\_OK  
Success
- E\_NOINTERFACE  
The feature is not available for the selected camera
- E\_INVALIDARG  
The value is not part of the enumerated set
- E\_FAIL  
Failure to set the feature value

#### Example

---

The following VB example demonstrates the use of a combo box to switch between different gain channels:

```
Private Sub Form_Load()  
Lst = ActiveGigel.GetEnumList("GainSelector")  
For i = 0 To UBound(Lst)  
Combol.AddItem (Lst(i))  
Next  
Combol.ListIndex = ActiveGigel.GainSelector  
End Sub  
  
Private Sub Combol_Click()  
ActiveGigel.GainSelector = Combol.Text  
End Sub
```

### Remarks

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *GainSelector* feature per GenICam standard.

### 3.1.45 Gamma

#### Description

Returns or sets the gamma correction factor for the image.

#### Syntax

[VB]  
`objActiveGige.Gamma [= Value]`

[C/C++]  
`HRESULT get_Gamma( float *pValue );`  
`HRESULT put_Gamma( float Value );`

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current gamma factor  
*Value* [in]  
The gamma factor to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example sets the Gamma factor to 1.2

```
ActiveGige1.Gamma = 1.2
```

#### Remarks

This property changes the gamma correction factor of the camera's circuitry. The gamma correction modifies an image by applying standard, nonlinear gamma curves to the intensity scale. Increasing the gamma value will lighten the video and increase the contrast in its darker areas. The valid range of the gamma values can be obtained by the [GetFeatureMin](#) and [GetFeatureMax](#) methods.

Note that the property is available only if the currently selected camera supports the *Gamma* feature.

---





### 3.1.46 Heartbeat

#### Description

Returns or sets the heartbeat timeout for the camera, in milliseconds.

#### Syntax

[VB]  
`objActiveGige.Heartbeat [= Value]`

[C/C++]  
`HRESULT get_Heartbeat( long *pValue );`  
`HRESULT put_Heartbeat( long Value );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current heartbeat timeout of the camera.  
*Value* [in]  
The heartbeat timeout to be set. The minimum value is 500 ms.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
Invalid property value.

#### Example

This VB example sets the heartbeat timeout for the camera into 10 sec:

```
ActiveGigel.Heartbeat=10000
```

#### Remarks

The heartbeat mechanism allows a GigE Vision application to maintain its connection with the camera by periodically exchanging data with it. The heartbeat timeout indicates the maximum interval during which the camera will expect the primary application to send a data request. If the application remains inactive for the period larger than the heartbeat timeout, the camera will close its connection with the application and make itself available to other applications.

The **Heartbeat** property should not be confused with the Heartbeat setting available in the [IP configuration utility](#). The latter one is provided for setting the rate at which ActiveGigE-based

---

---

applications send heartbeat signals to all cameras, while the **Heartbeat** property sets the timeout period of the current camera by modifying/reading its heartbeat timeout register.

The default value of the heartbeat timeout for a typical GigE Vision camera is 3000 ms. Setting the heartbeat timeout to a large value is recommended for debugging a GigE Vision application. This will prevent the camera from closing its connection with the application during long periods of inactivity related to the debugging process.

### 3.1.47 HotPixelCorrect

#### Description

Enables/disables the hot pixel correction mode. Used in combination with [HotPixelLevel](#).

#### Syntax

[VB]

```
objActiveGige.Integrate [= Value]
```

[C/C++]

```
HRESULT get_HotPixelCorrect ( bool *pCorrect );  
HRESULT put_HotPixelCorrect( bool Correct );
```

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pCorrect* [out,retval]

Pointer to the Boolean that is TRUE if the hot pixel correction is enabled, or FALSE otherwise

*Correct* [in]

Set to TRUE to enable the hot pixel correction, or set to FALSE to disable it

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example activates the hot pixel correction mode with a 15% threshold:

```
ActiveGige1.HotPixelLevel = 15  
ActiveGige1.HotPixelCorrect = True
```

#### Remarks

Hot pixels are individual pixels that appear much brighter than the rest of an image. They are associated with elements on a camera sensor that have higher than normal rates of charge leakage. Hot pixels are especially noticable in a low-light situation when the shutter and/or gain are set to high values.

The hot pixel correction algorithm is based on the "top hat" technique that treats a pixel intensity as an elevation of a surface. Any pixel that protrudes through the "crown of the hat" which height is defined by a value of [HotPixelLevel](#) is considered to be noise and replaced by the mean value under the "brim of the hat". This effectively removes hot pixels from the image.

---

### 3.1.48 HotPixelLevel

#### Description

Returns or sets the threshold level to be used in the [hot pixel correction](#) mode, in percents.

#### Syntax

[VB]

```
objActiveGige.HotPixelLevel [= Value]
```

[C/C++]

```
HRESULT get_HotPixelLevel( long *pValue );  
HRESULT put_HotPixelLevel( long Value );
```

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pValue* [out,retval]

Pointer to the currently selected threshold level for the hot pixel correction mode.

*Value* [in]

The threshold level to be set.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid property value.

#### Example

This VB example activates the hot pixel correction mode with a 15% threshold:

```
ActiveGigel.HotPixelLevel = 15  
ActiveGigel.HotPixelCorrect = True
```

#### Remarks

The hot pixel level indicates the minimum difference in the intensity between a pixel and its neighborhood in order for the pixel to be considered hot. The value of the hot pixel level is given in percents of the maximum intensity for the given type of image. If the pixel falls into a hot-pixel category, its value will be replaced by the average intensity of the adjacent pixels.

### 3.1.49 Integrate

#### Description

Enables/disables the frame integration operation and selects the integration mode. The frame integration allows you to average or add frames "on the fly" without sacrificing the frame rate. Used in combination with

#### Syntax

[VB]

```
objActiveGige.Integrate [= Value]
```

[C/C++]

```
HRESULT get_Integrate ( short *pIntegrate );  
HRESULT put_Integrate( short Integrate );
```

#### Data Type [VB]

Integer

#### Parameters [C/C++]

*pIntegrate* [out,retval]

Pointer to the ordinal number of the currently selected Integrate filter, zero if Integrate conversion is disabled.

*Integrate* [in]

0 - Frame integration is disabled.

1 - Running Average mode. Each output frame is the result of averaging a selected number of previously captured frames.

2 - Running Accumulation mode. Each output frame is the sum of a selected number of previously captured frame.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example activates the running average mode with a 16-frame window:

```
ActiveGige1.IntegrateWnd=16  
ActiveGige1.Integrate = 1
```

#### Remarks

The frame integration is especially useful for suppressing the noise in the video and increasing its contrast in a low-light situation.

---

### 3.1.50 IntegrateWnd

#### Description

Returns or sets the number of frames to be used in the [Integrate](#) operation.

#### Syntax

[VB]  
`objActiveGige.IntegrateWnd [= Value]`

[C/C++]  
`HRESULT get_IntegrateWnd( long *pValue );`  
`HRESULT put_IntegrateWnd( long Value );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the currently selected number of frames for the frame integration  
*Value* [in]  
The number of frames to be set

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
Invalid property value.

#### Example

This VB example activates the running average mode with a 16-frame window:

```
ActiveGigel.IntegrateWnd=16  
ActiveGigel.Integrate = 1
```

#### Remarks

The size of the integration windows indicates the number of consecutive frames used for the Running Average or Running Accumulation operations. Increasing the size of the integration window lowers the video noise, but increases a motion-related blurring.

### 3.1.51 LensCorrect

#### Description

Enables/disables the lens distortion correction. Used in combination with [SetLensDistortion](#).

#### Syntax

[VB]  
`objActiveGige.LensCorrect [= Value]`

[C/C++]  
`HRESULT get_LensCorrect ( bool *pCorrect );`  
`HRESULT put_LensCorrect( bool Correct );`

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pCorrect* [out,retval]  
Pointer to the Boolean that is TRUE if the lens distortion correction is enabled, or FALSE otherwise  
*Correct* [in]  
Set to TRUE to enable the lens distortion correction, or set to FALSE to disable it

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

The following VB examples activates the lens distortion correction and uses two scroll bars to adjust the distortion parameters in real time :

```
Private Sub Form_Load()  
ActiveGige1.Acquire=True  
HScroll11.Min = -1000  
HScroll11.Max = 1000  
HScroll12.Min = -1000  
HScroll12.Max = 1000  
ActiveGige1.SetLensDistortion 0,0,False  
ActiveGige1.LensCorrect=True  
ActiveGige1.Acquire=True  
End Sub  
  
Private Sub HScroll11_Scroll()  
ActiveGige1.SetLensDistortion HScroll11.Value/1000., HScroll12.Value/1000.,  
False  
End Sub  
  
Private Sub HScroll12_Scroll()
```

---



```
ActiveGigel.SetLensDistortion HScroll1.Value/1000., HScroll2.Value/1000.,  
False  
End Sub
```

**Remarks**

The lens distortion correction is used to compensate for the barrel or pincushion image distortion caused by camera lenses. The barrel distortion makes straight lines at the edges of the image bow outwards and it is commonly seen on wide angle lenses with short focal length. The pincushion distortion makes straight lines at the edges of the image bow inwards, and it is commonly seen on telephoto lenses with long focal length. Before activating the lens distortion correction, you must set up the distortion parameters by calling [SetLensDistortion](#).

### 3.1.52 LineFormat

#### Description

Returns or sets (if possible) the current electrical format of the selected I/O line. Can be one of the following values:

*"NoConnect"*

The line is not connected.

*"TriState"*

The line is currently in the Tri-State mode (not driver).

*"TTL"*

The line is currently accepting or sending TTL level signals.

*"LVDS"*

The line is currently accepting or sending LVDS level signals.

*"RS422"*

The line is currently accepting or sending RS422 level signals..

*"OptoCoupled",...*

The line is opto-coupled.

#### Syntax

[VB]

```
objActiveGige.LineFormat [= Value]
```

[C/C++]

```
HRESULT get_LineFormat( string *pValue );
HRESULT put_LineFormat( string Value );
```

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the string specifying the line format.

*Value* [in]

The line format to be set.

#### Return Values

S\_OK

Success

E\_NOINTERFACE

The feature is not available for the selected camera

E\_INVALIDARG

The value is not part of the enumerated set

E\_FAIL

Failure to set the feature value

#### Example

The following VB example demonstrates the use of a combo box for line format selection:

---

```
Private Sub Form_Load()  
Lst = ActiveGigel.GetEnumList("LineFormat")  
For i = 0 To UBound(Lst)  
Combol.AddItem (Lst(i))  
Next  
Combol.ListIndex = ActiveGigel.LineFormat  
End Sub  
  
Private Sub Combol_Click()  
ActiveGigel.LineFormat = Combol.Text  
End Sub
```

### Remarks

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *LineFormat* feature per GenICam standard.

### 3.1.53 LineInverter

#### Description

Returns or sets the inverted state of the electrical input or output signal on the selected I/O line.

#### Syntax

[VB]  
`objActiveGige.LineInverter [= Value]`

[C/C++]  
`HRESULT get_LineInverter( VARIANT_BOOL *pValue );`  
`HRESULT put_LineInverter( VARIANT_BOOL Value );`

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pValue* [out, retval]  
Pointer to the boolean value specifying the state of the line.  
*Value* [in]  
FALSE if the line signal is not inverted, TRUE if the line signal is inverted.

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is not part of the enumerated set  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example sets an output of an inverted pulse coming from the Timer1 on the physical line 2 of the camera connector :

```
ActiveGige1.LineSelector="Line2"  
ActiveGige1.LineMode="Output"  
ActiveGige1.LineInverter="True"  
ActiveGige1.LineSource="Timer1Output"
```

#### Remarks

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *LineInverter*

---

---

feature per GenICam standard.

---

### 3.1.54 LineMode

#### Description

Returns or sets the input or output mode for the currently selected I/O line. Can be one of the following values:

*"Input"*

The selected physical line is used to input an electrical signal.

*"Output"*

The selected physical line is used to output an electrical signal.

#### Syntax

[VB]

```
objActiveGige.LineMode [= Value]
```

[C/C++]

```
HRESULT get_LineMode( string *pValue );  
HRESULT put_LineMode( string Value );
```

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the string specifying the line mode.

*Value* [in]

The line mode to be set.

#### Return Values

S\_OK

Success

E\_NOINTERFACE

The feature is not available for the selected camera

E\_INVALIDARG

The value is not part of the enumerated set

E\_FAIL

Failure to set the feature value

#### Example

The following VB example demonstrates the use of a combo box for line mode selection:

```
Private Sub Form_Load()  
Lst = ActiveGigel.GetEnumList("LineMode")  
For i = 0 To UBound(Lst)  
Combo1.AddItem (Lst(i))  
Next  
Combo1.ListIndex = ActiveGigel.LineMode
```

---

```
End Sub
```

```
Private Sub Comb1_Click()  
ActiveGigel.LineMode = Comb1.Text  
End Sub
```

### Remarks

When a line supports input and output modes, the default state is "Input" to avoid possible electrical contention.

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *LineMode* feature per GenICam standard.

### 3.1.55 LineSelector

#### Description

Returns or sets the physical line (or pin) of the external device connector to configure. The values are strings specifying the index of the physical line and associated I/O control block, such as "Line0", "Line1", "Line2".

#### Syntax

[VB]

```
objActiveGige.LineSelector [= Value]
```

[C/C++]

```
HRESULT get_LineSelector( string *pValue );  
HRESULT put_LineSelector( string Value );
```

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the string specifying the physical line to configure.

*Value* [in]

The line selection to be set

#### Return Values

S\_OK

Success

E\_NOINTERFACE

The feature is not available for the selected camera

E\_INVALIDARG

The value is not part of the enumerated set

E\_FAIL

Failure to set the feature value

#### Example

The following VB example sets an output of an inverted pulse coming from the Timer1 on the physical line 2 of the camera connector :

```
ActiveGige1.LineSelector="Line2"  
ActiveGige1.LineMode="Output"  
ActiveGige1.LineFormat="TTL"  
ActiveGige1.LineInverter="True"  
ActiveGige1.LineSource="Timer1Output"
```

#### Remarks

When a specific line is selected, all the other line features ([LineMode](#), [LineFormat](#), [LineSource](#),

---



---

[LineInverter](#)) will be applied to its associated I/O control block and will condition the resulting input or output signal.

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *LineSelector* feature per GenICam standard.

---

### 3.1.56 LineSource

#### Description

Returns or sets the source of the signal to output on the selected I/O line when the line mode is Output. Can be one of the following values:

*"Off"*  
Line output is disabled (Tri-State).

*"AcquisitionTriggerWait"*  
Camera is currently waiting for a trigger to capture one frame or series of frames.

*"AcquisitionActive"*  
Camera is currently doing an acquisition of one frame or series of frames.

*"FrameTriggerWait"*  
Camera is currently waiting for a frame trigger.

*"FrameActive"*  
Camera is currently doing a capture of a frame.

*"Timer1Active", "Timer2Active",...*  
The chosen timer is in the active state.

*"Counter1Active", "Counter2Active",...*  
The chosen counter is in the active state.

*"UserOutput1Active", "UserOutput2Active",...*  
The chosen user output bit state as defined by its current [UserOutputValue](#).

#### Syntax

[VB]

```
objActiveGige.LineSource [= Value]
```

[C/C++]

```
HRESULT get_LineSource( string *pValue );
HRESULT put_LineSource( string Value );
```

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]  
Pointer to the string specifying the line source.

*Value* [in]  
The line source to be set

#### Return Values

S\_OK  
Success

E\_NOINTERFACE  
The feature is not available for the selected camera

E\_INVALIDARG  
The value is not part of the enumerated set

E\_FAIL  
Failure to set the feature value

### Example

The following VB example demonstrates the use of a combo box for line source selection:

```
Private Sub Form_Load()  
Lst = ActiveGigel.GetEnumList("LineSource")  
For i = 0 To UBound(Lst)  
Combol.AddItem (Lst(i))  
Next  
Combol.ListIndex = ActiveGigel.LineSource  
End Sub  
  
Private Sub Combol_Click()  
ActiveGigel.LineSource = Combol.Text  
End Sub
```

### Remarks

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *LineSource* feature per GenICam standard.

### 3.1.57 LUTMode

#### Description

Enables/disables the lookup table operation on the video frames. The operation transforms the value of each pixel based on a corresponding factor in the LUT array.

#### Syntax

[VB]  
`objActiveGige.LUTMode [= Value]`

[C/C++]  
`HRESULT get_LUTMode ( bool *pLUT );`  
`HRESULT put_LUTMode( bool LUT );`

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pLUT* [out,retval]  
Pointer to the Boolean that is TRUE if the lookup table operation is enable, or FALSE otherwise  
*LUT* [in]  
Set to TRUE to enable the lookup table operation, or set to FALSE to disable it.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

The following VB example enables the Window/Level operation and sets the minimum and maximum levels for the histogram scaling to 20% and 70% :

```
ActiveGige1.LUTMode = True  
ActiveGige1.SetLevels 20, 70
```

#### Remarks

This property works in combination with the [SetLUT](#), [SetLevels](#) or [SetGains](#) methods.

---

### 3.1.58 Magnification

#### Description

Returns or sets the zoom factor for the live video display.

#### Syntax

[VB]  
`objActiveGige.Magnification [= Value]`

[C/C++]  
`HRESULT get_Magnification( float *pMagnification );`  
`HRESULT put_Magnification( float Magnification );`

#### Data Type [VB]

Float

#### Parameters [C/C++]

*pMagnification* [out,retval]

Pointer to the current zoom factor

*Magnification* [in]

Floating point value specifying the zoom factor to be set. If 0, the image will be fit to the size of the control window. If -1, the image will be displayed in the full-screen mode. If -2, the image conversion and display will be disabled.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid property value.

#### Example

This VB example sets the zoom factor to 0.25:

```
ActiveGigel.Magnification = 0.25  
MsgBox ActiveGigel.Magnification
```

#### Remarks

This property adjusts the magnification of the live video display. It does not change the content of the image data, but only its appearance in the *ActiveGige* control window. The valid property values are from 0 to 100. If the **Magnification** property is set to zero, the video image will be fit to the size of the control window. In this case the display might not retain the original proportions of the video frame. If the property is set to -1, the image will be displayed in the full-screen mode retaining the original proportions.

Note that the [Acquire](#) and [Display](#) properties must be set to TRUE in order for the live video to be

displayed in the control window.

Note that setting **Magnification** properties to -2 will disable an image conversion pipeline that is normally applied to each raw frame. This will significantly increase the performance of your application as long as it does not utilize *ActiveGigE*'s built-in image decoding/display and uses [GetRawData](#) to access the raw image data.

### 3.1.59 MonitorSync

#### Description

Enables/disables the monitor synchronization feature for the live video display in the control window.

#### Syntax

[VB]

```
objActiveGige.MonitorSync [= Value]
```

[C/C++]

```
HRESULT get_MonitorSync ( bool *pValue );  
HRESULT put_MonitorSync( bool Value );
```

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pValue* [out,retval]

Pointer to the Boolean that is TRUE if the monitor synchronization is enable, or FALSE otherwise

*Value* [in]

Set to TRUE to enable the monitor synchronization, or set to FALSE to disable it

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example enables the monitor synchronization:

```
ActiveGigel.MonitorSync = True  
MsgBox ActiveGigel.MonitorSync
```

#### Remarks

Enabling the monitor synchronization option causes the camera frame rate to exactly match the current refresh rate of the system monitor thus eliminating all the display artifacts related to the digital image transfer, such as the display tearing and syncopation. The monitor synchronization algorithm uses a patent-pending technique from A&B Software. For more details refer to [Software overcomes display artifacts in digital image transfer](#).

Note that this property will be unavailable if the camera does not support the software trigger.





### 3.1.60 Multicast

#### Description

Enables/disables the multicast mode.

#### Syntax

```
[VB]  
objActiveGige.Multicast [= Value]
```

```
[C/C++]  
HRESULT get_Multicast( bool *pMulticase );  
HRESULT put_Multicast( bool Multicast );
```

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pMulticast*[out,retval]  
Pointer to the Boolean that is TRUE if the multicast mode is enable, or FALSE otherwise  
*Multicast* [in]  
Set to TRUE to enable the multicast, or set to FALSE to disable it.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

The following VB example sets the camera into the multicast mode and starts the video acquisition:

```
ActiveGigel.Multicast = True  
ActiveGigel.Acquire = True
```

#### Remarks

The **Multicast** mode allows multiple computers and applications on the network to receive the video feed from the same camera. An *ActiveGige* based application that first opens the camera becomes the [master](#) application. Once the master application enables the **Multicast** mode and starts the acquisition, all other *ActiveGige* based applications launched in the network will acquire and display the video in the slave mode.

**Multicast** is usually used for distributed monitoring or distributed processing. In the latter case one computer may record video to the disk while another one can process and analyze images.

A default IP address of the multicast end-point is assigned by the [IP configuration utility](#), but can also be set programmatically with the [SetDestinationIP](#) method.

Note that each computer that needs to receive the video feed from a camera must be able to "see" the

camera on the network. This is usually done by connecting the camera and computers to the same Gigabit switch and setting the IP address of each network card to the subnet of the camera (typically 169.254.x.x).

Note that any change in the image format performed by the master application in the **Multicast** mode (such as [Format](#), [SizeX](#), [SizeY](#), [PacketSize](#)) are not transferred to the slave applications and can cause image corruption and errors if done during the video acquisition.

Note that certain GigE Vision cameras may not support **Multicast**.

---

### 3.1.61 Overlay

#### Description

Enables/disables the overlay.

#### Syntax

[VB]  
`objActiveGige.Overlay [= Value]`

[C/C++]  
`HRESULT get_Overlay ( bool *pOverlay );`  
`HRESULT put_Overlay( bool Overlay );`

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pOverlay* [out,retval]  
Pointer to the Boolean that is TRUE if the overlay is enable, or FALSE otherwise  
*Overlay* [in]  
Set to TRUE to enable the overlay, or set to FALSE to disable it.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

The following VB example overlays a red circle on the live video:

```
ActiveGigel.Acquire = True
ActiveGigel.OverlayEllipse 100,100,200,200
ActiveGigel.OverlayColor=RGB(255,0,0)
ActiveGigel.Overlay= True
```

#### Remarks

Overlay feature allows you to display custom graphics and text on the live video image. Setting this property to true causes ActiveGige to show the overlay. Disabling this property hides the overlay. Note that hiding the overlay does not erase graphics and text from it. To clear the overlay, use [OverlayClear](#). Also see [OverlayColor](#), [OverlayPixel](#), [OverlayLine](#), [OverlayRectangle](#), [OverlayEllipse](#), [OverlayText](#).

### 3.1.62 OverlayColor

#### Description

Returns or sets the color of the overlay graphics.

#### Syntax

[VB]

```
objActiveGige.OverlayColor [= Color]
```

[C/C++]

```
HRESULT get_OverlayColor(OLE_COLOR& pColor);  
HRESULT put_OverlayColor(OLE_COLOR Color);
```

#### Data Type [VB]

RGB color

#### Parameters [C/C++]

*pColor* [out,retval]

Pointer to the current overlay color

*Color* [in]

The overlay color to be set

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example overlays a red circle on the live video:

```
ActiveGige1.Acquire = True  
ActiveGige1.OverlayEllipse 100,100,200,200  
ActiveGige1.OverlayColor=RGB(255,0,0)  
ActiveGige1.Overlay= True
```

#### Remarks

Overlay feature allows you to display custom graphics and text on the live video image. Also see [OverlayColor](#), [OverlayPixel](#), [OverlayLine](#), [OverlayRectangle](#), [OverlayEllipse](#), [OverlayText](#).

---

### 3.1.63 OverlayFont

#### Description

Returns or sets the font for [OverlayText](#).

#### Syntax

[VB]

```
objActiveGige.OverlayFont [= Font]
```

[C/C++]

```
HRESULT get_OverlayFont(IFontDisp* *pFont);  
HRESULT put_OverlayFont(IFontDisp* Font);
```

#### Data Type [VB]

StdFont

#### Parameters [C/C++]

*pFont* [out,retval]

Pointer to the *IFontDisp* interface object corresponding to the current overlay font

*Font* [in]

*IFontDisp* interface object corresponding to the overlay font to be set

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example overlays a string of text on the live video:

```
Dim Font As New StdFont  
Font.Name = "Arial"  
Font.Size = 18  
Font.Bold = True  
ActiveGigel.OverlayFont = Font  
ActiveGigel.OverlayText 10, 100, "ActiveGige rules!"  
ActiveGigel.OverlayColor = RGB(255, 0, 0)  
ActiveGigel.Overlay = True
```

#### Remarks

Also see [OverlayColor](#), [OverlayText](#).

### 3.1.64 PacketSize

#### Description

Returns or sets the size of image data packets in bytes.

#### Syntax

[VB]  
`objActiveGige.PacketSize [= Value]`

[C/C++]  
`HRESULT get_PacketSize( long *pValue );`  
`HRESULT put_PacketSize( long Value );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current packet size  
*Value* [in]  
The packet size to be set

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
Invalid property value.

#### Example

This VB example sets the current packet size to 8000 bytes:

```
ActiveGigel.PacketSize = 8000  
MsgBox ActiveGigel.PacketSize
```

#### Remarks

To lower the overhead of packet transmission, it is recommended to set this property to the maximum packet size allowed by your network card and network configuration (typically 1500 if Jumbo packets are disabled and 9014 if Jumbo packets are enabled). You can enable Jumbo packets in many network cards from the Device Manager by right-clicking the network card and selecting Properties. If your network card does not support Jumbo Frames, set this property to 1500.

For more information refer to *"GigE Vision Video Streaming and Device Control Over Ethernet"*

---

*Standard*" published by the Automated Imaging Association.

### 3.1.65 Palette

#### Description

Returns or sets the ordinal number of the currently selected live video palette. The values from 0 to 7 correspond to the following predefined palettes:

- 0 - *Gray*  
Applies the standard 256-level grayscale palette. This is a regular mode of viewing a grayscale video.
- 1 - *Inverse*  
Applies the inverted 256-level grayscale palette. The video will be displayed in the negative format.
- 2 - *Saturated*  
Applies the grayscale palette with colored upper entries. The saturated palette allows you to control the dynamic range of the video signal by bringing it slightly below the saturation level of the video camera or video amplifier. To achieve the maximum dynamic range, adjust the intensity of the light source and/or the gain and zero level of the video amplifier so that the red color corresponding to the brightest pixel values just barely shows up.
- 3 - *Rainbow*  
Applies a color palette where the entries are evenly distributed along the Hue axis. This allows for assigning different color pigments to different levels of intensity.
- 4 - *Spectra*  
Applies a color palette where the entries are distributed along the Hue and Luminance axes. That allows for assigning different color pigments to different levels of intensity while preserving the luminance scale.
- 5 - *Isodense*  
Applies the 256-level grayscale palette, each 8-th entry of which is colored. The isodense palette allows you to clearly see transitions between different levels of intensities as isolines on a topographic map.
- 6 - *Multiphase*  
Applies the multiphase palette. Entries in the multiphase palette are at opposite ends of the color model so even small changes in gray levels are highlighted.
- 7 - *Random*  
Applies the random color palette whose entries are filled with random values each time you select it from the menu.
- 8 - *Threshold*  
Applies the grayscale palette with colored luminance range of interest. Works in combination with [SetROI](#).

#### Syntax

[VB]  
`objActiveGige.Palette [= Value]`

[C/C++]  
`HRESULT get_Palette( long *pPalette );`  
`HRESULT put_Palette( long Palette );`

#### Data Type [VB]

Long

#### Parameters [C/C++]



*pPalette* [out,retval]  
Pointer to the ordinal number of the currently selected palette  
*Palette* [in]  
The number of the palette to be selected

### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
Invalid property value.

### Example

This VB example demonstrates the use of a combo box for changing display palettes on the live video:

```
Private Sub Form_Load()  
    ActiveGigel.Acquire = True  
    Combol.AddItem ("Gray")  
    Combol.AddItem ("Inverse")  
    Combol.AddItem ("Saturated")  
    Combol.AddItem ("Rainbow")  
    Combol.AddItem ("Spectra")  
    Combol.AddItem ("Isodense")  
    Combol.AddItem ("Multiphase")  
    Combol.AddItem ("Random")  
    Combol.ListIndex = 0  
End Sub  
  
Private Sub Combol_Click()  
    ActiveGigel.Palette = Combol.ListIndex  
End Sub
```

### Remarks

The **Palette** property is used for viewing monochrome video in pseudo-colors. The property is only valid for a monochrome [Mode](#). If you use property pages in your development environment, the **Palette** property field will be presented as a list box containing the names of all available palettes.

Note that the [Acquire](#) and [Display](#) properties must be set to TRUE in order for the live video to be displayed in the control window.

### 3.1.66 Privilege

#### Description

Selects or returns the level of the camera control privilege for an *ActiveGige*-based process. Can be one of the following values:

*0 - No Access (read-only)*

Process has no access to the camera. Another process has exclusive control over the camera.

*1 - Monitor Access*

Process has secondary access to the camera. It can read the camera settings and receive the video stream (when the primary application is using [Multicast](#)), but cannot modify the camera settings. A process can be allowed monitor access if there is no other active process with exclusive access to the same camera.

*2 - Control Access*

Process has primary control over the camera. Other processes can read from the camera, but cannot modify its settings..A process can be allowed control access if there is no other active process with exclusive or control access to the same camera.

*3 - Exclusive Access*

Process has exclusive control over the camera. No other process can access the camera. A process can be allowed exclusive access only if there is no other active process with exclusive or control access to the same camera.

*4 - Control with Switchover Access*

Process has primary control over the camera. Other processes can read from the camera, but cannot modify its settings..A process can be allowed control access if there is no other active process with exclusive or control access to the same camera, or if the process obtains the right credentials by submitting a correct Control Switchover Key to the camera via the [SetControlKey](#) method.

#### Syntax

[VB]

```
objActiveGige.Privilege [= Value]
```

[C/C++]

```
HRESULT get_Privilege ( short *pValue );
HRESULT put_Privilege( short Value );
```

#### Data Type [VB]

Integer

#### Parameters [C/C++]

*Value* [in]

Pointer to the ordinal number of the currently selected camera control privilege.

*Value* [in]

The privilege level to be set.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

### Example

This VB example changes the privilege of an application to the Monitor Access:

```
ActiveGigel.Privilege = 1
```

### Remarks

This property can be used for transferring primary control over the camera from one application (or/and computer) to another. This can be achieved by changing the privilege of a primary application from the Control to Monitor access and then changing the privilege of a secondary application from the Monitor to Control access. If your camera supports the Control with Switchover access, any application can become a primary one by submitting a correct access key. See [SetControlKey](#) for more details.

By default the first *ActiveGige*-based application that connects to a camera receives the *Control Access* privilege, while applications that connect to the camera afterwards receive the *Monitor Access* privilege.

### 3.1.67 Rotate

#### Description

Rotates the image at the specified angle. The following angle values are allowed:

- 0 - No image rotation is performed
- 90 - The image is rotated 90° counterclockwise
- 180 - The image is rotated 180°
- 270 - The image is rotated -90° clockwise

#### Syntax

[VB]

```
objActiveGige.Rotate [= Value]
```

[C/C++]

```
HRESULT get_Rotate( long *pRotate );  
HRESULT put_Rotate( long Rotate );
```

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pRotate* [out,retval]

Pointer to the rotational angle.

*Rotate* [in]

The rotational angle to be set.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example demonstrates the use of a combo box for rotating the live video:

```
Private Sub Form_Load()  
ActiveGige1.Acquire = True  
Combol.AddItem ("0°")  
Combol.AddItem ("90°")  
Combol.AddItem ("180°")  
Combol.AddItem ("270°")  
Combol.ListIndex = 0  
End Sub  
  
Private Sub Combol_Click()  
ActiveGige1.Rotate = Combol.ListIndex  
End Sub
```

---

## Remarks

Image rotation affects the way the video is displayed in the control window as well as actual order of pixels in the image frame. If the rotation angle is different from zero, the data returned by [GetData](#) and other data access methods will be rotated as well.

Note that the [Flip](#) operation has precedence over rotation. If both **Flip** and **Rotate** properties are non-zero, the frame will be flipped first and the resulting image rotated.

If the value of the angle is different from 0, 90, 180 and 270, it will be substituted with the nearest allowable value.

### 3.1.68 ScrollBars

#### Description

Enables/disables the scroll bars on the control window.

#### Syntax

[VB]

```
objActiveGige.ScrollBars [= Value]
```

[C/C++]

```
HRESULT get_ScrollBars ( bool *pScrollBars );  
HRESULT put_ScrollBars( bool ScrollBars );
```

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pScrollBars* [out,retval]

Pointer to the Boolean that is TRUE if the scroll bars are enable, or FALSE otherwise

*ScrollBars* [in]

Set to TRUE to enable the scroll bars, or set to FALSE otherwise

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example enables scroll bars on the control window:

```
ActiveGige1.ScrollBars = True  
MsgBox ActiveGige1.ScrollBars
```

#### Remarks

If this property is set to TRUE and the video width or/and height (see [SizeX](#) and [SizeY](#)) exceed the size of the control window, a scroll bar(s) will be displayed on the border of the control window allowing you to pan the live video. The current position of scroll bars can be retrieved or modified via the [ScrollX](#) and [ScrollY](#) properties. When the scroll bars are moved, the [Scroll](#) event will be raised.

Note that the [Acquire](#) and [Display](#) properties must be set to TRUE in order for the live video to be displayed in the control window.

---



### 3.1.69 ScrollX

#### Description

Returns or sets the current horizontal scroll position for the live video display.

#### Syntax

[VB]

```
objActiveGige.ScrollX [= Value]
```

[C/C++]

```
HRESULT get_ScrollX( long *pScrollX );  
HRESULT put_ScrollX( long ScrollX );
```

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pScrollX* [out,retval]

Pointer to the current horizontal scroll position.

*ScrollX* [in]

The horizontal scroll position to be set.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid property value.

#### Example

This VB example sets the current horizontal scroll position to 512:

```
ActiveGigel.ScrollX = 512  
MsgBox ActiveGigel.ScrollX
```

#### Remarks

The **ScrollX** property is only valid if the [ScrollBars](#) are present in the control window.

Note that the value returned by this property refers to the image coordinate system, not to the screen coordinates.

---



### 3.1.70 ScrollY

#### Description

Returns or sets the current vertical scroll position for the live video display.

#### Syntax

[VB]  
`objActiveGige.ScrollY [= Value]`

[C/C++]  
`HRESULT get_ScrollY( long *pScrollY );`  
`HRESULT put_ScrollY( long ScrollY );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pScrollY* [out,retval]  
Pointer to the current vertical scroll position.  
*ScrollY* [in]  
The vertical scroll position to be set.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
Invalid property value.

#### Example

This VB example sets the current vertical scroll position to 512:

```
ActiveGigel.ScrollY = 512  
MsgBox ActiveGigel.ScrollY
```

#### Remarks

The **ScrollY** property is only valid if the [ScrollBars](#) are present in the control window.

Note that the value returned by this property refers to the image coordinate system, not to the screen coordinates.

### 3.1.71 SizeX

#### Description

Returns or sets the width of the partial scan window.

#### Syntax

[VB]  
`objActiveGige.SizeX [= Value]`

[C/C++]  
`HRESULT get_SizeX( long *pSizeX );`  
`HRESULT put_SizeX( long SizeX );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pSizeX* [out,retval]  
Pointer to the currently selected image width  
*SizeX* [in]  
The image width to be selected

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
Invalid property value.

#### Example

This VB example sets the current image width to 512:

```
ActiveGigel.SizeX = 512  
MsgBox ActiveGigel.SizeX
```

#### Remarks

This property allows you to set the horizontal size of the scan window in pixels. Only certain values of the width can be allowed for the partial scan window depending on the camera. If **SizeX** is set to a value which is not supported by the camera, it will be reset to the nearest allowable width. This property is directly associated with the *Width* feature of the camera.

---

### 3.1.72 SizeY

#### Description

Returns or sets the height of the partial scan window.

#### Syntax

[VB]  
`objActiveGige.SizeY [= Value]`

[C/C++]  
`HRESULT get_SizeY( long *pSizeY );`  
`HRESULT put_SizeY( long SizeY );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pSizeY* [out,retval]  
Pointer to the currently selected image height  
*SizeY* [in]  
The image height to be selected

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
Invalid property value.

#### Example

This VB example sets the current image height to 512:

```
ActiveGigel.SizeY = 512  
MsgBox ActiveGigel.SizeY
```

#### Remarks

This property allows you to set the vertical size of the scan window in pixels. Only certain values of the height can be allowed for the partial scan window depending on the camera. If the **SizeY** is set to a value which is not supported by the camera, it will be reset to the nearest allowable height. This property is directly associated with the *Height* feature of the camera.

Note that if images are transmitted by a linescan camera, the actual vertical size of each image can vary and be less than **SizeY**. In this case **SizeY** will represent the maximum vertical size of the scan

window, while the actual vertical size of the current image can be obtained by calling [GetHeight](#).

---

### 3.1.73 OffsetX

#### Description

Returns or sets the horizontal offset of the partial scan window.

#### Syntax

[VB]

```
objActiveGige.OffsetX [= Value]
```

[C/C++]

```
HRESULT get_OffsetX( long *pOffsetX );
```

```
HRESULT put_OffsetX( long OffsetX );
```

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pOffsetX* [out,retval]

Pointer to the currently set horizontal offset

*SizeX* [in]

The horizontal offset to be set

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid property value.

#### Example

This VB example sets the horizontal offset of the video window to 64:

```
ActiveGigel.OffsetX = 64  
MsgBox ActiveGigel.OffsetX
```

#### Remarks

This property allows you to set the horizontal offset of the top left corner of the scan window in pixels. Only certain values **OffsetX** can be allowed depending on the camera. If the property is set to a value which is not supported by the camera, it will be reset to the nearest allowable one.

### 3.1.74 OffsetY

#### Description

Returns or sets the vertical offset of the partial scan window.

#### Syntax

[VB]  
`objActiveGige.OffsetY [= Value]`

[C/C++]  
`HRESULT get_OffsetY( long *pOffsetY );`  
`HRESULT put_OffsetY( long OffsetY );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pOffsetY* [out,retval]  
Pointer to the currently set vertical offset  
*OffsetY* [in]  
The vertical offset to be set

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
Invalid property value.

#### Example

This VB example sets the vertical offset of the video window to 64 pixels:

```
ActiveGigel.OffsetY = 64  
MsgBox ActiveGigel.OffsetY
```

#### Remarks

This property allows you to set the vertical offset of the top left corner of the scan window in pixels. Only certain values **OffsetY** can be allowed depending on the camera. If the property is set to a value which is not supported by the camera, it will be reset to the nearest allowable one.

---

### 3.1.75 Timeout

#### Description

Returns or sets the current acquisition timeout in seconds.

#### Syntax

[VB]  
`objActiveGige.Timeout [= Value]`

[C/C++]  
`HRESULT get_Timeout( long *pTimeout );`  
`HRESULT put_Timeout( long Timeout );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pTimeout* [out,retval]  
Pointer to the currently set timeout.  
*Timeout* [in]  
The timeout to be set.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example sets the current timeout to 10 sec:

```
ActiveGigel.Timeout = 10  
MsgBox ActiveGigel.Timeout
```

#### Remarks

The **Timeout** property lets you set the number of seconds to wait for a frame to be acquired. Typically used to assign the timeout when the [Trigger](#) mode is active. If the timeout expires, the [Timeout](#) event will be raised.

### 3.1.76 TestImageSelector

#### Description

Returns or sets the mode of generating internal test images. Can be one of the following values:

- "Off"  
Test image mode is disabled. Image is coming from the sensor.
- "Black"  
Generates the darkest possible image
- "White"  
Generates the brightest possible image
- "GrayHorizontalRamp"  
Generates the horizontal wedge going from the darkest to the brightest possible intensity
- "GrayVerticalRamp"  
Generates the vertical wedge going from the darkest to the brightest possible intensity
- "GrayHorizontalRampMoving"  
Generates the horizontal wedge going from the darkest to the brightest possible intensity and moving from left to right
- "GrayVerticalRampMoving"  
Generates the vertical wedge going from the darkest to the brightest possible intensity and moving from top to bottom
- "HorizontalLineMoving"  
A moving horizontal line is superimposed on the live image
- "VerticalLineMoving"  
A moving vertical line is superimposed on the live image
- "FrameCounter"  
A frame counter is superimposed on the live image

#### Syntax

[VB]  
`objActiveGige.TestImageSelector [= Value]`

[C/C++]  
`HRESULT get_TestImageSelector( string *pValue );`  
`HRESULT put_TestImageSelector( string Value );`

#### Data Type [VB]

String

#### Parameters [C/C++]

- pValue* [out, retval]  
Pointer to the string specifying the test image selector setting
- Value* [in]  
The test image selection to be set

#### Return Values

- S\_OK  
Success
- E\_NOINTERFACE  
The feature is not available for the selected camera
- E\_INVALIDARG



The value is not part of the enumerated set  
E\_FAIL  
Failure to set the feature value

### Example

The following VB example demonstrates the use of a combo box to switch between available test images.

```
Private Sub Form_Load()  
Lst = ActiveGigel.GetEnumList("TestImageSelector")  
For i = 0 To UBound(Lst)  
Combol.AddItem (Lst(i))  
Next  
Combol.ListIndex = ActiveGigel.TestImageSelector  
End Sub  
  
Private Sub Combol_Click()  
ActiveGigel.TestImageSelector = Combol.Text  
End Sub
```

### Remarks

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *TestImageSelector* feature per GenICam standard.

### 3.1.77 Trigger

#### Description

Enables/disables the selected trigger.

#### Syntax

[VB]  
`objActiveGige.Trigger [= Value]`

[C/C++]  
`HRESULT get_Trigger ( bool *pTrigger );`  
`HRESULT put_Trigger( bool Trigger );`

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pTrigger* [out,retval]  
Pointer to the Boolean that is TRUE if the trigger mode is enable, or FALSE otherwise  
*Trigger* [in]  
Set to TRUE to enable the trigger mode, or set to FALSE otherwise

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
Triggering is not available for the selected camera  
E\_FAIL  
Failure.

#### Example

This VB example activates the trigger mode:

The following VB example sets a hardware trigger which will start the capture of each frame on the rising edge of the signal coming from the physical input Line1:

```
ActiveGigel.TriggerSelector = "FrameStart"  
ActiveGigel.Trigger = True  
ActiveGigel.TriggerActivation = "RisingEdge"  
ActiveGigel.TriggerSource = "Line1"  
ActiveGigel.Acquire = True
```

#### Remarks

This property corresponds to the *TriggerMode* feature per GenICam standard.

---

---

Before turning the trigger on, select a corresponding trigger configuration with [TriggerSelector](#).

---

### 3.1.78 TriggerActivation

#### Description

Returns or sets the activation mode for the selected trigger configuration. Can be one of the following values:

*"RisingEdge"*  
Trigger will assert on the rising edge of the source signal

*"FallingEdge"*  
Trigger will assert on the falling edge of the source signal

*"AnyEdge"*  
Trigger will assert on both edges of the source signal

*"LevelHigh"*  
Trigger will be valid as long as the level of the source signal is high

*"LevelLow"*  
Trigger will be valid as long as the level of the source signal is low

#### Syntax

[VB]  
`objActiveGige.TriggerActivation [= Value]`

[C/C++]  
`HRESULT get_TriggerActivation( string *pValue );`  
`HRESULT put_TriggerActivation( string Value );`

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]  
Pointer to the string specifying the trigger activation setting

*Value* [in]  
The trigger activation mode to be set

#### Return Values

`S_OK`  
Success

`E_NOINTERFACE`  
The feature is not available for the selected camera

`E_INVALIDARG`  
The value is not part of the enumerated set

`E_FAIL`  
Failure to set the feature value

#### Example

The following VB example sets a hardware trigger which will start the capture of each frame on the rising edge of the signal coming from the physical input Line1:

```
ActiveGigel.TriggerSelector = "FrameStart"  
ActiveGigel.Trigger = True  
ActiveGigel.TriggerActivation = "RisingEdge"  
ActiveGigel.TriggerSource = "Line1"  
ActiveGigel.Acquire = True
```

### Remarks

Before setting up this property, select a corresponding trigger configuration with [TriggerSelector](#).

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *TriggerActivation* feature per GenICam standard.

### 3.1.79 TriggerDelay

#### Description

Returns or sets the absolute value of the trigger delay in microseconds or raw units.

#### Syntax

[VB]  
`objActiveGige.TriggerDelay [= Value]`

[C/C++]  
`HRESULT get_TriggerDelay( float *pValue );`  
`HRESULT put_TriggerDelay( float Value );`

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current trigger delay value  
*Value* [in]  
The trigger delay value to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example sets a hardware trigger which will start the capture of each frame on the rising edge of the signal coming from the physical input Line1:

```
ActiveGigel.TriggerSelector = "FrameStart"  
ActiveGigel.Trigger = True  
ActiveGigel.TriggerActivation = "RisingEdge"  
ActiveGigel.TriggerDelay = 10000.  
ActiveGigel.TriggerSource = "Line1"  
ActiveGigel.Acquire = True
```

#### Remarks

Trigger delay changes the time after the trigger reception before effectively activating it. Prior to setting

---

---

up this property, select a corresponding trigger configuration with [TriggerSelector](#). The valid property range can be retrieved by the [GetTriggerDelayMin](#) and [GetTriggerDelayMax](#) methods.

Note that the property is available only if the currently selected camera supports one of the following GenICam features: *TriggerDelay*, *TriggerDelayAbs*, *TriggerDelayRaw*.

### 3.1.80 TriggerDelayAbs

#### Description

Returns or sets the absolute value of the trigger delay in microseconds. This property is deprecated and replaced by [TriggerDelay](#).

#### Syntax

[VB]  
`objActiveGige.TriggerDelayAbs [= Value]`

[C/C++]  
`HRESULT get_TriggerDelayAbs( float *pValue );`  
`HRESULT put_TriggerDelayAbs( float Value );`

#### Data Type [VB]

Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current trigger delay value  
*Value* [in]  
The trigger delay value to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example sets a hardware trigger which will start the capture of each frame on the rising edge of the signal coming from the physical input Line1:

```
ActiveGigel.TriggerSelector = "FrameStart"  
ActiveGigel.Trigger = True  
ActiveGigel.TriggerActivation = "RisingEdge"  
ActiveGigel.TriggerDelayAbs = 10000.  
ActiveGigel.TriggerSource = "Line1"  
ActiveGigel.Acquire = True
```

#### Remarks

---



---

Trigger delay changes the time after the trigger reception before effectively activating it. Prior to setting up this property, select a corresponding trigger configuration with [TriggerSelector](#). The valid property range can be retrieved by the [GetFeatureMin](#) and [GetFeatureMax](#) methods.

Note that the property is available only if the currently selected camera supports the *TriggerDelay* or *TriggerDelayAbs* feature per GenICam standard.

---

### 3.1.81 TriggerDelayRaw

#### Description

Returns or sets the raw value of the trigger delay. This property is deprecated and replaced by [TriggerDelay](#).

#### Syntax

[VB]  
`objActiveGige.TriggerDelayRaw [= Value]`

[C/C++]  
`HRESULT get_TriggerDelayRaw( long *pValue );`  
`HRESULT put_TriggerDelayRaw( long Value );`

#### Data Type [VB]

Long

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current trigger delay value  
*Value* [in]  
The trigger delay value to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_INVALIDARG  
The value is out of range  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example sets a hardware trigger which will start the capture of each frame on the rising edge of the signal coming from the physical input Line1:

```
ActiveGigel.TriggerSelector = "FrameStart"  
ActiveGigel.Trigger = True  
ActiveGigel.TriggerActivation = "RisingEdge"  
ActiveGigel.TriggerDelayRaw = 10000  
ActiveGigel.TriggerSource = "Line1"  
ActiveGigel.Acquire = True
```

#### Remarks

---

---

Trigger delay changes the time after the trigger reception before effectively activating it. Prior to setting up this property, select a corresponding trigger configuration with [TriggerSelector](#).

Note that the property is available only if the currently selected camera supports the *TriggerDelayRaw* feature per GenICam standard.

### 3.1.82 TriggerSelector

#### Description

Returns or sets the configuration of a trigger signal. Can be one of the following values:

*"AcquisitionStart"*

Trigger will start the acquisition of one or several frames according to [AcquisitionMode](#)

*"AcquisitionEnd"*

Trigger will stop the acquisition of one or several frames according to [AcquisitionMode](#)

*"AcquisitionActive"*

Trigger will control the duration of the acquisition of one or several frames

*"FrameStart"*

Trigger will start the acquisition of a frame

*"FrameEnd"*

Trigger will end the acquisition of a frame (used in line scan cameras)

*"FrameActive"*

Trigger will control the duration of a frame (used in line scan cameras)

*"LineStart"*

Trigger will start the capture of one line of a frame (used in line scan cameras)

*"ExposureStart"*

Trigger will start the exposure of a frame (or line )

*"ExposureEnd"*

Trigger will end the exposure of a frame (or line )

*"ExposureActive"*

Trigger will control the duration of the exposure of a frame (or line )

#### Syntax

[VB]

```
objActiveGige.TriggerSelector [= Value]
```

[C/C++]

```
HRESULT get_TriggerSelector( string *pValue );
HRESULT put_TriggerSelector( string Value );
```

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the string specifying the trigger selector setting

*Value* [in]

The trigger selection to be set

#### Return Values

S\_OK

Success

E\_NOINTERFACE

The feature is not available for the selected camera

E\_INVALIDARG

The value is not part of the enumerated set

**E\_FAIL**

Failure to set the feature value

**Example**

The following VB example sets a hardware trigger which will start the capture of each frame on the rising edge of the signal coming from the physical input Line1:

```
ActiveGigel.TriggerSelector = "FrameStart"  
ActiveGigel.Trigger = True  
ActiveGigel.TriggerActivation = "RisingEdge"  
ActiveGigel.TriggerSource = "Line1"  
ActiveGigel.Acquire = True
```

**Remarks**

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *TriggerSelector* feature per GenICam standard.

### 3.1.83 TriggerSource

#### Description

Returns or sets the internal signal or physical input to be used as the trigger source for the selected trigger configuration. Can be one of the following values:

*"Software"*

Trigger signal will be generated by software using the [SoftTrigger](#) command.

*"Line0", "Line1", "Line2",...*

The physical line or pin to be used as external trigger source.

*"Timer1Start", "Timer2Start", ... "Timer1End", "Timer2End",...*

The Timer signal to be used as internal trigger source

*"Counter1Start", "Counter2Start", ... "Counter1End", "Counter2End",...*

The Counter signal to be used as internal trigger source

*"UserInput0", "UserInput1", "UserInput2",...*

The User Output bit signal to be used as internal trigger source

#### Syntax

[VB]

```
objActiveGige.TriggerSource [= Value]
```

[C/C++]

```
HRESULT get_TriggerSource( string *pValue );
HRESULT put_TriggerActivation( string Value );
```

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the string specifying the current trigger source

*Value* [in]

The trigger source to be set

#### Return Values

S\_OK

Success

E\_NOINTERFACE

The feature is not available for the selected camera

E\_INVALIDARG

The value is not part of the enumerated set

E\_FAIL

Failure to set the feature value

#### Example

The following VB example sets a hardware trigger which will start the capture of each frame on the rising edge of the signal coming from the physical input Line1:

```
ActiveGigel.TriggerSelector = "FrameStart"  
ActiveGigel.Trigger = True  
ActiveGigel.TriggerActivation = "RisingEdge"  
ActiveGigel.TriggerSource = "Line1"  
ActiveGigel.Acquire = True
```

### Remarks

Before setting up this property, select a corresponding trigger configuration with [TriggerSelector](#).

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *TriggerSource* feature per GenICam standard.

### 3.1.84 UserOutputSelector

#### Description

Selects the bit of the User Output register to be set by [UserOutputValue](#). Can be one of the following values:

"UserOutput0"  
Selects Bit 0 of the User Output register.  
"UserOutput1"  
Selects Bit 1 of the User Output register.  
"UserOutput2"  
Selects Bit 2 of the User Output register.  
.....

#### Syntax

[VB]  
`objActiveGige.UserOutputSelector [= Value]`

[C/C++]  
`HRESULT get_UserOutputSelector( string *pValue );`  
`HRESULT put_UserOutputSelector( string Value );`

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]  
Pointer to the string specifying the bit in the User Output register.  
*Value* [in]  
The bit to be set

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the use of a combo box for line selection:

```
Private Sub Form_Load()  
Lst = ActiveGige1.GetEnumList("UserOutputSelector")  
For i = 0 To UBound(Lst)  
Combo1.AddItem (Lst(i))  
Next  
Combo1.ListIndex = ActiveGige1.UserOutputSelector
```



```
End Sub
```

```
Private Sub Comb1_Click()  
ActiveGigel.UserOutputSelector = Comb1.Text  
End Sub
```

### Remarks

This property works in combination with [UserOutputValue](#). Using a corresponding [LineSource](#), the bits from the User Output register can be directed to a physical output line.

Note that the property is available only if the currently selected camera supports the *UserOutputSelector* feature per GenICam standard.

### 3.1.85 UserOutputValue

#### Description

Returns or sets the value of the selected bit of the User Output register.

#### Syntax

[VB]  
`objActiveGige.UserOutputValue [= Value]`

[C/C++]  
`HRESULT get_UserOutputValue( VARIANT_BOOL *pValue );`  
`HRESULT put_UserOutputValue( VARIANT_BOOL Value );`

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pValue* [out, retval]  
Pointer to the boolean value of the selected bit in the User Output register.  
*Value* [in]  
TRUE sets bit to High, FALSE sets bit to Low

#### Return Values

S\_OK  
Success  
E\_NOINTERFACE  
The feature is not available for the selected camera  
E\_FAIL  
Failure to set the feature value

#### Example

The following VB example demonstrates the setup of the bits in the User Output register:

```
ActiveGige1.UserOutputSelector="UserOutput0"  
ActiveGige1.UserOutputValue=TRUE  
ActiveGige1.UserOutputSelector="UserOutput1"  
ActiveGige1.UserOutputValue=FALSE
```

#### Remarks

This property works in combination with [UserOutputSelector](#). Using a corresponding [LineSource](#), the bits from the User Output register can be directed to a physical output line.

Note that the property is available only if the currently selected camera supports the *UserOutputValue* feature per GenICam standard.

---

### 3.1.86 UserSetSelector

#### Description

Returns or assigns the user set to load, save or configure. User sets allow for loading or saving factory or user-defined settings. Loading the factory default user set guarantees a state where a continuous acquisition can be started using only the mandatory features. Can be one of the following values:

*"Default" (or "Factory")*  
Selects the factory settings user set.  
*"UserSet1"*  
Selects the first user set.  
*"UserSet2"*  
Selects the second user set  
.....

#### Syntax

[VB]  
`objActiveGige.UserSetSelector [= Value]`

[C/C++]  
`HRESULT get_UserSetSelector( string *pValue );`  
`HRESULT put_UserSetSelector( string Value );`

#### Data Type [VB]

String

#### Parameters [C/C++]

*pValue* [out, retval]  
Pointer to the string specifying the user set.  
*Value* [in]  
The user set to be selected.

#### Return Values

`S_OK`  
Success  
`E_NOINTERFACE`  
The feature is not available for the selected camera  
`E_INVALIDARG`  
The value is not part of the enumerated set  
`E_FAIL`  
Failure to set the feature value

#### Example

The following VB example demonstrates how to load the first user set.

```
ActiveGig1.UserSetSelector="UserSet1"  
ActiveGig1.SetFeatureString( "UserSetLoad" , "Execute" )
```

**Remarks**

This feature should be used in combination with "UserSetLoad" and "UserSetSave" commands. To execute a command, use [SetFeatureString](#).

Depending on a camera model, other device-specific values may be available for this feature. The list of valid values can be retrieved using [GetEnumList](#).

Note that the property is available only if the currently selected camera supports the *UserSetSelector* feature per GenICam standard.

---

### 3.1.87 WebStream

#### Description

Enables/disables the RTSP web streaming. Used in combination with [SetWebStreamer](#).

#### Syntax

```
[VB]
objActiveGige.WebStream [= Value]
```

```
[C/C++]
HRESULT get_WebStream ( bool *pValue);
HRESULT put_WebStream( bool Value );
```

#### Data Type [VB]

Boolean

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the Boolean that is TRUE if the web streaming is enabled, or FALSE otherwise  
*Value* [in]  
Set to TRUE to enable the web streaming, or set to FALSE to disable it

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example sets the web streamer's parameters and activates the streaming:

```
sourceAddr="192.168.0.5:8554/ActiveGige.sdp"
destAddr="192.168.0.10"
streamFPS=15
streamQuality=4
ActiveGigel.SetWebStreamer sourceAddr, destAddr, streamFPS, streamQuality
ActiveGigel.Acquire=True
ActiveGigel.WebStream=True
```

#### Remarks

The web streaming option allows you to automatically convert video outputted by the camera into the H.264 compression format and transmit it over the wireless or wired network using the RTSP protocol to a remote playback devices, such as PCs, tablets and smartphones.

Before enabling the **WebStream** property, you should configure the web streaming parameters by calling [SetWebStreamer](#).

The [Acquire](#) property must be set to TRUE prior to activating the web streaming.

To watch a transmitted video on a remote playback device, use an RTSP/RTP client such as VLC Media Player (for Windows, Linux and MAC) or Fresh Video Player (for iOS devices). For more information refer to [SetWebStreamer](#).

*Note - the web streaming module utilizes Intel's H.264 encoder which takes advantage of Intel's hardware acceleration. Therefore running your ActiveGigE-based web streaming application on a PC with Intel graphics chipset will substantially improve the encoding performance.*

---

## 3.2 Methods

*ActiveGige* provides the following methods to a container application:

### Acquisition

<a href="#">Grab</a>	Grabs a single frame into the internal memory
<a href="#">SoftTrigger</a>	Generates an internal trigger signal
<a href="#">SetDestinationIP</a>	Sets the destination IP address and port for the specified stream channel
<a href="#">SetStreamChannel</a>	Sets the stream channel on which the video data will be received

### Information

<a href="#">GetCameraList</a>	Returns the list of GigE Vision devices connected to the system
<a href="#">GetCameraIPList</a>	Returns the list of IP addresses of connected GigE Vision devices
<a href="#">GetCameraMACList</a>	Returns the list of MAC addresses of connected GigE Vision devices
<a href="#">GetFormatList</a>	Returns the list of pixel formats supported by the currently selected camera
<a href="#">GetCameraIP</a>	Returns the IP address of the currently selected device
<a href="#">GetCameraMAC</a>	Returns the MAC address of the currently selected device
<a href="#">GetBitsPerChannel</a>	Returns the bit depth of each component of the internal image
<a href="#">GetBytesPerPixel</a>	Returns the byte depth of the internal image
<a href="#">GetWidthMax</a>	Returns the maximum available horizontal size of the video frame
<a href="#">GetHeightMax</a>	Returns the maximum available vertical size of the video frame
<a href="#">IsMasterApplication</a>	Checks if the application has an exclusive control over the selected device
<a href="#">GetOptimalPacketSize</a>	Returns the optimal packet size for the current network configuration

**Features**

<a href="#"><u>GetFeatureList</u></a>	Returns the array of features grouped under the specified category
<a href="#"><u>IsFeatureAvailable</u></a>	Checks availability of the specified camera feature
<a href="#"><u>GetFeature</u></a>	Returns the numerical value of the specified camera feature
<a href="#"><u>SetFeature</u></a>	Sets the numerical value of the specified camera feature
<a href="#"><u>GetFeature64</u></a>	Returns the numerical value of the specified 64-bit camera feature
<a href="#"><u>SetFeature64</u></a>	Sets the numerical value of the specified 64-bit camera feature
<a href="#"><u>GetFeatureString</u></a>	Returns the string value of the specified camera feature
<a href="#"><u>SetFeatureString</u></a>	Sets the string value of the specified camera feature
<a href="#"><u>GetFeatureArray</u></a>	Returns the array of values associated with the specified camera feature
<a href="#"><u>SetFeatureArray</u></a>	Sets the array of values associated with the specified camera feature
<a href="#"><u>GetFeatureMin</u></a>	Returns the minimum value allowed for the specified camera feature
<a href="#"><u>GetFeatureMax</u></a>	Returns the maximum value allowed for the specified camera feature
<a href="#"><u>GetFeatureIncrement</u></a>	Returns the increment value for the specified integer feature
<a href="#"><u>GetEnumList</u></a>	Returns the array of string values representing the specified enumerated feature
<a href="#"><u>GetFeatureAccess</u></a>	Returns the information on the access to the specified camera feature
<a href="#"><u>GetFeatureDependents</u></a>	Returns the names of features dependent on the specified camera feature
<a href="#"><u>GetFeatureTip</u></a>	Returns the short description of the specified camera feature
<a href="#"><u>GetFeatureDescription</u></a>	Returns the detailed description of the specified camera feature
<a href="#"><u>GetFeatureType</u></a>	Returns the type of the specified camera feature
<a href="#"><u>GetFeatureVisibility</u></a>	Returns the information on the suggested visibility for the specified camera feature
<a href="#"><u>GetFeatureRepresentation</u></a>	Returns the information on the suggested representation for the specified camera feature



<a href="#"><u>GetAcquisitionFrameRateMax</u></a>	Returns the maximum value allowed for the camera's frame rate
<a href="#"><u>GetAcquisitionFrameRateMin</u></a>	Returns the minimum value allowed for the camera's frame rate
<a href="#"><u>GetBalanceRatioMax</u></a>	Returns the maximum value allowed for the white balance ratio
<a href="#"><u>GetBalanceRatioMin</u></a>	Returns the minimum value allowed for the white balance ratio
<a href="#"><u>GetBlackLevelMax</u></a>	Returns the maximum value allowed for the camera's black level
<a href="#"><u>GetBalanceRatioMin</u></a>	Returns the minimum value allowed for the camera's black level
<a href="#"><u>GetExposureTimeMax</u></a>	Returns the maximum value allowed for the camera's exposure time
<a href="#"><u>GetExposureTimeMin</u></a>	Returns the minimum value allowed for the camera's exposure time
<a href="#"><u>GetGainMax</u></a>	Returns the maximum value allowed for the camera's gain
<a href="#"><u>GetGainMin</u></a>	Returns the minimum value allowed for the camera's gain
<a href="#"><u>GetTriggerDelayMax</u></a>	Returns the maximum value allowed for the camera's trigger delay
<a href="#"><u>GetTriggerDelayMin</u></a>	Returns the minimum value allowed for the camera's trigger delay

**Image Access**

<a href="#"><u>GetPixel</u></a>	Returns the pixel value at the specified coordinates
<a href="#"><u>GetRGBPixel</u></a>	Returns the array of RGB values at the specified coordinates
<a href="#"><u>GetImageLine</u></a>	Returns the array of pixel values in the specified horizontal line
<a href="#"><u>GetComponentLine</u></a>	Returns the array of pixel values of the color component in the specified horizontal line
<a href="#"><u>GetImageWindow</u></a>	Returns the 2D array of pixel values in the specified rectangular area of the current frame
<a href="#"><u>SetImageWindow</u></a>	Copies the 2D array of pixel values to the selected window of the current frame
<a href="#"><u>GetImageData</u></a>	Returns the two dimensional array of pixel values in the currently acquired frame
<a href="#"><u>GetComponentData</u></a>	Returns the two dimensional array of pixel values in the specified color component
<a href="#"><u>GetRawData</u></a>	Returns the 2D array of raw values in the currently acquired data buffer
<a href="#"><u>GetImagePointer</u></a>	Returns the memory pointer to the specified pixel
<a href="#"><u>GetDIB</u></a>	Returns the handler to a Device Independent Bitmap
<a href="#"><u>GetPicture</u></a>	Returns the Picture object corresponding to the currently acquired frame
<a href="#"><u>GetChunkPointer</u></a>	Returns the pointer to the data associated with the specified chunk feature
<a href="#"><u>GetChunkSize</u></a>	Returns the size of the data associated with the specified chunk feature

**Image and Video Capture**

<a href="#">SaveImage</a>	Saves the current frame buffer in the specified image file
<a href="#">LoadImage</a>	Loads and displays the image from the specified image file
<a href="#">StartCapture</a>	Starts video capture to the specified AVI file or series of image files
<a href="#">StopCapture</a>	Stops video capture
<a href="#">GetCodecList</a>	Returns the names of video codecs available in the system
<a href="#">GetCodec</a>	Return the name of the currently selected video codec
<a href="#">SetCodec</a>	Sets the video codec to be used for the AVI capture
<a href="#">ShowCodecDlg</a>	Shows the configuration dialog of the currently selected video codec
<a href="#">ShowCompressionDlg</a>	Shows the video compression dialog

**Advanced Video Recording (DVR version only)**

<a href="#"><u>CreateVideo</u></a>	Creates an AVI file for the video recording and preallocates its size
<a href="#"><u>StartVideoCapture</u></a>	Starts time-lapse video capture to the previously created AVI file
<a href="#"><u>StopVideoCapture</u></a>	Stops video capture to the AVI file
<a href="#"><u>SetCodecProperties</u></a>	Sets the generic parameters of the currently selected compression codec
<a href="#"><u>GetCodecProperties</u></a>	Returns the generic parameters of the currently selected compression codec
<a href="#"><u>GetAudioList</u></a>	Returns the names of audio recording devices available in the system
<a href="#"><u>SetAudioSource</u></a>	Sets the index of the audio recording device to be used during the AVI capture
<a href="#"><u>GetAudioSource</u></a>	Returns the index of the currently selected audio recording device for the AVI capture
<a href="#"><u>ShowAudioDlg</u></a>	Displays the audio Input dialog for adjusting the mixer properties of the audio source
<a href="#"><u>SetAudioLevel</u></a>	Sets the recording level of the currently selected audio device
<a href="#"><u>GetAudioLevel</u></a>	Returns the recording level of the currently selected audio device
<a href="#"><u>SetWebStreamer</u></a>	Configures parameters of the RTSP web streaming

**Sequence Capture (DVR version only)**

<a href="#"><u>CreateSequence</u></a>	Allocates memory for capturing a sequence of frames
<a href="#"><u>StartSequenceCapture</u></a>	Starts acquiring a sequence of frames into the memory
<a href="#"><u>StopSequenceCapture</u></a>	Stops acquiring a sequence of frames into the memory
<a href="#"><u>GetSequenceFrameCount</u></a>	Returns the current number of frames in the memory sequence
<a href="#"><u>SaveSequence</u></a>	Saves the memory sequence in the specified video file
<a href="#"><u>LoadSequence</u></a>	Loads the specified video file into the memory sequence
<a href="#"><u>GetSequenceWindow</u></a>	Returns 2D-array of pixel values of the selected window in a frame in the memory sequence
<a href="#"><u>GetSequenceRawData</u></a>	Returns 2D-array of raw pixel values in a frame in the memory sequence
<a href="#"><u>GetSequencePixel</u></a>	Returns the pixel value in the selected coordinates of the frame in the memory sequence
<a href="#"><u>GetSequencePointer</u></a>	Returns the memory pointer to a selected pixel of the frame in the memory sequence
<a href="#"><u>GetSequencePicture</u></a>	Returns the Picture object corresponding to a frame in the memory sequence
<a href="#"><u>GetSequenceTimestamp</u></a>	Returns the timestamp of the selected frame in the memory sequence

**Video and Sequence Playback (DVR version only)**

<a href="#"><u>OpenVideo</u></a>	Opens the specified video file or memory sequence for the playback
<a href="#"><u>PlayVideo</u></a>	Plays the currently open video file or memory sequence
<a href="#"><u>StopVideo</u></a>	Stops the playback of the video file or memory sequence
<a href="#"><u>CloseVideo</u></a>	Closes the currently open video file
<a href="#"><u>GetVideoFrameCount</u></a>	Returns the number of the frame in the currently open video file or memory sequence
<a href="#"><u>GetVideoPosition</u></a>	Returns the position of the current frame in the open video file or memory sequence
<a href="#"><u>SetVideoPosition</u></a>	Seeks, extracts and display the specified frame from the open video file or sequence
<a href="#"><u>GetVideoFPS</u></a>	Returns the frame rate of the currently opened video file or memory sequence
<a href="#"><u>SetVideoFPS</u></a>	Sets the frame rate of the currently opened video file or memory sequence
<a href="#"><u>GetVideoVolume</u></a>	Returns the playback volume level of the currently open AVI file
<a href="#"><u>SetVideoVolume</u></a>	Sets the playback volume level of the currently open AVI file
<a href="#"><u>SetVideoSync</u></a>	Sets the playback synchronization mode of the currently open AVI file
<a href="#"><u>TriggerVideo</u></a>	Advances the AVI playback to the next frame

**Drawing**

<a href="#">Draw</a>	Displays the current frame in <i>ActiveGige</i> window.
<a href="#">OverlayClear</a>	Clears graphics and text from the overlay
<a href="#">OverlayEllipse</a>	Draws an empty or filled ellipse in the overlay
<a href="#">OverlayLine</a>	Draws a line in the overlay
<a href="#">OverlayPixel</a>	Draws a pixel in the overlay
<a href="#">OverlayRectangle</a>	Draws an empty or filled rectangle in the overlay
<a href="#">OverlayText</a>	Draws a string of text in the overlay
<a href="#">DrawPixel</a>	Draws a pixel in the current image frame
<a href="#">DrawLine</a>	Draws a line in the current image frame
<a href="#">DrawRectangle</a>	Draws a rectangle in the current image frame
<a href="#">DrawEllipse</a>	Draws an ellipse in the current image frame
<a href="#">DrawText</a>	Draws a string of text in the current image frame
<a href="#">DrawAlphaClear</a>	Clears the alpha plane
<a href="#">DrawAlphaPixel</a>	Draws a pixel in the alpha plane
<a href="#">DrawAlphaLine</a>	Draws a line in the alpha plane
<a href="#">DrawAlphaRectangle</a>	Draws an empty or filled rectangle in the alpha plane
<a href="#">DrawAlphaEllipse</a>	Draws an empty or filled ellipse in the alpha plane
<a href="#">DrawAlphaText</a>	Draws a srting of text in the alpha plane

**Image Processing**

<a href="#"><u>SaveBkg</u></a>	Stores a dark or bright background image on the hard drive
<a href="#"><u>SetROI</u></a>	Sets the rectangular region of interest and luminance range
<a href="#"><u>GetROI</u></a>	Returns the settings of the currently selected ROI
<a href="#"><u>GetImageStat</u></a>	Returns the array containing statistical data of the current image frame
<a href="#"><u>GetHistogram</u></a>	Returns the histogram of the current image frame
<a href="#"><u>SetLUT</u></a>	Assigns the array of values for the software lookup table
<a href="#"><u>GetLUT</u></a>	Returns the array of values for the software lookup table
<a href="#"><u>SetGains</u></a>	Set the levels for the software gain control
<a href="#"><u>SetLevels</u></a>	Set the minimum and maximum levels for the Window/Level operation
<a href="#"><u>GetLevels</u></a>	Returns the minimum and maximum levels for the Window/Level operation
<a href="#"><u>SetColorMatrix</u></a>	Sets the matrix coefficients for the color correction operation
<a href="#"><u>SetLensDistortion</u></a>	Sets distortion parameters for the lens distortion correction
<a href="#"><u>GetBarcode</u></a>	Returns the character string decoded from a barcode found in the current frame



**File Access**

<a href="#"><u>GetFileList</u></a>	Returns the array of names of all files hosted in the device
<a href="#"><u>GetFileSize</u></a>	Returns the size of the specified file in the device
<a href="#"><u>GetFileAccessMode</u></a>	Returns the access mode in which the specified file can be opened in the device
<a href="#"><u>ReadFile</u></a>	Transfers the indicated amount of bytes from the specified file in the device
<a href="#"><u>WriteFile</u></a>	Transfers the indicated amount of bytes to the specified file in the device
<a href="#"><u>GetFileTransferProgress</u></a>	Returns the progress of the current file access operation in percent

**Utilities**

<a href="#"><u>GetFPS</u></a>	Returns the actual frame rate of the camera
<a href="#"><u>GetFPSAcquired</u></a>	Returns the acquired (displayed) frame rate of the application
<a href="#"><u>GetTimestamp</u></a>	Returns the timestamp of the last captured frame
<a href="#"><u>GetBlockId</u></a>	Returns the block ID of the last captured frame
<a href="#"><u>ShowProperties</u></a>	Displays property pages in run-time
<a href="#"><u>ReadRegister</u></a>	Reads a 32-bit value from the specified bootstrap register of the currently selected camera
<a href="#"><u>WriteRegister</u></a>	Write a 32-bit value to the specified bootstrap register of the currently selected camera
<a href="#"><u>ReadBlock</u></a>	Reads the block of data from the camera starting from the specified bootstrap address
<a href="#"><u>WriteBlock</u></a>	Writes the block of data to the camera starting from the specified bootstrap address
<a href="#"><u>LoadSettings</u></a>	Loads previously saved camera settings from the data file
<a href="#"><u>SaveSettings</u></a>	Stores the camera settings in the data file
<a href="#"><u>SetControlKey</u></a>	Sets the numerical access key for the Control with Switchover access to the camera
<a href="#"><u>SetDeviceKey</u></a>	Sets the device key register for action commands
<a href="#"><u>SetActionConditions</u></a>	Configures the asserting conditions of the specified action task
<a href="#"><u>SendActionCommand</u></a>	Broadcasts the action command to all available network interfaces
<a href="#"><u>GetActionAcknowledgeInfo</u></a>	Returns the array containing IP addresses of devices that acknowledged action command

### 3.2.1 CloseVideo

#### Description

Closes the video file currently opened by [OpenVideo](#).

#### Syntax

[VB]  
`objActiveGige.CloseVideo`

[C/C++]  
`HRESULT CloseVideo( );`

#### Parameters [C/C++]

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example opens an AVI file, plays it and closes the file when the playback is finished using the [PlayCompleted](#) event.

```
Private Sub Play_Click()  
ActiveGige1.OpenVideo "C:\\video1.avi"  
ActiveGige1.PlayVideo  
End Sub
```

```
Private Sub ActiveGige1_PlayCompleted (ByVal Frames As Long)  
ActiveGige1.CloseVideo  
End Sub
```

#### Remarks

If the video file is currently being played, this method will stop the playback and close the file.

### 3.2.2 CreateSequence

#### Description

Allocates memory for capturing a sequence of frames. Used in combination with [StartSequenceCapture](#) and [StopSequenceCapture](#).

#### Syntax

[VB]  
`objActiveGige.CreateSequence Frames`

[C/C++]  
`HRESULT CreateSequence( long Frames);`

#### Data Types [VB]

*Frames* : long

#### Parameters [C/C++]

*Frames* [in]  
The maximum number of frames to be allocated for the memory sequence.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.  
E\_OUTOFMEMORY  
Not enough memory

#### Example

This VB example demonstrates how to allocate enough memory for the sequence capture:

```
Private Sub Form_Load()  
ActiveGige1.CreateSequence 1000  
ActiveGige1.Acquire = True  
End Sub  
  
Private Sub StartButton_Click()  
ActiveGige1.StartSequenceCapture 800  
End Sub  
  
Private Sub StopButton_Click()  
ActiveGige1.StopSequenceCapture  
End Sub
```

#### Remarks

Using this method allows you to avoid an unnecessary delay when calling [StartSequenceCapture](#). If

---

---

enough memory has been allocated with **CreateSequence**, the [StartSequenceCapture](#) will start acquiring frames into the memory immediately after being called.

The amount of memory allocated with `CreateSequence` must not exceed the limit allowed for an application in your operating system. Such an amount is typically limited to 2 GB for a 32-bit version of Windows. If you use Windows 64-bit, it is recommended to keep the amount of allocated memory below the physical size of RAM in order to avoid the use of the virtual (hard-drive) space.

If the amount of memory you try to allocate exceeds the limit allowed for your application, this method will return an error.

---

### 3.2.3 CreateVideo

#### Description

Creates an AVI file for the video recording and preallocates its size. Used in combination with [StartVideoCapture](#) and [StopVideoCapture](#).

#### Syntax

[VB]  
`objActiveGige.CreateVideo File [, Size = 0]`

[C/C++]  
`HRESULT CreateVideo( bstr File, long Size);`

#### Data Types [VB]

*File* : String  
*Size* : Long (optional)

#### Parameters [C/C++]

*File* [in]  
The string containing the path to the AVI file to be created.  
*Size* [in]  
The size of the file to allocate, in bytes.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure  
E\_INVALIDARG  
Invalid file name  
E\_OUTOFMEMORY  
Not enough space to allocate

#### Example

This VB example demonstrates how to preallocate an AVI file with the size of 80 MBytes and perform the video capture.

```
Private Sub LoadForm()  
ActiveGige1.CreateVideo "c:\\mycapture.avi", 80000000  
ActiveGige1.Acquire = True  
End Sub
```

```
Private Sub StartButton_Click()  
ActiveGige1.StartVideoCapture  
End Sub
```

---

```
Private Sub StopButton_Click()  
ActiveGige1.StopVideoCapture  
End Sub
```

**Remarks**

Use this method to create an AVI file and perform an initialization of the currently selected video codec prior to calling [StartVideoCapture](#).

It is recommended to allocate enough space to accommodate the length of a typical video recording. If the allocated size is not sufficient, the video capture engine will append the AVI file on the fly, but this may reduce the recording throughput and result to frames being dropped.

### 3.2.4 Draw

#### Description

Displays the current frame in *ActiveGige* window.

#### Syntax

[VB]  
`objActiveGige.Draw`

[C/C++]  
`HRESULT Draw();`

#### Parameters

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example uses the [FrameAcquired](#) event to invert pixel value in the bottom left corner of the current frame and display the processed frame in real time.

```
Private Sub Form_Load()  
ActiveGigel.Display = False  
ActiveGigel.Acquire = True  
End Sub  
  
Private Sub ActiveGigel_FrameAcquired()  
a = ActiveGigel.GetImageData  
For x = 0 To 200  
For y = 0 To 200  
a(x, y) = 255 - a(x, y)  
Next  
Next  
ActiveGigel.Draw  
End Sub
```

#### Remarks

Use this method when the automatic live display is disabled ([Display](#) is set to *False*). This is typically done when you want to perform real time image processing and display the processed image in the control window.

---



### 3.2.5 DrawAlphaClear

#### Description

Clears graphics and text from the alpha plane and resets the plane to the fully transparent state.

#### Syntax

[VB]  
`objActiveGige.DrawAlphaClear`

[C/C++]  
`HRESULT DrawAlphaClear();`

#### Parameters

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

The following VB example moves a transparent red rectangle over the live image by repeatedly erasing and drawing it:

```
Dim x As Integer

Private Sub Form_Load()
    x = 0
    ActiveGigel.Acquire = True
    ActiveGigel.Alpha = True
End Sub

Private Sub ActiveGigel_FrameAcquired()
    ActiveGigel.DrawAlphaClear
    ActiveGigel.DrawAlphaRectangle x, 10, x + 50, 80, 0, 255,0,0,50
    x = x + 2
    If x = ActiveGigel.SizeX Then
        x = 0
    End If
End Sub
```

#### Remarks

To create animation effects, use this method in combination with [DrawAlphaPixel](#), [DrawAlphaLine](#), [DrawAlphaRectangle](#), [DrawAlphaEllipse](#), [DrawAlphaText](#). For more information on the alpha-plane drawing refer to [Alpha](#).

### 3.2.6 DrawAlphaEllipse

#### Description

Draws an empty or filled ellipse in the alpha plane over the video window.

#### Syntax

[VB]

```
objActiveGige.DrawAlphaElliplse X1, Y1, X2, Y2, Width, Red , Green, Blue [, Opacity]
```

[C/C++]

```
HRESULT DrawAlphaEllipse( short X1, short Y2, short X2, short Y2, short Width, long Red , long Green, long Blue [, short Opacity ] );
```

#### Data Types [VB]

*X1, Y1, X2, Y2* : Integer

*Width*: Integer

*Red, Green, Blue*: Long

*Opacity*: Integer

#### Parameters [C/C++]

*X1* [in], *Y1* [in]

Coordinates of the top left corner of the ellipse's bounding rectangle relative to the image origin.

*X2* [in], *Y2* [in]

Coordinates of the bottom right corner of the ellipse's bounding rectangle relative to the image origin.

*Width* [in]

Width of the line in pixels. If zero, a filled ellipse will be drawn.

*Red* [in], *Green* [in], *Blue* [in]

Values specifying the color or intensity of the ellipse.

*Opacity* [in]

Value in the range 1-100 specifying the percentage of opacity at which the ellipse will be blended with the underlying pixels in the image.

If 0, the pixels of the ellipse will be reset to the fully transparent state.

If omitted, default value of 100 is used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example draws a semi-transparent filled red ellipse in the alpha-plane over the live video:

```
ActiveGigel.Acquire = True
ActiveGigel.DrawAlphaEllipse 100,100,200,200,0,255,0,0,50
```

---

```
ActiveGigel.Alpha = True
```

**Remarks**

To draw multiple ellipses, call this method several times. For more information on the alpha-plane drawing refer to [Alpha](#).

### 3.2.7 DrawAlphaLine

#### Description

Draws a line in the alpha plane over the video window.

#### Syntax

[VB]

```
objActiveGige.DrawAlphaLine X1, Y1, X2, Y2, Width, Red , Green, Blue [,
Opacity]
```

[C/C++]

```
HRESULT DrawAlphaLine( short X1, short Y1, short X2, short Y2, short Width,
long Red , long Green, long Blue [, short Opacity ]);
```

#### Data Types [VB]

*X1, Y1, X2, Y2* : Integer

*Width*: Integer

*Red, Green, Blue*: Long

*Opacity*: Integer

#### Parameters [C/C++]

*X1* [in], *Y1* [in], *X2* [in], *Y2* [in]

Coordinates of the starting and ending point of the line relative to the image origin.

*Width* [in]

Width of the line in pixels.

*Red* [in], *Green* [in], *Blue* [in]

Values specifying the color or intensity of the line.

*Opacity* [in]

Value in the range 1-100 specifying the percentage of opacity at which the line will be blended with the underlying pixels in the image.

If 0, the pixels of the line will be reset to the fully transparent state.

If omitted, default value of 100 is used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example draws a solid green line of the width of 3 in the alpha-plane over the live video:

```
ActiveGige1.Acquire = True
ActiveGige1.DrawAlphaLine 100,100,200,200,3,0,255,0
ActiveGige1.Alpha = True
```

---

**Remarks**

To draw multiple lines, call this method several times. For more information on the alpha-plane drawing refer to [Alpha](#).

### 3.2.8 DrawAlphaPixel

#### Description

Draws a pixel in the alpha plane over the live video.

#### Syntax

[VB]

```
objActiveGige.DrawAlphaPixel X, Y, Red, Green, Blue [,Opacity ]
```

[C/C++]

```
HRESULT DrawAlphaPixel( short X, short Y, long Red,long Green, long Blue  
[,short Opacity ]);
```

#### Data Types [VB]

*X, Y*: Integer

*Red, Green, Blue*: Long

*Opacity*: Integer

#### Parameters [C/C++]

*X* [in], *Y* [in]

Coordinates of the pixel relative to the image origin.

*Red* [in], *Green* [in], *Blue* [in]

Values specifying the color or intensity of the pixel.

*Opacity* [in]

Value in the range 1-100 specifies the percentage of opacity at which the pixel will be blended with the underlying pixels in the image.

If 0, the pixel will be reset to the fully transparent state.

If omitted, default value of 100 is used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example draws a blue pixel with the 80% opacity in the alpha-plane:

```
ActiveGige1.Acquire = True  
ActiveGige1.DrawAlphaPixel 100,100,200,200,0,0,255,80  
ActiveGige1.Alpha = True
```

#### Remarks

To draw multiple pixels, call this method repeatedly. For more information on the alpha-plane drawing refer to [Alpha](#).

---

### 3.2.9 DrawAlphaRectangle

#### Description

Draws an empty or filled rectangle in the alpha plane over the video window.

#### Syntax

[VB]

```
objActiveGige.DrawAlphaRectangle X1, Y1, X2, Y2, Width, Red , Green, Blue  
[, Opacity]
```

[C/C++]

```
HRESULT DrawAlphaRectangle( short X1, short Y2, short X2, short Y2, short  
Width, long Red , long Green, long Blue [, short Opacity ] );
```

#### Data Types [VB]

*X1, Y1, X2, Y2* : Integer

*Width*: Integer

*Red, Green, Blue*: Long

*Opacity*: Integer

#### Parameters [C/C++]

*X1* [in], *Y1* [in]

Coordinates of the top left corner of the rectangle's bounding rectangle relative to the image origin.

*X2* [in], *Y2* [in]

Coordinates of the bottom right corner of the rectangle's bounding rectangle relative to the image origin.

*Width* [in]

Width of the line in pixels. If zero, a filled rectangle will be drawn.

*Red* [in], *Green* [in], *Blue* [in]

Values specifying the color or intensity of the rectangle.

*Opacity* [in]

Value in the range 1-100 specifying the percentage of opacity at which the ellipse will be blended with the underlying pixels in the image.

If 0, the pixels of the rectangle will be reset to the fully transparent state.

If omitted, default value of 100 is used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example draws a semi-transparent filled red rectangle in the alpha-plane over the live video:

```
ActiveGigel.Acquire = True  
ActiveGigel.DrawAlphaRectangle 100,100,200,200,0,255,0,0,50
```

```
ActiveGigel.Alpha = True
```

**Remarks**

To draw multiple rectangles, call this method several times. For more information on the alpha-plane drawing refer to [Alpha](#).

---



### 3.2.10 DrawAlphaText

#### Description

Embeds a string of text in the alpha plane over the live video.

#### Syntax

[VB]

```
objActiveGige.DrawAlphaText X, Y, Text, Red, Green, Blue [,Opacity ]
```

[C/C++]

```
HRESULT DrawAlphaText( short X, short Y, bstr Text, long Red, long Green,  
long Blue [,short Opacity ] );
```

#### Data Types [VB]

*X, Y*: Integer

*Text*: String

*Red, Green, Blue*: Long

*Opacity*: Integer

#### Parameters [C/C++]

*X* [in], *Y* [in]

Coordinates of the beginning of the text relative to the image origin.

*Text* [in]

The string containing the text to be drawn.

*Red* [in], *Green* [in], *Blue* [in]

Values specifying the color or intensity of the text.

*Opacity* [in]

Value in the range 1-100 specifies the percentage of opacity at which the ellipse will be blended with the underlying pixels in the image.

If 0, the pixels of the text area will be reset to the fully transparent state.

If omitted, default value of 100 is used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example draws a semi-transparent yellow text in the alpha plane:

```
ActiveGigel.Acquire = True  
ActiveGigel.DrawAlphaText 50,100, "ActiveGige SDK for 1394 cameras", 255,255,0,50  
ActiveGigel.Alpha = True
```

#### Remarks

To select the font for drawing text strings in the alpha plane, use the [OverlayFont](#) property. To draw

multiple strings, call this method several times. For more information on the alpha-plane drawing refer to [Alpha](#).

---

### 3.2.11 DrawEllipse

#### Description

Draws an empty or filled ellipse in the current image frame.

#### Syntax

[VB]

```
objActiveGige.DrawElliplse X1, Y1, X2, Y2, Width, Red [,Green, Blue,  
Opacity ]
```

[C/C++]

```
HRESULT DrawEllipse( short X1, short Y1, short X2, short Y2, short Width,  
long Red [,long Green, long Blue, short Opacity ]);
```

#### Data Types [VB]

*X1, Y1, X2, Y2* : Integer

*Width*: Integer

*Red, Green, Blue*: Long

*Opacity*: Integer

#### Parameters [C/C++]

*X1* [in], *Y1* [in]

Coordinates of the top left corner of the ellipse's bounding rectangle relative to the image origin.

*X2* [in], *Y2* [in]

Coordinates of the bottom right corner of the ellipse's bounding rectangle relative to the image origin.

*Width* [in]

Width of the line in pixels. If zero, a filled ellipse will be drawn.

*Red* [in], *Green* [in], *Blue* [in]

Values specifying the color or intensity of the ellipse. If *Green* and *Blue* arguments are omitted, they will be assigned the value of *Red*.

*Opacity* [in]

An optional integer between 0 and 100 specifying the percentage of opacity at which the ellipse will be blended with the underlying pixels in the image. If omitted, default value of 100 is used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example embeds a semi-transparent filled ellipse into the live video:

```
Private Sub ActiveGige1_FrameAcquired()  
    ActiveGige1.DrawEllipse 50,100, 300,200, 0, 0, 0, 255, 50  
End Sub
```

**Remarks**

To draw multiple ellipses, call this method several times.

---

### 3.2.12 DrawLine

#### Description

Draws a line in the current image frame.

#### Syntax

[VB]

```
objActiveGige.DrawLine X1, Y1, X2, Y2, Width, Red [,Green, Blue, Opacity ]
```

[C/C++]

```
HRESULT DrawLine( short X1, short Y1, short X2, short Y2, short Width, long Red [,long Green, long Blue, short Opacity ]);
```

#### Data Types [VB]

*X1, Y1, X2, Y2* : Integer

*Width*: Integer

*Red, Green, Blue*: Long

*Opacity*: Integer

#### Parameters [C/C++]

*X1* [in], *Y1* [in], *X2* [in], *Y2* [in]

Coordinates of the starting and ending point of the line relative to the image origin.

*Width* [in]

Width of the line in pixels.

*Red* [in], *Green* [in], *Blue* [in]

Values specifying the color or intensity of the line. If *Green* and *Blue* arguments are omitted, they will be assigned the value of *Red*.

*Opacity* [in]

An optional integer between 0 and 100 specifying the percentage of opacity at which the line will be blended with the underlying pixels in the image. If omitted, default value of 100 is used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example embeds a line into the live video:

```
Private Sub ActiveGige1_FrameAcquired()  
    ActiveGige1.DrawLine 50,100, 300,200, 3, 255  
End Sub
```

#### Remarks

To draw multiple lines, call this method several times.

### 3.2.13 DrawPixel

#### Description

Draws a pixel in the current image frame.

#### Syntax

[VB]

```
objActiveGige.DrawPixel X, Y, Red [,Green, Blue, Opacity ]
```

[C/C++]

```
HRESULT DrawPixel( short X, short Y, long Red [,long Green, long Blue,  
short Opacity ]);
```

#### Data Types [VB]

*X, Y*: Integer

*Red, Green, Blue*: Long

*Opacity*: Integer

#### Parameters [C/C++]

*X* [in], *Y* [in]

Coordinates of the pixel relative to the image origin.

*Red* [in], *Green* [in], *Blue* [in]

Values specifying the color or intensity of the pixel. If *Green* and *Blue* arguments are omitted, they will be assigned the value of *Red*.

*Opacity* [in]

An optional integer between 0 and 100 specifying the percentage of opacity at which the pixel will be blended with the underlying pixel in the image. If omitted, default value of 100 is used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example embeds a pixel into the live video:

```
Private Sub ActiveGige1_FrameAcquired()  
    ActiveGige1.DrawPixel 50,100, 255  
End Sub
```

#### Remarks

To draw multiple pixels, call this method several times.

---

### 3.2.14 DrawRectangle

#### Description

Draws an empty or filled rectangle in the current image frame.

#### Syntax

[VB]

```
objActiveGige.DrawRectangle X1, Y1, X2, Y2, Width, Red [,Green, Blue,  
Opacity ]
```

[C/C++]

```
HRESULT DrawRectangle( short X1, short Y1, short X2, short Y2, short Width,  
long Red [,long Green, long Blue, short Opacity ]);
```

#### Data Types [VB]

*X1, Y1, X2, Y2* : Integer

*Width*: Integer

*Red, Green, Blue*: Long

*Opacity*: Integer

#### Parameters [C/C++]

*X1* [in], *Y1* [in]

Coordinates of the top left corner of the rectangle relative to the image origin.

*X2* [in], *Y2* [in]

Coordinates of the bottom right corner of the rectangle relative to the image origin.

*Width* [in]

Width of the line in pixels. If zero, a filled rectangle will be drawn.

*Red* [in], *Green* [in], *Blue* [in]

Values specifying the color or intensity of the rectangle. If *Green* and *Blue* arguments are omitted, they will be assigned the value of *Red*.

*Opacity* [in]

An optional integer between 0 and 100 specifying the percentage of opacity at which the rectangle will be blended with the underlying pixels in the image. If omitted, default value of 100 is used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example embeds a semi-transparent filled rectangle into the live video:

```
Private Sub ActiveGige1_FrameAcquired()  
    ActiveGige1.DrawRectangle 50,100, 300,200, 0, 255, 0, 0, 50  
End Sub
```

#### Remarks

To draw multiple rectangles, call this method several times.



### 3.2.15 DrawText

#### Description

Embeds a string of text in the current image frame.

#### Syntax

[VB]

```
objActiveGige.DrawText X, Y, Text, Red [,Green, Blue, Opacity ]
```

[C/C++]

```
HRESULT DrawText( short X, short Y, bstr Text, long Red [,long Green, long Blue, short Opacity ]);
```

#### Data Types [VB]

*X, Y*: Integer

*Text*: String

*Red, Green, Blue*: Long

*Opacity*: Integer

#### Parameters [C/C++]

*X* [in], *Y* [in]

Coordinates of the beginning of the text relative to the image origin.

*Text* [in]

The string containing the text to be drawn.

*Red* [in], *Green* [in], *Blue* [in]

Values specifying the color or intensity of the text. If *Green* and *Blue* arguments are omitted, they will be assigned the value of *Red*.

*Opacity* [in]

An optional integer between 0 and 100 specifying the percentage of opacity at which the text will be blended with the underlying pixels in the image. If omitted, default value of 100 is used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example embeds a string of text into the live video:

```
Private Sub ActiveGige1_FrameAcquired()  
    ActiveGige1.DrawText 50,100, "ActiveGige SDK for GigE Vision cameras", 255  
End Sub
```

#### Remarks

To select the font for drawing text strings, use [Font](#). To draw multiple strings, call this method several times.

### 3.2.16 GetActionAcknowledgeInfo

#### Description

Returns the array containing IP addresses of devices that acknowledged the [Action Command](#).

#### Syntax

[VB]

```
Value=objActiveGige.GetActionAcknowledgeInfo
```

[C/C++]

```
HRESULT GetActionAcknowledgeInfo ( VARIANT* pIP );
```

#### Data Types [VB]

*Return value:* Variant (Array of Long values)

#### Parameters [C/C++]

*pIP* [out,retval]

Pointer to the SAFEARRAY (unsigned long) containing the IP addresses of devices having acknowledged the Action command.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This MFC code fragment send the action command to the network and retrieves the IP addresses of devices that acknowledged the action command:

```
#include <Winsock2.h>
#include <sstream>

m_ActiveGige->SendActionCommand(0, 0, 1, 3, 0);

VARIANT ipList = m_ActiveGige->GetActionAcknowledgeInfo();

long nElementCount=0;
char *SafeArray;
unsigned long* pBuffer = new unsigned long[nElementCount];

SafeArrayGetUBound(ipList.parray,1,&nElementCount);
SafeArrayAccessData (ipList.parray, (void**)&SafeArray);
memcpy(pBuffer,SafeArray, nElementCount * sizeof(int));
SafeArrayUnaccessData (ipList.parray);
stringstream ss;
for (int i=0; i<nElementCount; i++)
```

```
{  
  in_addr ip; ip.S_un.S_addr = pBuffer[i];  
  ss << inet_ntoa(ip) << "\n";  
}
```

**Remarks**

When device asserts the action command, it typically sends an acknowledge response back to the application. This method allows you to check the amount of devices that acknowledged the action command and retrieve their IP addresses. For more information refer to [SendActionCommand](#).

### 3.2.17 GetAcquisitionFrameRateMax

#### Description

Returns the maximum value allowed for the camera's frame rate.

#### Syntax

[VB]

```
Value=objActiveGige.GetAcquistionFrameRateMax
```

[C/C++]

```
HRESULT GetAcquisitionFrameRateMax( float* pValue );
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the maximum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGige1.GetAcquisitionFrameRateMin  
Slider1.Max = ActiveGige1.GetAcquisitionFrameRateMax  
Slider1.Value = ActiveGige1.AcquisitionFrameRate  
End Sub
```

#### Remarks

Note that this method is available only if the currently selected camera supports one of the following GenICam features: *AcquisitionFrameRate*, *AcquisitionFrameRateAbs*, *AcquisitionFrameRateRaw*.

---

### 3.2.18 GetAcquisitionFrameRateMin

#### Description

Returns the minimum value allowed for the camera's frame rate.

#### Syntax

[VB]

```
Value=objActiveGige.GetAcquistionFrameRateMin
```

[C/C++]

```
HRESULT GetAcquisitionFrameRateMin( float* pValue );
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the minimum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGigel.GetAcquisitionFrameRateMin  
Slider1.Max = ActiveGigel.GetAcquisitionFrameRateMax  
Slider1.Value = ActiveGigel.AcquisitionFrameRate  
End Sub
```

#### Remarks

Note that this method is available only if the currently selected camera supports one of the following GenICam features: *AcquisitionFrameRate*, *AcquisitionFrameRateAbs*, *AcquisitionFrameRateRaw*.

### 3.2.19 GetAudioLevel

#### Description

Returns the recording level of the currently selected audio device.

#### Syntax

[VB]  
`Value=objActiveGige.GetAudioLevel`

[C/C++]  
`HRESULT GetAudioLevel(short* pValue );`

#### Data Types [VB]

*Return value:* Integer

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the current audio recording level, in percent.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure

This VB example initiates an AVI recording with a sound track and uses a scroll bar to adjust the audio recording level.

```
Private Sub Form_Load()  
ActiveGige1.SetAudioSource 0  
HScroll1.Min=0  
HScroll1.Max=100  
HScroll1.Value=ActiveGige1.GetAudioLevel  
End Sub  
  
Private Sub Capture_Click  
ActiveGige1.StartCapture "c:\\capture_with_sound.avi"  
End Sub  
  
Private Sub HScroll1_Scroll()  
ActiveGige1.SetAudioLevel HScroll1.Value  
End Sub
```

#### Remarks

In order for this method to work, the audio recording device must be selected with [SetAudioSource](#).

---

### 3.2.20 GetAudioList

#### Description

Returns the array of strings containing the names of audio recording devices available in the system.

#### Syntax

[VB]

```
Value=objActiveGige.GetAudioList()
```

[C/C++]

```
HRESULT GetAudioList( VARIANT* pList );
```

#### Data Types [VB]

*Return value:* Variant (SAFEARRAY)

#### Parameters [C/C++]

*pList* [out,retval]

Pointer to the SAFEARRAY of strings containing available categories

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example initializes a combo box with the list of available audio recording devices:

```
AudioLst = ActiveGigel.GetAudioList
For i = 0 To UBound(AudioLst)
    Combo1.AddItem (AudioLst(i))
Next
Combo1.ListIndex = 0
```

#### Remarks

Note that in C/C++ applications it is required to call `SafeArrayDestroy()` to delete the SAFEARRAY returned by **GetAudioList**.

### 3.2.21 GetAudioSource

#### Description

Gets the index of the currently selected audio recording device for the AVI capture.

#### Syntax

[VB]

```
Value = objActiveGige.GetAudioSource
```

[C/C++]

```
HRESULT GetAudioSource(short* Index);
```

#### Data Types [VB]

*Return value:* Integer

#### Parameters [C/C++]

*pValue* [out,retval]

Pointer to the index of the currently selected audio device.

#### Return Values

S\_OK

Success

E\_FAIL

Failure to set the codec

#### Example

The following VB example prints the name of the currently selected audio source:

```
AudioLst = ActiveGigel.GetAudioList  
index=ActiveGigel.GetAudioSource  
MsgBox AudioLst(index)
```

#### Remarks

If the return value is -1, the audio recording is disabled.

---



### 3.2.22 GetBalanceRatioMax

#### Description

Returns the maximum value allowed for the [BalanceRatio](#) property. Used for the white balance control.

#### Syntax

[VB]

```
Value=objActiveGige.GetBalanceRatioMax
```

[C/C++]

```
HRESULT GetBalanceRatioMax( float* pValue );
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the maximum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGigel.GetBalanceRatioMin  
Slider1.Max = ActiveGigel.GetBalanceRatioMax  
Slider1.Value = ActiveGigel.BalanceRatio  
End Sub
```

#### Remarks

Note that this method is available only if the currently selected camera supports one of the following GenICam features: *BalanceRatio*, *BalanceRatioAbs*, *BalanceRatioRaw*.

### 3.2.23 GetBalanceRatioMin

#### Description

Returns the minimum value allowed for the [BalanceRatio](#) property. Used for the white balance control.

#### Syntax

[VB]

```
Value=objActiveGige.GetBalanceRatioMin
```

[C/C++]

```
HRESULT GetBalanceRatioMin( float* pValue );
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the minimum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGigel.GetBalanceRatioMin  
Slider1.Max = ActiveGigel.GetBalanceRatioMax  
Slider1.Value = ActiveGigel.BalanceRatio  
End Sub
```

#### Remarks

Note that this method is available only if the currently selected camera supports one of the following GenICam features: *BalanceRatio*, *BalanceRatioAbs*, *BalanceRatioRaw*.

---

### 3.2.24 GetBarcode

#### Description

Returns the character string decoded from a barcode found in the current frame. The following 1D and 2D barcode symbologies are supported: UPC-A, UPC-E, EAN-8, Code 128, Code 39, Interleaved 2/5, QR Code, DataMatrix, PDF417

#### Syntax

[VB]

```
Value=objActiveGige.GetBarcode([ Type ])
```

[C/C++]

```
HRESULT GetFeatureArray(short Type, bstr* pValue);
```

#### Data Types [VB]

*Type*: Integer

*Return value*: String

#### Parameters [C/C++]

*Type* [in]

An optional short value specifying the type of the barcode symbology:

0 - *All [default]*

Will attempt to decode all supported barcode types.

1 - *Linear*

Will attempt to decode all supported 1D barcode types.

2 - *QR*

Will attempt to decode the QR code.

3 - *DataMatrix*

Will attempt to decode the DataMatrix code.

4 - *PDF417*

Will attempt to decode the PDF417 code.

*pValue* [out,retval]

Pointer to the bstr value containing the decoded string.

#### Return Values

S\_OK

Success

#### Example

The following VB example continuously analyzes the incoming image frames for all supported barcode types and displays the string decoded:

```
Private Sub Form_Load()  
ActiveGigel.Acquire = True
```

```
End Sub
```

```
Private Sub ActiveGigel_FrameAcquired()  
Label1.Caption = ActiveGigel.GetBarcode()  
End Sub
```

### Remarks

To increase the decoding speed, it is recommended to use this method with the *Type* argument indicating the specific barcode type.

If several barcodes are present in the current frame, ActiveGigE will attempt to decode the first one found, in the top-to-bottom left-to-right order.

Note that this function does not return an error code. If the barcode has not been decoded, the returned string will be empty. If a critical error occur during the decoding, the string will contain one blank (space) character.

---

### 3.2.25 GetBitsPerChannel

#### Description

Returns the number of bits per color component of a pixel.

#### Syntax

[VB]  
`Value=objActiveGige.GetBitsPerChannel()`

[C/C++]  
`HRESULT GetBitsPerChannel(long* pValue );`

#### Data Types [VB]

*Return value:* Long

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the number of bytes per pixel

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure

#### Example

This VB example reads and displays the pixel depth:

```
value=ActiveGigel.GetBitsPerChannel()  
MsgBox value
```

#### Remarks

The method returns the pixel depth of the internal image buffer of the *ActiveGige* object after the unpacking and color interpolation is performed. Depending on the selected format, the following value will be returned:

Pixel Format	Bits per channel
Mono8, Mono8s,	8
Mono10, Mono10Packed	10
Mono12, Mono12Packed	12
Mono14	14
Mono16	16
Bayer**8, Bayer**8Packed, RGB8, BGR8, RGBa8, BGRa8, YUV411_8_UYYVYY, YUV422_8_UYVY, YUV8_YUV, RGB8Planar	8
Bayer**10, Bayer**10Packed, RGB10, BGR10, BGR10V1Packed, BGR10V2Packed, RGB10Planar	10
Bayer**12, Bayer**12Packed, RGB12, BGR12, RGB12Planar	12
Bayer**16, RGB16Planar	16
Coord3D_ABC16	16
Coord3D_ABC32f	32

---

### 3.2.26 GetBlackLevelMax

#### Description

Returns the maximum value allowed for the camera's black level..

#### Syntax

[VB]

```
Value=objActiveGige.GetBlackLevelMax
```

[C/C++]

```
HRESULT GetBlackLevelMax( float* pValue );
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the maximum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGigel.GetBlackLevelMin  
Slider1.Max = ActiveGigel.GetBlackLevelMax  
Slider1.Value = ActiveGigel.BlackLevel  
End Sub
```

#### Remarks

Note that this method is available only if the currently selected camera supports one of the following GenICam features: *BlackLevel*, *BlackLevelAbs*, *BlackLevelRaw*.

### 3.2.27 GetBlackLevelMin

#### Description

Returns the minimum value allowed for the camera's black level..

#### Syntax

[VB]

```
Value=objActiveGige.GetBlackLevelMin
```

[C/C++]

```
HRESULT GetBlackLevelMin( float* pValue );
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the minimum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGigel.GetBlackLevelMin  
Slider1.Max = ActiveGigel.GetBlackLevelMax  
Slider1.Value = ActiveGigel.BlackLevel  
End Sub
```

#### Remarks

Note that this method is available only if the currently selected camera supports one of the following GenICam features: *BlackLevel*, *BlackLevelAbs*, *BlackLevelRaw*.

---



### 3.2.28 GetBlockId

#### Description

Returns the block ID of the last acquired frame.

#### Syntax

[VB]

```
Value=objActiveGige.GetBlockId
```

[C/C++]

```
HRESULT GetBlockId(long* pValue);
```

#### Data Types [VB]

*Return value:* Long

#### Parameters [C/C++]

*pValue* [out,retval]

Pointer to the timestamp value

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example displays the block ID of each frame in a text box.

```
Private Sub ActiveGigel_FrameAcquired()  
    Label1.Caption = ActiveGigel.GetBlockId  
End Sub
```

#### Remarks

Block ID is an ordinal number (starting from 1) assigned by the camera to each acquired frame. Block IDs can be used to identify missing frames.

If this method is used in the [FrameAcquired](#) event handler, it must be called immediately after the event has been received. This will guarantee that the value of the block ID will not be taken from the next frame while the current frame is being processed.

### 3.2.29 GetBytesPerPixel

#### Description

Returns the number of bytes per pixel for the current video [Format](#).

#### Syntax

[VB]  
`Value=objActiveGige.GetBytesPerPixel()`

[C/C++]  
`HRESULT GetBytesPerPixel(long* pValue );`

#### Data Types [VB]

*Return value:* Long

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the number of bytes per pixel

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure

#### Example

This VB example reads and displays the pixel depth:

```
value=ActiveGige1.GetBytesPerPixel()  
MsgBox value
```

#### Remarks

The method returns the pixel depth of the internal image buffer of the ActiveGige object, which can be different from the pixel depth of the raw image outputted by the camera. For instance, the YUV 4:1:1 and YUV 4:2:2 images are always converted to the RGB 8:8:8 video, therefore the number of bytes per pixel reported for these formats will be 3. Also, when the [Bayer](#) color conversion is activated for Mono 8 and Mono 10/12/16 formats, the resulting video will have 3 or 6 bytes per pixel accordingly.

---

### 3.2.30 GetCameraIP

#### Description

Returns the IP address of the currently selected device in Internet standard dotted format.

#### Syntax

[VB]

```
Value=objActiveGige.GetCameraIP
```

[C/C++]

```
HRESULT GetCameraIP(bstr* pValue );
```

#### Data Types [VB]

*Return value:* String

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the string value specifying the IP address

#### Return Values

S\_OK

Success

E\_FAIL

Failed to read the feature

#### Example

The following VB example prints the IP address if the camera:

```
MsgBox ActiveGige1.GetCameraIP
```

#### Remarks

The IP address is returned in the Internet standard dotted format such as 169.254.102.77

### 3.2.31 GetCameraIPList

#### Description

Returns the array of strings containing the IP addresses of connected GigE Vision™ devices in Internet standard dotted format.

#### Syntax

[VB]

```
Value=objActiveGige.GetCameraIPList()
```

[C/C++]

```
HRESULT GetCameraIPList( VARIANT* pList );
```

#### Data Types [VB]

*Return value:* Variant (Array of strings)

#### Parameters [C/C++]

*pList* [out,retval]

Pointer to the SAFEARRAY containing IP addresses

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example initializes a combo box with IP addresses and uses it to switch between the cameras:

```
Private Sub Form_Load()  
CamIPLst = ActiveGigel.GetCameraIPList  
For i = 0 To UBound(CamIPLst)  
Combol.AddItem (CamIPLst(i))  
Next  
Combol.ListIndex = 0  
ActiveGigel.Acquire = True  
End Sub  
  
Private Sub Combol_Click()  
ActiveGigel.Camera = Combol.ListIndex  
End Sub
```

This MFC example fills out a combo box with IP addresses:

```
VARIANT m_CamArray=m_ActiveGige.GetCameraIPList();  
SAFEARRAY *pArray=m_CamArray.parray;  
UINT nCam=pArray->rgsabound[0].cElements;
```

---

```
CString strCamera;  
CComboBox *pCamera=(CComboBox*)GetDlgItem(IDC_CAMERA);  
for(UINT i=0;i<nCam;i++)  
{  
    CString str; str.Format("Camera %d",i);  
    pCamera->AddString(str);  
}  
int iCam=m_ActiveGige.GetCamera();  
pCamera->SetCurSel(iCam);  
  
SafeArrayDestroy(pArray);
```

### Remarks

The IP addresses are returned in the Internet standard dotted format such as 169.254.102.77

The index of an element in the IP list can be used as an argument of the [Camera](#) property to select a specific device.

Note that in C/C++ applications it is required to call SafeArrayDestroy() to delete the SAFEARRAY returned by **GetCameraIPList**.

### 3.2.32 GetCameraList

#### Description

Returns the array of strings containing the names of GigE Vision™ devices connected to the system. The devices are listed in the alphabetical order "model + serial number".

#### Syntax

[VB]  
`Value=objActiveGige.GetCameraList()`

[C/C++]  
`HRESULT GetCameraList( VARIANT* pList );`

#### Data Types [VB]

*Return value:* Variant (Array of strings)

#### Parameters [C/C++]

*pList* [out,retval]  
Pointer to the SAFEARRAY containing camera names

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example initializes a combo box with camera names and uses it to switch between the cameras:

```
Private Sub Form_Load()  
CamLst = ActiveGigel.GetCameraList  
For i = 0 To UBound(CamLst)  
Combol.AddItem (CamLst(i))  
Next  
Combol.ListIndex = 0  
ActiveGigel.Acquire = True  
End Sub  
  
Private Sub Combol_Click()  
ActiveGigel.Camera = Combol.ListIndex  
End Sub
```

This MFC example fills out a combo box with camera names:

```
VARIANT m_CamArray=m_ActiveGige.GetCameraList();  
SAFEARRAY *pArray=m_CamArray.parray;
```

---

```
UINT nCam=pArray->rgsabound[0].cElements;

CString strCamera;
CComboBox *pCamera=(CComboBox*)GetDlgItem(IDC_CAMERA);
for(UINT i=0;i<nCam;i++)
{
    CString str; str.Format("Camera %d",i);
    pCamera->AddString(str);
}
int iCam=m_ActiveGige.GetCamera();
pCamera->SetCurSel(iCam);

SafeArrayDestroy(pArray);
```

### Remarks

The index of an element in the camera list can be used as an argument of the [Camera](#) property to select a specific camera.

Note that in C/C++ applications it is required to call `SafeArrayDestroy()` to delete the `SAFEARRAY` returned by **GetCameraList**.

### 3.2.33 GetCameraMAC

#### Description

Returns the MAC address of the currently selected device in the string form.

#### Syntax

[VB]

```
Value=objActiveGige.GetCameraMAC
```

[C/C++]

```
HRESULT GetCameraMAC(bstr* pValue );
```

#### Data Types [VB]

*Return value:* String

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the string value specifying the MAC address

#### Return Values

S\_OK

Success

E\_FAIL

Failed to read the feature

#### Example

The following VB example prints the MAC address if the camera:

```
MsgBox ActiveGige1.GetCameraMAC
```

#### Remarks

The MAC address is returned in standard dashed format such as 00-05-5D-24-6E-5B.

---



### 3.2.34 GetCameraMACList

#### Description

Returns the array of strings containing the MAC addresses of connected GigE Vision™ devices in the string form.

#### Syntax

[VB]

```
Value=objActiveGige.GetCameraMACList()
```

[C/C++]

```
HRESULT GetCameraMACList( VARIANT* pList );
```

#### Data Types [VB]

*Return value:* Variant (Array of strings)

#### Parameters [C/C++]

*pList* [out,retval]

Pointer to the SAFEARRAY containing MAC addresses

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example initializes a combo box with MAC addresses and uses it to switch between the cameras:

```
Private Sub Form_Load()  
CamMACLst = ActiveGigel.GetCameraMACList  
For i = 0 To UBound(CamMACLst)  
Combol.AddItem (CamMACLst(i))  
Next  
Combol.ListIndex = 0  
ActiveGigel.Acquire = True  
End Sub  
  
Private Sub Combol_Click()  
ActiveGigel.Camera = Combol.ListIndex  
End Sub
```

This MFC example fills out a combo box with MAC addresses:

```
VARIANT m_CamArray=m_ActiveGige.GetCameraMACList();  
SAFEARRAY *pArray=m_CamArray.parray;
```

```
UINT nCam=pArray->rgsabound[0].cElements;

CString strCamera;
CComboBox *pCamera=(CComboBox*)GetDlgItem(IDC_CAMERA);
for(UINT i=0;i<nCam;i++)
{
    CString str; str.Format("Camera %d",i);
    pCamera->AddString(str);
}
int iCam=m_ActiveGige.GetCamera();
pCamera->SetCurSel(iCam);

SafeArrayDestroy(pArray);
```

### Remarks

The IP addresses are returned in the standard dashed format such as 00-05-5D-24-6E-5B.

The index of an element in the MAC list can be used as an argument of the [Camera](#) property to select a specific device.

Note that in C/C++ applications it is required to call SafeArrayDestroy() to delete the SAFEARRAY returned by **GetCameraMACList**.

---

### 3.2.35 GetChunkPointer

#### Description

Returns the pointer to the data associated with the specified chunk feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetChunkPointer( Name )
```

[C/C++]

```
HRESULT GetChunkPointer( bstr Name, VARIANT* pValue );
```

#### Data Types [VB]

*Return value:* Variant (pointer)

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the chunk feature

*pValue* [out,retval]

Pointer to the variant containing the pointer to the data associated with the feature

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid input arguments.

#### Example

This fragment of the C++ code grabs a frame, retrieves a pointer to the chunk data and copies them into a byte array.

```
VARIANT var; LONG var_size;  
char buffer[255]; short v;
```

```
pActiveGige->SetFeature( OLESTR("ChunkEnable"), 1);  
pActiveGige->Grab(&v);
```

```
pActiveGige->GetChunkPointer( OLESTR( "ChunkInputStatusAtLineTriggerValue" ), &var);  
pActiveGige->GetChunkSize(OLESTR( "ChunkInputStatusAtLineTriggerValue" ), &var_size);  
memcpy( buffer, (BYTE*)var.byref, var_size);
```

#### Remarks

Chunks are tagged blocks of auxiliary data that are transmitted along with each image frame. The **GetImagePointer** method provides the most efficient way to quickly access the chunk data in pointer-aware programming languages. It is especially useful for accessing data associated with array-type features.

This method should be used in combination with [GetChunkSize](#).

Note that when several chunk features share the same Chunk ID, this method will return the pointer to the beginning of the data block identified by the given Chunk ID. For more information on the chunk data refer to *"GigE Vision Video Streaming and Device Control Over Ethernet Standard"* published by the Automated Imaging Association.

---

### 3.2.36 GetChunkSize

#### Description

Returns the size of the data associated with the specified chunk feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetChunkSize( Name )
```

[C/C++]

```
HRESULT GetChunkSize( bstr Name, long* pValue );
```

#### Data Types [VB]

*Return value:* Variant (pointer)

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the chunk feature

*pValue* [out,retval]

Pointer to the size of the chunk data in bytes.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid input arguments.

#### Example

This fragment of the C++ code grabs a frame, retrieves a pointer to the chunk data and copies them into a byte array.

```
VARIANT var; LONG var_size;  
char buffer[255]; short v;
```

```
pActiveGige->SetFeature( OLESTR("ChunkEnable"), 1);  
pActiveGige->Grab(&v);
```

```
pActiveGige->GetChunkPointer( OLESTR( "ChunkInputStatusAtLineTriggerValue" ), &var);  
pActiveGige->GetChunkSize(OLESTR( "ChunkInputStatusAtLineTriggerValue" ), &var_size);  
memcpy( buffer, (BYTE*)var.byref, var_size);
```

**Remarks**

Chunks are tagged blocks of auxiliary data that are transmitted along with each image frame. The **GetChunkSize** and [GetChunkPointer](#) methods provides the most efficient way to quickly access the chunk data in pointer-aware programming languages. It is especially useful for accessing data associated with array-type features.

Note that when several chunk features share the same Chunk ID, this method will return the size of the data block identified by the given Chunk ID. For more information on the chunk data refer to "*GigE Vision Video Streaming and Device Control Over Ethernet Standard*" published by the Automated Imaging Association.

---

### 3.2.37 GetCodec

#### Description

Gets the name of the currently selected codec (video compressor) for the AVI capture.

#### Syntax

[VB]

```
Value = objActiveGige.GetCodec
```

[C/C++]

```
HRESULT GetCodec();
```

#### Data Types [VB]

*Return value:* String

#### Parameters [C/C++]

*pValue* [out,retval]

Pointer to the bstr value containing the name of the currently selected codec.

#### Return Values

S\_OK

Success

E\_FAIL

Failure to get the codec

#### Example

The following VB example prints the name of the currently selected codec:

```
MsgBox ActiveGige1.GetCodec
```

### 3.2.38 GetCodecList

#### Description

Returns the array of strings containing the names of AVI codecs (compressors) available in the system.

#### Syntax

[VB]

```
Value=objActiveGige.GetCodecList()
```

[C/C++]

```
HRESULT GetCodecList( VARIANT* pList );
```

#### Data Types [VB]

*Return value:* Variant (SAFEARRAY)

#### Parameters [C/C++]

*pList* [out,retval]

Pointer to the SAFEARRAY of strings containing available categories

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example initializes a combo box with the list of available codecs:

```
CodecLst = ActiveGige1.GetCodecList
For i = 0 To UBound(CodecLst)
  Combol.AddItem (CodecLst(i))
Next
Combol.ListIndex = 0
```

#### Remarks

Note that in C/C++ applications it is required to call SafeArrayDestroy() to delete the SAFEARRAY returned by **GetCodecList**.

---



### 3.2.39 GetCodecProperties

#### Description

Returns the generic parameters of the currently selected compression codec.

#### Syntax

[VB]

```
value=objActiveGige.GetCodecProperties
```

[C/C++]

```
HRESULT GetCodecProperties(VARIANT* pProperties);
```

#### Data Types [VB]

*Return value:* Variant (Array of Long values)

#### Parameters [C/C++]

*pProperties* [out,retval]

Pointer to the SAFEARRAY (integer) containing the values of the codec properties. The values are written in the array as follows:

Properties [0] – *Quality*. Numerical value of the quality parameter used by the codec, if supported.

Properties [1] – *DataRate*. Data rate in kb/sec used by the codec, if supported.

Properties [2] – *KeyFrameRate*. Amount of frames after which a key frame is recorded, if supported.

Properties [3] – *PFramesPerKey*. Rate of predicted (P) frames per key frame, if supported.

Properties [4] – *WindowSize*. Amount of frames over which the compressor maintains the average data rate.

#### Return Values

S\_OK

Success

E\_FAIL

Failure to retrieve the codec's properties

#### Example

The following VB example displays the quality and datarate settings of the MJPEG codec:

```
ActiveGige1.SetCodec "MJPEG Compressor"  
CodecParams=ActiveGige1.GetCodecProperties  
LabelQuality.Caption=CodecParams[0]  
LabelDatarate.Caption=CodecParams[1]
```

#### Remarks

This method allows you to retrieve only the basic compression properties which may not be supported by certain codecs. To access the internal parameters of a codec, use [ShowCodecDialog](#).

### 3.2.40 GetComponentData

#### Description

Returns the two-dimensional array of pixel values in the specified color component of the current frame.

#### Syntax

[VB]

```
Value=objActiveGige.GetComponentData( Component )
```

[C/C++]

```
HRESULT GetComponentData( short Component, VARIANT* pArray );
```

#### Data Types [VB]

*Y*: Integer

*Component*: Integer

*Return value*: Variant (SAFEARRAY)

#### Parameters [C/C++]

*Component* [in]

The index of the selected color component. Must be one of the following values:

0 - returns the luminance data

1 - returns the red component data

2 - returns the green component data

3 - returns the blue component data

*pArray* [out,retval]

Pointer to the SAFEARRAY containing the pixel values in the frame

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid input argument.

#### Example

This VB example grabs a frame, retrieves its green component and displays the value of the specified element of the array.

```
Dim pix as Long
ActiveGigel.Grab
dataArray=ActiveGigel.GetComponentData(1)
x=16
y=32
pix=dataArray(x,y)
```

---

```
if pix < 0 then  
pix=65535-pix  
endif
```

### Remarks

The array returned by **GetComponentData** has the dimensions 0 to [SizeX](#) - 1, 0 to [SizeY](#) - 1. The type of data in the array depends on the format of the video acquired. For the 24-bit video the method will return an array of bytes. If the video is of the high-bit depth, the array will contain integer (word) values.

Note that modifying elements of the array will not change actual pixel values in the frame buffer. The image data in the array are stored in the standard order from top to bottom, therefore the first element of the array is first pixel of the top line of the image. This is opposite to the order of rows in the array returned by [GetImageData](#).

If the video has a grayscale format, the *Component* parameter will have no effect and the output array will contain the luminance data.

For integer (word) type of data you can receive negative numbers if pixel values exceed 32767. In C and C# you can convert signed integers to unsigned ones by using data casting. To get rid of negative values in VB and Delphi, subtract them from 65535 (see the example above).

Note that in C/C++ applications it is required to call `SafeArrayDestroy()` to delete the `SAFEARRAY` returned by **GetComponentData**.

### 3.2.41 GetComponentLine

#### Description

Returns the array of pixel values of the specified color component at the specified horizontal line of the current frame.

#### Syntax

[VB]

```
Value=objActiveGige.GetComponentLine( Y, Component )
```

[C/C++]

```
HRESULT GetComponentLine( short Y, short Component, VARIANT* pArray );
```

#### Data Types [VB]

*Y*: Integer

*Component*: Integer

*Return value*: Variant (SAFEARRAY)

#### Parameters [C/C++]

*Y* [in]

The y-coordinate of the line in the image

*Component* [in]

The index of the selected color component. Must be one of the following values:

0 - returns the luminance data

1 - returns the red component data

2 - returns the green component data

3 - returns the blue component data

*pArray* [out,retval]

Pointer to the SAFEARRAY containing the pixel values in the line

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid input argument.

#### Example

This VB example grabs a frame, retrieves the 32th row of green pixels and displays the value of 10th pixel in the row.

```
ActiveGige1.Grab  
Line=ActiveGige1.GetComponentLine(32,2)  
MsgBox Line(10)
```

---

## Remarks

The array returned by **GetComponentLine** has the dimension range from 0 to [SizeX](#) - 1. The type of data in the array depends on the format of the video acquired. For the 24-bit video the method will return an array of bytes. If the video is of the high-bit depth, the array will contain integer (word) values. If the video has a grayscale format, the *Component* parameter will have no effect and the output array will contain the luminance data (see [GetLine](#)).

Note that modifying elements of the array will not change actual pixel values in the frame buffer.

The value of the y coordinate must not exceed the height of the video frame, or the error will occur.

Note that in C/C++ applications it is required to call `SafeArrayDestroy()` to delete the `SAFEARRAY` returned by **GetComponentLine**.

### 3.2.42 GetDIB

#### Description

Returns the handler to a Device Independent Bitmap.

#### Syntax

[VB]  
`Value=objActiveGige.GetDIB()`

[C/C++]  
`HRESULT GetDIB(HGLOBAL* pDib);`

#### Data Types [VB]

*Return value:* long

#### Parameters [C/C++]

*X* [in]  
The x-coordinate of the pixel  
*Y* [in]  
The y-coordinate of the pixel  
*pDib* [out,retval]  
Pointer to the handler to *ActiveGige*'s display DIB

#### Return Values

*S\_OK*  
Success  
*E\_FAIL*  
Failure.

#### Example

This C example demonstrates how to retrieve and display *ActiveGige*'s DIB

```
BITMAPINFO *pDIB;  
pActiveGige->GetDIB((ULONG*)&pDIB);  
RECT rect; long sx,sy;  
GetWindowRect(hWnd,&rect);  
pActiveGige->GetSizeX(&sx);  
pActiveGige->GetSizeY(&sy);  
BYTE* pData=(BYTE*)pDIB+sizeof(BITMAPINFOHEADER) + pDIB->bmiHeader.biClrUsed * sizeof(RGBQUAD);  
SetDIBitsToDevice(hDC, 0, 0, sx, sy, 0, 0, 0, sy, pData, pDIB,  
DIB_RGB_COLORS);
```

#### Remarks

The **GetDIB** method provides the most efficient way to display the internal image buffer when you do

---

---

not want to use *ActiveGige*'s live display. *ActiveGige* uses packed DIBs, the pixel data immediately following the BITMAPINFO structure. The method returns a GlobalAlloc handler which contains the pointer to a DIB. The application must not free the global handler after using the DIB data.

Note that a DIB contains only 8- and 24-bit pixel values, even if the current video mode is a 16- or 48-bit one. To access the actual pixel data, use the [GetImageData](#) or [GetPointer](#) methods.

### 3.2.43 GetEnumList

#### Description

Returns the array of string values representing the specified enumerated feature of the camera.

#### Syntax

[VB]

```
Value=objActiveGige.GetEnumList( Name )
```

[C/C++]

```
HRESULT GetEnumList( bstr Name, VARIANT* pInfo );
```

#### Data Types [VB]

*Name*: String

*Return value*: Variant (Array)

#### Parameters [C/C++]

*Name* [in]

String containing the name of the enumerated feature

*pInfo* [out,retval]

Pointer to the SAFEARRAY containing the allowed string values of the feature:

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_INVALIDARG

Feature is not of enumerated type

E\_FAIL

Failed to read feature.

#### Example

This VB example uses a combo box to switch between different auto exposure modes:

```
Private Sub Form_Load()  
Lst = ActiveGigel.GetEnumList("ExposureAuto")  
For i = 0 To UBound(Lst)  
Combol.AddItem (Lst(i))  
Next  
Combol.ListIndex = ActiveGigel.GetFeature("ExposureAuto")  
ActiveGigel.Acquire = True  
End Sub
```

```
Private Sub Combol_Click()  
ActiveGigel.SetFeatureString Combol.Text  
End Sub
```

---



**Remarks**

The index of an element in the string list can be used as an argument of [SetFeature](#) to set a specific value for the feature.

Note that in C/C++ applications it is required to call `SafeArrayDestroy()` to delete the `SAFEARRAY` returned by **GetEnumList**.

If the specified feature is not of the enumerated type or not supported by the camera, the method will return an error.

### 3.2.44 GetExposureTimeMax

#### Description

Returns the maximum value allowed for the camera exposure time, in microseconds or raw units..

#### Syntax

[VB]

```
Value=objActiveGige.GetExposureTimeMax
```

[C/C++]

```
HRESULT GetExposureTimeMax( float* pValue );
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the maximum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGigel.GetExposureTimeMin  
Slider1.Max = ActiveGigel.GetExposureTimeMax  
Slider1.Value = ActiveGigel.ExposureTime  
End Sub
```

#### Remarks

Note that this method is available only if the currently selected camera supports one of the following GenICam features: *ExposureTime*, *ExposureTimeAbs*, *ExposureTimeRaw*.

---

### 3.2.45 GetExposureTimeMin

#### Description

Returns the minimum value allowed for the camera exposure time, in microseconds or raw units..

#### Syntax

[VB]

```
Value=objActiveGige.GetExposureTimeMin
```

[C/C++]

```
HRESULT GetExposureTimeMin( float* pValue );
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the minimum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGigel.GetExposureTimeMin  
Slider1.Max = ActiveGigel.GetExposureTimeMax  
Slider1.Value = ActiveGigel.ExposureTime  
End Sub
```

#### Remarks

Note that this method is available only if the currently selected camera supports one of the following GenICam features: *ExposureTime*, *ExposureTimeAbs*, *ExposureTimeRaw*.

### 3.2.46 GetFeature

#### Description

Returns the numerical value of the specified camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeature( Name )
```

[C/C++]

```
HRESULT GetFeature(bstr Name, float* pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: Single

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pValue* [out, retval]

Pointer to the numerical value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failed to read the feature

#### Example

The following VB example demonstrates the use of a scroll control for real-time adjustment of the "GainRaw" feature.

```
Private Sub Form_Load()  
ActiveGigel.Acquire=True  
HScroll1.Value = ActiveGigel.GetFeature("GainRaw")  
HScroll1.Min = ActiveGigel.GetFeatureMin("GainRaw")  
HScroll1.Max = ActiveGigel.GetFeatureMax("GainRaw")  
End Sub
```

```
Private Sub HScroll1_Scroll()  
ActiveGigel.SetFeature("GainRaw", HScroll1.Value)  
End Sub
```

---

**Remarks**

Depending on the [Type](#) of the feature **GetFeature** will return the following values:

<b>Feature Type</b>	<b>Return value</b>
Integer	Integer value converted to floating point
Float	Floating point value
Boolean	0 if False, 1 if True
Enumerated	Ordinal number in the enumeration list
String	Zero value
Command	Zero value

If the currently selected camera does not support the specified feature, the method will generate an error.

To retrieve the string value of a feature, use [GetFeatureString](#).

### 3.2.47 GetFeature64

#### Description

Returns the numerical value of the specified 64-bit camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeature64( Name )
```

[C/C++]

```
HRESULT GetFeature64(bstr Name, double* pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: Double

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pValue* [out, retval]

Pointer to the numerical value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failed to read the feature

#### Example

The following VB example demonstrates how to retrieve the value of the ChunkTimeStamp feature.

```
Private Sub Form_Load()  
ActiveGigel.Acquire=True  
ActiveGigel.SetFeature("ChunkModeActive",True)  
ActiveGigel.SetFeature("ChunkModeEnable",True)  
End Sub  
  
Private Sub ActiveGigel_FrameAcquired()  
Label1.Caption = ActiveGigel.GetFeature64("ChunkTimeStamp")  
End Sub
```

---

**Remarks**

It is recommended to use this method only for those integer and floating point features whose length is 64-bit.

If the currently selected camera does not support the specified feature, the method will generate an error.

### 3.2.48 GetFeatureAccess

#### Description

Returns the information on availability and access to the specified camera feature.

#### Syntax

[VB]  
`Value=objActiveGige.GetFeatureAccess ( Name )`

[C/C++]  
`HRESULT GetFeatureAccess( bstr Name, bstr pValue );`

#### Data Types [VB]

*Name*: String

*Return value*: String

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pValue* [out, retval]

Pointer to the string representing the access to the feature in the currently selected camera:

"NI" - feature is not implemented

"NA" - feature is not available at the moment

"WO" - feature has the write-only access

"RO" - feature has the read-only access

"RW" - feature has the full access

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

This VB example displays the access mode of the feature:

```
MsgBox GetFeatureAccess("DeviceFirmwareVersion")
```

#### Remarks

Note that the access to a feature can change depending on the value of other features.

This function should be used for features included in the camera xml file. If you want to check

---



---

availability of a feature which may be not present in the xml file, use [IsFeatureAvailable](#).

---

### 3.2.49 GetFeatureArray

#### Description

Returns an array of values associated with the specified camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeatureArray( Name, ElementSize )
```

[C/C++]

```
HRESULT GetFeatureArray(bstr Name, short ElementSize, VARIANT* pArray);
```

#### Data Types [VB]

*Name*: String

*ElementSize*: Integer

*Return value*: Variant (SAFEARRAY)

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*ElementSize* [in]

A short value specifying the size of each array's element, in bytes

*pArray* [out, retval]

Pointer to the SAFEARRAY containing the values of the buffer

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failed to read the feature

#### Example

The following VB example creates an inverse lookup table and copies it into the camera using the "LUTValueAll" feature:

```
Dim LUT(255) As Long
For i = 0 To 255
    LUT(i) = 255-i
Next
ActiveGige1.SetFeatureArray "LUTValueAll", 4, LUT
ActiveGige1.SetFeature "LUTEnable", True
```

#### Remarks

---

---

This method sets the content of buffers associated with features of the IRegister type. For more information refer to GenICam standard specifications.

Note that the size of an element in the binary buffer linked to an IRegister feature cannot be determined automatically. Therefore, the *ElementSize* parameter is not optional and must specify the size of each element of the array in bytes.

### 3.2.50 GetFeatureDependents

#### Description

Returns the array of strings containing the names of camera features dependent on the specified feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeatureDependents( Name )
```

[C/C++]

```
HRESULT GetFeatureList( bstr Name, VARIANT* pList );
```

#### Data Types [VB]

*Name*: String

*Return value*: Variant (SAFEARRAY)

#### Parameters [C/C++]

*Name* [in]

Name of the feature for which the list of dependent features is requested.

*pList* [out,retval]

Pointer to the SAFEARRAY of strings containing dependent features.

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Category is not supported by the camera

E\_FAIL

Failure.

#### Example

This VB example retrieves the list of features dependent on the "TriggerSelector" feature:

```
FeatureLst = ActiveGige1.GetFeatureDependents("TriggerSelector")
```

#### Remarks

Use this method to determine the features whose values may change when the value of the specified feature is changing. Your application can use it to update the values of the controls linked to the dependant features.

Note that in C/C++ applications it is required to call SafeArrayDestroy() to delete the SAFEARRAY returned by **GetFeatureDependents**.

---

### 3.2.51 GetFeatureDescription

#### Description

Returns the detailed description of the specified camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeatureDescription ( Name )
```

[C/C++]

```
HRESULT GetFeatureDescription(bstr Name, bstr* pType );
```

#### Data Types [VB]

*Name*: String

*Return value*: String

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pType* [out, retval]

Pointer to a string specifying the description of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure

#### Example

This VB example displays the description of the specified feature:

```
MsgBox ActiveGigel.GetFeatureDescription( "ExposureAuto" )
```

#### Remarks

Feature descriptions are retrieved from an XML file embedded in a camera per GenICam standard.

### 3.2.52 GetFeatureList

#### Description

Returns the array of strings containing the names of camera features and categories grouped under the specified category.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeatureList( Category )
```

[C/C++]

```
HRESULT GetFeatureList( bstr Category, VARIANT* pList );
```

#### Data Types [VB]

*Category*: String

*Return value*: Variant (Array of strings)

#### Parameters [C/C++]

*Category* [in]

Name of the category or sub-category for the list of features. If this parameter is "Root", the list of upper-level categories and features will be returned.

*pList* [out,retval]

Pointer to the SAFEARRAY of strings containing the list of features under the specified category. An empty array indicates that the name of a feature is used instead of the name of a category.

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Category is not supported by the camera

E\_FAIL

Failure.

#### Example

This VB example initializes a combo box with the names of the camera features grouped under the "AnalogControls" category:

```
Private Sub Form_Load()  
FeatureLst = ActiveGige1.GetFeatureList("AnalogControls")  
For i = 0 To UBound(FeatureLst)  
Combol.AddItem (FeatureLst(i))  
Next  
Combol.ListIndex = 0  
ActiveGige1.Acquire = True  
End Sub
```

---

This MFC example fills out a combo box with camera features grouped under the "TriggerAcquisition" category:

```
VARIANT m_FeatureArray=m_ActiveGige.GetFeatureList("TriggerAcquisition");
SAFEARRAY *pArray=m_FeatureArray.parray;
UINT nFeatures=pArray->rgsabound[0].cElements;

CString strFeature;
CComboBox *pFeature=(CComboBox*)GetDlgItem(IDC_FEATURE);
for(UINT i=0;i<nFeatures;i++)
{
    pFeature->AddString(strFeature);
}
int iFeature=m_ActiveGige.GetFeature();
pFeature->SetCurSel(iFeature);

SafeArrayDestroy(pArray);
```

### Remarks

To get or set the value of a specific camera feature, use [GetFeature](#), [GetFeatureString](#), [SetFeature](#), [SetFeatureString](#)

Note that in C/C++ applications it is required to call `SafeArrayDestroy()` to delete the `SAFEARRAY` returned by **GetFeatureList**.

### 3.2.53 GetFeatureIncrement

#### Description

Returns the increment value for the specified integer feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeatureIncrement ( Name )
```

[C/C++]

```
HRESULT GetFeatureIncrement( bstr Name, long* pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: Long

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pValue* [out, retval]

Pointer to the increment value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_INVALIDARG

Feature is not of integer type

E\_FAIL

The gain control is not available for the selected camera

#### Example

This VB example uses the increment value of the horizontal image size to initialize the scroll control:

```
Private Sub Form_Load()  
HScroll1.Min = ActiveGige1.GetFeatureMin ("Width")  
HScroll1.Max = ActiveGige1.GetFeatureMax ("Width")  
HScroll1.SmallChange = ActiveGige1.GetFeatureIncrement ("Width")  
HScroll1.Value = ActiveGige1.GetFeature ("Width")  
End Sub
```

#### Remarks

The increment of the feature defines the minimum value at which the feature can increase or

---



decrease. Available only for integer features.

### 3.2.54 GetFeatureMin

#### Description

Returns the minimum numerical value allowed for the specified camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeatureMin ( Name )
```

[C/C++]

```
HRESULT GetFeatureMin( bstr Name, float* pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: Single

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pValue* [out, retval]

Pointer to the minimum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum gain values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGige1.GetFeatureMin("GainRaw")  
Slider1.Max = ActiveGige1.GetFeatureMax("GainRaw")  
Slider1.Value = ActiveGige1.GetFeature("GainRaw")  
End Sub
```

#### Remarks

Depending on the [Type](#) of the feature **GetFeatureMin** will return the following values:

---

---

Feature Type	Return value
Integer	Minimum allowed value converted to floating point
Float	Minimum allowed value
Boolean	Zero value
Enumerated	Zero value
String	Zero value
Command	Zero value

If the currently selected camera does not support the specified feature, the method will generate an error.

### 3.2.55 GetFeatureMax

#### Description

Returns the maximum numerical value allowed for the specified camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeatureMin ( Name )
```

[C/C++]

```
HRESULT GetFeatureMin( bstr Name, float* pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: Single

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pValue* [out, retval]

Pointer to the maximum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum gain values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGige1.GetFeatureMin("GainRaw")  
Slider1.Max = ActiveGige1.GetFeatureMax("GainRaw")  
Slider1.Value = ActiveGige1.GetFeature("GainRaw")  
End Sub
```

#### Remarks

Depending on the [Type](#) of the feature **GetFeatureMax** will return the following values:

---

---

Feature Type	Return value
Integer	Maximum allowed value converted to floating point
Float	Maximum allowed value
Boolean	1.
Enumerated	Maximum ordinal number in the enumeration list
String	Zero value
Command	1.

If the currently selected camera does not support the specified feature, the method will generate an error.

---

### 3.2.56 GetFeatureRepresentation

#### Description

Returns the information on the suggested representation for the specified camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeatureAccess ( Name )
```

[C/C++]

```
HRESULT GetFeatureAccess( bstr Name, bstr pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: String

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pValue* [out, retval]

Pointer to the string specifying the representation of the feature:

"Linear" - feature should be represented by a slider with the linear behaviour

"Logarithmic" - feature should be represented by a slider with the logarithmic behaviour

"Boolean" - feature should be represented by a check box

"PureNumber" - feature should be represented by a text box with decimal display

"HexNumber" - feature should be represented by a text box with hexadecimal display

"Undefined" - no representation information available for the feature

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

This VB example displays the representation of the feature:

```
MsgBox GetFeatureRepresentation( "ExposureTimeAbs" )
```

#### Remarks

Per GenICam standard, the representation of a feature gives developers a hint about how to display and control a specific camera feature in the GUI.

---

### 3.2.57 GetFeatureString

#### Description

Returns the string value of the specified camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeatureString( Name )
```

[C/C++]

```
HRESULT GetFeatureString(bstr Name, bstr* pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: String

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pValue* [out, retval]

Pointer to the string value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failed to read the feature

#### Example

The following VB example prints the value of the "GainAuto" feature:

```
MsgBox ActiveGigel.GetFeatureString("GainAuto")
```

#### Remarks

Depending on the [Type](#) of the feature the return value has the following meaning:

Feature Type	Value
Integer	Integer value in form of string
Float	Floating point value in form of string
Boolean	"False" or "True"
Enumerated	Current string value
String	Current string value
Command	Zero value

If the currently selected camera does not support the specified feature, the method will generate an error.



### 3.2.58 GetFeatureTip

#### Description

Returns the tooltip (short description) for the specified camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeatureTip ( Name )
```

[C/C++]

```
HRESULT GetFeatureTip(bstr Name, bstr* pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: String

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pValue* [out, retval]

Pointer to a string specifying the short description of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure

#### Example

This VB example displays the short description of the specified feature:

```
MsgBox ActiveGigel.GetFeatureTip( "ExposureAuto" )
```

#### Remarks

Feature tooltips are retrieved from an XML file embedded in a camera per GenICam standard.

### 3.2.59 GetFeatureType

#### Description

Returns the type of the specified camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeatureType ( Name )
```

[C/C++]

```
HRESULT GetFeatureType(bstr Name, bstr* pType );
```

#### Data Types [VB]

*Name*: String

*Return value*: String

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pType* [out, retval]

Pointer to a string specifying the type of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure

#### Example

This VB example prints the type of the specified feature:

```
MsgBox ActiveGigel.GetFeatureType( "ExposureAuto" )
```

#### Remarks

Depending on the type of the feature the method returns the following string values:

---

---

Feature Type	Value
Integer	"Int"
Float	"Float"
Boolean	"Bool"
Enumerated	"Enum"
String	"String"
Command	"Command"
Feature category	"Category"
Unrecognized type	"Unknown"

---

### 3.2.60 GetFeatureVisibility

#### Description

Returns the information on the recommended visibility for the specified camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.GetFeatureAccess ( Name )
```

[C/C++]

```
HRESULT GetFeatureAccess( bstr Name, bstr pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: String

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pValue* [out, retval]

Pointer to the string specifying the visibility of the feature:

"Beginner" - access to the feature should be available for all users

"Expert" - access to the feature should be available for intermediate-level users

"Guru" - access to the feature should be limited to advanced-level users

"Invisible" - access to the feature should be available only in the API (no GUI visibility)

"Undefined" - no visibility information available for the feature

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

This VB example displays the visibility of the feature:

```
MsgBox GetFeatureVisibility("GevTimestampControl")
```

#### Remarks

Per GenICam standard, the visibility of a feature gives developers a hint about how to limit the access to a specific feature depending on the user's qualification.

---

### 3.2.61 GetFileAccessMode

#### Description

Returns the access mode in which the specified file can be opened in the device.

#### Syntax

[VB]

```
Value=objActiveGige.GetFileAccessMode ( Name )
```

[C/C++]

```
HRESULT GetFileAccessMode( bstr Name, bstr pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: String

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the file in the device

*pValue* [out, retval]

Pointer to the string representing the access mode to the file in the currently selected device:

"WO" - file has the write-only access

"RO" - file has the read-only access

"RW" - file has both read and write access

#### Return Values

S\_OK

Success

E\_NOINTERFACE

File with the specified name does not exist in the device

E\_FAIL

Failure

#### Example

This VB example displays the access mode of the file:

```
MsgBox GetFileAccessMode( "UserSet1" )
```

#### Remarks

The File Access functionality allows an application to read and write files hosted on the device. The data in those files may contain look-up tables, configuration sets, firmware, and other information.

This method lets you determine whether the file can be accessed for [Reading](#), [Writing](#), or for both operations.

### 3.2.62 GetFileList

#### Description

Returns the array of strings containing the names of all files in the device that can be accessed via the File Access functionality.

#### Syntax

[VB]

```
Value=objActiveGige.GetFileList
```

[C/C++]

```
HRESULT GetFileList ( VARIANT* pFiles );
```

#### Data Types [VB]

*Return value:* Variant (Array of strings)

#### Parameters [C/C++]

*pFiles* [out,retval]

Pointer to the SAFEARRAY of strings containing the list of files hosted in the camera. An empty array indicates that the File Access is not supported by the camera.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example initializes a combo box with the names of files hosted on the camera and displays the size of the selected file:

```
Private Sub Form_Load()  
FileList = ActiveGige.GetFileList  
For i = 0 To UBound(FileList)  
Combo1.AddItem (FileList(i))  
Next  
End Sub  
  
Private Sub Combo1_Click()  
FileName=Combo1.Text  
Label1.Caption = ActiveGige.GetFileSize (FileName)  
End Sub
```

#### Remarks

---

The File Access functionality allows an application to read and write files hosted on the device. The data in those files may contain look-up tables, configuration sets, firmware, and other information.

This method lets you determine if the device hosts any files and retrieve their names.

---

### 3.2.63 GetFileSize

#### Description

Returns the size of the specified file in the device in bytes.

#### Syntax

[VB]

```
Value=objActiveGige.GetFileSize( Name )
```

[C/C++]

```
HRESULT GetFileSize(bstr Name, long* pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: Single

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the file in the device

*pValue* [out, retval]

Pointer to the size of the file in bytes

#### Return Values

S\_OK

Success

E\_NOINTERFACE

File with the specified name does not exist in the device

E\_FAIL

Failure

#### Example

This VB example initializes a combo box with the names of files hosted on the camera and displays the size of the selected file:

```
Private Sub Form_Load()  
FileList = ActiveGige1.GetFileList  
For i = 0 To UBound(FileList)  
Combol.AddItem (FileList(i))  
Next  
End Sub  
  
Private Sub Combol_Click()  
FileName=Combol.Text  
Label1.Caption = ActiveGige1.GetFileSize (FileName)  
End Sub
```

---



**Remarks**

The File Access functionality allows an application to read and write files hosted on the device. The data in those files may contain look-up tables, configuration sets, firmware, and other information.

The size of the file can be used to reserve a proper amount of memory for the data transfer or to determine the maximum amount of bytes that can be written into the file.

---

### 3.2.64 GetFileTransferProgress

#### Description

Returns the progress of the current file access operation in percent.

#### Syntax

[VB]

```
Value=objActiveGige.GetFileTransferProgress( Name )
```

[C/C++]

```
HRESULT GetFileTransferProgress(bstr Name, float* pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: Single

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the file

*pValue* [out, retval]

Pointer to the value indicating the progress of the current file access operation in percent

#### Return Values

S\_OK

Success

E\_NOINTERFACE

File with the specified name does not exist in the device

E\_FAIL

Failed

#### Example

The following VB example demonstrates the use of a scroll control for real-time adjustment of the "GainRaw" feature.

```
Private Sub Form_Load()  
ActiveGigel.Acquire=True  
HScroll1.Value = ActiveGigel.GetFeature("GainRaw")  
HScroll1.Min = ActiveGigel.GetFeatureMin("GainRaw")  
HScroll1.Max = ActiveGigel.GetFeatureMax("GainRaw")  
End Sub
```

```
Private Sub HScroll1_Scroll()  
ActiveGigel.SetFeature("GainRaw", HScroll1.Value)  
End Sub
```

---

---

**Remarks**

### 3.2.65 GetFormatList

#### Description

Returns the array of strings containing the descriptions of pixel formats supported by the currently selected camera.

#### Syntax

[VB]  
`Value=objActiveGige.GetFormatList()`

[C/C++]  
`HRESULT GetFormatList( VARIANT* pList );`

#### Data Types [VB]

*Return value:* Variant (SAFEARRAY)

#### Parameters [C/C++]

*pList* [out,retval]  
Pointer to the SAFEARRAY of strings containing available formats

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example initializes a combo box with the descriptions of available pixel formats and uses it to select a specific format:

```
Private Sub Form_Load()  
FmtLst = ActiveGigel.GetFormatList  
For i = 0 To UBound(FmtLst)  
Combol.AddItem (FmtLst(i))  
Next  
Combol.ListIndex = 0  
ActiveGigel.Acquire = True  
End Sub  
  
Private Sub Combol_Click()  
ActiveGigel.Format = Combol.ListIndex  
End Sub
```

This MFC example fills out a combo box with available video Formats:

```
VARIANT m_FormatArray=m_ActiveGige.GetFormatList();  
SAFEARRAY *pArray=m_FormatArray.parray;
```

---

```
UINT nFormats=pArray->rgsabound[0].cElements;

CString str;
CComboBox *pFormat=(CComboBox*)GetDlgItem(IDC_FORMAT);
for(UINT i=0;i<nFormats;i++)
{
    pFormat->AddString(strFormat);
}
int iFormat=m_ActiveGige.GetFormat();
pFormat->SetCurSel(iFormat);

SafeArrayDestroy(pArray);
```

### Remarks

The format list is built in the acceding order by browsing through all pixel formats supported by the current camera. The index of an element in the Format list can be used as an argument of the [Format](#) property to select a specific pixel format.

Note that in C/C++ applications it is required to call `SafeArrayDestroy()` to delete the `SAFEARRAY` returned by **GetFormatList**.

### 3.2.66 GetFPSAcquired

#### Description

Returns the acquired (displayed) frame rate in frames per second.

#### Syntax

[VB]

```
Value=objActiveGige.GetFPSAcquired
```

[C/C++]

```
HRESULT GetFPSAcquired(float* pValue);
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the fps value

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example displays the acquired frame rate.

```
Private Sub ActiveGigel_FrameAcquired()  
    Label1.Caption = ActiveGigel.GetFPSAcquired  
End Sub
```

#### Remarks

This method returns the frame rate based on the arrival of the video frames into the application buffer. The acquired frame rate depends on the CPU power, performance of the graphic card, color conversion required for a given video mode and custom image processing performed on each frame. The acquired frame rate is equal to the frequency at which the [FrameAcquired](#) event is called in your application. When the acquired frame rate is lower than [camera frame rate](#), *ActiveGige* will fire the [FrameDropped](#) events.

---

### 3.2.67 GetFPS

#### Description

Returns the actual frame rate of the camera in frames per second.

#### Syntax

[VB]

```
Value=objActiveGige.GetFPS
```

[C/C++]

```
HRESULT GetFPS(float* pValue);
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out,retval]

Pointer to the fps value

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example displays the actual frame rate of the camera.

```
Private Sub ActiveGigel_FrameAcquired()  
    Label1.Caption = ActiveGigel.GetFPS  
End Sub
```

#### Remarks

This method returns the actual frame rate of the camera. Note that the actual frame rate may differ from the [acquired frame rate](#). *ActiveGige* calculates the actual frame rate using the [frame timestamps](#). If the camera doesn't support timestamps, the frame rate is calculated based on the arrival of the video packets to the system memory.

### 3.2.68 GetGainMax

#### Description

Returns the maximum value allowed for the camera's gain.

#### Syntax

[VB]

```
Value=objActiveGige.GetGainMax
```

[C/C++]

```
HRESULT GetGainMax( float* pValue );
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the maximum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGige1.GetGainMin  
Slider1.Max = ActiveGige1.GetGainMax  
Slider1.Value = ActiveGige1.Gain  
End Sub
```

#### Remarks

Note that this method is available only if the currently selected camera supports one of the following GenICam features: *Gain*, *GainAbs*, *GainRaw*.

---



### 3.2.69 GetGainMin

#### Description

Returns the minimum value allowed for the camera's gain.

#### Syntax

[VB]

```
Value=objActiveGige.GetGainMin
```

[C/C++]

```
HRESULT GetGainMin( float* pValue );
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the minimum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGigel.GetGainMin  
Slider1.Max = ActiveGigel.GetGainMax  
Slider1.Value = ActiveGigel.Gain  
End Sub
```

#### Remarks

Note that this method is available only if the currently selected camera supports one of the following GenICam features: *Gain*, *GainAbs*, *GainRaw*.

### 3.2.70 GetHeight

#### Description

Returns the actual height of the image.

#### Syntax

[VB]

```
Value=objActiveGige.GetHeight()
```

[C/C++]

```
HRESULT GetHeight(long* pValue );
```

#### Data Types [VB]

*Return value:* Long

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the height of the image

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

This C++ example reserves an array of the full frame size, grabs a frame, retrieves a pointer to the frame and copies the content of the frame into array using the actual height of the image.

```
int iSizeX=m_ActiveGige.GetSizeX();
int iSizeY=m_ActiveGige.GetSizeY();
BYTE* pArray=new BYTE [iSizeX*iSizeY];
int iHeight=m_ActiveGige.GetHeight();
m_ActiveGige.Grab()
for (int i=0; i<iHeight; i++)
{
    VARIANT v=m_ActiveGige.GetImagePointer(0,i);
    memcpy(pArray+iSizeX*i,v.pvData,iWidth);
}
```

#### Remarks

This method allows you to determine the height of the image is a situation when it can be smaller than the Y size of the internal frame buffer defined by the [SizeY](#) property. This typically happens when a linescan camera used in the start-stop trigger mode transfers images of a variable height. Knowing the

---

---

actual size of the image in the internal buffer allows you to process or display only a valid portion of the frame buffer and disregard an unused part.

---

### 3.2.71 GetHeightMax

#### Description

Returns the maximum available vertical size of the video frame.

#### Syntax

[VB]

```
Value=objActiveGige.GetHeightMax( )
```

[C/C++]

```
HRESULT GetHeightMax(long* pValue );
```

#### Data Types [VB]

*Return value:* Long

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the maximum vertical size of the video frame

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

This VB example uses the maximum video width and height to initialize slider controls:

```
Private Sub Form_Load()  
Slider1.Min = 0  
Slider1.Max = ActiveGige1.GetWidthMax  
Slider2.Min = 0  
Slider2.Max = ActiveGige1.GetHeightMax  
Slider1.Value = ActiveGige1.SizeX  
Slider2.Value = ActiveGige1.SizeY  
End Sub
```

#### Remarks

This method allows you to determine the maximum height of the video frame for the currently selected horizontal [Binning](#) and [Decimation](#). If Binning and Decimation are set to 1, the maximum height of the video frame is typically equal to the vertical resolution of the sensor.

---

### 3.2.72 GetHistogram

#### Description

Returns the histogram of the current image frame over a selected [ROI](#) .

#### Syntax

[VB]

```
Value=objActiveGige.GetHistogram(Component [,Bins, Step ])
```

[C/C++]

```
HRESULT GetHistogram( VARIANT* pHistogram, short Component, long Bins=256,  
short Step=1 );
```

#### Data Types [VB]

*Channel* : Integer

*Bins* : Long

*Step* : Integer

*Return value*: Variant (Array)

#### Parameters [C/C++]

*Component* [in]

Enumerated integer specifying the color component for which the histogram data are obtained. This parameter is disregarded for grayscale images. Must be one of the following values:

- 0 - collects the histogram of the luminance of the image
- 1 - collects the histogram of the red component of the image
- 2 - collects the histogram of the green component of the image
- 3 - collects the histogram of the blue component of the image

*Bins* [in]

Number of intervals used for histogram collection. It is recommended to set this parameter to a number of possible intensity levels in the image or to an integer fraction of that value, otherwise some precision will be lost due to averaging that occurs within the consolidated bins. A typical number of histogram bins for 8-bit and 16-bit images will be 256.

*Step* [in]

An optional integer between 1 and 16 specifying the sampling step, which is used to collect pixel values for the histogram calculations. If this number is 1, every pixel is taken into consideration. If the number is 2, every second pixel in a row will be skipped, as well as every second row in the image, etc. Increasing the sampling step raises the speed of the histogram calculations, but sacrifices some accuracy.

*pHistogram* [out,retval]

Pointer to the SAFEARRAY containing the histogram values.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

### Example

This VB example retrieves the histogram of the red component per each frame acquired:

```
Private Sub ActiveGigel_FrameAcquired()  
ActiveGigel.SetROI 100,100,500, 400  
HistArray=ActiveGigel.GetHistogram (1, 256)  
End Sub
```

### Remarks

The histogram is calculated over the rectangular region of interest defined by [SetROI](#). The luminance thresholds of the current ROI are disregarded. If the ROI is not set, the whole image frame will be used.

---

### 3.2.73 GetImageData

#### Description

Returns the two-dimensional array of pixel values in the currently acquired frame.

#### Syntax

[VB]

```
Value=objActiveGige.GetImageData
```

[C/C++]

```
HRESULT GetImageData( VARIANT* pArray );
```

#### Data Types [VB]

*Return value:* Variant (SAFEARRAY)

#### Parameters [C/C++]

*pArray* [out,retval]

Pointer to the SAFEARRAY containing the pixel values in the frame

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the [FrameAcquired](#) event to invert pixel value in the bottom left corner of the current frame and display the processed frame in real time.

```
Private Sub Form_Load()  
ActiveGigel.Display = False  
ActiveGigel.Acquire = True  
End Sub  
  
Private Sub ActiveGigel_FrameAcquired()  
a = ActiveGigel.GetImageData  
For x = 0 To 200  
For y = 0 To 200  
a(x, y) = 255 - a(x, y)  
Next  
Next  
ActiveGigel.Draw  
End Sub
```

#### Remarks

**GetImageData** does not copy the image data, so the array returned contains the actual image buffer of the currently acquired frame. Modifying elements of the array will change actual pixel values in the image buffer. This can be used to perform custom image processing and display a processed image in real time. Note that this feature cannot be used in .NET as its framework creates a copy of the array returned. For direct access to *ActiveGigE*'s image frame in VB.NET and C# use [GetImagePointer](#).

Images in *ActiveGigE* are stored bottom up, therefore the first element of the array is first pixel of the bottom line of the image. The type of data and dimensions of the array returned by **GetImageData** depends on the output format of the video, its horizontal width [SizeX](#) and the number of lines acquired, as specified in the following table:

Camera Pixel Format	Output Format	Data type	Dimensions
Mono8	8-bit monochrome	Byte	0 to SizeX -1, 0 to Lines - 1
Mono10, Mono12, Mono16	16-bit gray monochrome	Integer (word)	0 to SizeX -1, 0 to Lines - 1
YUV411, YUV422, YUV444, RGB8, BGR8, Bayer8	24-bit RGB	Byte	0 to SizeX * 3 - 1, 0 to Lines - 1
RGB10, RGB12, RGB16, BGR10, BGR12, Bayer10, Bayer12, Bayer16	48-bit RGB	Integer (word)	0 to SizeX * 3 - 1, 0 to Lines - 1

For integer (word) type of data you can receive negative numbers if pixel values exceed 32767. In C and C# you can convert signed integers to unsigned ones by using data casting. To get rid of negative values in VB and Delphi, subtract them from 65535.

Note that if images are transmitted by a linescan camera, the number of lines in each image can vary and be less than the vertical size of the image frame defined by the [SizeY](#) property. See [GetHeight](#) for more details.



### 3.2.74 GetImageLine

#### Description

Returns the array of pixel values at the specified horizontal line of the currently acquired frame.

#### Syntax

[VB]

```
Value=objActiveGige.GetImageLine( Y )
```

[C/C++]

```
HRESULT GetImageLine( short Y, VARIANT* pArray );
```

#### Data Types [VB]

Y: Integer

*Return value:* Variant (SAFEARRAY)

#### Parameters [C/C++]

Y [in]

The y-coordinate of the line in the image

pArray [out,retval]

Pointer to the SAFEARRAY containing the pixel values in the line

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid input argument.

#### Example

This VB example grabs a frame, retrieves the 33th row of pixels and displays the value of 11th pixel in the row.

```
Dim pix as Long
ActiveGige1.Grab
Line=ActiveGige1.GetImageLine(32)
pix=Line(10)
if pix < 0 then
pix=65535-pix
endif
MsgBox Line(10)
```

#### Remarks

The array returned by **GetImageLine** is a copy of the actual raw of pixels. Modifying elements of the array will not change actual pixel values in the frame buffer. The type of data and dimension of the array returned by **GetImageLine** depends on the output format of the video as specified in the following table:

Camera Pixel Format	Output Format	Data type	Dimension
Mono8	8-bit monochrome	Byte	0 to SizeX -1
Mono10, Mono12, Mono16	16-bit gray monochrome	Integer (word)	0 to SizeX -1
YUV411, YUV422, YUV444, RGB8, BGR8, Bayer8	24-bit RGB	Byte	0 to SizeX * 3 - 1
RGB10, RGB12, RGB16, BGR10, BGR12, Bayer10, Bayer12, Bayer16	48-bit RGB	Integer (word)	0 to SizeX * 3 - 1

The value of the y coordinate must not exceed the height of the video frame, or the error will occur. For integer (word) type of data you can receive negative numbers if pixel values exceed 32767. In C and C# you can convert signed integers to unsigned ones by using data casting. To get rid of negative values in VB and Delphi, subtract them from 65535 (see example above).

Note that in C/C++ applications it is required to call `SafeArrayDestroy()` to delete the `SAFEARRAY` returned by **GetImageLine**.

### 3.2.75 GetImagePointer

#### Description

Returns the pointer to the pixel at the specified coordinates of the current frame.

#### Syntax

[VB]

```
Value=objActiveGige.GetImagePointer( X, Y )
```

[C/C++]

```
HRESULT GetImagePointer( short X, short Y, VARIANT* pValue );
```

#### Data Types [VB]

X: Integer

Y: Integer

Return value: Variant (pointer)

#### Parameters [C/C++]

X [in]

The x-coordinate of the pixel

Y [in]

The y-coordinate of the pixel

pValue [out,retval]

Pointer to the variant containing the pointer to the specified memory location

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid input arguments.

#### Example

This C++ example grabs a frame, retrieves a pointer to the 32th line in the frame memory and copies the pixels values into a byte array . A wrapper class CActiveGige is used to access the *ActiveGige* control:

```
BYTE line[4096];  
int iWidth=m_ActiveGige.GetSizeX();  
int iHeight=m_ActiveGige.GetSizeY();  
m_ActiveGige.Grab()  
VARIANT v=m_ActiveGige.GetImagePointer(0,iHeight-1-32);  
memcpy(&line,v.pvVal,iWidth);
```

This C# example uses the [FrameAcquired](#) event to retrieve a pointer to the 32th line in the frame

memory and change pixel values in the line to zeros:

```
private void axActiveGigE1_FrameAcquired(object sender,
AxActiveGigELib._IActiveGigEEvents_FrameAcquiredEvent e)
{
    int sx=axActiveGigE1.SizeX;
    int sy=axActiveGigE1.SizeY;
    object obj=axActiveGigE1.GetImagePointer(0, sy-1-32);
    int a = (int)obj;
    byte* ptr= (byte*)a;
    for (int x=0; x<sx; x++, ptr++)
        *ptr=0;
}
```

### Remarks

The **GetImagePointer** method provides the most efficient way to quickly access the internal image buffer in pointer-aware programming languages. Unlike [GetImageData](#), this method doesn't modify original pixel values of high-bit depth images. The number of bytes in each line of the image buffer depends on the format and horizontal size of the video as specified in the following table:

Camera Pixel Format	Output Format	Line width in bytes
Mono8	8-bit monochrome	SizeX
Mono10, Mono12, Mono16	16-bit monochrome	SizeX * 2
YUV411, YUV422, YUV444, RGB8, BGR8, Bayer8	24-bit RGB	SizeX * 3
RGB10, RGB12, RGB16, BGR10, BGR12, Bayer10, Bayer12, Bayer16	48-bit RGB	SizeX * 6

Images in *ActiveGigE* are stored bottom up, therefore the zero vertical coordinate corresponds to the bottom line of the image.

The values of the x and y coordinates must not exceed the width and height of the video frame, or the error will occur.

Note that if images are transmitted by a linescan camera, the number of lines in each image can vary and be less than the vertical size of the image frame defined by the [SizeY](#) property. See [GetHeight](#) for more details.

### 3.2.76 GetImageStat

#### Description

Returns the array containing statistical data of the current image frame over a selected [ROI](#).

#### Syntax

[VB]

```
Value=objActiveGige.GetImageStat( Component [,Step] )
```

[C/C++]

```
HRESULT GetImageStat( VARIANT* pStat, short Component, short Step=1 );
```

#### Data Types [VB]

*Channel* : Integer

*Bins* : Long

*Step* : Integer

*Return value*: Variant (Array of Single values)

#### Parameters [C/C++]

*Component* [in]

Enumerated integer specifying the color component for which the image statistics is obtained. This parameter is disregarded for grayscale images. Must be one of the following values:

- 0 - collects the statistics of the luminance of the image
- 1 - collects the statistics of the red component of the image
- 2 - collects the statistics of the green component of the image
- 3 - collects the statistics of the blue component of the image

*Step* [in]

An optional integer between 1 and 16 specifying the sampling step, which is used to collect pixel values for the calculations. If this number is 1, every pixel is taking into consideration. If the number is 2, every second pixel in a row will be skipped, as well as every second row in the image, etc. Increasing the sampling step raises the speed of the image statistics collection, but sacrifices some accuracy.

*pStat* [out,retval]

Pointer to the SAFEARRAY (float) containing the image statistics. The statistics is written to the array as follows:

- Stat [0] – Mean value
- Stat [1] – Standard deviation
- Stat [2] – Pixel count
- Stat [3] – Minimum
- Stat [4] – Maximum
- Stat [5] – Median
- Stat [6] – Skewness
- Stat [7] – Kurtosis

#### Return Values

S\_OK

Success  
E\_FAIL  
Failure.

### Example

This VB example displays the mean value and standard deviation of the red component per each frame acquired:

```
Private Sub ActiveGigel_FrameAcquired()  
ActiveGigel.SetROI 100,100,500, 400  
StatArray=ActiveGigel.GetImageStat 1  
LabelMean.Caption=StatArray[0]  
LabelStd.Caption=StatArray[1]  
End Sub
```

### Remarks

The image statistics is calculated over the rectangular region of interest and luminance range defined by [SetROI](#). If the ROI is not set, the whole size and luminance range of the image will be used.

---

### 3.2.77 GetImageWindow

#### Description

Returns the two-dimensional array of pixel values corresponding to the selected window in the currently acquired frame.

#### Syntax

[VB]

```
Value=objActiveGige.GetImageWindow (X, Y, Width, Height)
```

[C/C++]

```
HRESULT GetImageWindow( short X, short Y, short Width, short Height,  
VARIANT* pArray );
```

#### Data Types [VB]

*X, Y, Width, Height*: Integer

*Return value*: Variant (SAFEARRAY)

#### Parameters [C/C++]

*X [in], Y[in]*

The x- and y-coordinates of the top left pixel of the window in the entire frame

*Width [in], Height [in]*

The horizontal and vertical size of the window in pixels.

*pArray [out,retval]*

Pointer to the SAFEARRAY containing the pixel values in the frame

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the [FrameAcquired](#) event to increase the brightness in the central area of the live image:

```
Private Sub Form_Load()  
ActiveGigel.Display = False  
ActiveGigel.Acquire = True  
End Sub  
  
Private Sub ActiveGigel_FrameAcquired()  
xc = ActiveGigel.SizeX / 2  
yc = ActiveGigel.SizeY / 2  
w = ActiveGigel.GetImageWindow(xc - 70, yc - 50, 140, 100)  
For y = 0 To UBound(w, 2)  
For x = 0 To UBound(w, 1)
```

```

pix = w(x, y) + 50
If pix > 255 Then
pix = 255
End If
w(x, y) = pix
Next
Next
ActiveGigel.SetImageWindow xc - 70, yc - 50, w
ActiveGigel.Draw
End Sub

```

### Remarks

The image data in the array are stored in the standard order from top to bottom, therefore the first element of the array is the top left pixel of the window. This is opposite to the order of rows in the arrays returned by [GetImageData](#). The type of data and dimensions of the array returned by **GetImageWindow** depends on the output format of the video, the width and height of the window and the number of lines acquired, as specified in the following table:

Camera Pixel Format	Output Format	Data type	Dimensions
Mono8	8-bit monochrome	Byte	0 to SizeX - 1, 0 to Lines - 1
Mono10, Mono12, Mono16	16-bit gray monochrome	Integer (word)	0 to SizeX - 1, 0 to Lines - 1
YUV411, YUV422, YUV444, RGB8, BGR8, Bayer8	24-bit RGB	Byte	0 to SizeX * 3 - 1, 0 to Lines - 1
RGB10, RGB12, RGB16, BGR10, BGR12, Bayer10, Bayer12, Bayer16	48-bit RGB	Integer (word)	0 to SizeX * 3 - 1, 0 to Lines - 1

If the dimensions of the window are too large to accommodate the frame size, they will be clipped to the frame boundaries.

For integer (word) type of data you can receive negative numbers if pixel values exceed 32767. In C and C# you can convert signed integers to unsigned ones by using data casting. To get rid of negative values in VB and Delphi, subtract them from 65535.

If images are transmitted by a linescan camera, the number of lines in each image can vary and be less than the vertical size of the image frame defined by the [SizeY](#) property. See [GetHeight](#) for more details.

Note that modifying elements of the array will not change actual pixel values in the frame buffer. Use [SetImageWindow](#) to transfer pixel values into *ActiveGige*.

Note that in C/C++ applications it is required to call `SafeArrayDestroy()` to delete the `SAFEARRAY` returned by **GetImageWindow**.



### 3.2.78 GetLevels

#### Description

Returns the array containing the current levels for the Window/Level operation.

#### Syntax

[VB]

```
value = objActiveGige.GetLevels
```

[C/C++]

```
HRESULT SetROI( VARIANT* pLevels );
```

#### Data Types [VB]

*Return value:* Variant (Array)

#### Parameters [C/C++]

*pLevels* [out,retval]

Pointer to the SAFEARRAY containing the current levels for the Window/Level operation:

ROI [0] - current minR level

ROI [1] - current maxR level

ROI [2] - current minG level

ROI [3] - current maxG level

ROI [4] - current minB level

ROI [5] - current maxB level

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example displays the currently selected levels for a monochrome image:

```
Levels=ActiveGigel.GetLevels
```

```
MsgBox "Min: ", Levels[0], "Max: ", Levels[1]
```

#### Remarks

For more information on the Window/Level parameters see [SetLevels](#).

### 3.2.79 GetLUT

#### Description

Returns the array containing the current lookup table.

#### Syntax

[VB]

```
value = objActiveGige.GetLUT
```

[C/C++]

```
HRESULT GetLUT( VARIANT* pLUT );
```

#### Data Types [VB]

*Return value:* Variant (Array)

#### Parameters [C/C++]

*pLUT* [out,retval]

Pointer to the SAFEARRAY of floating point values containing the current lookup table.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example displays several elements of the current lookup table:

```
LUT=ActiveGige1.GetLUT
MsgBox "LUT(100): ", LUT(100), "LUT(101): ", LUT(101), "LUT(102): ", LUT
(102)
```

#### Remarks

For more information on the lookup table refer to [SetLUT](#)

---

### 3.2.80 GetOptimalPacketSize

#### Description

Returns the optimal packet size for the video streaming in the current network configuration.

#### Syntax

[VB]

```
Value=objActiveGige.GetOptimalPacketSize
```

[C/C++]

```
HRESULT GetOptimalPacketSize( long* pValue );
```

#### Data Types [VB]

*Return value:* long

#### Parameters [C/C++]

*pValue* [out,retval]

Pointer to the optimal packet value.

#### Return Values

S\_OK

Success

E\_FAIL

Failure

E\_NOINTERFACE

Camera does not support packet size negotiation

#### Example

This VB example finds the optimal packet size and modifies the packet size register of the camera accordingly. If the optimal packet size negotiation failed, the packet size is set to 1500.

```
ActiveGigE1.Camera = 2
MTU=ActiveGigE1.GetOptimalPacketSize
if MTU=0 then
MTU=1500
endif
ActiveGigE1.PacketSize=MTU
```

#### Remarks

The optimal packet size is determined using the packet negotiation functionality of GigE Vision cameras. Network packets of different sizes are requested from the camera until the maximum packet size possible for the current configuration is found. If the camera does not support the packet size negotiation, this method will return zero.

The value of the optimal packet size is typically defined by the maximum packet size supported by the network card and camera, whichever is lower. Setting the packet size to the optimal value reduces the CPU load during the video transmission.

negotiation

---

### 3.2.81 GetPicture

#### Description

Returns a Picture object created from the currently acquired frame.

#### Syntax

[VB]

```
Value=objActiveGige.GetPicture()
```

[C/C++]

```
HRESULT GetPicture(IPictureDisp* *pPicture);
```

#### Data Types [VB]

*Return value:* Picture

#### Parameters [C/C++]

*pPicture* [out,retval]

Pointer to the *IPictureDisp* interface object

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the [FrameAcquired](#) event to display a live video in a PictureBox:

```
Private Sub ActiveGigel_FrameAcquired(ByVal Lines As Integer)
    Picture1.Picture=ActiveGigel.GetPicture
End Sub
```

This VB.NET example shows how to display a live video in a PictureBox:

```
Private Sub AxActiveGigel_FrameAcquired(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles AxActiveGigel.FrameAcquired
    If Not (PictureBox1.Image Is Nothing) Then PictureBox1.Image.Dispose()
    PictureBox1.Image = Bitmap.FromHbitmap(AxActiveGigel.GetPicture.Handle)
End Sub
```

This C# statement shows how to display an image in a PictureBox:

```
pictureBox1.Image =
Bitmap.FromHbitmap((System.IntPtr)axActiveGigel.GetPicture().Handle);
```

**Remarks**

The **GetPicture** method provides the most convenient graphic interface to *ActiveGige* internal image and allows you to display the last acquired frame in popular graphic controls such as PictureBox. Note that a Picture object contains only 8-bit pixel values, even if the current video mode is a 16- or 48-bit one. To access the actual pixel data, use the [GetImageData](#) or [GetPointer](#) methods.

In .NET programming environment calling **GetPicture** inside the [FrameAcquiredX](#) event handler may not work. Use the [FrameAcquired](#) event instead. It is also necessary to apply the Dispose() method to the PictureBox Image property before each subsequent call to **GetPicture** in order to prevent the memory leak.

---

### 3.2.82 GetPixel

#### Description

Returns the pixel value at the specified coordinates.

#### Syntax

[VB]

```
Value=objActiveGige.GetPixel( X, Y )
```

[C/C++]

```
HRESULT GetPixel( short X, short Y, long* pValue );
```

#### Data Types [VB]

X: Integer

Y: Integer

Return value: Long

#### Parameters [C/C++]

X [in]

The x-coordinate of the pixel

Y [in]

The y-coordinate of the pixel

pValue [out,retval]

Pointer to the pixel's value

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid input arguments.

#### Example

This VB example grabs a frame and displays the value of the pixel at the specified coordinates.

```
ActiveGigel.Grab  
value=ActiveGigel.GetPixel(64,32)  
MsgBox value
```

#### Remarks

The value returned by **GetPixel** depends on the format of the video acquired. For monochrome video the method will retrieve the data from the internal image memory. For the color video the RGB data in the specified coordinates will be converted to the luminance using the formula:  $L=(R+G+B) / 3$ .

The values of the x and y coordinates must not exceed the width and height of the video frame, or the error will occur.



### 3.2.83 GetRawData

#### Description

Returns the two-dimensional array of values in the currently acquired data buffer or pointer to this buffer.

#### Syntax

[VB]

```
Value=objActiveGige.GetRawData ([isPointer=False])
```

[C/C++]

```
HRESULT GetRawData( bool isPointer, VARIANT* pArray );
```

#### Data Types [VB]

*Return value:* Variant (SAFEARRAY)

#### Parameters [C/C++]

*isPointer* [in]

If false returns the array of raw data, otherwise returns pointer to data.

*pArray* [out,retval]

Pointer to the SAFEARRAY containing the raw data buffer.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example grabs a frame, retrieves the raw data array and displays the value of the specified element of the array.

```
Dim pix as Long
ActiveGigel.Grab
RawData=ActiveGigel.GetRawData
x=16
y=32
pix=RawData(x,y)
if pix < 0 then
pix=65535-pix
endif
MsgBox RawData(x,y)
```

#### Remarks

This function returns the frame data as they arrive from the camera (before being converted to a viewable format, such as RGB). **GetRawData** does not copy the image data, so the array returned contains the actual image buffer of the currently acquired frame. Modifying elements of the array will change actual pixel values in the image buffer. This can be used to perform custom image processing and display a processed image in real time. Note that this feature cannot be used in .NET as its framework creates a copy of the array returned. For direct access to raw data in C# set *isPointer* to True and use the pointer instead of an array.

Raw image data are stored in the standard order from top to bottom, therefore the first element of the array is first pixel of the top line of the image. The type of data and dimensions of the array returned by **GetRawData** depends on the current pixel format, horizontal width [SizeX](#) and the number of lines acquired, as specified in the following table:

Pixel Format	Data type	Dimensions
Mono8, Mono8s	Byte	0 to SizeX - 1, 0 to Lines - 1
Bayer**8	Byte	0 to SizeX - 1, 0 to Lines - 1
Mono10, Mono12, Mono16	Integer (word)	0 to SizeX - 1, 0 to Lines - 1
Bayer**10, Bayer**12, Bayer**16	Integer (word)	0 to SizeX - 1, 0 to Lines - 1
Mono10Packed, Mono12Packed	Byte	0 to SizeX*3/2 - 1, 0 to Lines - 1
YUV411_8_UYVYY	Byte	0 to SizeX*3/2 - 1, 0 to Lines - 1
YUV422_8_UYVY, YUV8_YUV	Integer (word)	0 to SizeX - 1, 0 to Lines - 1
RGB8, BGR8	Byte	0 to SizeX * 3 - 1, 0 to Lines - 1
RGBa8, BGRa8	Byte	0 to SizeX * 4 - 1, 0 to Lines - 1
RGB10, BGR10, RGB12, BGR12	Integer (word)	0 to SizeX * 3 - 1, 0 to Lines - 1
BGR10V1Packed, BGR10V2Packed	Long	0 to SizeX - 1, 0 to Lines - 1
RGB8Planar	Byte	0 to SizeX * 3 - 1, 0 to Lines - 1
RGB10Planar, RGB12Planar, RGB16Planar	Integer (word)	0 to SizeX * 3 - 1, 0 to Lines - 1
Coord3D_ABC16	Integer (word)	1, 0 to SizeY * 3 - 1
Coord3D_ABC32f	Float (dword)	1, 0 to SizeY * 3 - 1

For integer (word) type of data you can receive negative numbers if pixel values exceed 32767. In C and C# you can convert signed integers to unsigned ones by using data casting. To get rid of negative values in VB and Delphi, subtract them from 65535 (see the example above).

If the camera is streaming in the JPEG or H.264 compression mode, the raw data buffer will contain a compressed frame of a variable size not directly related to the dimensions of the image.

For 3D point cloud formats the raw data buffer will contain a linear array of XYZ coordinates with SizeY representing the maximum number of points in the array.

Note that setting both [Display](#) and [Magnification](#) properties to zero will disable an image conversion pipeline that is normally applied to each raw frame. This will significantly increase the performance of your application as long as it does not utilize *ActiveGigE's* built-in image decoding/display.

### 3.2.84 GetRGBPixel

#### Description

Returns the array of RGB values at the specified coordinates.

#### Syntax

[VB]

```
Value=objActiveGige.GetRGBPixel( X, Y )
```

[C/C++]

```
HRESULT GetRGBPixel( short X, short Y, VARIANT* pArray );
```

#### Data Types [VB]

X, Y: Integer

Return value: Variant (SAFEARRAY)

#### Parameters [C/C++]

X[in]

The x-coordinate of the pixel

Y[in]

The y-coordinate of the pixel

pArray[out,retval]

Pointer to the SAFEARRAY of the dimension of 3 containing long R, G and B values of the current pixel

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid input arguments.

#### Example

This VB example grabs a frame and displays the value of the G-value of the pixel at the specified coordinates.

```
ActiveGigel.Grab  
RGB=ActiveGigel.GetRGBPixel(64,32)  
MsgBox RGB(1)
```

#### Remarks

The values returned by **GetRGBPixel** depend on the format of the video acquired. For 24-bit video the method will retrieve the data from the internal image memory. For the grayscale video the R, G, and B

values will be the same and equal to the luminance value in the specified coordinates.

The values of the x and y coordinates must not exceed the width and height of the video frame, or the error will occur.

### 3.2.85 GetROI

#### Description

Returns the array containing the current ROI settings.

#### Syntax

[VB]

```
value = objActiveGige.GetROI
```

[C/C++]

```
HRESULT GetROI( VARIANT* pROI );
```

#### Data Types [VB]

*Return value:* Variant (Array)

#### Parameters [C/C++]

*pROI* [out,retval]

Pointer to the SAFEARRAY containing the current ROI settings as follows:

ROI [0] - horizontal coordinate of the top left corner of the ROI, relative to the image origin

ROI [1] - vertical coordinate of the top left corner of the ROI, relative to the image origin.

ROI [2] - horizontal coordinate of the bottom right corner of the ROI, relative to the image origin.

ROI [3] - vertical coordinate of the bottom right corner of the ROI, relative to the image origin.

ROI [4] - lower threshold of the luminance range for image statistics calculations

ROI [5] - higher threshold of the luminance range for image statistics calculations

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example displays the settings of the current ROI:

```
ROI=ActiveGigel.GetROI
```

```
MsgBox "X1: ", ROI[0], "Y1: ", ROI[1], "X2: ", ROI [2], "Y2: ", ROI [3]
```

#### Remarks

For more information on the region of interest see [SetROI](#), [GetImageStat](#), [GetHistogram](#).

### 3.2.86 GetSequenceFrameCount

#### Description

Returns the current number of frames in the memory sequence.

#### Syntax

[VB]

```
Value=objActiveGige.GetSequenceFrameCount( )
```

[C/C++]

```
HRESULT GetSequenceFrameCount(long* pValue );
```

#### Data Types [VB]

*Return value:* Long

#### Parameters [C/C++]

*pValue* [out,retval]

Pointer to the number of frames

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

This VB example displays the number of frames in the memory sequence.

```
value=ActiveGige1.GetSequenceFrameCount( )  
MsgBox value
```

#### Remarks

For more information on the sequence capture see [StartSequenceCapture](#).

---

### 3.2.87 GetSequencePicture

#### Description

Returns a Picture object representing a selected frame in the memory sequence.

#### Syntax

[VB]

```
Value=objActiveGige.GetSequencePicture( Frame )
```

[C/C++]

```
HRESULT GetSequencePicture(long iFrame, IPictureDisp* *pPicture);
```

#### Data Types [VB]

*Long*: Frame

*Return value*: Picture

#### Parameters [C/C++]

*iFrame*

The zero-based index of the frame for which a Picture object will be created

*pPicture* [out,retval]

Pointer to the *IPictureDisp* interface object

#### Return Values

S\_OK

Success

E\_FAIL

Failure

E\_INVALIDARG

Invalid frame index

#### Example

This VB example demonstrated how to display the 15th frame from the memory sequence in a picture box:

```
Picture1.Picture=ActiveGigel.GetSequencePicture (15)
```

#### Remarks

The **GetSequencePicture** method provides allows you to display frames from the memory sequence in popular graphic controls such as PictureBox. Note that a Picture object contains only 8-bit pixel values, even if the current video mode is a 16- or 48-bit one. To access the actual pixel data, use the [GetSequenceData](#) or [GetSequencePointer](#) methods.

### 3.2.88 GetSequencePixel

#### Description

Returns the pixel value at the specified coordinates.

#### Syntax

[VB]

```
Value=objActiveGige.GetSequencePixel( Frame, X, Y )
```

[C/C++]

```
HRESULT GetSequencePixel( long iFrame, short X, short Y, long* pValue );
```

#### Data Types [VB]

*Long*: Frame

*X*: Integer

*Y*: Integer

*Return value*: Long

#### Parameters [C/C++]

*iFrame* [in]

The zero-based index of the frame in the memory sequence

*X* [in]

The x-coordinate of the pixel

*Y* [in]

The y-coordinate of the pixel

*pValue* [out,retval]

Pointer to the pixel's value

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid input arguments.

#### Example

This VB example displays the value of the pixel at the specified coordinates of the memory sequence frame #0.

```
ActiveGigel.Grab  
value=ActiveGigel.GetSequencePixel(0,64,32)  
MsgBox value
```

#### Remarks

---



---

The value returned by **GetSequencePixel** depends on the format of the video acquired. For monochrome video the method will return the actual pixel value. For the color video the RGB data in the specified coordinates will be converted to the luminance using the formula:  $L=(R+G+B) / 3$ .

The values of the x and y coordinates must not exceed the width and height of the video frame, or the error will occur.

---

### 3.2.89 GetSequencePointer

#### Description

Returns the pointer to the pixel at the specified coordinates of the selected frame in the memory sequence.

#### Syntax

[VB]

```
Value=objActiveGige.GetSequencePointer( Frame, X, Y )
```

[C/C++]

```
HRESULT GetSequencePointer( long iFrame, short X, short Y, VARIANT* pValue );
```

#### Data Types [VB]

*Frame*: Long

*X*: Integer

*Y*: Integer

*Return value*: Variant (pointer)

#### Parameters [C/C++]

*iFrame* [in]

The zero-based index of the frame in the memory sequence

*X* [in]

The x-coordinate of the pixel

*Y* [in]

The y-coordinate of the pixel

*pValue* [out,retval]

Pointer to the variant containing the pointer to the specified memory location

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid input arguments.

#### Example

This C++ example grabs a frame, retrieves a pointer to the 32th line in the frame # 7 of the memory sequence and copies the pixels values into a byte array . A wrapper class CActiveGige is used to access the *ActiveGige* control:

```
BYTE line[4096];  
int iWidth=m_ActiveGige.GetSizeX;  
int iHeight=m_ActiveGige.GetSizeY;
```

---

```
VARIANT v=m_ActiveGige.GetSequencePointer(7,0,iHeight-1-32);  
memcpy(&line,v.pcVal,iWidth);
```

### Remarks

The **GetSequencePointer** method provides the most efficient way to quickly access the memory sequence data in pointer-aware programming languages. The number of bytes in each line of the image buffer depends on the format and horizontal size of the video as specified in the following table:

Camera Pixel Format	Output Format	Line width in bytes
Mono8	8-bit monochrome	SizeX
Mono10, Mono12, Mono16	16-bit monochrome	SizeX * 2
YUV411, YUV422, YUV444, RGB8, BGR8, Bayer8	24-bit RGB	SizeX * 3
RGB10, RGB12, RGB16, BGR10, BGR12, Bayer10, Bayer12, Bayer16	48-bit RGB	SizeX * 6

Note that this method automatically converts the raw sequence data into standard monochrome or RGB output formats. To access the raw image data, use [GetSequenceRawData](#).

Images in *ActiveGige* are stored bottom up, therefore the zero vertical coordinate corresponds to the bottom line of the image.

The values of the x and y coordinates must not exceed the width and height of the video frame, or the error will occur.

### 3.2.90 GetSequenceRawData

#### Description

Returns the two-dimensional array of raw values in the selected frame buffer of the memory sequence or pointer to this buffer.

#### Syntax

[VB]

```
Value=objActiveGige.GetSequenceRawData (Frame [, isPointer=False])
```

[C/C++]

```
HRESULT GetSequenceRawData( long iFrame, bool isPointer, VARIANT* pArray );
```

#### Data Types [VB]

*Frame*: Long

*Return value*: Variant (SAFEARRAY)

#### Parameters [C/C++]

*iFrame* [in]

The zero-based index of the frame in the memory sequence

*isPointer* [in]

If false returns the array of raw data, otherwise returns pointer to data.

*pArray* [out,retval]

Pointer to the SAFEARRAY containing the raw data buffer.

#### Return Values

S\_OK

Success

E\_FAIL

Failure

E\_INVALIDARG

Invalid input arguments

#### Example

This VB retrieves the raw data array from frame #4 in the memory sequence and displays the value of the specified element of the array.

```
Dim pix as Long
RawData=ActiveGigel.GetRawSequenceData
x=16
y=32
pix=RawSequenceData(4,x,y)
if pix < 0 then
pix=65535-pix
endif
```

---

`MsgBox RawData(x,y)`

### Remarks

This function returns the sequence data as they have been recorded (not converted). Raw image data are stored in the standard order from top to bottom, therefore the first element of the array is first pixel of the top line of the image. The type of data and dimensions of the array returned by **GetSequenceRawData** depends on the current pixel format, horizontal width [SizeX](#) and the number of lines acquired, as specified in the following table:

Pixel Format	Data type	Dimensions
Mono8, Mono8Signed	Byte	0 to SizeX - 1, 0 to Lines - 1
Bayer**8	Byte	0 to SizeX - 1, 0 to Lines - 1
Mono10, Mono12, Mono16	Integer (word)	0 to SizeX - 1, 0 to Lines - 1
Bayer**10, Bayer**12, Bayer**16	Integer (word)	0 to SizeX - 1, 0 to Lines - 1
Mono10Packed, Mono12Packed	Byte	0 to SizeX*3/2 - 1, 0 to Lines - 1
YUV411Packed	Byte	0 to SizeX*3/2 - 1, 0 to Lines - 1
YUV422Packed, YUV444Packed	Integer (word)	0 to SizeX - 1, 0 to Lines - 1
RGB8Packed, BGR8Packed	Byte	0 to SizeX * 3 - 1, 0 to Lines - 1
RGBA8Packed, BGRA8Packed	Byte	0 to SizeX * 4 - 1, 0 to Lines - 1
RGB10Packed, BGR10Packed, RGB12Packed, BGR12Packed	Integer (word)	0 to SizeX * 3 - 1, 0 to Lines - 1
BGR10V1Packed, BGR10V2Packed	Long	0 to SizeX - 1, 0 to Lines - 1
RGB8Planar	Byte	0 to SizeX * 3 - 1, 0 to Lines - 1
RGB10Planar, RGB12Planar, RGB16Planar	Integer (word)	0 to SizeX * 3 - 1, 0 to Lines - 1

For integer (word) type of data you can receive negative numbers if pixel values exceed 32767. In C and C# you can convert signed integers to unsigned ones by using data casting. To get rid of negative values in VB and Delphi, subtract them from 65535 (see the example above).

### 3.2.91 GetSequenceTimestamp

#### Description

Returns the timestamp of the selected frame in the memory sequence, in seconds.

#### Syntax

[VB]

```
Value=objActiveGige.GetSequenceTimestamp (Frame)
```

[C/C++]

```
HRESULT GetSequenceTimestamp(long iFrame, double* pValue);
```

#### Data Types [VB]

*Frame*: Long

*Return value*: Double

#### Parameters [C/C++]

*iFrame* [in]

The zero-based index of the frame in the memory sequence

*pValue* [out,retval]

Pointer to the timestamp value

#### Return Values

S\_OK

Success

E\_FAIL

Failure

E\_INVALIDARG

Invalid frame index

#### Example

This VB example displays the timestamp value of each frame of the sequence in a text box.

```
Private Sub ActiveGigel_FrameLoaded()  
    Label1.Caption = ActiveGigel.GetSequenceTimestamp  
End Sub
```

#### Remarks

If your GigE Vision™ camera support the timestamp function, it will mark each frame with a very accurate time value indicating the precise moment at which the frame was acquired by the camera relative to the moment at which the camera was powered up.

If the camera doesn't support timestamps, this method will return the time passed since January 1, 1970.

---

### 3.2.92 GetSequenceWindow

#### Description

Returns the two-dimensional array of pixel values corresponding to the selected window in the frame of the memory sequence.

#### Syntax

[VB]

```
Value=objActiveGige.GetSequenceWindow (Frame, X, Y, Width, Height)
```

[C/C++]

```
HRESULT GetSequenceWindow( long iFrame, short X, short Y, short Width,  
short Height, VARIANT* pArray );
```

#### Data Types [VB]

*X, Y, Width, Height*: Integer

*Return value*: Variant (SAFEARRAY)

#### Parameters [C/C++]

*iFrame* [in]

The zero-based index of the frame in the memory sequence

*X* [in], *Y* [in]

The x- and y-coordinates of the top left pixel of the window in the entire frame

*Width* [in], *Height* [in]

The horizontal and vertical size of the window in pixels.

*pArray* [out,retval]

Pointer to the SAFEARRAY containing the pixel values in the frame

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example fills a 3D array (x, y, frame) with pixel values from the memory sequence.

```
For z= 0 To ActiveGigel.GetSequenceFrameCount-1  
sx=ActiveGigel.SizeX  
sy=ActiveGigel.SizeY  
a = ActiveGigel.GetSequenceWindow (z, 0, 0, sx, sy)  
For x = 0 To sx-1  
For y = 0 To sy-1  
M(x,y,z) = a(x, y)  
Next
```

[Next](#)  
[Next](#)

### Remarks

The image data in the array are stored in the standard order from top to bottom, therefore the first element of the array is the top left pixel of the window. The type of data and dimensions of the array returned by **GetSequenceWindow** depends on the output format of the video, the width and height of the window and the number of lines acquired, as specified in the following table:

Camera Pixel Format	Output Format	Data type	Dimensions
Mono8	8-bit monochrome	Byte	0 to SizeX - 1, 0 to Lines - 1
Mono10, Mono12, Mono16	16-bit gray monochrome	Integer (word)	0 to SizeX - 1, 0 to Lines - 1
YUV411, YUV422, YUV444, RGB8, BGR8, Bayer8	24-bit RGB	Byte	0 to SizeX * 3 - 1, 0 to Lines - 1
RGB10, RGB12, RGB16, BGR10, BGR12, Bayer 10, Bayer 12, Bayer16	48-bit RGB	Integer (word)	0 to SizeX * 3 - 1, 0 to Lines - 1

If the dimensions of the window are too large to accommodate the frame size, they will be clipped to the frame boundaries.

For integer (word) type of data you can receive negative numbers if pixel values exceed 32767. In C and C# you can convert signed integers to unsigned ones by using data casting. To get rid of negative values in VB and Delphi, subtract them from 65535.

Note that in C/C++ applications it is required to call SafeArrayDestroy() to delete the SAFEARRAY returned by **GetSequenceWindow**.



### 3.2.93 GetTimestamp

#### Description

Returns the timestamp of the last acquired frame in seconds.

#### Syntax

[VB]

```
Value=objActiveGige.GetTimestamp
```

[C/C++]

```
HRESULT GetTimestamp(double* pValue);
```

#### Data Types [VB]

*Return value:* Double

#### Parameters [C/C++]

*pValue* [out,retval]

Pointer to the timestamp value

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example displays the timestamp value of each frame in a text box.

```
Private Sub ActiveGigel_FrameAcquired()  
    Label1.Caption = ActiveGigel.GetTimestamp  
End Sub
```

#### Remarks

If your GigE Vision™ camera support the timestamp function, it will mark each frame with a very accurate time value indicating the precise moment at which the frame was acquired by the camera relative to the moment at which the camera was powered up. Timestamps can be used to imprint a text with the timeline information into images during the AVI capture.

If this method is used in the [FrameAcquired](#) event handler, it must be called immediately after the event has been received. This will guarantee that the value of the timestamp will not be taken from the next frame while the current frame is being processed.

If the camera doesn't support timestamps, this method will return zero.

### 3.2.94 GetTriggerDelayMax

#### Description

Returns the maximum value allowed for the camera's trigger delay.

#### Syntax

[VB]

```
Value=objActiveGige.GetTriggerDelayMax
```

[C/C++]

```
HRESULT GetTriggerDelayMax( float* pValue );
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the maximum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGigel.GetTriggerDelayMin  
Slider1.Max = ActiveGigel.GetTriggerDelayMax  
Slider1.Value = ActiveGigel.TriggerDelay  
End Sub
```

#### Remarks

Note that this method is available only if the currently selected camera supports one of the following GenICam features: *TriggerDelay*, *TriggerDelayAbs*, *TriggerDelayRaw*.

---

### 3.2.95 GetTriggerDelayMin

#### Description

Returns the minimum value allowed for the camera's trigger delay.

#### Syntax

[VB]

```
Value=objActiveGige.GetTriggerDelayMin
```

[C/C++]

```
HRESULT GetTriggerDelayMin( float* pValue );
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the minimum value of the feature

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failure to read the value

#### Example

This VB example uses the minimum and maximum values to initialize the slider control:

```
Private Sub Form_Load()  
Slider1.Min = ActiveGigel.GetTriggerDelayMin  
Slider1.Max = ActiveGigel.GetTriggerDelayMax  
Slider1.Value = ActiveGigel.TriggerDelay  
End Sub
```

#### Remarks

Note that this method is available only if the currently selected camera supports one of the following GenICam features: *TriggerDelay*, *TriggerDelayAbs*, *TriggerDelayRaw*.

### 3.2.96 GetVideoFPS

#### Description

Returns the playback frame rate of the currently open video file or memory sequence. The default frame rate at which the video was recorded can be changed by calling [SetVideoFPS](#).

#### Syntax

[VB]

```
Value=objActiveGige.GetVideoFPS
```

[C/C++]

```
HRESULT GetVideoFPS(float* pValue);
```

#### Data Types [VB]

*Return value:* Single

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the fps value

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example opens an AVI file and displays its frame rate.

```
ActiveGige1.OpenVideo "c:\\video1.avi"  
MsgBox ActiveGige1.GetVideoFPS
```

---

### 3.2.97 GetVideoFrameCount

#### Description

Returns the number of frames in the currently [open video](#) file.

#### Syntax

[VB]  
`Value=objActiveGige.GetVideoFrameCount()`

[C/C++]  
`HRESULT GetVideoFrameCount(long* pValue );`

#### Data Types [VB]

*Return value:* Long

#### Parameters [C/C++]

*pValue* [out,retval]  
Pointer to the number of frames

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure

#### Example

This VB example displays the number of frames in a TIFF video file.

```
ActiveGigel.OpenVideo "c:\\sequence.tif"  
value=ActiveGigel.GetVideoFrameCount()  
MsgBox value
```

### 3.2.98 GetVideoPosition

#### Description

Returns the zero-based position of the current frame in the open video file or sequence.

#### Syntax

[VB]

```
Value=objActiveGige.GetVideoPosition
```

[C/C++]

```
HRESULT GetVideoPosition(long* pFrame );
```

#### Data Types [VB]

*Return value:* Long

#### Parameters [C/C++]

*pFrame* [out,retval]

Pointer to the zero-based index of the current frame.

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

This VB example uses the [FrameLoaded](#) event to display the number of a currently played frame.

```
Private Sub ActiveGigel_FrameLoaded()  
frameindex = ActiveGigel.GetVideoPosition + 1  
LabelCount.Caption = frameindex
```

#### Remarks

When the video file or sequence has just been opened, the video position is reset to zero.

---

### 3.2.99 GetVideoVolume

#### Description

Returns the volume level of the currently open AVI file.

#### Syntax

[VB]

```
Value=objActiveGige.GetVideoVolume
```

[C/C++]

```
HRESULT GetVideoVolume(short* pValue );
```

#### Data Types [VB]

*Return value:* Integer

#### Parameters [C/C++]

*pValue* [out,retval]

Pointer to the current volume, in percent.

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

This VB example displays the volume of the currently open video file.

```
ActiveGige.OpenVideo "C:\\\\video1.avi", True  
MsgBox ActiveGigel.GetVideoVolume
```

#### Remarks

In order for this method to work, the AVI file must be recorded with the sound and [opened](#) with the sound option enabled.

### 3.2.100 GetWidthMax

#### Description

Returns the maximum available horizontal size of the video frame.

#### Syntax

[VB]

```
Value=objActiveGige.GetWidthMax( )
```

[C/C++]

```
HRESULT GetWidthMax(long* pValue );
```

#### Data Types [VB]

*Return value:* Long

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the maximum horizontal size of the video frame

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

This VB example uses the maximum video width and height to initialize slider controls:

```
Private Sub Form_Load()  
Slider1.Min = 0  
Slider1.Max = ActiveGige1.GetWidthMax  
Slider2.Min = 0  
Slider2.Max = ActiveGige1.GetHeightMax  
Slider1.Value = ActiveGige1.SizeX  
Slider2.Value = ActiveGige1.SizeY  
End Sub
```

#### Remarks

This method allows you to determine the maximum width of the video frame for the currently selected horizontal [Binning](#) and [Decimation](#). If Binning and Decimation are set to 1, the maximum width of the video frame is typically equal to the horizontal resolution of the sensor.

---



### 3.2.101 Grab

#### Description

Grabs a single frame into the internal memory. If the [Display](#) property is enabled, the frame will be automatically displayed in the control window.

#### Syntax

[VB]  
`objActiveGige.Grab`

[C/C++]  
`HRESULT Grab();`

#### Parameters

*None*

#### Return Values

S\_OK  
Success  
E\_ABORT  
Timeout occurred  
E\_ACCESSDENIED  
Camera not found  
E\_FAIL  
Failure

#### Example

This VB example grabs a frame and saves it as a tiff file

```
Private Sub Form_Load()  
ActiveGigel.Acquire=False  
End Sub  
  
Private Sub Button_Click()  
ActiveGigel.Grab  
ActiveGigel.SaveImage "image1.tif"  
End Sub
```

#### Remarks

The **Grab** method will wait for the current frame to be acquired before returning. When the acquisition of the current frame is complete, the [FrameAcquired](#) event will be raised.

Note that the use of the Grab method is incompatible with the [Acquire](#) property set to TRUE. Make sure to set Acquire to FALSE when using **Grab**.

### 3.2.102 IsFeatureAvailable

#### Description

Checks the availability of the specified camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.IsFeatureAvailable ( Name )
```

[C/C++]

```
HRESULT IsFeatureAvailable( bstr Name, BOOL pValue );
```

#### Data Types [VB]

*Name*: String

*Return value*: String

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*pValue* [out, retval]

Pointer to the boolean value which is TRUE if the feature is available and FALSE otherwise

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

This VB example uses the availability of the feature to enable or disable a slider:

```
if IsFeatureAvailable("ExposureTimeAbs") then
Slider1.Enable(TRUE)
else
Slider1.Enable(FALSE)
endif
```

#### Remarks

A feature is considered available when its access flag is RO or RW. See [GetFeatureAccess](#) for more details.

---

### 3.2.103 IsMasterApplication

#### Description

Checks if the application has an exclusive control over the selected camera.

#### Syntax

[VB]

```
Value=objActiveGige.IsMasterApplication
```

[C/C++]

```
HRESULT IsMasterApplication( BOOL pValue );
```

#### Data Types [VB]

*Return value:* Boolean

#### Parameters [C/C++]

*pValue* [out, retval]

Pointer to the boolean value which is TRUE if the application has an exclusive control and FALSE otherwise

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

This VB example checks the status of the camera control to enable or disable the exposure slider:

```
if ActiveGigE1.IsMasterApplicatoin then
Slider1.Enable(TRUE)
else
Slider1.Enable(FALSE)
endif
```

#### Remarks

This method is usually used in combination with the [Multicast](#) property. When the camera is set in the Multicast mode, only the master application (the one that was launched first and initiated the multicast) can control the camera settings. Other applications that typically run on other computers in the network can display the video and report camera parameters, but cannot modify them. The

**IsMasterApplication** method can be used to find out if an application works in the slave mode and disable some user interface elements.

In the unicast mode this method can be used to display an error message when a second application or process is trying to get an access to the same camera.

### 3.2.104 LoadImage

#### Description

Loads the image from the specified file into the internal frame buffer and displays it in *ActiveGige* window. The method supports BMP, TIFF and JPEG formats.

#### Syntax

[VB]  
`objActiveGige.LoadImage File`

[C/C++]  
`HRESULT LoadImage( bstr File );`

#### Data Types [VB]

*File*: String

#### Parameters [C/C++]

*File* [in]  
The string containing the file name and path

#### Return Values

S\_OK  
Success

E\_FAIL  
Failure.

E\_INVALIDARG  
Invalid file name.

#### Example

This VB example shows how to load an image file.

```
ActiveGige1.LoadImage "C:\\myframe.jpg"
```

#### Remarks

When the image has been loaded into the internal buffer, the [FrameLoaded](#) event will be fired.

The image format is defined by the file extension indicated in the *File* argument. Use "bmp" for a BMP file, "tif" for TIFF, and "jpg" for JPEG. If none of these extensions are found in the *File* string, an error will occur.

---

### 3.2.105 LoadSettings

#### Description

Loads previously stored camera settings from the data file.

#### Syntax

[VB]

```
objActiveGige.LoadSettings( Name )
```

[C/C++]

```
HRESULT LoadSettings( BSTR Name );
```

#### Data Types [VB]

*Profile*: String

*Return value*: None

#### Parameters [C/C++]

*Name* [in]

The name of the profile or path to the file in which the settings are stored.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example restores previously saved camera settings:

This VB example shows how to restore previously saved camera settings using two options - profile name and file path:

```
Private Sub LoadSettings_Profile()  
    ActiveGigel.LoadSettings ( "FluorescentHighRes" )  
End Sub  
  
Private Sub LoadSettings_File()  
    ActiveGigel.LoadSettings ( "C:/CamSettings/HighRes.dat" )  
End Sub
```

#### Remarks

If no colon or slash symbols are found in the Name argument, *ActiveGigE* will treat it as a profile name and will attempt to find the profile with the camera settings in the Profile subfolder of the ActiveGigE folder. Otherwise the exact path to the data file will be used.

This method verifies if the currently selected camera model coincides with the model for which the settings were saved. If the model is different, the function will return an error.

Also see [SaveSettings](#).

---

### 3.2.106 LoadSequence

#### Description

Loads a fragment of the specified AVI or TIFF file into the memory sequence.

#### Syntax

[VB]

```
objActiveGige.LoadSequence File [,Start, End ]
```

[C/C++]

```
HRESULT LoadSequence( bstr File, long Start, long End );
```

#### Data Types [VB]

*File*: String

*Start*: Long

*End*: Long

#### Parameters [C/C++]

*File* [in]

The string containing the file name and path. The extension of the file must be "avi" or "tif".

*Start* [in]

The zero-based index of the first frame of the video fragment to be loaded. If omitted, frame 0 will be used.

*End* [in]

The zero-based index of the last frame of the video fragment to be loaded. If omitted, the last frame will be used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid file name or file not found.

#### Example

This VB example loads a video sequence from a TIFF file into the memory and plays it.

```
ActiveGige1.LoadSequence "video1.tif"
```

```
ActiveGige1.OpenVideo "ram"
```

```
ActiveGige1.PlayVideo
```

#### Remarks

If the system memory doesn't have enough capacity to accommodate the specified video fragment, the video will be truncated to fit the maximum memory size available for the application.

### 3.2.107 OpenVideo

#### Description

Opens a specified video file (AVI or TIF) or previously recorded memory sequence for a playback. Used in combination with [PlayVideo](#), [StopVideo](#), [CloseVideo](#).

#### Syntax

[VB]  
`objActiveGige.OpenVideo Source [, Sound = False ]`

[C/C++]  
`HRESULT OpenVideo (bstr Source, bool Sound );`

#### Data Types [VB]

*Source* : String

*Sound* : Boolean (optional)

#### Parameters [C/C++]

*File* [in]

The string containing the path to the AVI or TIF file; or "RAM" for a memory sequence.

*Sound* [in]

Optional boolean parameter for AVI files. If TRUE, the playback will be performed with the sound.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid file name.

#### Example

This VB example opens and plays a memory sequence.

```
Private Sub Play_Click()  
ActiveGige1.OpenVideo "ram"  
ActiveGige1.PlayVideo  
End Sub
```

#### Remarks

For the *Sound* option to work, the AVI must contain the sound track.

If an AVI file is open with the *Sound* enabled, its maximum playback frame rate will be limited to x 16 of the recorded frame rate, and the synchronous playback will not be available.

---



Only one video file or memory sequence can be open at a time by one *ActiveGige* object. To play back several video sources, use several *ActiveGige* objects.

### 3.2.108 OverlayClear

#### Description

Clears graphics and text from the overlay.

#### Syntax

[VB]  
`objActiveGige.OverlayClear`

[C/C++]  
`HRESULT OverlayClear();`

#### Parameters

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

The following VB example moves a red rectangle over the live image by repeatedly erasing and drawing it:

```
Private Sub Form_Load()  
    x = 0  
    ActiveGigel.Acquire = True  
    ActiveGigel.OverlayColor = RGB(255, 0, 0)  
    ActiveGigel.Overlay = True  
End Sub  
  
Dim x As Integer  
Private Sub ActiveGigel_FrameAcquired()  
    ActiveGigel.OverlayClear  
    ActiveGigel.OverlayRectangle x, 10, x + 50, 80, 3  
    x = x + 2  
    If x = ActiveGigel.SizeX Then  
        x = 0  
    End If  
End Sub
```

#### Remarks

To create animation effects, use this method in combination with [OverlayColor](#), [OverlayPixel](#), [OverlayLine](#), [OverlayRectangle](#), [OverlayEllipse](#), [OverlayText](#).

---

### 3.2.109 OverlayEllipse

#### Description

Draws an empty or filled ellipse in the overlay.

#### Syntax

[VB]

```
objActiveGige.OverlayEllipse StartX, StartY, EndX, EndY [,Width = 1]
```

[C/C++]

```
HRESULT OverlayEllipse( short StartX, short StartY, short EndX, short EndY,  
short Width);
```

#### Data Types [VB]

*StartX, StartY, EndX, EndY*: Integer

*Width*: Integer (optional)

#### Parameters [C/C++]

*StartX* [in], *StartY* [in]

Pixel coordinates of the top left corner of the ellipse's bounding rectangle, relative to the image origin.

*EndX* [in], *EndY* [in]

Pixel coordinates of the bottom right corner of the ellipse's bounding rectangle, relative to the image origin.

*Width* [in]

Width of the outline of the ellipse in pixels. If zero, a filled ellipse is drawn.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example overlays a filled red ellipse on the live video:

```
ActiveGigel.Acquire = True  
ActiveGigel.OverlayEllipse 100,100,200,200,0  
ActiveGigel.OverlayColor=RGB(255,0,0)  
ActiveGigel.Overlay= True
```

#### Remarks

To draw a filled ellipse, use *Width=0*. Also see [OverlayColor](#), [OverlayClear](#).

### 3.2.110 OverlayLine

#### Description

Draws a line in the overlay.

#### Syntax

[VB]

```
objActiveGige.OverlayLine StartX, StartY, EndX, EndY [,Width = 1]
```

[C/C++]

```
HRESULT OverlayLine( short StartX, short StartY, short EndX, short EndY,  
short Width);
```

#### Data Types [VB]

*StartX, StartY, EndX, EndY*: Integer

*Width*: Integer (optional)

#### Parameters [C/C++]

*StartX* [in], *StartY* [in]

Pixel coordinates of the starting point of the line, relative to the image origin.

*EndX* [in], *EndY* [in]

Pixel coordinates of the end point of the line, relative to the image origin.

*Width* [in]

Width of the line in pixels.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example overlays a filled red line of width 3 on the live video:

```
ActiveGigel.Acquire = True  
ActiveGigel.OverlayLine 100,100,200,200,3  
ActiveGigel.OverlayColor=RGB(255,0,0)  
ActiveGigel.Overlay= True
```

#### Remarks

To draw multiple lines, call this method several times. Also see [OverlayColor](#), [OverlayClear](#).

---

### 3.2.111 OverlayPixel

#### Description

Draws a pixel in the overlay.

#### Syntax

[VB]

```
objActiveGige.OverlayPixel X, Y
```

[C/C++]

```
HRESULT OverlayPixel( short X, short Y );
```

#### Data Types [VB]

X, Y: Integer

#### Parameters [C/C++]

X [in], Y [in]

Coordinates of the pixel relative to the image origin.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example overlays four red pixels on the live video:

```
ActiveGigel.Acquire = True
ActiveGigel.OverlayPixel 100,100
ActiveGigel.OverlayPixel 100,200
ActiveGigel.OverlayPixel 200,100
ActiveGigel.OverlayPixel 200,200
ActiveGigel.OverlayColor=RGB(255,0,0)
ActiveGigel.Overlay= True
```

#### Remarks

To draw custom shapes, call this method multiple times. Also see [OverlayColor](#), [OverlayClear](#).

### 3.2.112 OverlayRectangle

#### Description

Draws an empty or filled rectangle in the overlay.

#### Syntax

[VB]

```
objActiveGige.OverlayRectangle StartX, StartY, EndX, EndY [,Width = 1]
```

[C/C++]

```
HRESULT OverlayRectangle( short StartX, short StartY, short EndX, short EndY, short Width);
```

#### Data Types [VB]

*StartX, StartY, EndX, EndY*: Integer

*Width*: Integer (optional)

#### Parameters [C/C++]

*StartX* [in], *StartY* [in]

Pixel coordinates of the top left corner of the rectangle, relative to the image origin.

*EndX* [in], *EndY* [in]

Pixel coordinates of the bottom right corner of the rectangle, relative to the image origin.

*Width* [in]

Width of the outline of the rectangle in pixels. If zero, a filled rectangle is drawn.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example overlays a filled red rectangle on the live video:

```
ActiveGigel.Acquire = True  
ActiveGigel.OverlayRectangle 100,100,200,200,0  
ActiveGigel.OverlayColor=RGB(255,0,0)  
ActiveGigel.Overlay= True
```

#### Remarks

To draw a filled rectangle, use *Width=0*. To draw multiple rectangles, call this method several times. Also see [OverlayColor](#), [OverlayClear](#).

---

### 3.2.113 OverlayText

#### Description

Draws a string of text in the overlay.

#### Syntax

[VB]

```
objActiveGige.OverlayText X, Y, Text
```

[C/C++]

```
HRESULT OverlayText( short X, short Y, bstr Text );
```

#### Data Types [VB]

*X*, *Y*: Integer

*Text*: String

#### Parameters [C/C++]

*X* [in], *Y* [in]

Coordinates of the text relative to the image origin.

*Text* [in]

The string containing the text to be drawn.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example overlays a string of text on the live video:

```
Dim Font As New StdFont
Font.Name = "Arial"
Font.Size = 18
Font.Bold = True
ActiveGigel.OverlayFont = Font
ActiveGigel.OverlayText 10, 100, "ActiveGige rules!"
ActiveGigel.OverlayColor = RGB(255, 0, 0)
ActiveGigel.Overlay = True
```

#### Remarks

To select the font for drawing text strings, use [OverlayFont](#). To draw multiple strings, call this method several times. Also see [OverlayColor](#), [OverlayClear](#).

### 3.2.114 PlayVideo

#### Description

Plays back the currently open video file (AVI or TIF) or memory sequence in the *ActiveGige* window.

#### Syntax

[VB]

```
objActiveGige.PlayVideo [ Start, End, Increment ]
```

[C/C++]

```
HRESULT PlayVideo( long Start, long End, long Increment );
```

#### Data Types [VB]

*Start*: Long

*End*: Long

*Increment*: Long

#### Parameters [C/C++]

*Start* [in]

The zero-based index of the frame of the video file or sequence at which the playback will start. If omitted, the whole file or sequence will be played forward. If -1, the whole file or sequence will be played backward.

*End* [in]

The zero-based index of the frame at which the playback will stop. If omitted, the playback will stop at the last frame. If *End* < *Start*, the specified fragment will be played backward.

*Increment* [in]

The increment of frames during the playback. If 2, every second frame will be played. If 3, every third frame will be played, and so on. If 0, 1 or omitted, the playback of every frame will occur.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example opens an AVI file and plays its fragment while updating the frame count.

```
Private Sub Play_Click()  
ActiveGige1.OpenVideo "C:\\video1.avi"  
ActiveGige1.PlayVideo 0, 50  
End Sub
```

```
Dim frameplayed As Integer
```

```
Private Sub ActiveGige1_FrameLoaded()  
frameplayed = frameplayed + 1  
LabelCount.Caption = frameplayed
```

---



End Sub

```
Private Sub ActiveGige1_PlayCompleted (ByVal Frames As Long)
ActiveGige1.CloseVideo
End Sub
```

### Remarks

The video will be played at the frame rate at which it was recorded unless the [SetVideoFPS](#) method is applied.

If the video file was recorded with compression, a corresponding codec must be installed on the system in order to play it back. Note that the playback frame rate of a compressed video file can become significantly reduced if the video is played backward or the *Increment* parameter used.

Per each frame of the video file or sequence played, the [FrameLoaded](#) event will be fired. When the playback is finished, the [PlayCompleted](#) event will be fired.

During the playback frames are being extracted from the video file or memory sequence and loaded into the internal *ActiveGige* buffer. The pixels of the currently loaded frame can be accessed via *ActiveGige* image access functions ([GetImageData](#), [GetImagePointer](#), [GetImageWindow](#) etc).

During the playback of a memory sequence all current conversion properties (such as [Bayer](#), [WindowLevel](#), [BkgCorrect](#), [Rotate](#)) will be applied to the frames of the sequence in real time.

Calling this method will automatically turn off the live video by setting the [Acquire](#) property to FALSE.

### 3.2.115 ReadBlock

#### Description

Reads the block of data from the internal camera memory starting from the specified bootstrap address.

#### Syntax

[VB]

```
objActiveGige.ReadBlock Offset, Buffer, nBytes
```

[C/C++]

```
HRESULT ReadBlock( long Offset, VARIANT Buffer, long nBytes );
```

#### Data Types [VB]

*Offset*: Long

*Buffer*: Variant (pointer)

*nBytes*: Long

#### Parameters [C/C++]

*Offset* [in]

The 32-bit offset into the camera register address space.

*Buffer* [in]

Variant containing pointer to a buffer that will receive the data read from the specified offset of the camera memory.

*nBytes* [in]

The number of bytes to read from the specified offset

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This C++ example reads a block of 128 bytes from a specified offset in the camera address space:

```
unsigned char buffer[128]  
VARIANT v;  
v.pvVal=buffer;  
ActiveGige.ReadBlock(0xA0000,v,128);
```

#### Remarks

Using this method provides a much faster access to the internal camera memory than repetitive calls to [ReadRegister](#).

---

If the specified offset does not exist or the camera cannot accomodate the size of the data block, this method will return an error.

### 3.2.116 ReadFile

#### Description

Transfers the indicated amount of bytes from the specified file in the device to the data array in memory.

#### Syntax

[VB]

```
Value=objActiveGige.ReadFile Name, Buffer, Length
```

[C/C++]

```
HRESULT ReadFile( bstr Name, long Offset, long Length , VARIANT* pData);
```

#### Data Types [VB]

*Name*: String

*Offset*: Long

*Length*: Long

*Return value*: Variant (Array of Bytes)

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the file in the device

*Offset* [in]

The transfer starting position from the beginning of the file in bytes

*Length* [in]

The number of bytes to transfer from the specified file

*pData* [out]

Pointer to SAFEARRAY of bytes containing data transferred from the file

#### Return Values

S\_OK

Success

E\_NOINTERFACE

File with the specified name does not exist in the device

E\_FAIL

Failure.

#### Example

This VB code transfers data from the "UserSet1" file in the device to the DataArray in memory.

```
length=ActiveGigel.GetFileSize("UserSet1")  
DataArray=ActiveGigel.ReadFile("UserSet1",0,length)
```

#### Remarks

---

---

The File Access functionality allows an application to read and write files hosted on the device. The data in those files may contain look-up tables, configuration sets, firmware, and other information.

Once you transferred the file data from the device to memory, you can save them in a file on a hard drive. Note that in order for this method to work, the [Access Mode](#) of the file should be "R" or "RW".

### 3.2.117 ReadRegister

#### Description

Reads a 32-bit integer value from the specified bootstrap register of the current camera.

#### Syntax

[VB]

```
Value=objActiveGige.ReadRegister( Reg )
```

[C/C++]

```
HRESULT ReadRegister( long Reg, long* pValue );
```

#### Data Types [VB]

*Reg*: Long

*Return value*: Long

#### Parameters [C/C++]

*Reg* [in]

The 32-bit offset of the register in the camera's address space

*pValue* [out,retval]

Pointer to the registers's 32-bit value

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Examples

This C++ example reads the value of the heartbeat timeout from the corresponding GigE Vision™ register :

```
value=ActiveGige.ReadRegister(0x00D8);
```

This VB example reads the value of the custom camera feature:

```
value=ActiveGige.ReadRegister(&HA080)
```

#### Remarks

The list of standard bootstrap registers can be found in "*GigE Vision Video Streaming and Device Control Over Ethernet Standard*" published by the Automated Imaging Association.

This method can be used to access custom camera attributes not available through GigE Vision™ feature interface. Refer to the camera documentation for the list of the manufacturer-specific registers.

---

### 3.2.118 SaveBkg

#### Description

Stores a dark or bright background image on the hard drive. The background image is produced as a result of temporal frame averaging.

#### Syntax

[VB]  
`objActiveGige.SaveBkg Type [, Frames = 8 ]`

[C/C++]  
`HRESULT SaveBkg( short Type, short Frames );`

#### Data Types [VB]

*Type*: Integer  
*Frames*: Integer

#### Parameters [C/C++]

*Type* [in]  
Enumerated integer indicating the type of background to be saved: 1 - Dark Field, 2 - Bright Field  
*Frames* [in]  
Number of consecutive frames to be averaged for background calculation.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

The following VB example uses two buttons to save dark and bright background images:

```
Private Sub SaveDark_Click  
ActiveGige1.SaveBkg (1)  
End Sub
```

```
Private Sub SaveBright_Click  
ActiveGige1.SaveBkg (2)  
End Sub
```

#### Remarks

In general, to perform the background correction you have to prepare two auxiliary images: dark field and bright field. Dark field is a background image captured with no light transmitted through the camera lens; it is a measure of the dark current in the CCD. The bright field is a background image captured with the maximum light transmitted and no objects in the field of view; it is a measure of the difference in pixel-to-pixel sensitivity as well as the uniformity of illumination. While capturing the bright

field image you should adjust the light intensity so that it stays just below the saturation level of the camera. That will provide the maximum dynamic range for the image acquisition. To control the saturation for monochrome cameras, use the **Saturated [Palette](#)**. For more information on the background correction see [BkgCorrect](#).

Background data are stored as raw image files in the Bkgnd subfolder of ActiveGige program directory. See [BkgName](#) for more details.

When the background saving is complete, the [FrameRecorded](#) event is fired.

Note that this function will only work if the [Acquire](#) property is set to True or the [Grab](#) method is repeatedly called.

---



### 3.2.119 SaveImage

#### Description

Saves the current frame buffer in the specified image file. The method supports RAW, BMP, TIFF, DPX and JPEG formats with a selectable compression.

#### Syntax

[VB]

```
objActiveGige.SaveImage File [, Compression]
```

[C/C++]

```
HRESULT SaveImage( bstr File, long Compression );
```

#### Data Types [VB]

*File*: String

*Compression*: Integer (optional)

#### Parameters [C/C++]

*File* [in]

The string containing the file name and path

*Compression* [in]

The file compression ratio

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid file name.

#### Example

This VB example saves the current frame in a JPEG file with the specified compression quality.

```
ActiveGigel.SaveImage "C:\\myframe.jpg", 75
```

#### Remarks

The image file format in which the frame will be saved is defined by the file extension indicated in the *File* argument. Use "raw" for a RAW file, "bmp" for a BMP file, "tif" for TIFF, "jpg" for JPEG and "dpx" for DPX. If none of these extensions are found in the *File* string, an error will occur.

When the image saving is complete, the [FrameRecorded](#) event is fired.

The way the *Compression* argument is treated depends on the file format.

**BMP files:**

*Compression* = 0 - no compression

*Compression* = 1 - RLE compression (not implemented in this version)

**TIFF files:**

*Compression* = 0 - no compression

*Compression* = 1 - LZW compression

*Compression* = 2 - PackBits compression

**JPEG files:**

*Compression* is an integer value in the range 0-100 specifying the quality of the image. Lower values correspond to a lower quality with a higher compression, while higher values correspond to a higher quality with a lower compression.

**DPX files:**

*Compression* = 0 - RGB order

*Compression* = 1 - BRG order

**RAW files:**

*Compression* has no effect.

If *Compression* parameter is omitted, BMP and TIFF files will be recorded with no compression, while JPEG files will be recorded with quality of 75. DPX files are recorded in the 10-bit packed RGB or monochrome format.

When the RAW format is selected, the resulting file will contain undecoded frame data as they arrived from the camera. See [GetRawData](#) for more details.

---

### 3.2.120 SaveSettings

#### Description

Stores the current camera settings in a data file.

#### Syntax

[VB]  
`objActiveGige.SaveSettings( Name )`

[C/C++]  
`HRESULT SaveSettings( BSTR Name );`

#### Data Types [VB]

*Profile*: String

*Return value*: None

#### Parameters [C/C++]

*Name* [in]

The name of the profile or path to the file in which the settings will be saved.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example shows how to save the current camera settings using two options - profile name and file path:

```
Private Sub SaveSettings_Profile()  
    ActiveGigel.SaveSettings ( "FluorescentHighRes" )  
End Sub  
  
Private Sub SaveSettings_File()  
    ActiveGigel.SaveSettings ( "C:/CamSettings/HighRes.dat" )  
End Sub
```

#### Remarks

If no colon or slash symbols are found in the Name argument, *ActiveGigE* will treat it as a profile name and will attempt to save the camera settings in the Profile subfolder of the ActiveGigE folder. Otherwise the settings will be saved using the exact path to the file.

Note that due to User Account Control (UAC) your application may not have a permission to write data into the Program File folder where the ActiveGigE/Profile subfolder is located. In this case you should

create a folder that has both read and write permissions and store the camera settings in it by using the full path and file name as an argument.

Also see [LoadSettings](#).

---

### 3.2.121 SaveSequence

#### Description

Saves the current memory sequence or its fragment in the specified AVI or TIFF file.

#### Syntax

[VB]

```
objActiveGige.SaveSequence File [, Start, End, FPS ])
```

[C/C++]

```
HRESULT SaveSequence( bstr File, long Start, long End, float FPS );
```

#### Data Types [VB]

*File*: String

*Start*: Long

*End*: Long

*FPS*: Single

#### Parameters [C/C++]

*File* [in]

The string containing the file name and path. The extension of the file must be "avi" or "tif".

*Start* [in]

The zero-based index of the first frame of the fragment of the sequence to be saved. If omitted, frame 0 will be used.

*End* [in]

The zero-based index of the fragment of the sequence to be saved. If omitted, the last frame will be used.

*FPS* [in]

The frame rate to be assigned to the saved video. If zero or omitted, the recorded frame rate will be used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid file name.

#### Example

This VB example records 1000 frames into the memory sequence and uses the [CaptureCompleted](#) event to save it in an AVI file.

```
Private Sub StartButton_Click()  
ActiveGige1.StartSequenceCapture 1000  
End Sub
```

```
Private Sub ActiveGigE1_CaptureCompleted(ByVal Frames As Long)
SaveSequence "C:\\video.avi"
End Sub
```

**Remarks**

If the sequence is saved in the AVI format, the currently selected [compression codec](#) will be used.

If the external device (typically, a hard drive or memory card) doesn't have enough space to store the whole sequence, it will be truncated to accommodate the remaining space.

---

### 3.2.122 SendActionCommand

#### Description

Broadcasts the action command to all available network interfaces

#### Syntax

[VB]

```
objActiveGige.SendActionCommand DeviceKey, GroupKey[, GroupMask,  
WaitForAck, SceduledTime]
```

[C/C++]

```
HRESULT SendActionCommand (long DeviceKey, long GroupKey, long GroupMask,  
short WaitForAck, double ScheduledTime)
```

#### Data Types [VB]

*DeviceKey, GroupKey, GroupMask*: Long

*WaitForAck*: Integer

*ScheduledTime*: Double

#### Parameters [C/C++]

*DeviceKey* [in]

The 32-bit value of the device key in the action command. In order for the action command to be asserted by a device, this parameter should match the device's device key. The device key on the device are typically configured using the ActionDeviceKey feature of the GenICam interface or via the [SetDeviceKey](#) method of *ActiveGigE*.

*GroupKey* [in]

The 32-bit value of the group key in the action command. In order for the action command to be asserted by a device, this parameter should match the device's group key for at least one action supported by the device. The group keys on the device are typically configured using the ActionGroupKey feature of the GenICam interface or via the [SetActionConditions](#) method of *ActiveGigE*. Different actions on the device can be assigned different group keys.

*GroupMask* [in]

The 32-bit value of the group mask in the action command. In order for the action command to be asserted by a device, the result of the AND-wise application of this mask against the device's group mask must be non-zero. The group masks on the device are typically configured using the ActionGroupKey feature of the GenICam interface or via the [SetActionConditions](#) method of *ActiveGigE*. Different actions on the device can be assigned different group masks. The default value of this parameter is 0xFFFFFFFF.

*WaitForAck* [in]

The number of expected acknowledge responses from devices. Each devices which asserts the action command sends an acknowledge response back to the application. If no acknowledgements are received within the *Action timeout* period or the amount received is less than the value of this parameter, ActiveGigE will resend the Action command. If no acknowledgments are received after all Action retry attempts have been exhausted, SendActionCommands will return an error. If the value of WaitForAck is set to zero, no acknowledge responses will be requested from devices and the method will return immediately.

*ScheduledTime* [in]

The double value indicating the scheduled time of the action in seconds. The time must be in the same time domain as the device's timestamps. Upon receiving a scheduled action command, the device will put it in its action queue and perform the associated action as close to the scheduled

timestamp as possible. If the device does not support the scheduled action functionality or if this parameter is zero (default value), the device will execute the action command upon reception.

### Return Values

S\_OK  
Success  
E\_FAIL

The application has not received the requested amount of acknowledge responses.

### Example

The following VB example configures two camera for the same action conditions and broadcasts an action command to trigger a frame acquisition on both cameras. Both cameras are assumed to have the software trigger function linked to the "Action1" task.

```
Private Sub Form_Load()  
ActiveGigE1.SetDeviceKey 24  
ActiveGigE1.SetActionConditions 1,11,7  
ActiveGigE1.SetFeature "TriggerSource", "Action1"  
ActiveGigE1.Trigger=True  
ActiveGigE1.Acquire=True  
  
ActiveGigE2.SetDeviceKey 24  
ActiveGigE2.SetActionConditions 1,11,7  
ActiveGigE2.SetFeature "TriggerSource", "Action1"  
ActiveGigE2.Trigger=True  
ActiveGigE2.Acquire=True  
End Sub  
  
Private Sub OnButtonActionCommand()  
ActiveGigE1.SendActionCommand 24,11,7  
End Sub
```

### Remarks

The following four conditions must be met for an action command to be asserted by the device:

- 1) *ActiveGigE* or a third-party application/receiver must have a master control over the device.
- 2) The *DeviceKey* parameter of the **SendActionCommand** and the [Device Key](#) assigned to the device must be equal.
- 3) The *GroupKey* parameter of the **SendActionCommand** and the [Group Key](#) assigned to the corresponding device action must be equal.
- 4) The logical AND-wise operation of the *GroupMask* parameter of the **SendActionCommand** and the [Group Mask](#) assigned to the corresponding device action must be non-zero. In other words, they must have at least one common bit at the same position.

The *Timeout Period* and the number of *Retry Attempts* for Action commands can be configured in the Advanced panel of the [GEV Configurator](#).

You can get the amount of acknowledge responses received from devices and their associated IP addresses by calling [GetActionAcknowledgeInfo](#).

Note that scheduled action commands can only be supported by devices compliant with the GigE Vision 2.x specifications. A typical application of this functionality is to simultaneously trigger multiple devices which are using the IEEE 1588 Precision Time Protocol (PTP) for the clock synchronization. The precision achievable in such a setup can be as high as 1 microsecond. For more information refer to "*GigE Vision Video Streaming and Device Control Over Ethernet Standard, version 2.0*" published by



---

the Automated Imaging Association.

---

### 3.2.123 SetActionConditions

#### Description

Configures the device for the asserting conditions of the specified action task.

#### Syntax

[VB]

```
objActiveGige.SetActionConditions ActionIndex, GroupKey, GroupMask
```

[C/C++]

```
HRESULT SetActionConditions (short ActionIndex, long GroupKey, long GroupMask)
```

#### Data Types [VB]

*ActionIndex*: Short

*GroupKey*: Long

*GroupMask*: Long

#### Parameters [C/C++]

*ActionIndex* [in]

The 1-based integer index of the device's action task to which further parameters will be applied. If the device supports only one action task, this parameter must be set to 1.

*GroupKey* [in]

The 32-bit value of the group key for the specified *ActionIndex*. The device will only assert an [Action Command](#) sent by an application if the device's group key matches the group key parameter transmitted in the action command. Different action tasks on the device can be assigned different group keys.

*GroupMask* [in]

The 32-bit value of the group mask for the specified *ActionIndex*. The device will only assert an [Action Command](#) sent by an application if the AND-wise application of the device's group mask against the group mask in the action command results in a non-zero value. Different action tasks on the device can be assigned different group masks.

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

The following VB example configures the camera for two different action conditions. The camera is assumed to support two action tasks, with the group key 11 being assigned to "Action1" and group key 12 to "Action2". An action command broadcast upon a button click will cause the camera to perform the "Action1" task.

```
Private Sub Form_Load()  
ActiveGigEl.SetDeviceKey 24
```

---

```
ActiveGigEl.SetActionConditions 1,11,7
ActiveGigEl.SetActionConditions 2,12,7
ActiveGigEl.SetFeature "TriggerSource","Action1"
ActiveGigEl.Trigger=True
ActiveGigEl.Acquire=True
End Sub

Private Sub OnButtonActionCommand()
ActiveGigEl.SendActionCommand 24,11,7
End Sub
```

## Remarks

The following four conditions must be met for an action command to be asserted by the device:

- 1) *ActiveGigE* or a third-party application/receiver must have a master control over the device.
- 2) The Device Key parameter of the action command sent by an application and the [DeviceKey](#) assigned to the device must be equal.
- 3) The Group Key parameter of the of the action command sent by an application and the *GroupKey* assigned to the corresponding device action must be equal.
- 4) The logical AND-wise operation of the Group Mask parameter of the action command sent by an application and the *GroupMask* assigned to the corresponding device action must be non-zero. In other words, they must have at least one common bit at the same position.

For more information on the Action functionality refer to [SendActionCommand](#).

### 3.2.124 SetAudioLevel

#### Description

Sets the recording level of the currently selected audio device.

#### Syntax

[VB]  
`objActiveGige.SetAudioLevel Value`

[C/C++]  
`HRESULT SetAudioLevel(short Value);`

#### Data Types [VB]

*Value*: Integer

#### Parameters [C/C++]

*Value* [in]  
The recording level to be set, in percent.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example initiates an AVI recording with a sound track and uses a scroll bar to adjust the audio recording level.

```
Private Sub Form_Load()  
    ActiveGige1.SetAudioSource 0  
    HScroll1.Min=0  
    HScroll1.Max=100  
    HScroll1.Value=ActiveGige1.GetAudioLevel  
End Sub  
  
Private Sub Capture_Click  
    ActiveGige1.StartCapture "c:\\capture_with_sound.avi"  
End Sub  
  
Private Sub HScroll1_Scroll()  
    ActiveGige1.SetAudioLevel HScroll1.Value  
End Sub
```

#### Remarks

In order for this method to work, the audio recording device must be selected with [SetAudioSource](#).

---

### 3.2.125 SetAudioSource

#### Description

Sets the index of the audio recording device to be used during the AVI capture.

#### Syntax

```
[VB]
objActiveGige.SetAudioSource Index

[C/C++]
HRESULT SetAudioSource(short Index);
```

#### Data Types [VB]

*Source*: Integer

#### Parameters [C/C++]

*Index* [in]  
Zero-based index of the selected audio source in the system list of audio recording devices. If -1 (default), no audio track will be recorded.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure to set the audio source

#### Example

This VB example initializes a combo box with the descriptions of available audio recording devices and uses it to select a specific device:

```
AudioLst = ActiveGige1.GetAudioList
For i = 0 To UBound(AudioLst)
    Comb1.AddItem (AudioLst(i))
Next
Comb1.ListIndex = 0
ActiveGige1.SetAudioSource 0

Private Sub Comb1_Click()
    ActiveGige1.SetAudioSource Comb1.ListIndex
End Sub

Private Sub Capture_Click
    ActiveGige1.StartCapture "c:\\capture_with_sound.avi"
End Sub
```

**Remarks**

This method allows you to add the audio track to your AVI file and select a specific audio recording device. If only one recording device (such as a microphone) is present in the system, use 0 as the value of *Index*. To disable the audio recording, call **SetAudioSource** with the value of *Index* -1. Note that by default the audio recording is disabled in *ActiveGige*.

To initiate the AVI recording, use [StartCapture](#).

---

### 3.2.126 SetCodec

#### Description

Sets the name of the codec (video compressor) to be used during the AVI capture.

#### Syntax

[VB]  
`objActiveGige.SetCodec Name`

[C/C++]  
`HRESULT SetCodec(bstr Name);`

#### Data Types [VB]

*Name*: String

#### Parameters [C/C++]

*Name* [in]  
String specifying the name of the feature

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure to set the codec

#### Example

The following VB example demonstrates the use of the DivX codec:

```
Private Sub Form_Load()  
ActiveGige1.SetCodec "DivX 5.0.2 Codec"  
ActiveGige1.Acquire=True  
End Sub  
  
Private Sub StartButton_Click()  
ActiveGige1.StartCapture "c:\mycapture.avi"  
End Sub  
  
Private Sub StopButton_Click()  
ActiveGige1.StopCapture  
End Sub
```

The following VB example demonstrates how to set up a codec by its index in the system:

```
CodecList=ActiveGige1.GetCodecList  
CodecName=CodecList(9)  
ActiveGige1.SetCodec CodecName
```

**Remarks**

The name of the codec must be precise in order for **SetCodec** to work. An extra space in the name of the codec may cause this function to fail. To avoid errors, It is recommended to use an index of the codec rather than its name, as shown in the second example above.

*ActiveGige* video recording engine includes a proprietary "Raw Uncompressed" codec. When the camera streams video in the Bayer format and the "Raw Uncompressed" codec is selected, *ActiveGige* records the video in its original raw format while displaying it in color. This reduces the disk space requirements by three times without any degradation in the image quality. In addition, when high-depth pixel formats are used (such as Bayer10, Bayer12, Bayer16, Mono10, Mono 12, Mono16), the recorded video will be packed into the 12-bit per pixel format as opposed to 16 bit, thus saving extra 25% of the disk space. To decode and play back AVI files recorded with the "Raw Uncompressed" codec, use video playback methods of the DVR version of *ActiveGige*.



### 3.2.127 SetCodecProperties

#### Description

Sets the generic parameters of the currently selected compression codec.

#### Syntax

[VB]

```
objActiveGige.SetCodecProperties Quality [, DataRate, KeyFrameRate,  
PFramesPerKey, WindowSize ])
```

[C/C++]

```
HRESULT SetCodecProperties(long Quality = -1 [, long DataRate = -1, long  
KeyFrameRate = -1,  
long PFramesPerKey = -1, long WindowSize = -1 ]);
```

#### Data Types [VB]

*Quality*: Integer

*DataRate*: Integer

*KeyFrameRate*: Integer

*PFramesPerKey*: Integer

*WindowSize*: Integer

#### Parameters [C/C++]

*Quality* [in]

Numerical value of the quality parameter used by the codec, if supported.

*DataRate* [in]

Data rate in kb/sec used by the codec, if supported.

*KeyFrameRate* [in]

Amount of frames after which a key frame is recorded, if supported.

*PFramesPerKey* [in]

Rate of predicted (P) frames per key frame, if supported.

*WindowSize* [in]

Amount of frames over which the compressor maintains the average data rate.

#### Return Values

S\_OK

Success

E\_FAIL

Failure to set the codec's properties

#### Example

The following VB example sets the quality and datarate settings of the MJPEG codec:

```
ActiveGige1.SetCodec "MJPEG Compressor"  
ActiveGige1.SetCodecProperties 50, 1000
```

**Remarks**

This method allows you to set only the basic compression properties which may not be supported by certain codecs. To adjust the internal parameters of a codec, use [ShowCodecDialog](#).

If some parameters of this methods are omitted or set to -1, corresponding properties of the compression codec will not be changed.

---

### 3.2.128 SetColorMatrix

#### Description

Sets the matrix coefficients for the color correction operation. Used in combination with the [ColorCorrect](#) property.

#### Syntax

[VB]

```
objActiveGige.SetColorMatrix rr, rg, rb, gr, gg, gb, br, bg, bb
```

[C/C++]

```
HRESULT SetColorMatrix (float rr, float rg, float rb,  
                        float gr, float gg, float gb,  
                        float br, float bg, float bb);
```

#### Data Types [VB]

rr, rg, rb, gr, gg, gb, br, bg, bb: Single

#### Parameters [C/C++]

rr[in], rg[in], rb[in], gr[in], gg[in], gb[in], br[in], bg[in], bb[in]  
Coefficients of the color correction matrix. Must be in the range from -2.0 to +2.0

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example defines the color correction matrix and turns on the color correction:

```
ActiveGigel.SetColorMatrix 0.85, 0.1, 0.05, 0.05, 0.98, -0.03, 0.13, -0.15,  
1.02  
ActiveGigel.ColorCorrect=True
```

#### Remarks

The relations between the original color components of each pixel (R,G,B) and resulting components (R',G',B') are defined by the following formulas:

$$\begin{aligned} R' &= R*rr + G*rg + B*rb \\ G' &= R*gr + G*gg + B*gb \\ B' &= R*br + G*bg + B*bb \end{aligned}$$

Color correction is used to eliminate the overlap in the color channels caused by the fact that the light

intended to be detected only by pixels of a certain color is partially seen by pixels of other colors. For example, the red light is partially seen by green and blue elements of the CCD. A properly adjusted color correction matrix can eliminate this overlap.

In order for the white balance to work properly, the sum of each row in the color correct matrix (i.e.  $rr+rg+rb$ ) should be equal to 1.

---

### 3.2.129 SetControlKey

#### Description

Sets the numerical access key for the Control with Switchover access to the camera. Used in combination with the [Privilege](#) property set to 4.

#### Syntax

[VB]  
`objActiveGige.SetControlKey Value`

[C/C++]  
`HRESULT SetControlKey(long Value);`

#### Data Types [VB]

*Value*: Long

#### Parameters [C/C++]

*Value* [in]  
The 16-bit integer access key.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example shows a simple application that takes the control over the camera by submitting a predefined access key.

```
Private Sub Form_Load()  
ActiveGige1.Multicast=True  
ActiveGige1.SetControlKey "1234"  
ActiveGige1.Privilege=2  
ActiveGige1.Acquire=True  
End Sub
```

#### Remarks

When a primary application issues the **SetControlKey** command followed by setting the [Privilege](#) property to 4, the camera will be assigned the Control with Switchover access with a corresponding key code, provided the camera supports this type of access. When a secondary applications issues the SetControlKey command followed by setting the [Privilege](#) property to 4, it will become a primary application as long as the control key code is correct and the camera supports the Control with Switchover access.

When the primary control over the camera is switched to another application, the application that

previously controlled the camera will be sent a [message](#) with Event ID #7 that can be used to perform a set of actions associated with the lost of primary control. For more information refer to "*GigE Vision Video Streaming and Device Control Over Ethernet Standard*" published by the Automated Imaging Association.

---

### 3.2.130 SetDestinationIP

#### Description

Sets the destination IP address and port for the specified stream channel.

#### Syntax

[VB]  
`objActiveGige.SetDestinationIP Channel, IP, Port`

[C/C++]  
`HRESULT SetDestinationIP(long Channel, bstr IP, long Port );`

#### Data Types [VB]

*Channel*: Integer

*IP*: String

*Port*: Integer

#### Parameters [C/C++]

*Channel* [in]

Index of the streaming channel for which the destination address will be set

*IP* [in]

String specifying the destination IP address in the in the IPv4 format (four decimals in the range of 0-255 separated by dots). If empty, the video will be streamed to the IP address of the host application.

*Port* [in]

Destination port number. If 0, the value of 49152 will be used.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example sets different multicast end points for cameras #0 and #1:

```
ActiveGige1.SetCamera(0)
ActiveGige1.SetDestinationIP 0, "239.0.1.1", 0
ActiveGige1.Multicast=True
ActiveGige1.Acquire=True
ActiveGige2.SetCamera(1)
ActiveGige2.SetDestinationIP 0, "239.0.1.2", 0
ActiveGige2.Multicast=True
ActiveGige2.Acquire=True
```

This VB example sets the destination address for stream channel #1, links it with video source #1 and

activates streaming:

```
ActiveGigE1.SetDestinationIP 1, "192.168.12.12", 41000
ActiveGigE1.SetFeatureInt "SourceSelector", 1
ActiveGigE1.SetFeature "GevStreamChannelSelector", 1
ActiveGigE1.SetFeature "AcquisitionStart", True
```

### Remarks

Use this method to set the IP address and port for the multicast streaming if you need it to be different from the default address assigned by the [IP configuration utility](#). The valid range of multicast addresses is 239.0.0.0 - 239.255.255.255. See [Multicast](#) for more details.

If your camera supports multiple stream channels, calling **SetDestinationIP** is mandatory prior to starting the video acquisition on a channel other than channel #0. If the IP parameter is an empty string, the video on the specified channel will be streamed to the IP address of your main application (the one that called this method) allowing you to receive video streams from several channels at the same destination. Using a specific IP as a parameter will allow you to stream video on a specified channel to a network destination different from the one of your ActiveGigE-based main application. It is typically used for re-routing the second-channel video to a different computer or/and client application. Note that you should call [SetStreamChannel](#) with a specific stream channel index in order for ActiveGigE to receive video data at a corresponding destination point.



### 3.2.131 SetDeviceKey

#### Description

Configures the action device key register of the device (camera).

#### Syntax

[VB]  
`objActiveGige.SetDeviceKey DeviceKey`

[C/C++]  
`HRESULT SetDeviceKey (short DeviceKey)`

#### Data Types [VB]

*DeviceKey*: Long

#### Parameters [C/C++]

*DeviceKey* [in]

The 32-bit value of the device key for the current device. The device will only assert an [Action Command](#) sent by an application if the device's *DeviceKey* matches the device key parameter transmitted in the action command.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

The following VB example configures the camera for a specific the device key and action conditions, and then fires a corresponding action command.

```
Private Sub Form_Load()  
ActiveGigEl.SetDeviceKey 24  
ActiveGigEl.SetActionConditions 1,0,7  
End Sub  
  
Private Sub OnButtonActionCommand()  
ActiveGigEl.SendActionCommand 24,0,7  
End Sub
```

#### Remarks

The following four conditions must be met for an action command to be asserted by the device:

- 1) *ActiveGigE* or a third-party application/receiver must have a master control over the device.
- 2) The Device Key parameter of the action command sent by an application and the *DeviceKey*

assigned to the device must be equal.

3) The Group Key parameter of the of the action command sent by an application and the *GroupKey* assigned to the corresponding device action must be equal.

4) The logical AND-wise operation of the Group Mask parameter of the action command sent by an application and the *GroupMask* assigned to the corresponding device action must be non-zero. In other words, they must have at least one common bit at the same position.

For more information on the Action functionality refer to [SetActionConditions](#) and [SendActionCommand](#).

---

### 3.2.132 SetImageWindow

#### Description

Copies pixel values from the two-dimensional array into the selected window of the current frame.

#### Syntax

[VB]

```
objActiveGige.SetImageWindow X, Y, A
```

[C/C++]

```
HRESULT SetImageWindow( short X, short Y, VARIANT Data );
```

#### Data Types [VB]

X,Y: Integer

A: Variant (SAFEARRAY)

#### Parameters [C/C++]

X[in], Y[in]

The x- and y- frame coordinates at which the top left corner of the window will be copied.

Data[in]

SAFEARRAY variant containing the pixel values to be copied.

#### Return Values

S\_OK

Success

E\_FAIL

Failure

E\_INVALIDARG

Input array has wrong data type

#### Example

This VB example uses the [FrameAcquired](#) event to increase the brightness in the central area of the live image:

```
Private Sub Form_Load()  
ActiveGigel.Display = False  
ActiveGigel.Acquire = True  
End Sub  
  
Private Sub ActiveGigel_FrameAcquired()  
xc = ActiveGigel.SizeX / 2  
yc = ActiveGigel.SizeY / 2  
w = ActiveGigel.GetImageWindow(xc - 70, yc - 50, 140, 100)  
For y = 0 To UBound(w, 2)  
For x = 0 To UBound(w, 1)  
pix = w(x, y) + 50  
If pix > 255 Then
```

```

pix = 255
End If
w(x, y) = pix
Next
Next
ActiveGigE1.SetImageWindow xc - 70, yc - 50, w
ActiveGigE1.Draw
End Sub

```

### Remarks

The array submitted to **SetImageWindow** must have the type and dimensions corresponding of those of the frame buffer, as specified in the following table:

Camera Pixel Format	Output Format	Data type	Horizontal Dimension
Mono8	8-bit monochrome	Byte	Width
Mono10, Mono12, Mono16	16-bit gray monochrome	Integer (word)	Width
YUV411, YUV422, YUV444, RGB8, BGR8	24-bit RGB	Byte	Width*3
RGB10, RGB12, RGB16, BGR10, BGR12	48-bit RGB	Integer (word)	Width*3

where *Width* is the intended horizontal size of the window in pixels. If the dimensions of the window are too large to accommodate the frame size, they will be clipped to the frame boundaries.

For real-time image processing **SetImageWindow** should be used in conjunction with the [Draw](#) method.

---

### 3.2.133 SetFeature

#### Description

Sets the numerical value of the specified camera feature.

#### Syntax

[VB]

```
objActiveGige.SetFeature Name, Value
```

[C/C++]

```
HRESULT SetFeature(bstr Name, float Value );
```

#### Data Types [VB]

*Name*: String

*Value*: Single

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*Value* [in]

Numerical value of the feature to be set

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_INVALIDARG

Feature is not of numerical type

E\_OUTOFMEMORY

Value is out of range

E\_FAIL

Failure to set the feature

#### Example

The following VB example demonstrates the use of a scroll control for real-time adjustment of the "GainRaw" feature.

```
Private Sub Form_Load()  
ActiveGige1.Acquire=True  
HScroll11.Value = ActiveGige1.GetFeature("GainRaw")  
HScroll11.Min = ActiveGige1.GetFeatureMin("GainRaw")  
HScroll11.Max = ActiveGige1.GetFeatureMax("GainRaw")  
End Sub  
  
Private Sub HScroll11_Scroll()
```

```
ActiveGigE1.SetFeature("GainRaw", HScroll11.Value)  
End Sub
```

**Remarks**

Depending on the [Type](#) of the feature the *Value* argument has the following meaning:

Feature Type	Value
Integer	Integer value converted to floating point
Float	Floating point value
Boolean	0. if False, 1. if True
Enumerated	Ordinal number in the enumeration list
String	N/A
Command	Use non-zero value to execute

If the currently selected camera does not support the specified feature or if the feature is read-only, the method will generate an error.

To set the string value of a feature, use [SetFeatureString](#).

---

### 3.2.134 SetFeature64

#### Description

Sets the numerical value of the specified 64-bit camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.SetFeature64 Name, Value
```

[C/C++]

```
HRESULT SetFeature64(bstr Name, double Value );
```

#### Data Types [VB]

*Name*: String

*Value*: Double

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*Value* [in]

Numerical value of the feature to be set

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failed to read the feature

#### Example

The following VB example demonstrates how to set the value of a 64-bit feature.

```
Dim dx As Double  
dx=3.141592653589793238  
ActiveGige1.SetFeature64 "LargeFeature",dx
```

#### Remarks

It is recommended to use this method only for those integer and floating point features whose length is 64-bit.

If the currently selected camera does not support the specified feature, the method will generate an error.



### 3.2.135 SetFeatureArray

#### Description

Sets an array of values associated with the specified camera feature.

#### Syntax

[VB]

```
Value=objActiveGige.SetFeatureArray( Name, ElementSize, Buffer )
```

[C/C++]

```
HRESULT SetFeatureArray(bstr Name, short ElementSize, VARIANT* Data);
```

#### Data Types [VB]

*Name*: String

*ElementSize*: Integer

*Buffer*: Variant (SAFEARRAY)

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*ElementSize* [in]

A short value specifying the size of each element of the array, in bytes

*Data* [out, retval]

SAFEARRAY variant containing the values to be copied to the feature's buffer

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_FAIL

Failed to read the feature

#### Example

The following VB example retrieves the camera LUT array and displays the value of the 16th element:

```
LUT=ActiveGigel.GetFeatureArray( "LUTValueAll", 4)  
Label.Caption=LUT(16)
```

#### Remarks

This method retrieves the content of buffers associated with features of the IRegister type. For more information refer to GenICam standard specifications.

Note that the size of an element in the binary buffer returned by an IRegister feature cannot be

determined automatically. Therefore, the *ElementSize* parameter is not optional and must specify the size of each element of the array in bytes.

### 3.2.136 SetFeatureString

#### Description

Sets the string value of the specified camera feature.

#### Syntax

[VB]

```
objActiveGige.SetFeatureString Name, Value
```

[C/C++]

```
HRESULT SetFeatureString(bstr Name, bstr Value );
```

#### Data Types [VB]

*Name*: String

*Value*: String

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the feature

*Value* [in]

String value of the feature to be set

#### Return Values

S\_OK

Success

E\_NOINTERFACE

Feature does not exist or not available

E\_OUTOFMEMORY

Value is not valid for the feature

E\_FAIL

Failed to set the feature

#### Example

This VB example uses a combo box to switch between different auto exposure modes:

```
Private Sub Form_Load()  
Lst = ActiveGigel.GetEnumList("ExposureAuto")  
For i = 0 To UBound(Lst)  
Combol.AddItem (Lst(i))  
Next  
Combol.ListIndex = ActiveGigel.GetFeature("ExposureAuto")  
ActiveGigel.Acquire = True  
End Sub  
  
Private Sub Combol_Click()  
ActiveGigel.SetFeatureString Combol.Text  
End Sub
```

**Remarks**

Depending on the [Type](#) of the feature the *Value* argument has the following meaning:

<b>Feature Type</b>	<b>Value</b>
Integer	Integer value in form of string
Float	Floating point value in form of string
Boolean	"0" or "False", "1" or "True"
Enumerated	String value from the enumerated set
String	String value
Command	"1" or "Execute"

If the currently selected camera does not support the specified feature or if the value is not valid for the feature, the method will generate an error.

---

### 3.2.137 SetGains

#### Description

Sets the levels for the software gain control. If the [LUTMode](#) property is enabled, this operation multiplies the value of each pixel by a given floating point factor thus changing the contrast of the video or its color components.

#### Syntax

[VB]  
`objActiveGige.SetGains fR [, fG, fB, fGb]`

[C/C++]  
`HRESULT SetGains (float fR [, float fG, float fB, float fGb])`

#### Data Types [VB]

*fR, fG, fB, fGb*: Single

#### Parameters [C/C++]

*fR* [in]  
Gain factor for the red channel. If the rest of arguments are -1 or omitted, this factor will be applied to all color channels in the video.

*fG* [in]  
Gain factor for the green channel. If four arguments are used, this factor will be applied to Gr pixels of the raw Bayer image.

*fR* [in]  
Gain factor for the blue channel.

*fGb* [in]  
Gain factor for the Gb pixels of the raw Bayer image. If -1 or omitted, fGb=fG.

#### Return Values

S\_OK  
Success

E\_FAIL  
Failure.

#### Example

The following VB example uses three sliders to increase the gain for R, G and B channels up to the factor of 10:

```
Private Sub Form_Load()  
ActiveGigel.Acquire=True  
HScroll1.Min = 1  
HScroll1.Max = 10  
HScroll2.Min = 1  
HScroll2.Max = 10  
HScroll3.Min = 1  
HScroll3.Max = 10  
ActiveGigel.LUTMode=True
```

```
End Sub
```

```
Private Sub HScroll1_Scroll()  
L1 = HScroll1.Value  
L2 = HScroll2.Value  
L3 = HScroll3.Value  
ActiveGigE1.SetGains L1, L2, L3  
End Sub
```

```
Private Sub HScroll2_Scroll()  
L1 = HScroll1.Value  
L2 = HScroll2.Value  
L3 = HScroll3.Value  
ActiveGigE1.SetGains L1, L2, L3  
End Sub
```

### Remarks

The gain adjustment will be applied to the image only when [LUTMode](#) is enabled. Otherwise the original pixel values will remain intact.

---

### 3.2.138 SetLensDistortion

#### Description

Sets distortion parameters for the lens distortion correction.

#### Syntax

[VB]

```
objActiveGige.SetLensDistortion Alpha, Beta, Interpolate
```

[C/C++]

```
HRESULT SetLensDistortion (float Alpha, float Beta, bool Interpolate)
```

#### Data Types [VB]

*Alpha*: Single

*Beta*: Single

*Interpolate*: Boolean

#### Parameters [C/C++]

*Alpha* [in]

Floating point value of the radial distortion coefficient. The allowable range of values is from -1.0 to +1.0.

*Beta* [in]

Floating point value of the tangential distortion coefficient. The allowable range of values is from -1.0 to +1.0.

*Interpolate* [in]

If TRUE, the bilinear interpolation will be used for the lens distortion correction.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB examples activates the lens distortion correction and uses two scroll bars to adjust the distortion parameters in real time :

```
Private Sub Form_Load()  
ActiveGigel.Acquire=True  
HScroll1.Min = -1000  
HScroll1.Max = 1000  
HScroll2.Min = -1000  
HScroll2.Max = 1000  
ActiveGigel.SetLensDistortion 0,0,False  
ActiveGigel.LensCorrect=True  
ActiveGigel.Acquire=True  
End Sub
```

```
Private Sub HScroll1_Scroll()  
ActiveGig1.SetLensDistortion HScroll1.Value/1000., HScroll2.Value/1000.,  
False  
End Sub  
  
Private Sub HScroll2_Scroll()  
ActiveGig1.SetLensDistortion HScroll1.Value/1000., HScroll2.Value/1000.,  
False  
End Sub
```

### Remarks

The lens distortion correction is used to compensate for the barrel or pincushion image distortion caused by camera lenses. The barrel distortion makes straight lines at the edges of the image bow outwards and it is commonly seen on wide angle lenses with short focal length. The pincushion distortion makes straight lines at the edges of the image bow inwards, and it is commonly seen on telephoto lenses with long focal length. Some lenses can have a combination of both types of distortion.

Use negative values for Alpha and Beta parameters to correct for the barrel distortion, and positive values to correct for the pincushion distortion. A combination of positive and negative values can be used to correct for the complex type of distortion.

The most practical way to select the distortion parameters is to point the camera to a square grid pattern and empirically adjust the values of both coefficients so that the lines at the edges of the frame become as close to straight ones as possible.

---



### 3.2.139 SetLevels

#### Description

Sets the levels for the Window/Level operation. If the [WindowLevel](#) property is enabled, the operation performs a linear scaling of the histogram of each video frame thus optimizing the contrast, brightness and color of the video.

The lower limit indicates the darkest pixel value that will be mapped to the black (zero) level of the image or its color component. The upper limit indicates the brightest pixel value that will be mapped to the maximum pixel value of the image or its color component. Increasing the lower limit will make the shadows of the image darker. Decreasing the upper limit will make the highlights of the image brighter.

This method also allows you to assign the limits automatically by selecting the *Auto Fit* or/and *Auto White Balance* options. The Auto Fit optimizes the values of the lower and upper limits by providing the maximum contrast between the brightest and darkest pixel in the image. The Auto White Balance optimizes the values of the limits of each color component so that the average color of the image (or selected ROI) becomes gray.

#### Syntax

[VB]

```
objActiveGige.SetLevels minR, maxR [, minB, maxG, minB , maxB]
```

[C/C++]

```
HRESULT SetLevels(long minR, long maxR, long minG, long maxG, long minB,  
long maxB);
```

#### Data Types [VB]

*minR, maxR*: Long

*minG, maxG, minB, maxB*: Long (optional)

#### Parameters [C/C++]

*minR* [in], *maxR* [in]

Lower and upper limits for a monochrome image or for the red channel of a color image. The values are given in percents of the maximum pixel value for the currently selected image format.

If *minR* is set to -1, all the limits will be set automatically based on the Auto Fit algorithm.

If *minR* is set to -2, all the limits will be adjusted automatically based on the Auto White Balance algorithm.

*minB* [in], *maxB* [in], *minG* [in], *maxG* [in]

Lower and upper limits for the green and blue channels. If these parameters are omitted or set to zero, the green and blue channels are scaled proportionally with the red channel thus preserving the colors on the image. These parameters are ignored for a monochrome video.

#### Return Values

S\_OK

Success  
E\_FAIL  
Failure.

### Example

The following VB example uses a slider to adjust the brightness and contrast of the video without changing its color:

```
Private Sub Form_Load()  
ActiveGigE1.Acquire=True  
HScroll11.Min = 0  
HScroll11.Max = 100  
HScroll12.Min = 0  
HScroll12.Max = 100  
ActiveGigE1.WindowLevel=True  
End Sub  
  
Private Sub HScroll11_Scroll()  
L1 = HScroll11.Value  
L2 = HScroll12.Value  
ActiveGigE1.SetLevels L1, L2  
End Sub  
  
Private Sub HScroll12_Scroll()  
L1 = HScroll11.Value  
L2 = HScroll12.Value  
ActiveGigE1.SetLevels L1, L2  
End Sub
```

### Remarks

The limits for the Window/Level operation are given in percents of the maximum pixel value for the currently selected image format. The following table shows the maximum pixel value for standard GigE Vision formats

Pixel Format	Maximum pixel value
Mono8	255
Mono10, Mono10Packed	1023
Mono12, Mono12Packed	4095
Mono14	16383
Mono16	65535
Bayer**8, RGB8Packed, BGR8Packed, RGBA8Packed, BGRA8Packed, YUV411Packed, YUV422Packed, YUV444Packed, RGB8Planar	255
Bayer**10, RGB10Packed, BGR10Packed, BGR10V1Packed, BGR10V2Packed, RGB10Planar	1023
Bayer**12, RGB12Packed, BGR12Packed, RGB12Planar	4095
RGB16Planar	65535

The *Auto Fit* and *Auto White Balance* algorithms work on the currently selected [SetROI](#). When applying the Auto White Balance, make sure to select the ROI corresponding to the gray area in the image.

---

To retrieve the values of the limits assigned by **SetLevels**, use the [GetLevels](#) method.

### 3.2.140 SetLUT

#### Description

Assigns the array of values for the software lookup table. If the [LUTMode](#) is active, the pixel values in the incoming frames will be transformed based on a corresponding factor in the LUT array.

#### Syntax

[VB]  
`objActiveGige.SetROI LUT, Channels`

[C/C++]  
`HRESULT SetROI(VARIANT LUT, short Channels);`

#### Data Types [VB]

*LUT*: Variant (Array)

*Channels*: Integer

#### Parameters [C/C++]

##### *LUT*

SAFEARRAY of floating point factors to be used in the lookup table. The value of 1.0 corresponds to the maximum pixel value in the currently selected video format.

##### *Channels*

Number of color channels in the submitted array. Can be one of the following values:

- 1 - the array contains a one-channel LUT which will be used for all color components
- 3 - the array contains a three-channel LUT where the first 1/3 of elements represent the red channel, the second 1/3 of elements - green channel and the last 1/3 of elements - blue channel.
- 4 - the array contains a four-channel LUT used for the operation on a raw Bayer video. The first 1/4 of elements represent the R channel, the second 1/4 of elements - Gr channel, the third 1/4 of elements - B channel and the last 1/4 of elements - Gb channel.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

The following VB example prepares a 3-channel linear LUT with an amplified green channel and applies it to the video:

```
Dim LUT(256 * 3) As Single
For i = 0 To 255
    LUT(i) = i/255.
Next
For i = 256 To 511
    LUT(i) = (i-256)*2/255.
Next
For i = 512 To 767
    LUT(i) = (i-512)/255.
```

---

---

[Next](#)

[ActiveGige1.SetLUT A, 3](#)

### Remarks

The LUT processing in ActiveGige works in two stages. An array submitted by **SetLUT** and containing coefficients in the range 0 - 1.0 is converted to the internal LUT array with the number of elements and their values corresponding to the actual range of pixels in the currently selected format. This guarantees that the input LUT will have an identical effect on images of different types and pixel depths.

For example, if a submitted array has 256 elements with values gradually increasing from 0 to 0.5 and the currently selected format is Mono12 (for which the maximum pixel value is 4095), it will be converted to the internal LUT with 4096 elements and values gradually increasing from 0 to 2047. If the format is changed to Mono8, the internal LUT will automatically shrink to 256 elements with values gradually increasing from 0 to 255. If [LUTMode](#) is active, the value of each pixel will be mapped to the value of a corresponding element in the internal LUT.

For a single-channel LUT the input value of a pixel is used directly as an index in the LUT array. Multi-channel LUTs are logically divided into several subsequent LUTs starting from the red channel array. Therefore, if a submitted lookup table contains 768 elements and 3 channels, the first 256 elements will be used to index and remap red pixels, the second 256 elements - green pixels, and the last 256 elements - blue pixels.

To obtain the values of the current lookup table, use the [GetLUT](#) method.

### 3.2.141 SetROI

#### Description

Sets the rectangular region of interest and luminance range for the [histogram](#) and [image statistics](#) calculations.

#### Syntax

[VB]

```
objActiveGige.SetROI X1, Y1, X2, Y2 [,L1 , L2]
```

[C/C++]

```
HRESULT SetROI(long X1, long Y1, long X2, long Y2, long L1, long L2);
```

#### Data Types [VB]

X1, Y1, X2, Y2: Long  
L1, L2: Long (optional)

#### Parameters [C/C++]

X1 [in], Y1 [in]

Pixel coordinates of the top left corner of the ROI, relative to the image origin.

X2 [in], Y2 [in]

Pixel coordinates of the bottom right corner of the ROI, relative to the image origin.

L1 [in], L2 [in]

Optional threshold values specifying the luminance range for image statistics calculations. The luminance range of interest is defined as follows:

L1<L2    Only those pixel values greater or equal to L1 and less or equal to L2 are included into the calculations

L2<L1    Only those pixel values less than L2 or greater than L1 are included into the calculations

L1=L2    Only those pixel values equal to L1 are included into the calculations

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

The following VB example uses a slider to adjust a luminance range for the ROI:

```
Private Sub Form_Load()  
ActiveGige1.Palette = 8  
HScroll11.Min = 0  
HScroll11.Max = 255  
X1 = 0  
Y1 = 0  
X2 = ActiveGige1.SizeX
```

---

```
Y2 = ActiveGigel.SizeY
L1 = 0
L2 = 255
End Sub

Private Sub HScrollThreshold1_Scroll()
L1 = HScrollThreshold1.Value
LabelThreshold1.Caption = L1
ActiveGigel.SetROI X1, Y1, X2, Y2, L1, L2
End Sub
```

**Remarks**

If all four parameters are zero, the ROI will be reset to the maximum image area.

### 3.2.142 SetStreamChannel

#### Description

Sets the stream channel on which the video data will be received.

#### Syntax

[VB]  
`objActiveGige.SetStreamChannel Value`

[C/C++]  
`HRESULT SetStreamChannel(long Value);`

#### Data Types [VB]

*Value*: Integer

#### Parameters [C/C++]

*Value* [in]  
The stream channel to be set (0, 1...)

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example sets both the video source and stream channel on the camera to #1 and then initiates video acquisition on channel #1:

```
ActiveGigel.SetFeatureInt "SourceSelector", 1           'select Source1 as
video source on camera
ActiveGigel.SetFeature "GevStreamChannelSelector", 1   'select stream
channel #1 for this video source
ActiveGigel.SetDestinationIP 1, "", 0                  'set our host IP as
destination for channel #1
ActiveGigel.SetStreamChannel 1                         'tell ActiveGigE to
receive video on channel #1
ActiveGigel.Acquire = True
```

#### Remarks

This method should always be used in combination with [SetDestinationIP](#).

Calling **SetStreamChannel** assumes that the camera is already set up for streaming video data on the selected channel. This method does not modify camera features and registers. In order to prepare camera for multi-channel streaming, set corresponding camera's features prior to using **SetStreamChannel**, as shown in the example above.

---



Note that this method should be used only if the camera supports multiple stream channels.

### 3.2.143 SetVideoFPS

#### Description

Sets the playback frame rate of the currently open video file or memory sequence.

#### Syntax

[VB]  
`objActiveGige.SetVideoFPS Value`

[C/C++]  
`HRESULT SetVideoFPS(float Value);`

#### Data Types [VB]

*Value*: Single

#### Parameters [C/C++]

*Value* [in]  
The fps value to be set

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example opens an AVI file, doubles its frame rate and starts the playback.

```
ActiveGige1.OpenVideo "c:\\video1.avi"  
fps=ActiveGige1.GetVideoFPS * 2  
ActiveGige1.SetVideoFPS fps  
ActiveGige1.PlayVideo
```

#### Remarks

The actual frame rate during the playback of a video file may be lower than the set frame rate. This depends on the hard drive performance and the use of compression during the recording. If an AVI file is [open](#) with the sound option, the playback rate cannot be higher than 16 times of the recorded fps.

---

### 3.2.144 SetVideoPosition

#### Description

Seeks the specified frame in the currently open video file or memory sequence, extracts it and displays in the *ActiveGige* window.

#### Syntax

[VB]

```
objActiveGige.SetVideoPosition Frame [, Mode=0 ]
```

[C/C++]

```
HRESULT SetVideoPosition(long Frame, short Mode);
```

#### Data Types [VB]

*Frame*: Long

*Mode*: Integer

#### Parameters [C/C++]

*Frame* [in]

The zero-based index of the frame to set.

*Mode* [in]

If 0, *Frame* will be used as an absolute zero-based position of the frame in the file or sequence.

If 1, *Frame* will be used as an incremental move relative to the current frame position (i.e.

Frame=2 will move the video position two frames forward, while Frame=-2 will move it two frames back.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example shows how to use a scroll bar to move between the frames in the AVI file.

```
Private Sub Form_Load()  
ActiveGige1.OpenVideo "C:\\video1.avi"  
HScroll1.Min=0  
HScroll1.Max=ActiveGige1.GetVideoFrameCount - 1  
End Sub
```

```
Private Sub HScroll1_Scroll()  
frameposition = HScroll1.Value  
ActiveGige1.SetVideoPosition frameposition  
End Sub
```

#### Remarks

This method extracts the specified frame from the video file or memory sequence and loads it into the

internal *ActiveGigE* buffer. The [FrameLoaded](#) event will be fired, when **SetVideoPosition** is called successfully. The pixels of the loaded frame can be accessed via *ActiveGigE* image access functions ([GetImageData](#), [GetImagePointer](#), [GetImageWindow](#) etc).

The speed at which a random frame can be extracted from a video file depends on the performance of the hard drive and the compression used during the recording.

If the frame is extracted from a memory sequence, all current conversion properties (such as [Bayer](#), [WindowLevel](#), [BkgCorrect](#), [Rotate](#)) will be applied to the frame for the display.

Calling this method will automatically turn off the live video by setting the [Acquire](#) property to FALSE.

### 3.2.145 SetVideoSync

#### Description

Sets the playback synchronization mode (freerun or triggered) for the currently open video file or memory sequence. Used in combination with [PlayVideo](#).

#### Syntax

[VB]

```
objActiveGige.SetVideoSync Mode
```

[C/C++]

```
HRESULT SetVideoSync(short Mode);
```

#### Data Types [VB]

*Mode*: Integer

#### Parameters [C/C++]

*Mode* [in]

The synchronization mode of the currently open video source. Select one of the following values:  
0 - internal (freerun) synchronization. Video will be played by *ActiveGige* at the currently set frame rate.

1 - external (triggered) synchronization. Video will advanced to the next frame when [TriggerVideo](#) is called.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example shows how to synchronously play two AVI files using two *ActiveGige* objects.

```
Private Sub Form_Load()  
ActiveGige1.OpenVideo "C:\\video1.avi"  
ActiveGige2.OpenVideo "C:\\video2.avi"  
ActiveGige1.SetVideoSync 1  
ActiveGige2.SetVideoSync 1  
ActiveGige1.PlayVideo  
ActiveGige2.PlayVideo  
fps=ActiveGige1.GetVideoFPS  
Timer1.Interval=1000/fps  
End Sub
```

```
Private Sub Timer1_Timer()  
ActiveGige1.TriggerVideo  
ActiveGige2.TriggerVideo  
End Sub
```

**Remarks**

When the video file or sequence has just been opened, the playback is set to the internal synchronization.

Setting the video playback to the external synchronization mode is necessary when two video files or sequences need to be played synchronously.

Note that the external synchronization cannot be applied to an AVI files opened with the sound option.

---

### 3.2.146 SetVideoVolume

#### Description

Sets the audio volume level of the AVI file currently open for a playback.

#### Syntax

```
[VB]
objActiveGige.SetVideoVolume Value

[C/C++]
HRESULT SetVideoVolume(short Value);
```

#### Data Types [VB]

*Value*: Integer

#### Parameters [C/C++]

*Value* [in]  
The volume to be set, in percent.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example uses a scroll bar to adjust the video volume.

```
Private Sub Form_Load()
    HScroll1.Min=0
    HScroll1.Max=100
    ActiveGige1.OpenVideo "C:\\video1.avi", True
    HScroll1.Value=ActiveGige1.GetVideoVolume
    ActiveGige1.PlayVideo
End Sub

Private Sub HScroll1_Scroll()
    ActiveGige1.SetVideoVolume HScroll1.Value
End Sub
```

#### Remarks

In order for this method to work, the AVI file must be recorded with the sound and [opened](#) with the sound option enabled.

### 3.2.147 SetWebStreamer

#### Description

Configures parameters of the RTSP web streaming.

#### Syntax

[VB]

```
objActiveGige.SetWebStreamer srcAddress, dstAddress, frameRate, quality[,  
tunnelOverHttp]
```

[C/C++]

```
HRESULT SetDestinationIP(BSTR srcAddress, BSTR dstAddress, float frameRate,  
short quality, bool tunnelOverHttp);
```

#### Data Types [VB]

*dstAddress*: String

*srcAddress*: String

*frameRate*: Float

*quality*: Integer

*tunnelOverHttp*: Boolean

#### Parameters [C/C++]

*srcAddress* [in]

String specifying the source address of the web streaming in the following format:

"xxx.xxx.xxx.xxx[:pppp/filename]", where

xxx.xxx.xxx.xxx - the IP address of the local network interface to be used for the web streaming (typically a local area interface)

pppp - the local source port to be used for the web streaming; if omitted, port 8554 will be used

filename - the file name assigned to the stream; if omitted, "ActiveGige.sdp" will be used

*dstAddress* [in]

String specifying the IP address of a remote playback device. For the multicast streaming, use an address in the range 239.0.0.0 - 239.255.255.255

*frameRate* [in]

Floating point value specifying the frame rate limit for the web stream.

*quality* [in]

Integer value in the range 0-6 specifying the H.264 compression quality. The higher the quality is, the higher the streaming bandwidth will be.

*tunnelOverHttp* [in]

Enabling this boolean value will reroute the web streaming via HTTP ports 80, 8000 or 8080

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

---



## Example

This VB example sets the web streamer's parameters and activates the streaming. The port number is omitted in the source address, so the default port 8554 is used:

```
sourceAddr="192.168.0.5/test.sdp"  
destAddr="192.168.0.10"  
streamFPS=15  
streamQuality=4  
ActiveGigel.SetWebStreamer sourceAddr, destAddr, streamFPS, streamQuality  
ActiveGigel.Acquire=True  
ActiveGigel.WebStream=True
```

## Remarks

The web streaming option allows you to automatically convert video outputted by the camera into the H.264 compression format and transmit it over the wireless or wired network using the RTSP protocol to a remote playback devices, such as PCs, tablets and smartphones. Use the **SetWebStreamer** method to configure parameters of the web streamer before turning the [web streaming](#) on.

When specifying the source address, make sure to use the IP address of a local network interface which is usually different from the interface(s) to which your GigE Vision cameras are connected. The port number and/or filename can be omitted in the source address string, in which case default values will be used.

When specifying the destination address, use the intranet IP address of a remote device or network interface on a remote PC (you can look up IP addresses assigned to different devices on your network by accessing your router's web interface). If you want to multicast the web stream to all devices on your network, enter a multicast IP address such as 239.0.0.1

If the value of the frameRate parameter exceeds the camera's fps, the camera's fps will be used as the frame rate of the web video.

To watch a transmitted video on a remote playback device, use an RTSP/RTP client such as VLC Media Player or Fresh Video Player.

The following steps describe how to use VLC Media Player on a remote Windows-based client PC to display a web video stream transmitted by an ActiveGigE-based streaming application:

- Run VLC media player on a client PC (the IP address of the client PC must be same as the destination address used in your streaming application).
- Select Media -> Open Network Stream in the player's menu. The Open Media dialog box will be displayed.
- In the Network URL field enter the full RTSP web address matching the source address used in your ActiveGigE-based application, for example:  
rtsp://192.168.0.5:8554/test.asp
- Click the Play button. Within a few seconds a decoded video should appear on the player's screen.

### 3.2.148 ShowAudioDlg

#### Description

Displays the audio Input dialog for adjusting the mixer properties of the audio source.

#### Syntax

[VB]  
`objActiveGige.ShowAudioDlg`

[C/C++]  
`HRESULT ShowAudioDlg( );`

#### Parameters

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure

#### Example

This VB example selects the audio source and displays the audio input dialog:

```
ActiveGige1.SetAudioSource 0  
ActiveGige1.ShowAudioDlg
```

#### Remarks

The options available in the audio input dialog will depend on the configuration of your audio devices.

---

### 3.2.149 ShowCodecDlg

#### Description

Displays the configuration dialog for the currently selected codec. The dialog is provided by the codec's manufacturer.

#### Syntax

[VB]  
`objActiveGige.ShowCodecDlg`

[C/C++]  
`HRESULT ShowCodecDlg( );`

#### Parameters

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure

#### Example

This VB example displays the configuration dialog for the DivX codec:

```
ActiveGige1.SetCodec "DivX 5.0.2 Codec"  
ActiveGige1.ShowCodecDlg
```

#### Remarks

The settings selected in the codec configuration dialog will be remembered as long as the corresponding ActiveGige control remains in the memory.

### 3.2.150 ShowCompressionDlg

#### Description

Displays the compression dialog for the AVI recording. The dialog allows you to select a desired codec and adjust its parameters.

#### Syntax

[VB]  
`objActiveGige.ShowCompressionDlg`

[C/C++]  
`HRESULT ShowCompressionDlg( );`

#### Parameters

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure

#### Example

This VB example demonstrates a simple video recording application

```
Private Sub Form_Load()  
ActiveGige1.Acquire=True  
End Sub
```

```
Private Sub CompressionButton_Click()  
ActiveGige1.ShowCompressionDlg  
End Sub
```

```
Private Sub StartButton_Click()  
ActiveGige1.StartCapture "c:\mycapture.avi"  
End Sub
```

```
Private Sub StopButton_Click()  
ActiveGige1.StopCapture  
End Sub
```

#### Remarks

The settings selected in the compression dialog are remembered as long as the corresponding *ActiveGige* control remains in the memory.

*ActiveGige* video recording engine includes a proprietary "Raw Uncompressed" codec. When the

---

---

camera streams video in the Bayer format and the "Raw Uncompressed" codec is selected, *ActiveGige* records the video in its original raw format while displaying it in color. This reduces the disk space requirements by three times without any degradation in the image quality. In addition, when high-depth pixel formats are used (such as Bayer10, Bayer12, Bayer16, Mono10, Mono 12, Mono16), the recorded video will be packed into the 12-bit per pixel format as opposed to 16 bit, thus saving extra 25% of the disk space. To decode and play back AVI files recorded with the "Raw Uncompressed" codec, use video playback methods of the DVR version of *ActiveGige*.

---

### 3.2.151 ShowProperties

#### Description

Displays *ActiveGige* property pages in runtime mode.

#### Syntax

[VB]

`objActiveGige.ShowProperties [ EnableCamList, Page ]`

[C/C++]

`HRESULT ShowProperties( bool bEnableCamList, short iPage );`

#### Parameters

*EnableCamList*: Boolean

*Page*: Integer

#### Parameters [C/C++]

*bEnableCamList* [in]

Set to TRUE to enable the camera selection box in the Source property page, or set to FALSE to disable it. Default value - TRUE.

*iPage* [in]

The index of the property page to be activated. Use one of the following values to activate a specific property page:

0 or omitted - the property page which was active last time the property pages were used.

1 - the Source page

2 - the Format page

3 - the Analog page

4 - the Input/Output page

5 - the Advanced page

6 - the Display page

#### Return Values

S\_OK

Success

E\_FAIL

Failure

#### Example

This VB example demonstrates how to display the property pages in runtime mode.

```
Private Sub Properties_Click()  
ActiveGige1.ShowProperties False  
End Sub
```

#### Remarks

---

Setting the *EnableCamList* parameter to False allows you to disable the camera selection box in the **Source** property page thus preventing a user from switching to another camera. This is a recommended scenario in case your application uses several *ActiveGige* objects configured for different cameras. See [PropertyPages](#) for more information.

---

### 3.2.152 SoftTrigger

#### Description

Generates an internal trigger signal.

#### Syntax

[VB]  
`objActiveGige.SoftTrigger`

[C/C++]  
`HRESULT SoftTrigger();`

#### Parameters

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example demonstrates the use of the software trigger:

```
Private Sub Form_Load()  
ActiveGigel.Acquire = True  
ActiveGigel.TriggerSource = "Software"  
ActiveGigel.Trigger = True  
End Sub  
  
Private Sub SoftTrig_Click()  
ActiveGigel.SoftTrigger  
End Sub
```

#### Remarks

Use **SoftTrigger** to simulate the trigger signal. Note that this method will only have effect if [TriggerSource](#) is set to the "Software".

---



### 3.2.153 StartCapture

#### Description

Starts time-lapse video capture to the specified AVI file or series of image files (bmp, tif or jpg). Use [StopCapture](#) to end video capture.

#### Syntax

[VB]

```
objActiveGige.StartCapture File [, Timelapse = 0, Playrate = 0 ]
```

[C/C++]

```
HRESULT StartCapture( bstr File, float Timelapse, float Playrate );
```

#### Data Types [VB]

*File* : String

*Timelapse* : Single (optional)

*Playrate* : Single (optional)

#### Parameters [C/C++]

*File* [in]

The string containing the path to the avi file or image file (bmp, tif or jpg).

*Timelapse* [in]

The interval between consecutive frames in seconds

*Playrate* [in]

The frame rate at which AVT file will be played in frames per second

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid file name.

#### Example

This VB example demonstrates how to capture the video to an AVI file at the current frame rate:.

```
Private Sub StartButton_Click()  
ActiveGige1.StartCapture "c:\mycapture.avi"  
End Sub
```

```
Private Sub StopButton_Click()  
ActiveGige1.StopCapture  
End Sub
```

This C# example demonstrates how to capture a series of TIFF images at 0.5 sec interval:

```
private void startbutton_Click(object sender, System.EventArgs e)
{
    axActiveGige1.StartCapture("c:\\images\\myframe.tif", 0.5, 0.);
}

private void stopbutton_Click(object sender, System.EventArgs e)
{
    axActiveGige1.StopCapture();
}
```

### Remarks

The [Acquire](#) property must be set to TRUE prior to calling this method.

To record compressed AVI files, use [ShowCompressionDlg](#), [SetCodec](#) and [ShowCodecDlg](#). If no codec has been selected, *ActiveGige* will record AVI files in the uncompressed format, 8- or 24-bits per pixel.

If bmp, tif or jpg extension is specified for the file path, the file name is used as a template to which the ordinal number of the frame captured is appended. In the C# example above consecutive frames will be stored to files "myframe00001.tif", "myframe00002.tif" and so on. Note that TIF format can store 16- and 48-bit per pixel images.

If *Timelapse* is not specified, the video will be recorded at the maximum frame rate defined by the camera settings and system throughput.

If *Playrate* is not specified, the video will be played back at the same rate it was captured.

If the external device (typically, a hard drive or memory card) doesn't have enough space, the capture will stop automatically when the device is full and the [CaptureCompleted](#) event will be fired.

---

### 3.2.154 StopCapture

#### Description

Stops video capture to an AVI file or series of images.

#### Syntax

[VB]  
`objActiveGige.StopCapture`

[C/C++]  
`HRESULT StopCapture();`

#### Parameters

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure

#### Example

This VB example demonstrates how to capture the video to an AVI file with 0.5 sec time lapse between the frames.

```
Private Sub StartButton_Click()  
ActiveGige1.StartCapture "c:\mycapture.avi", 0.5  
End Sub
```

```
Private Sub StopButton_Click()  
ActiveGige1.StopCapture  
End Sub
```

#### Remarks

Use this method to end the video capture initiated by [StartCapture](#). The [CaptureCompleted](#) event will be generated when this method is called.

### 3.2.155 StartSequenceCapture

#### Description

Starts acquiring a sequence of frames into the memory. Use [StopSequenceCapture](#) to end the sequence capture process.

#### Syntax

[VB]

```
objActiveGige.StartSequenceCapture Frames [, Timelapse = 0 ]
```

[C/C++]

```
HRESULT StartSequenceCapture( long Frames, float Timelapse);
```

#### Data Types [VB]

*Frames* : Long

*Timelapse* : Single (optional)

#### Parameters [C/C++]

*Frames* [in]

The number of frames to capture.

Frames>0 - sequence capture will stop automatically after the specified number of frames is reached, unless [StopSequenceCapture](#) is called before that.

Frames<0 - loop recording mode; when the specified number of frames is reached, the capture will continue in a loop until [StopSequenceCapture](#) is called.

*Timelapse* [in]

The interval between consecutive frame acquisitions in seconds. If zero or not specified, the sequence will be captured at the current frame rate.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid file name.

#### Example

This VB example demonstrates how to capture the video loop into the system memory at the current frame rate. The maximum sequence size is set to 1000 frames:

```
Private Sub StartButton_Click()  
ActiveGige1.StartSequenceCapture -1000  
End Sub
```

```
Private Sub StopButton_Click()  
ActiveGige1.StopSequenceCapture  
End Sub
```

---

## Remarks

It is recommended to use [CreateSequence](#) for memory allocation before calling **StartSequenceCapture**. If [CreateSequence](#) was not called or did not allocate enough memory, **StartSequenceCapture** will have to reserve the necessary amount of memory itself, which will introduce a delay between the function call and actual start of the sequence capture.

During the sequence capture the [FrameRecorded](#) event will be fired per each frame recorded into the memory.

In the loop recording mode (*Frames* < 0) after the frame limit is reached, *ActiveGige* will maintain the specified number of recorded frames in the memory by removing a frame from the head of the sequence and adding the latest frame to its tail. Thus, in the example above, at any given moment the sequence will contain the most recent 1000 frames.

To reduce the memory consumption, *ActiveGige* records frames in a sequence in the original raw image format. When the sequence is played back, its frames are converted to a regular monochrome or RGB format on the fly. See [PlayVideo](#) for more details.

If *Timelapse* is not specified, the sequence will be recorded at the maximum frame rate defined by the camera settings and system throughput.

### 3.2.156 StopSequenceCapture

#### Description

Stops acquiring a sequence of frames into the memory.

#### Syntax

[VB]  
`objActiveGige.StopSequenceCapture`

[C/C++]  
`HRESULT StopSequenceCapture();`

#### Parameters

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure

#### Example

This VB example demonstrates how to end the sequence capture using a button click.

```
Private Sub StopButton_Click()  
ActiveGige1.StopSequenceCapture  
End Sub
```

#### Remarks

Use this method to explicitly end the sequence capture initiated by [StartSequenceCapture](#).

When the sequence capture stops, the [CaptureComplete](#) event will be fired.

---

### 3.2.157 StartVideoCapture

#### Description

Starts time-lapse video capture to the AVI file created by [CreateVideo](#). Use [StopVideoCapture](#) to end video capture.

#### Syntax

[VB]

```
objActiveGige.StartVideoCapture [Frames = 0 [, Timelapse = 0, Playrate = 0 ]
```

[C/C++]

```
HRESULT StartVideoCapture( long Frames, float Timelapse, float Playrate );
```

#### Data Types [VB]

*Frames* : Long

*Timelapse* : Single (optional)

*Playrate* : Single (optional)

#### Parameters [C/C++]

*Frames* [in]

The number of frames to capture. The video capture will stop automatically after the specified number of frames is reached or [StopVideoCapture](#) is called. If zero or omitted, the video capture will continue until the disk is full or [StopVideoCapture](#) is called.

*Timelapse* [in]

The interval between consecutive frames in seconds. If zero or omitted, the video will be recorded at the maximum frame rate define by the camera settings and system throughput.

*Playrate* [in]

The frame rate at which AVT file will be played in frames per second. If zero or omitted, the video will be played back at the same rate it was captured.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example demonstrates how to create an AVI file and capture 1000 frames into it.

```
Private Sub LoadForm()  
ActiveGige1.CreateVideo "c:\\mycapture.avi"  
ActiveGige1.Acquire = True  
End Sub
```

```
Private Sub StartButton_Click()  
ActiveGige1.StartVideoCapture 1000  
End Sub
```

**Remarks**

The advantage of using this method as opposed to [StartCapture](#) is that **StartVideoCapture** is called after an AVI file is already opened and preallocated with [CreateVideo](#). As a result, the video capture starts immediately with no delay.

The [Acquire](#) property must be set to TRUE prior to calling this method.

To record compressed AVI files, use [ShowCompressionDlg](#), [SetCodec](#) and [ShowCodecDlg](#). If no codec has been selected, *ActiveGige* will record AVI files in the uncompressed format, 8- or 24-bits per pixel.

If the external device (typically, a hard drive or memory card) does not have enough space, the capture will stop automatically when the device is full and the [CaptureCompleted](#) event will be fired.

---



### 3.2.158 StopVideoCapture

#### Description

Stops video capture to an AVI file.

#### Syntax

[VB]  
`objActiveGige.StopVideoCapture`

[C/C++]  
`HRESULT StopVideoCapture();`

#### Parameters

*None*

#### Return Values

`S_OK`  
Success  
`E_FAIL`  
Failure

#### Example

This VB example demonstrates how to capture the video to an AVI file with 0.5 sec time lapse between the frames.

```
Private Sub LoadForm()  
ActiveGige1.CreateVideo "c:\\mycapture.avi"  
ActiveGige1.Acquire = True  
End Sub
```

```
Private Sub StartButton_Click()  
ActiveGige1.StartVideoCapture 0, 0.5  
End Sub
```

```
Private Sub StopButton_Click()  
ActiveGige1.StopVideoCapture  
End Sub
```

#### Remarks

Use this method to end the video capture initiated by [StartVideoCapture](#). The [CaptureCompleted](#) event will be raised when this method is called.

### 3.2.159 StopVideo

#### Description

Stops the playback of a video file or memory sequence that has been initiated by [PlayVideo](#).

#### Syntax

[VB]  
`objActiveGige.StopVideo`

[C/C++]  
`HRESULT StopVideo( );`

#### Parameters [C/C++]

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example shows how to stop and resume the video playback using push buttons.

```
Private Sub Form_Load()  
ActiveGige1.OpenVideo "C:\\video1.avi"  
End Sub
```

```
Private Sub Play_Click()  
ActiveGige1.PlayVideo -1  
End Sub
```

```
Private Sub StopButton_Click()  
ActiveGige1.StopVideo  
End Sub
```

#### Remarks

When this method is called, the [PlayCompleted](#) event will be fired.

---

### 3.2.160 TriggerVideo

#### Description

Advances the playback of the video file or memory sequence to the next frame. Used in combination with [SetVideoSync](#) and [PlayVideo](#).

#### Syntax

[VB]  
`objActiveGige.TriggerVideo`

[C/C++]  
`HRESULT TriggerVideo();`

#### Parameters [C/C++]

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure

#### Example

This VB example shows how to synchronously play two AVI files using two *ActiveGige* objects.

```
Private Sub Form_Load()  
ActiveGige1.OpenVideo "C:\\video1.avi"  
ActiveGige2.OpenVideo "C:\\video2.avi"  
ActiveGige1.SetVideoSync 1  
ActiveGige2.SetVideoSync 1  
ActiveGige1.PlayVideo  
ActiveGige2.PlayVideo  
fps=ActiveGige1.GetVideoFPS  
Timer1.Interval=1000/fps  
End Sub
```

```
Private Sub Timer1_Timer()  
ActiveGige1.TriggerVideo  
ActiveGige2.TriggerVideo  
End Sub
```

#### Remarks

This method can only be used when the video synchronization has been set to the external mode(see [SetVideoSync](#)) and [PlayVideo](#) has been called.

### 3.2.161 WriteBlock

#### Description

Writes a block of data into the internal camera memory starting from the specified bootstrap address.

#### Syntax

[VB]

```
objActiveGige.WriteBlock Offset, Buffer, nBytes
```

[C/C++]

```
HRESULT WriteBlock( long Offset, Variant Buffer, long nBytes );
```

#### Data Types [VB]

*Offset*: Long

*Buffer*: Variant (pointer)

*nBytes*: Long

#### Parameters [C/C++]

*Offset* [in]

The 32-bit offset into the camera register address space.

*Buffer*

Variant containing pointer to a buffer that contains the data to be written into the camera memory starting from the specified offset

*nBytes*

The number of bytes to write to the specified offset

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This C++ example writes a block of 128 bytes starting from a specified offset in the camera address space:

```
unsigned char buffer[128]
for (int i=0;i<128;i++)
    buffer[i]=i;
VARIANT v;
v.pvData=buffer;
ActiveGige.WriteBlock(0xA0000,v,128);
```

#### Remarks

Using this method typically provides a much faster access to the internal camera memory than

---

---

repetitive calls to [WriteRegister](#).

If the specified offset does not exist or the camera cannot accomodate the size of the data block, this method will return an error.

### 3.2.162 WriteFile

#### Description

Transfers the indicated amount of bytes from the data array in memory to the specified file in the device.

#### Syntax

[VB]

```
objActiveGige.WriteFile Name, Offset, Length, Data
```

[C/C++]

```
HRESULT WriteFile( bstr Name, long Offset, long Length, VARIANT Data );
```

#### Data Types [VB]

*Name*: String

*Offset*: Long

*Length*: Long

*Data*: Variant (Array of Bytes)

#### Parameters [C/C++]

*Name* [in]

String specifying the name of the file in the device

*Offset* [in]

The transfer starting position from the beginning of the file in bytes

*Length* [in]

The number of bytes to transfer to the specified file

*Data* [in]

Pointer to SAFEARRAY of bytes containing data to be transferred to the file

#### Return Values

S\_OK

Success

E\_NOINTERFACE

File with the specified name does not exist in the device

E\_FAIL

Failure.

#### Example

This VB example reads a LUT array from the file on the device, inverts it and writes back to the device

```
w = ActiveGige1.ReadFile("LUTArray1", 0, 256)
For y = 0 To 255
    w(y) = 255- w(y)
Next
ActiveGige1.WriteFile "LutArray1",0,256,w
```

---

**Remarks**

The File Access functionality allows an application to read and write files hosted on the device. The data in those files may contain look-up tables, configuration sets, firmware, and other information.

You can use the WriteFile method to transfer the content of a file on a hard drive to the file in the device. Note that in order for this method to work, the [Access Mode](#) of the file should be "W" or "RW".

### 3.2.163 WriteRegister

#### Description

Writes a 32-bit integer value to the specified bootstrap register of the current camera.

#### Syntax

[VB]  
`objActiveGige.WriteRegister Reg, Value`

[C/C++]  
`HRESULT WriteRegister( long Reg, long Value );`

#### Data Types [VB]

*Reg*: Long

#### Parameters [C/C++]

*Reg* [in]  
The 32-bit offset of the register in the camera's address space  
*Value* [in]  
The value to be written in the register

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This C++ example changes the value of the heartbeat timeout using the corresponding GigE Vision™ register.

```
value=ActiveGige.WriteRegister(0x0938,5000);
```

This VB example modifies a value of the custom camera feature.

```
ActiveGige.WriteRegister &HA080,512
```

#### Remarks

The list of standard bootstrap registers can be found in *"GigE Vision Video Streaming and Device Control Over Ethernet Standard"* published by the Automated Imaging Association.

This method can be used to control custom camera attributes not available through GigE Vision™ feature interface. Refer to the camera documentation for the list of the manufacturer-specific registers.

---



---

## 3.3 Events

The following events are generated by *ActiveGige* control:

<a href="#"><u>FrameAcquired</u></a>	ID: 1	Called after a frame has been acquired and decoded
<a href="#"><u>FrameAcquiredX</u></a>	ID:10	Called after a frame has been acquired and decoded (multithreaded version)
<a href="#"><u>RawFrameAcquired</u></a>	ID:19	Called after a raw frame has arrived from the camera
<a href="#"><u>FrameReady</u></a>	ID:20	Called after a frame is about to be submitted for display
<a href="#"><u>FrameDropped</u></a>	ID: 3	Called if a frame is dropped during the video acquisition
<a href="#"><u>Timeout</u></a>	ID: 2	Called if the acquisition timeout has expired
<a href="#"><u>FormatChanged</u></a>	ID: 8	Called if the video size or pixel format has changed
<a href="#"><u>FrameRecorded</u></a>	ID:16	Called when a frame is added to a video file, image file or memory sequence
<a href="#"><u>FrameLoaded</u></a>	ID:15	Called when a frame is loaded from a video file, image file or memory sequence
<a href="#"><u>CaptureCompleted</u></a>	ID:17	Called when the capture into a video file or memory sequence is stopped
<a href="#"><u>PlayCompleted</u></a>	ID:18	Called when the playback of a video file or memory sequence is stopped
<a href="#"><u>EventMessage</u></a>	ID:13	Called when an asynchronous event arrives from the camera
<a href="#"><u>EventDataMessage</u></a>	ID:14	Called when an asynchronous data event arrives from the camera
<a href="#"><u>CameraPlugged</u></a>	ID:11	Called if a camera gets connected to the system
<a href="#"><u>CameraUnplugged</u></a>	ID:12	Called if a camera gets disconnected from the system
<a href="#"><u>MouseDown</u></a>	ID: 4	Called when the mouse button is pressed inside the control window
<a href="#"><u>MouseUp</u></a>	ID: 5	Called when the mouse button is released inside the control window
<a href="#"><u>MouseDownRight</u></a>	ID:21	Called when the right mouse button is pressed inside the control window
<a href="#"><u>MouseUpRight</u></a>	ID:22	Called when the right mouse button is released inside the control window
<a href="#"><u>MouseDownClick</u></a>	ID: 9	Called when the mouse button is double-clicked inside the control window
<a href="#"><u>MouseMove</u></a>	ID: 6	Called when the mouse button has moved inside the control window
<a href="#"><u>Scroll</u></a>	ID: 7	Called when the live video display has been scrolled

### 3.3.1 CameraPlugged

#### Description

This event is fired each time a GigE Vision™ camera is connected to the system .

#### Syntax

[VB]

```
Private Sub objActiveGige_CameraPlugged(ByVal Camera As Integer)
```

[C/C++]

```
HRESULT Fire_CameraPlugged(SHORT Camera);
```

#### Data Types [VB]

*Camera*: Integer

#### Parameters [C/C++]

*Camera*

The index of the camera that has been connected.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the **CameraPlugged** and **CameraUnplugged** events to detect changes in the number of GigE Vision™ cameras in the network:

```
Private Sub ActiveGige1_CameraPlugged(ByVal Camera As Integer)
    Dim Msg As String
    Msg = "Camera #" + Str(Camera) + " has been connected"
    MsgBox Msg
End Sub

Private Sub ActiveGige1_CameraUnplugged(ByVal Camera As Integer)
    Dim Msg As String
    Msg = "Camera #" + Str(Camera) + " has been disconnected"
    MsgBox Msg
End Sub
```

### 3.3.2 CameraUnplugged

#### Description

This event is fired each time a GigE Vision™ camera gets disconnected from the system.

#### Syntax

[VB]

```
Private Sub objActiveGige_CameraUnplugged(ByVal Camera As Integer)
```

[C/C++]

```
HRESULT Fire_CameraUnplugged(SHORT Camera);
```

#### Data Types [VB]

*Camera*: Integer

#### Parameters [C/C++]

*Camera*

The index of the camera that has been disconnected.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the **CameraUnplugged** to detect when the currently selected camera is unplugged:

```
Private Sub ActiveGigel_CameraUnplugged(ByVal Camera As Integer)
    If ActiveGigel.Camera = Camera Then
        MsgBox "Currently selected camera has been disconnected"
    End If
End Sub
```

---

### 3.3.3 CaptureCompleted

#### Description

This event is fired when the capture into the memory sequence or video file is stopped.

#### Syntax

[VB]

```
Private Sub objActiveGige_CaptureCompleted(ByVal Frames As Long)
```

[C/C++]

```
HRESULT Fire_CaptureCompleted(long Frames);
```

#### Data Types [VB]

*Frames*: long

#### Parameters [C/C++]

*Frames*

The number of frames recorded.

#### Example

This VB example records 1000 frames into the memory sequence and uses the **CaptureCompleted** event to save it in an AVI file.

```
Private Sub StartButton_Click()  
ActiveGige1.StartSequenceCapture 1000  
End Sub
```

```
Private Sub ActiveGigel_CaptureCompleted(ByVal Frames As Long)  
SaveSequence "C:\\video.avi"  
End Sub
```

#### Remarks

One of the following conditions must be met, before the **CaptureCompleted** event will be fired:

- Capture has stopped automatically, because the specified number of frames has been acquired.

- Capture to a file has stopped automatically because the device is full.

- Capture has been stopped explicitly by calling the [StopSequenceCapture](#) or [StopCapture](#) methods.

### 3.3.4 EventMessage

#### Description

This event is fired each time an asynchronous event arrives from the camera on the message channel.

#### Syntax

[VB]

```
Private Sub objActiveGige_EventMessage(ByVal eventID as long, ByVal blockID  
as long, ByVal timestamp as double,  
ByVal channelIndex as long)
```

[C/C++]

```
HRESULT Fire_CameraPlugged(long eventId, long blockId, double timestamp,  
long channelIndex);
```

#### Data Types [VB]

*eventID*: long  
*blockID*: long  
*timestamp*: double  
*channelIndex*: long

#### Parameters [C/C++]

*eventID*  
The 16-bit even identifier.  
*blockID*  
The 16-bit identifier of the data block associated with the event. 0 if there is no data block association.  
*timestamp*  
The timestamp representing the time at which the event was fired by the camera.  
*channelIndex*  
The index of the stream channel associated with the event.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example intercepts message events and displays the information associated with each event:

```
Private Sub Form_Load()  
ActiveGigel.SetFeatureString("EventNotification","On")  
ActiveGigel.Acquire = True  
End Sub
```

---

```
Private Sub ActiveGigE_EventMessage(ByVal eventID as long, ByVal  
blockID as long, ByVal timestamp as double, ByVal channelIndex as long)  
LabelEventID.Caption=eventID  
LabelBlockID.Caption=blockID  
LabelTimestamp.Caption=timestamp  
End Sub
```

### Remarks

The **EventMessage** events are generated only for the EVENT type messages. For the EVENTDATA messages *ActiveGigE* will fire [EventDataMessage](#) events.

If several events are encapsulated into one GigE Vision message, the **EventMessage** event will fire several times.

For more information on the GigE Vision messaging refer to "*GigE Vision Video Streaming and Device Control Over Ethernet Standard*" published by the Automated Imaging Association.

### 3.3.5 EventDataMessage

#### Description

This event is fired each time an asynchronous data event arrives from the camera on the message channel.

#### Syntax

[VB]

```
Private Sub objActiveGige_EventDataMessage(ByVal eventID as long, ByVal  
blockID as long, ByVal data as Variant,                               ByVal timestamp as double, ByVal  
channelIndex as long)
```

[C/C++]

```
HRESULT Fire_EventDataMessage(SHORT Camera);
```

#### Data Types [VB]

*eventID*: Long  
*blockID*: Long  
*data*: Variant (Array)  
*timestamp*: Double  
*channelIndex*: Long

#### Parameters [C/C++]

*eventID*  
The 16-bit even identifier.

*blockID*  
The 16-bit identifier of the data block associated with the event. 0 if there is no data block association.

*data*  
Pointer to SAFEARRAY of bytes contating the data encapsulated in the message.

*timestamp*  
The timestamp representing the time at which the event was fired by the camera.

*channelIndex*  
The index of the stream channel associated with the event.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example intercepts message events and displays the information associated with each event:

```
Private Sub Form_Load()
```

---



```
ActiveGigel.SetFeatureString("EventNotification","On")
ActiveGigel.Acquire = True
End Sub

Private Sub ActiveGigel_EventMessage(ByVal eventID as long, ByVal
blockID as long, ByVal A as Variant,
                                ByVal timestamp as double, ByVal
channelIndex as long)
LabelEventID.Caption=eventID
LabelBlockID.Caption=blockID
LabelA0.Caption=A(0)
LabelA0.Caption=A(1)
LabelA0.Caption=A(2)
LabelA0.Caption=A(3)
LabelTimestamp.Caption=timestamp
End Sub
```

### Remarks

The **EventMessage** events are generated only for the EVENTDATA type messages. For the EVENT messages *ActiveGigE* will fire [EventMessage](#) events.

For more information on the GigE Vision messaging refer to "*GigE Vision Video Streaming and Device Control Over Ethernet Standard*" published by the Automated Imaging Association.

### 3.3.6 FormatChanged

#### Description

This event is fired each time the frame size or pixel format of the camera changes.

#### Syntax

[VB]

```
Private Sub objActiveGige_FormatChanged()
```

[C/C++]

```
HRESULT Fire_FormatChanged();
```

#### Parameters [C/C++]

*None*

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the **FormatChanged** event to generate a sound signal:

```
Private Sub ActiveGige1_FormatChanged()  
    Beep  
End Sub
```

#### Remarks

The **FrameDropped** event is raised when the frame size or pixel format of the camera changes. Your application may use this event to perform certain actions when the video format is changed through the Property Pages of ActiveGige. See [ShowProperties](#) for more details.

---

### 3.3.7 FrameAcquired

#### Description

This event is fired each time a frame has been acquired, decoded and processed.

#### Syntax

```
[VB]
Private Sub objActiveGige_FrameAcquired()
```

```
[C/C++]
HRESULT Fire_FrameAcquired();
```

#### Parameters [C/C++]

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example uses the **FrameAcquired** event to access and display a pixel value in real time:

```
Private Sub ActiveGigel_FrameAcquired()
    Label1.Caption = ActiveGigel.GetPixel(16, 32)
End Sub
```

#### Remarks

One of the following conditions must be met, before the FrameAcquired event will be fired:

- The [Acquire](#) property has been set to TRUE
- The [Grab](#) method has been called.

The **FrameAcquired** event is fired from the interface thread to provide compatibility with all types of ActiveX containers. For applications created in VB.NET, C# and C++ it is recommended to use the [FrameAcquiredX](#) event.

### 3.3.8 FrameAcquiredX

#### Description

This event is fired each time a frame has been acquired, decoded and processed. Unlike [FrameAcquired](#) event it is fired from a processing thread providing a higher efficiency. May not work in certain containers.

#### Syntax

[VB]  
`Private Sub objActiveGige_FrameAcquiredX()`

[C/C++]  
`HRESULT Fire_FrameAcquiredX();`

#### Parameters

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB.NET example uses the **FrameAcquiredX** event to access and display a pixel value in real time:

```
Public Class Form1

    Delegate Sub UpdateFPSLabelCallback([text] As String)

    Private Sub AxActiveGigE1_FrameAcquiredX(sender As Object, e As
System.EventArgs) Handles AxActiveGigE1.FrameAcquiredX
        UpdateLabel(Format(AxActiveGigE1.GetPixel(16, 32)))
    End Sub

    Private Sub UpdateLabel(ByVal [text] As String)
        If Me.Label1.InvokeRequired Then
            Dim d As New UpdateLabelCallback(AddressOf UpdateLabel)
            Me.Invoke(d, New Object() {[text]})
        Else
            Me.Label1.Text = [text]
        End If
    End Sub

End Class
```

#### Remarks

This event is provided for applications that can process events fired from a processing thread

---

---

(VB.NET, C#, C++). For applications created in VB6, VBA, Delphi and Matlab use [FrameAcquired](#) event instead.

One of the following conditions must be met, before the **FrameAcquiredX** event will be fired:

- The [Acquire](#) property has been set to TRUE

- The [Grab](#) method has been called.

---

### 3.3.9 FrameDropped

#### Description

This event is fired each time a frame is dropped during the video acquisition.

#### Syntax

[VB]

```
Private Sub objActiveGige_FrameDropped()
```

[C/C++]

```
HRESULT Fire_FrameDropped();
```

#### Parameters [C/C++]

*None*

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the **FrameDropped** event to generate a sound signal:

```
Private Sub ActiveGige1_FrameDropped()  
    Beep  
End Sub
```

#### Remarks

The **FrameDropped** event is raised if an image frame is dropped during the data transmission. The most common reasons for this are the loss of UDP packets due to the network overload, extensive image processing of each frame by an application and limited bandwidth during the AVI recording.

---

### 3.3.10 FrameLoaded

#### Description

This event is fired each time a frame has been loaded from a video file or memory sequence during the playback ([PlayVideo](#)). It is also fired when the image is loaded from an image file ([LoadImage](#)).

#### Syntax

```
[VB]
Private Sub objActiveGige_FrameLoaded(ByVal Frame As Long)
[C/C++]
HRESULT Fire_FrameLoaded();
```

#### Data Types [VB]

*Frame*: long

#### Parameters [C/C++]

*Frame*  
The zero-based index of the last loaded frame.

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example uses the **FrameLoaded** event to update a sequence frame counter:

```
Private Sub ActiveGigel_FrameLoaded((ByVal Frame As Long))
    Label1.Caption = Frame
End Sub
```

#### Remarks

### 3.3.11 FrameRecorded

#### Description

This event is fired each time a frame has been added to a memory sequence ([StartSequenceCapture](#)) or video file ([StartCapture](#)). It is also fired when an image file has been saved ([SaveImage](#), [SaveBkg](#)).

#### Syntax

[VB]

```
Private Sub objActiveGige_FrameRecorded(ByVal Frame As Long)
```

[C/C++]

```
HRESULT Fire_FrameRecorded();
```

#### Data Types [VB]

*Frame*: long

#### Parameters [C/C++]

*Frame*

The zero-based index of the last recorded frame.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the **FrameRecorded** event to update a sequence frame counter:

```
Private Sub ActiveGige1_FrameRecorded(ByVal Frame As Long)
    Label1.Caption = Frame
End Sub
```

#### Remarks

---



### 3.3.12 FrameReady

#### Description

This event is fired each time a decoded frame is submitted for display.

#### Syntax

[VB]  
`Private Sub objActiveGige_FrameReady()`

[C/C++]  
`HRESULT Fire_FrameReady();`

#### Parameters [C/C++]

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This MFC fragment uses the **FrameReady** event to perform a post-processing of the color video:

```
void CGcamProDlg::OnFrameReadyActivegige1()  
{  
    BYTE *ptr=m_ActiveGige.GetImagePointer(0, m_ActiveGige.GetSizeY()-1);  
    int nSize=m_ActiveGige.GetSizeX()*m_ActiveGige.GetSizeY();  
    int max=255;  
    for(int y=0;y<nSize;y++,ptr++)  
        *ptr=(*ptr+16)*2;  
}
```

#### Remarks

The **FrameReady** event is fired from *ActiveGige*'s display thread which is running separately from the acquisition thread. This allows your post-processing routine to take advantage of a multi-processor or multi-core architecture. Note that If the post-processing procedure is not fast enough, it will only affect the display frame rate and will not result in missed frames in the acquisition thread.

In order to synchronize the post-processing with image display, the [Display](#) property should be set to False and [Draw](#) method used in the end of the event handler.

One of the following conditions must be met, before the **FrameReady** event will be fired:

- The [Acquire](#) property has been set to TRUE
- The [Grab](#) method has been called.

*Note - this event may not work in older development environments such as VB6*

---

### 3.3.13 MouseDblClick

#### Description

This event is fired each time the mouse button is double-clicked inside the control window. Returns the coordinates of a pixel pointed by the cursor.

#### Syntax

[VB]

```
Private Sub objActiveGige_MouseDblClick( ByVal X As Integer, ByVal Y As Integer )
```

[C/C++]

```
HRESULT Fire_MouseDblClick( SHORT X, SHORT Y );
```

#### Data Types [VB]

X: Integer

Y: Integer

#### Parameters [C/C++]

X [in]

The X-coordinate of the pixel pointed by the cursor.

Y [in]

The Y-coordinate of the pixel pointed by the cursor.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the **MouseDblClick** event to open the *Properties* dialog:

```
Private Sub ActiveGigel_MouseDblClick(ByVal X As Integer, ByVal Y As Integer)
    Dim Value As Integer
    Value = ActiveGigel.ShowProperties
End Sub
```

#### Remarks

Note that the coordinates returned by this event refer to the image coordinate system, not to the screen coordinates.

Set the [Acquire](#) and [ScrollBars](#) properties to TRUE in order to display a scrollable live video and get a visual access to all parts of the image.

### 3.3.14 MouseDown

#### Description

This event is fired each time the mouse button is pressed inside the control window. Returns the coordinates of a pixel pointed by the cursor.

#### Syntax

[VB]

```
Private Sub objActiveGige_MouseDown( ByVal X As Integer, ByVal Y As Integer )
```

[C/C++]

```
HRESULT Fire_MouseDown( SHORT X, SHORT Y );
```

#### Data Types [VB]

X: Integer

Y: Integer

#### Parameters [C/C++]

X [in]

The X-coordinate of the pixel pointed by the cursor.

Y [in]

The Y-coordinate of the pixel pointed by the cursor.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the **MouseDown** event to display the value of the pixel pointed by the cursor:

```
Private Sub ActiveGigel_MouseDown(ByVal X As Integer, ByVal Y As Integer)
    Dim Value As Integer
    Value = ActiveGigel.GetPixel(X, Y)
    MsgBox Value
End Sub
```

#### Remarks

Note that the coordinates returned by this event refer to the image coordinate system, not to the screen coordinates.

Set the [Acquire](#) and [ScrollBars](#) properties to TRUE in order to display a scrollable live video and get a visual access to all parts of the image.

---

### 3.3.15 MouseDownRight

#### Description

This event is fired each time the right mouse button is pressed inside the control window. Returns the coordinates of a pixel pointed by the cursor.

#### Syntax

[VB]

```
Private Sub objActiveGige_MouseDownRight( ByVal X As Integer, ByVal Y As Integer )
```

[C/C++]

```
HRESULT Fire_MouseDownRight( SHORT X, SHORT Y );
```

#### Data Types [VB]

X: Integer

Y: Integer

#### Parameters [C/C++]

X [in]

The X-coordinate of the pixel pointed by the cursor.

Y [in]

The Y-coordinate of the pixel pointed by the cursor.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the **MouseDownRight** event to display the value of the pixel pointed by the cursor:

```
Private Sub ActiveGigel_MouseDownRight(ByVal X As Integer, ByVal Y As Integer)
    Dim Value As Integer
    Value = ActiveGigel.GetPixel(X, Y)
    MsgBox Value
End Sub
```

#### Remarks

Note that the coordinates returned by this event refer to the image coordinate system, not to the screen coordinates.

Set the [Acquire](#) and [ScrollBars](#) properties to TRUE in order to display a scrollable live video and get a

visual access to all parts of the image.

---

### 3.3.16 MouseMove

#### Description

This event is fired each time the mouse has moved inside the control window. Returns the coordinates of a pixel pointed by the cursor.

#### Syntax

```
[VB]
Private Sub objActiveGige_MouseMove( ByVal X As Integer, ByVal Y As Integer
)
```

```
[C/C++]
HRESULT Fire_MouseMove( SHORT X, SHORT Y );
```

#### Data Types [VB]

X: Integer

Y: Integer

#### Parameters [C/C++]

X [in]  
The X-coordinate of the pixel pointed by the cursor.

Y [in]  
The Y-coordinate of the pixel pointed by the cursor.

#### Return Values

S\_OK  
Success

E\_FAIL  
Failure.

#### Example

This VB example uses the **MouseMove** event to display the value of the pixel pointed by the cursor:

```
Private Sub ActiveGigel_MouseMove(ByVal X As Integer, ByVal Y As
Integer)
Dim Value As Integer
Value = ActiveGigel.GetPixel(X, Y)
Label1.Caption = Value
End Sub
```

#### Remarks

Note that the coordinates returned by this event refer to the image coordinate system, not to the screen coordinates.

Set the [Acquire](#) and [ScrollBars](#) properties to TRUE in order to display a scrollable live video and get a visual access to all parts of the image.

### 3.3.17 MouseUp

#### Description

This event is fired each time the mouse button is released inside the control window. Returns the coordinates of a pixel pointed by the cursor.

#### Syntax

[VB]

```
Private Sub objActiveGige_MouseUp( ByVal X As Integer, ByVal Y As Integer )
```

[C/C++]

```
HRESULT Fire_MouseUp( SHORT X, SHORT Y );
```

#### Data Types [VB]

X: Integer

Y: Integer

#### Parameters [C/C++]

X [in]

The X-coordinate of the pixel pointed by the cursor.

Y [in]

The Y-coordinate of the pixel pointed by the cursor.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the **MouseUp** event to display the value of the pixel pointed by the cursor:

```
Private Sub ActiveGigel_MouseUp(ByVal X As Integer, ByVal Y As Integer)
    Dim Value As Integer
    Value = ActiveGigel.GetPixel(X, Y)
    MsgBox Value
End Sub
```

#### Remarks

Note that the coordinates returned by this event refer to the image coordinate system, not to the screen coordinates.

Set the [Acquire](#) and [ScrollBars](#) properties to TRUE in order to display a scrollable live video and get a visual access to all parts of the image.

---



### 3.3.18 MouseUpRight

#### Description

This event is fired each time the rightmouse button is released inside the control window. Returns the coordinates of a pixel pointed by the cursor.

#### Syntax

[VB]

```
Private Sub objActiveGige_MouseUpRight( ByVal X As Integer, ByVal Y As Integer )
```

[C/C++]

```
HRESULT Fire_MouseUpRight( SHORT X, SHORT Y );
```

#### Data Types [VB]

X: Integer

Y: Integer

#### Parameters [C/C++]

X [in]

The X-coordinate of the pixel pointed by the cursor.

Y [in]

The Y-coordinate of the pixel pointed by the cursor.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the **MouseUpRight** event to display the value of the pixel pointed by the cursor:

```
Private Sub ActiveGigel_MouseUpRight(ByVal X As Integer, ByVal Y As Integer)
    Dim Value As Integer
    Value = ActiveGigel.GetPixel(X, Y)
    MsgBox Value
End Sub
```

#### Remarks

Note that the coordinates returned by this event refer to the image coordinate system, not to the screen coordinates.

Set the [Acquire](#) and [ScrollBars](#) properties to TRUE in order to display a scrollable live video and get a visual access to all parts of the image.

### 3.3.19 PlayCompleted

#### Description

This event is fired when the playback of the memory sequence or video file is stopped.

#### Syntax

[VB]

```
Private Sub objActiveGige_PlayCompleted(ByVal Frames As Long)
```

[C/C++]

```
HRESULT Fire_PlayCompleted(long Frames);
```

#### Data Types [VB]

*Frames*: long

#### Parameters [C/C++]

*Frames*

The number of frames recorded.

#### Example

This VB example opens the memory sequence, plays it and uses the **PlayCompleted** event to turn the live video on.

```
Private Sub Play_Click()  
ActiveGige1.OpenVideo "ram"  
ActiveGige1.PlayVideo  
End Sub
```

```
Private Sub ActiveGigel_PlayCompleted(ByVal Frames As Long)  
ActiveGigel.CloseVideo  
ActiveGigel.Acquire=True  
End Sub
```

#### Remarks

One of the following conditions must be met, before the **PlayCompleted** event will be fired:

Playback has stopped automatically, because the specified number of frames has been reached.

Playback has been stopped explicitly by calling [StopVideo](#).

---

### 3.3.20 RawFrameAcquired

#### Description

This event is fired each time a raw frame has arrived from the camera, before it is submitted to *ActiveGige* decoding and processing chain. Can be used in combination with [GetRawData](#) to perform a preprocessing on raw images.

#### Syntax

[VB]

```
Private Sub objActiveGige_RawFrameAcquired()
```

[C/C++]

```
HRESULT Fire_RawFrameAcquired();
```

#### Parameters [C/C++]

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This MFC fragment uses the **RawFrameAcquired** event to perform a preprocessing (invert operation) of the raw video:

```
void CGcamProDlg::OnRawFrameAcquiredActivegigel()  
{  
    VARIANT v=m_ActiveGige.GetRawData(true);  
    int nSize=m_ActiveGige.GetSizeX()*m_ActiveGige.GetSizeY();  
    BYTE* ptr=(BYTE*)v.pvData;  
    int max=255;  
    for(int y=0;y<nSize;y++,ptr++)  
        *ptr=max-*ptr;  
}
```

#### Remarks

One of the following conditions must be met, before the **RawFrameAcquired** event will be fired:

- The [Acquire](#) property has been set to TRUE
- The [Grab](#) method has been called.

The internal ActiveGige processing chain will be blocked until the container application releases the event handler. Therefore the displayed frame rate can drop and frames can get missed, if the external pre-processing procedure is not fast enough.

### 3.3.21 Scroll

#### Description

This event is fired each time the live video display has been scrolled. Returns the horizontal and vertical scroll positions.

#### Syntax

[VB]

```
Private Sub objActiveGige_Scroll( ByVal X As Integer, ByVal Y As Integer )
```

[C/C++]

```
HRESULT Fire_Scroll( SHORT ScrollX, SHORT ScrollY );
```

#### Data Types [VB]

*ScrollX*: Integer

*ScrollY*: Integer

#### Parameters [C/C++]

*ScrollX* [in]

The X-coordinate of the pixel pointed by the cursor.

*ScrollY* [in]

The Y-coordinate of the pixel pointed by the cursor.

#### Return Values

S\_OK

Success

E\_FAIL

Failure.

#### Example

This VB example uses the **Scroll** event to display the position of the live video in the control window:

```
Private Sub ActiveGigel_Scroll(ByVal ScrollX As Integer, ByVal ScrollY  
As Integer)  
Label1.Caption = ScrollX  
Label2.Caption = ScrollY  
End Sub
```

#### Remarks

Note that the scroll positions returned by this event refer to the image coordinate system, not to the screen coordinates.

The [ScrollBars](#) properties to TRUE in order for the **Scroll** event to be fired.

---

### 3.3.22 Timeout

#### Description

This event is fired each time a timeout occurs during the acquisition of a frame.

#### Syntax

[VB]  
`Private Sub objActiveGige_Timeout()`

[C/C++]  
`HRESULT Fire_Timeout();`

#### Parameters [C/C++]

*None*

#### Return Values

S\_OK  
Success  
E\_FAIL  
Failure.

#### Example

This VB example uses the **Timeout** event to generate a sound signal:

```
Private Sub ActiveGige1_Timeout()  
    Beep  
End Sub
```

#### Remarks

The **Timeout** event is raised when the amount of seconds specified by the [Timeout](#) property passes after the [Grab](#) method has been called and no frame has been acquired. The event will be raised repeatedly if the control is set in the continuous acquisition mode ([Acquire](#) is TRUE). The most common reason for a timeout is a missing trigger signal in the [Trigger](#) mode.

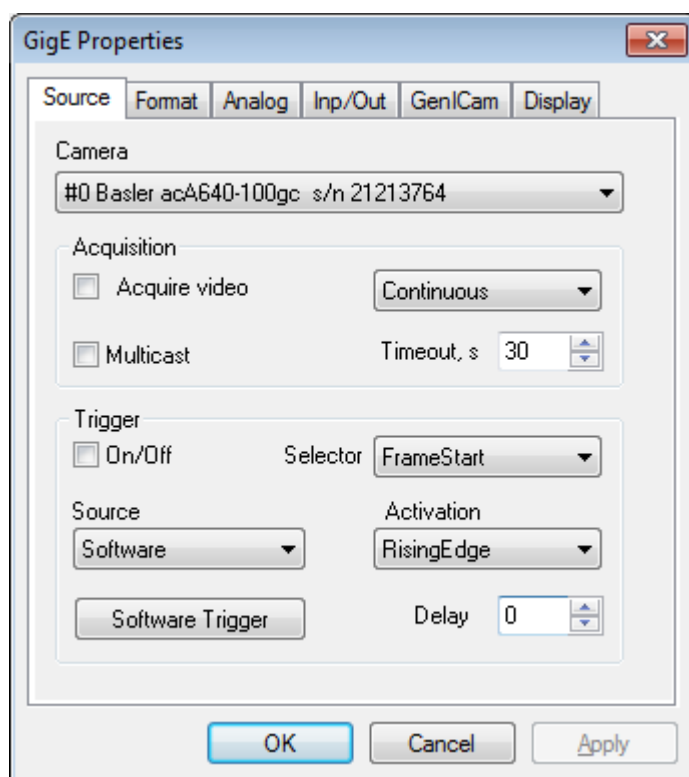
## 3.4 Property Pages

The following property pages are available in *ActiveGige* control:

<a href="#">Source</a>	Used to select the properties specifying the source for the video input
<a href="#">Format</a>	Used to select the properties specifying the format of the video
<a href="#">Analog</a>	Used to select the properties specifying the analog features of the video
<a href="#">Input/Output</a>	Used to select the properties specifying the input/output features of the video
<a href="#">GenICam</a>	Provides an access to the complete GenICam feature set of the camera
<a href="#">Display</a>	Used to select the properties specifying the display settings

### 3.4.1 Source

This property page is used to select the properties specifying the source for the video input.



Select from the following options:

#### Camera

Displays the vendor's name and model of the currently selected camera. If you have more than one GigE Vision™ camera connected to your system, you can switch to another camera by choosing the corresponding camera name in the list. Equivalent to the [Camera](#) property.

#### Acquire

Lets you enable the continuous acquisition mode. If this box is checked, the camera will continuously acquire the video into the internal image memory. If the control is visible, the live video will be displayed in the control window. Equivalent to the [Acquire](#) property.

#### Multicast

Check this box to enable the Multicast mode which allows multiple computers and applications on the network to receive the video feed from the same camera. Only the first application that connects to the camera can enable or disable the Multicast mode. All other *ActiveGige* based applications launched in the network can acquire and display the video in the slave mode, but will have no control over the camera settings. This option is equivalent to the [Multicast](#) property.

#### Timeout

Use this option to set the number of seconds to wait for a frame to be acquired. Typically used to assign the timeout when the [Trigger](#) mode is active. If the timeout expires, the [Timeout event](#) will be raised. Equivalent to the [Timeout](#) property.

#### Acquisition mode

Lets you select the desired acquisition mode from the list. The acquisition mode defines how many

frames will be captured when the **Acquire** option is enabled. Equivalent to the [AcquisitionMode](#) property.

**Trigger Source**

Lets you select the configuration of a trigger signal. The rest of the options in the Trigger group are dependent on this selection. Equivalent to the [TriggerSelector](#) property.

**Trigger**

Check this box to enable the selected trigger. This mode is typically used with an asynchronously resettable camera. An acquisition will occur upon receiving a signal from an external hardware **Trigger Source**. Equivalent to the [Trigger](#) property.

**Trigger Source**

Lets you select the source for the selected trigger. Depending on a camera, there may be one or more hardware trigger inputs as well as the software trigger. If the software trigger is available and selected, use the **Software Trigger** button to simulate the trigger event. If the camera doesn't support trigger source selection, this option will be unavailable. Equivalent to the [TriggerSource](#) property and [SoftTrigger](#) method.

**Trigger activation**

Lets you change the activation mode (trigger signal polarity) for the selected trigger. Equivalent to the [TriggerActivation](#) property.

**Trigger delay**

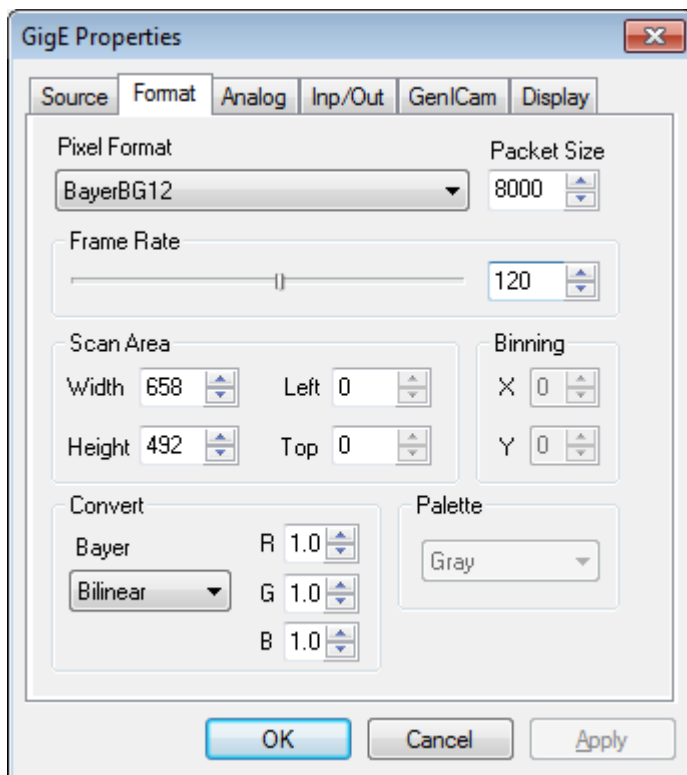
Lets you select the value for the trigger delay which defines the time between the arrival of the trigger signal and its effective activation. Equivalent to the [TriggerDelayRaw](#) property.

---



### 3.4.2 Format

This property page is used to select the properties specifying the format of the video stream.



Select from the following options:

#### Pixel Format

Use this option to select the desired pixel format from the list of modes available for the current camera. Equivalent to the [Format](#) property.

#### Packet size

Use this option to select the number of bytes in the image data packet. To lower the overhead of the packet transmission, it is recommended to set this property to the maximum packet size allowed by your network card and network configuration (typically 1500 if Jumbo packets are disabled and 9014 if Jumbo packets are enabled). Equivalent to the [PacketSize](#) property.

#### Frame rate

Use this option to set the acquisition frame rate in frames per seconds. Note that the actual frame rate can also depend on the exposure time. Equivalent to the [AcquisitionFrameRateAbs](#) property. If the camera doesn't support this property, the **Frame rate** option will be unavailable.

#### Scan Area

Lets you change the size and position of the image window on the camera's sensor. To modify the size and position of the window, enter the desired values for the image width, height, left coordinate and top coordinate in pixels. Equivalent to the [SizeX](#), [SizeY](#), [OffsetX](#), [OffsetY](#) properties.

#### Binning

Lets you change the number of horizontal and vertical photo-sensitive cells that must be combined together. This property has a net effect of increasing the intensity (or signal to noise ratio) of the

pixel value and reducing the horizontal size of the image. If the camera doesn't support the binning, this option will be unavailable. Equivalent to the [BinningX](#) and [BinningY](#) properties.

### Bayer

Select this option to activate the real-time color conversion of a monochrome raw video generated by a Bayer camera and select the specific Bayer conversion algorithm. Select one of the following options:

*None* - Bayer conversion is disabled. The camera will output a monochrome raw image.

*Nearest* - Nearest Neighbour filter. Missing pixels are substituted with adjacent pixels of the same color.

*Bilinear* - Bilinear filter. Calculates the values of missing pixels by performing bilinear interpolation of the adjacent pixels.

*Bilinear HQ* - High Quality Linear filter. Calculates the values of missing pixels based on the Malvar, He and Cutler algorithm.

See the [Bayer](#) property for more information.

### R, G, B, Y

Let you adjust the gain factors for individual color channels or the intensity factor for a monochrome video. Equivalent to the [SetGains](#) property.

### Palette

Lets you select one of a few predefined palette to be applied to a grayscale live video. The palettes represent choices that may be useful in viewing different kinds of video in pseudo-colors. Choose among the following palettes:

#### Gray

Applies the standard 256-level grayscale palette. This is a regular mode of viewing a grayscale video.

#### Inverse

Applies the inverted 256-level grayscale palette. The video will be displayed in the negative format.

#### Saturated

Applies the grayscale palette with colored upper entries. The saturated palette allows you to control the dynamic range of the video signal by bringing it slightly below the saturation level of the video camera or video amplifier. To achieve the maximum dynamic range, adjust the intensity of the light source and/or the gain and zero level of the video amplifier so that the red color corresponding to the brightest pixel values just barely shows up.

#### Rainbow

Applies a color palette where the entries are evenly distributed along the Hue axis. This allows for assigning different color pigments to different levels of intensity.

#### Spectra

Applies a color palette where the entries are distributed along the Hue and Luminance axes. That allows for assigning different color pigments to different levels of intensity while preserving the luminance scale.

#### Isodense

Applies the 256-level grayscale palette, each 8-th entry of which is colored. The isodense palette allows you to clearly see transitions between different levels of intensities as isolines on a topographic map.

#### Multiphase

Applies the multiphase palette. Entries in the multiphase palette are at opposite ends of the color model so even small changes in gray levels are highlighted.

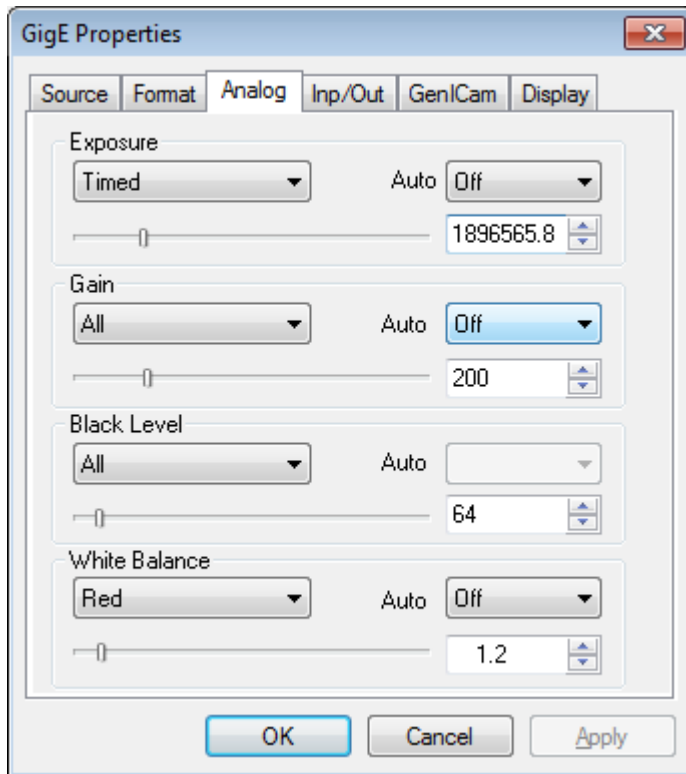
#### Random

Applies the random color palette whose entries are filled with random values each time you select it from the list.

This option is equivalent to the [Palette](#) property.

### 3.4.3 Analog

This property page is used to select the properties specifying the analog features of the camera.



Select from the following options:

#### Exposure Mode

Lets you select the operation mode of the exposure (shutter). See [ExposureMode](#) for more details.

#### Exposure Auto

Lets you select between the automatic (AE) or manual exposure control mode. Depending on the camera, the following selections can be available:

*Off* - exposure is manually controlled using **Exposure Value** option.

*Once* - the camera sets the optimal exposure level and returns to the "Off" state

*Continuous* - the camera constantly adjusts the exposure level

The availability and possible values of this option depend on the [ExposureAuto](#) property.

#### Exposure Value

Use the slider and spin controls to adjust the integration time of the incoming light. You can also enter the desired exposure value in the corresponding text box. Note that this option is available only for the cameras that support the manual exposure control. Equivalent to the [ExposureTimeRaw](#) or [ExposureTimeAbs](#) property.

#### Gain Selector

Lets you select the color channel to be controlled by the **Gain Value** and **Gain Auto** options. See [GainSelector](#) for more details.

#### Gain Auto

Lets you select between the automatic (AGC) or manual gain control mode. Depending on the camera, the following selections can be available:

- Off* - gain is manually controlled using **Gain Value** option
- Once* - the camera sets the optimal gain level and returns to the "Off" state
- Continuous* - the camera constantly adjusts the gain level

The availability and possible values of this option depend on the [GainAuto](#) property.

### Gain Value

Use the slider and spin controls to adjust the camera's video signal amplification. You can also enter the desired gain value in the corresponding text box. Note that this option is available only for the cameras that support the manual gain control. Equivalent to the [GainRaw](#) or [GainAbs](#) property.

### Black Level Selector

Lets you select the color channel to be controlled by the **Black Level Value** and **Black Level Auto** options. See [BlackLevelSelector](#) for more details.

### Black Level Auto

Lets you select between the automatic or manual black level control mode. Depending on the camera, the following selections can be available:

- Off* - black level is manually controlled using **Black Level Value** option
- Once* - the camera sets the optimal black level and returns to the "Off" state
- Continuous* - the camera constantly adjusts the black level

The availability and possible values of this option depend on the [BlackLevelAuto](#) property.

### Black Level Value

Use the slider and spin controls to adjust the black level (brightness) of the video signal. You can also enter the desired black level value in the corresponding text box. Note that this option is available only for the cameras that support the manual black level control. Equivalent to the [BlackLevelRaw](#) or [BlackLevelAbs](#) property.

### White Balance Selector

Lets you select the color channel to be controlled by the **White Balance Ratio** option. See [BalanceRatioSelector](#) for more details.

### White Balance Auto

Lets you select between the automatic (AWB) or manual white balance control mode. Depending on the camera, the following selections can be available:

- Off* - balance ration for a selected color channel is manually controlled using **White Balance Ratio** option.
- Once* - the camera sets the optimal white balance level and returns to the "Off" state
- Continuous* - the camera constantly adjusts the white balance level

The availability and possible values of this option depend on the [BalanceWhiteAuto](#) property.

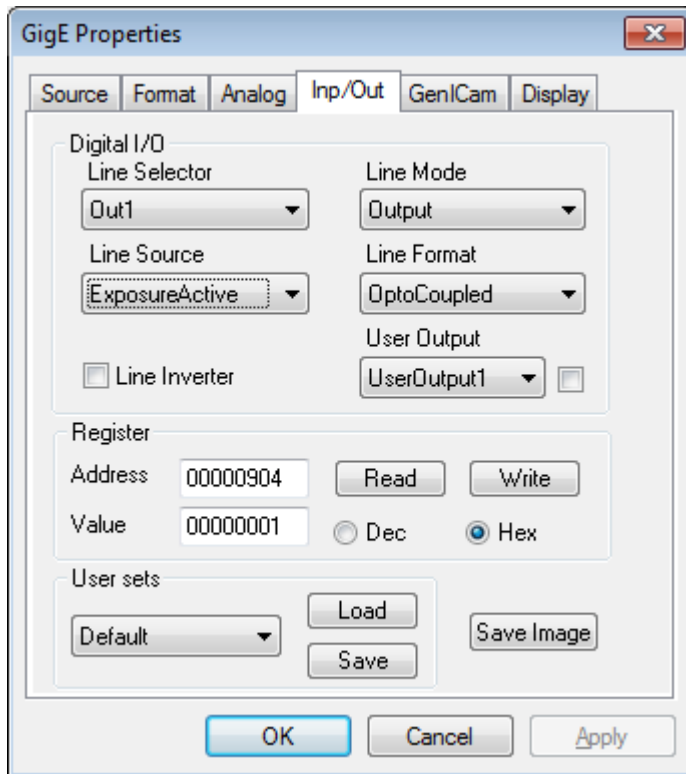
### White Balance Ratio

Use the slider and spin controls to adjust the ratio (amplification factor) of the selected color component. You can also enter the desired balance ratio in the corresponding text box. Note that this option is available only for the cameras that support the manual balance ratio control. Equivalent to the [BalanceRatioAbs](#) property.

---

### 3.4.4 Inp/Out

This property page is used to select the properties specifying input/output operations.



Select from the following options:

#### Line Selector

Lets you select the physical line (or pin) of the external device connector to configure. Equivalent to the [LineSelector](#) property.

#### Line Mode

Lets you select the input or output mode for the currently selected line. Equivalent to the [LineMode](#) property.

#### Line Source

Lets you select the source of the signal to output on the selected line when the line mode is *Output*. Equivalent to the [LineSource](#) property.

#### Line Format

Lets you select the electrical format (TTL, LVDS, OptoCoupled etc) of the selected line. Equivalent to the [LineFormat](#) property.

#### Line Inverter

Select this option to have the electrical signal on the selected line inverted. Equivalent to the [LineInverter](#) property.

#### User Output

Lets you configure the bits of the User Output register. Checking/unchecking the box will set the selected bit to the High or Low state respectively. Equivalent to the [UserOutputSelector](#) and [UserOutputValue](#) properties.

**Register**

Lets you perform reads and writes to a selected register in the camera bootstrap address space. To perform the read operation, enter the desired hexadecimal address to the **Address** field and click the **Read** button. The result will be displayed in the **Value** box. Depending on the **Dec/Hex** radio boxes, the result will be displayed either in decimal or hexadecimal form. To perform the write operation, enter a desired address and value to the **Address** and **Value** fields respectively, and then press the **Write** button. Equivalent to the [ReadRegister](#), [WriteRegister](#) methods.

**User set**

Lets you load or store camera settings under the specified user set. Use the list box to select the desired user set. Use the Save button to store the current camera settings in the selected set. Use the Load button to load the setting from the selected set into the camera. See [UserSetSelector](#) for details.

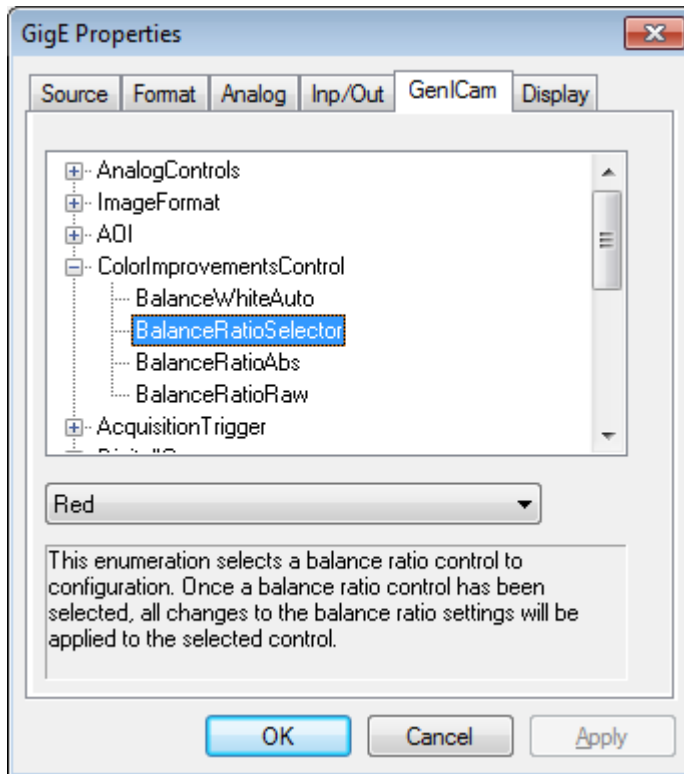
**Save image**

Lets you save the current frame buffer in an image file. When you click this button, the **Save As** dialog box will appear where you can select the file name and one of the image file formats: BMP, TIF and JPEG. Note that BMP and TIF files will be recorded with no compression while JPEG files will be recorded with quality 75. Equivalent to the [SaveImage](#) method.

---

### 3.4.5 GenICam

This property page is used to access all the camera features per GenICam standard.



Select from the following options:

#### GenICam Tree

Shows all the camera features sorted by categories as reported by the camera's XML file. GigE Vision™ standard, which is a subset of the GenICam standard, uses XML files to expose all the commands and features available for a camera. To access a specific feature, browse through the categories and highlight the desired feature.

#### Feature Control

Depending on the type of the feature highlighted in the tree, the following options will become available:

Feature type	GUI control	Action
Integer	Text box + spin control	Adjust the feature value or enter the desired value in the text box
Float	Text box + spin control	Adjust the feature value or enter the desired value in the text box
Boolean	List box	Select "On" or "Off" value
Enumerated	List box	Select among available feature values
Command	Push button	Click the button to execute the command

If the currently selected feature is read-only, the corresponding control will be disabled. If the feature is currently unavailable, the text box will display "Unavailable".

**Feature Description**

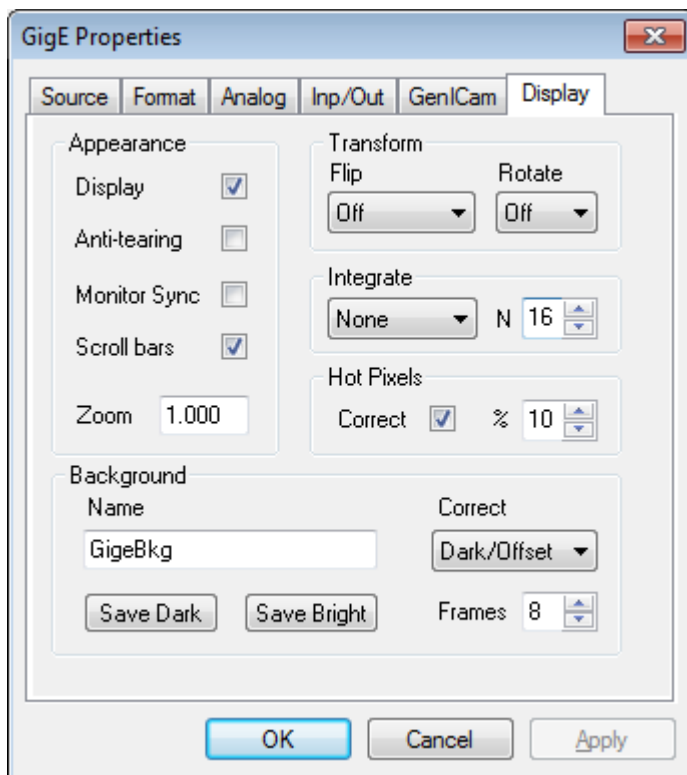
Shows the description of the currently selected feature. The availability of the feature description depends on the camera XML implementation.

---



### 3.4.6 Display

This property page is used to select the properties specifying the display settings of *ActiveGige*.



Select from the following options:

#### Display

Lets you enable or disable live display in the control window. You might want to disable the display option if you want to render captured frames by other means such as [Picture box](#). When using [DirectShow Video Capture Filter](#), setting this property to TRUE will activate the internal RGB24 conversion which is required by image data access and image analysis methods. Equivalent to the [Display](#) property.

#### Anti-tearing

Lets you enable or disable the anti-tearing feature. Anti-tearing removes horizontal tears in the live video caused by a difference between the camera frame rate and refresh rate of the monitor. Equivalent to the [AntiTearing](#) property.

#### Monitor Sync

Lets you enable or disable the monitor synchronization mode. If this box is checked, the camera frame rate will exactly match the refresh rate of the system monitor thus eliminating all the display artifacts related to the digital image transfer. Equivalent to the [MonitorSync](#) property.

#### Scroll bars

Lets you enable or disable the scroll bars in the control window. If this box is checked and the video width or/and height exceed the size of the control window, the scroll bar(s) will be displayed on the border of the control window allowing you to pan the live video. Equivalent to the [ScrollBars](#) property.

#### Digital zoom

Lets you adjust the magnification of the live video display. This option doesn't change the content of the image data, but only its appearance in the control window. If it is set to zero, the image will be fit to the size of the control window. In this case the display might not retain the original proportions of the video frame. Equivalent to the [Magnification](#) property. *Note - if the Display option is unchecked and Digital zoom is set to zero, ActiveGigE will enter a raw image transfer mode with the minimal CPU usage.*

**Flip**

Lets you select one of the flipping modes. Flipping affects the live video display as well as actual order of pixels in the frame buffer. Select among the following modes:

*Off*

No image flipping is performed.

*Horizontal*

The image is flipped horizontally.

*Vertical*

The image is flipped vertically.

*Diagonal*

The image is flipped horizontally and vertically.

This option is equivalent to the [Flip](#) property.

**Rotate**

Lets you rotate the image. Rotation affects the live video display as well as actual order of pixels in the frame buffer. Select one of the following modes:

*Off*

No image rotation is performed.

*90°*

The image is rotated counterclockwise.

*180°*

The image is rotated 180 degrees.

*270°*

The image is rotated clockwise.

This option is equivalent to the [Rotate](#) property.

**Integrate Mode**

Lets you select the frame integration operation mode. The frame integration allows you to average or add frames "on the fly" without sacrificing the frame rate. Equivalent to the [Integrate](#) property. Select one of the following modes:

*None*

Frame integration is disabled.

*Average*

Running Average mode. Each output frame is the result of averaging a selected number of previously captured frames.

*Add*

Running Accumulation mode. Each output frame is the sum of a selected number of previously captured frames.

**Integrate Window (N)**

Sets the number of frames for the integration. Equivalent to the [IntegrateWnd](#) property.

**Hot Pixel Correct**

Lets you enable or disable the hot pixel correction mode. If this box is checked, unusually bright

---

pixels will be effectively removed from the image. Hot pixels are associated with elements on a camera sensor that have higher than normal rates of charge leakage. For more details on hot pixel correction refer to [HotPixelCorrect](#).

**Hot Pixel Level (%)**

Sets the hot pixel correction level, in percent. Equivalent to the [HotPixelLevel](#) property.

**Background name**

Lets you enter the name prefix under which the background files are stored. See [BkgName](#) for more details.

**Save Dark**

Click this button to store a dark field background image on the hard drive. Dark field should be saved when no light transmitted through the camera lens. The dark field will be calculated by averaging the number of consecutive frames specified by the **Frames** option. Equivalent to the [SaveBkg](#) method.

**Save Bright**

Click this button to store a bright field background image on the hard drive. Bright field should be saved with the maximum light transmitted and no objects in the field of view. To achieve the best dynamic range, the light intensity should be adjusted so that it stays just below the saturation level of the camera. To control the saturation for monochrome cameras, use the **Saturated Palette**. The bright field will be calculated by averaging the number of consecutive frames specified by the **Frames** option.

**Correct**

Lets you select a background correction mode. Select *None* if you do not want the background correction to be performed. Select *Dark/Offset* to apply the dark-field background correction to each frame captured. Select *Flat/Gain* to apply the flat-field background correction to each frame captured. Make sure to save the bright and dark fields before using this option, otherwise certain background correction modes will not be available. For more details on background correction refer to [BkgCorrect](#).

## 4 DirectShow

*ActiveGige SDK* includes a *DirectShow Video Capture Filter* which allows Windows users to view/capture images from 1394 cameras and control camera parameters in DirectShow-compliant applications such as *AmCap*, *VirtualDub*, *Video Capturix* and others. The filter is automatically installed on your system during *ActiveGige* [Installation](#).

The filter has an output *Capture Pin* which can be connected to other DirectShow filters or graph. The pin supports all video modes provided by GigE Vision compliant cameras, with high-depth pixel modes automatically converted to 8-bit monochrome or 24-bit RGB format.

*ActiveGige Video Capture Filter* provides a programmatic access to the video acquisition and camera properties through a set of standard and custom DirectShow interfaces. For more information on DirectShow programming refer to [DirectShow Quick Reference Guide](#) and [DirectShow Interfaces](#).

---

## 4.1 Quick Reference Guide

### About DirectShow

Microsoft® DirectShow® application programming interface (API) is a media-streaming architecture for the Microsoft Windows® platform. It is embedded into the set of DirectX APIs. The purpose of this API is to capture, render and playback streaming media as Video or Audio streams. It manages both devices or files for capturing and rendering.

### Filters and Filter Graphs:

The building block of DirectShow is a software component called a *filter*. A filter is a software component that performs some operation on a multimedia stream. For example, DirectShow filters can

- read files
- get video from a video capture device
- decode various stream formats, such as MPEG-1 video
- pass data to the graphics or sound card

Filters receive input and produce output through their pin(s). *ActiveGige Video Capture Filter* provides a DirectShow interface to GigE Vision compliant cameras. It has one output Video Capture pin which can be connected to other filters. The pin supports all video modes provided by a camera, with high-depth pixel modes automatically converted to 8-bit monochrome or 24-bit RGB format.

One or more of the pins may be connected in a chain, so that the output from one filter becomes the input for another. A set of connected filters is called a *filter graph*.

Filters expose various interfaces and media type. Interfaces are a filter's set of method for a specific task, for example the CaptureGraphBuilder is an interface of GraphBuilder filter. Media type identifies what kind of data the upstream filter will deliver to the downstream filter, and the physical layout of the data.

Depending on the camera video mode, *ActiveGige Video Capture Filter* delivers the following media types:

If *Display* property of *IActiveGige* interface is set to FALSE:

Mono 8:	MEDIASUBTYPE_RGB8
Mono 16:	MEDIASUBTYPE_RGB8
RGB 24:	MEDIASUBTYPE_RGB24
Mono 8 debayered:	MEDIASUBTYPE_RGB24
RGB 48:	MEDIASUBTYPE_RGB24
Mono 16 debayered:	MEDIASUBTYPE_RGB24
YUV422:	MEDIASUBTYPE_UYVU
YUV411:	MEDIASUBTYPE_Y411
YUV444:	MEDIASUBTYPE_AYUV

If *Display* property of *IActiveGige* interface is set to TRUE:

For all video modes      MEDIASUBTYPE\_RGB24

### Writing a DirectShow Application:

A limited set of DirectShow interfaces are compatible with Visual Basic, but primarily DirectShow is a C/C++ COM interface. In order to build applications that use *ActiveGige Video Source Filter*, you will

need to install DirectX SDK and have a general understanding of COM client programming. A good starting point is Microsoft's Amcap sample application included in the DirectX SDK.

There are several tasks that any DirectShow application must perform.

- The application creates an instance of the Filter Graph Manager.
- The application uses the Filter Graph Manager to build a filter graph. The exact set of filters in the graph will depend on the application.
- The application uses the Filter Graph Manager to control the filter graph and stream data through the filters. Throughout this process, the application will also respond to events from the Filter Graph Manager.
- The application queries the interfaces exposed by the filters and uses their methods to control the properties of the filters.
- When processing is completed, the application releases the Filter Graph Manager and all of the filters.

Note that *ActiveGigE Video Source Capture Filter* provides two alternative ways of working with multiple cameras. By default, ActiveGigE installs one DirectShow device called "GigE Vision compliant camera". You can select a specific camera via the [Source](#) property page or through the [ActiveGigE](#) interface. Alternatively, you can use the [FilterConfig](#) utility to register several GigE Vision devices in the system, each one associated with a specific camera. Refer to [Working with multiple cameras](#) for more details.

For more information refer to the following topics:

[Retrieving the Filter](#)  
[Building the Graph](#)  
[Displaying the Preview](#)  
[Capturing to AVI](#)  
[Getting the Image Data](#)  
[Displaying Property Pages](#)

---

### 4.1.1 FilterConfig utility

Provided with *ActiveGige Video Capture Source Filter* is FilterConfig console utility located in the Driver subfolder of *ActiveGigE* folder (typically C:/Program Files/ActiveGigE/Driver). Depending on the platform on your DirectShow application, you should use either FilterConfig.exe or FilterConfig64.exe. The utility allows you to register several DirectShow devices when multiple GigE Vision cameras are used.

By default ActiveGigE registers a single DirectShow device called "*GigE Vision compliant camera*", in which case the selection of a specific camera is available via the [Source](#) property page or through the [IActiveGige](#) interface. You can run several instances of this device in your system or in your application provided each instance is configured for a different camera. If you execute several copies of a DirectShow-based video capture application such as Microsoft's Amcap, each of them will attempt to automatically configure itself for a different camera and memorize corresponding camera settings in the system registry upon exiting. The same automatic configuration routine will apply to multiple instances of the "*GigE Vision compliant camera*" device running in one application.

In certain cases however you may want to have each camera associated with a separate DirectShow device in the system. This may be necessary when a third-party DirectShow application has to be used with several GigE Vision cameras. To register several GigE Vision devices in the system, run the FilterConfig executable with a numerical argument indicating the amount of GigE Vision cameras in the system. For example, the following command will register three DirectShow devices "*GigE Vision Device 1*", "*GigE Vision Device 2*", "*GigE Vision Device 3*":

```
C:\Program Files\ActiveGigE\Driver> FilterConfig.exe 3
```

You can change the number of GigE Vision DirectShow devices listed in the system by calling FilterConfig with a different parameter. To reset the filter to a default single "GigE Vision compliant camera" device, run FilterConfig with the zero argument:

```
C:\Program Files\ActiveGigE\Driver> FilterConfig.exe 0
```

Note: Before using FilterConfig to register multiple devices, you must enumerate your cameras by opening several instances of Microsoft's AmCap application. Each instance will automatically connect to the next available camera and will memorize its device index upon exiting. Alternatively the GcamCap application (located in the ActiveGigE/Bin folder) can be used for the same purpose.

### 4.1.2 Retrieving the Filter

To instantiate *ActiveGige Video Capture Filter*, you need to enumerate all video capture filters using the system device enumerator and match the filter name with "GigE Vision compliant camera":

```

IBaseFilter* pFilter;
HRESULT hr;

//locate the camera filter using system device enumerator
CComPtr< ICreateDevEnum > pCreateDevEnum;
pCreateDevEnum.CoCreateInstance( CLSID_SystemDeviceEnum );
if( !pCreateDevEnum )
return E_FAIL;
// enumerate video capture devices
CComPtr< IEnumMoniker > pEm;
pCreateDevEnum->CreateClassEnumerator( CLSID_VideoInputDeviceCategory, &pEm, 0 );
if( !pEm )
return E_FAIL;

pCreateDevEnum.Release();
pEm->Reset();
int noCapDevice=0;

while(true)
{
    ULONG ulFetched = 0;
    CComPtr< IMoniker > pM;
    hr = pEm->Next( 1, &pM, &ulFetched );
    if( hr != S_OK )
        break;
    //get the property bag interface from the moniker
    CComPtr< IPropertyBag > pBag;
    hr = pM->BindToStorage( 0, 0, IID_IPropertyBag, (void**)&pBag );
    if( hr != S_OK )
    {
        pBag.Release();
        continue;
    }

    //retrieve the name of the device
    CComVariant var;
    var.vt = VT_BSTR;
    hr = pBag->Read(L"FriendlyName", &var, NULL );
    if( hr != S_OK )
    {
        SysFreeString(var.bstrVal);
        pBag.Release();
        continue;
    }

    //is this ActiveGige filter?
    if( !memcmp( W2CA(var.bstrVal), "GigE Vision compliant camera", 26 ) )
    {
        hr = pM->BindToObject( 0, 0, IID_IBaseFilter, (void**)pFilter );
    }
    noCapDevice++;
    pM.Release();
}

```



---

```
pBag.Release();
SysFreeString( var.bstrVal );
}
pEm.Release();
```

### 4.1.3 Building the Graph

DirectShow filters run in the context of the filter graph. The graph manages connections between filters from a source, such as a video capture filter, to a renderer, such as a video window, and any transformation filters in between. DirectShow provides **ICaptureGraphBuilder** and **ICaptureGraphBuilder2** interfaces containing methods for building and controlling a capture graph.

```
CComPtr< IGraphBuilder > pGraph;  
CComPtr< ICaptureGraphBuilder2 > pBuilder;  
HRESULT hr = S_OK;  
  
//create filter graph  
hr = pGraph.CoCreateInstance( CLSID_FilterGraph );  
if( FAILED(hr) )  
{  
    Error( "Failed to create a filter graph." );  
    return FALSE;  
}  
  
//create a capture graph builder  
hr = pBuilder.CoCreateInstance( CLSID_CaptureGraphBuilder2 );  
if( FAILED(hr) )  
{  
    Error( "Failed to create a capture graph builder." );  
    return FALSE;  
}  
  
//add ActiveGige filter to the graph  
hr = pGraph->AddFilter(pFilter, L"GigE Vision camera" );  
if( FAILED(hr) )  
{  
    Error( "Failed to add the camera to graph." );  
    return FALSE;  
}  
  
//specify the filter graph to the graph builder  
hr = pBuilder->SetFiltergraph (pGraph);
```

---

#### 4.1.4 Displaying the Preview

To build a video preview graph, call the *RenderStream* method of **ICaptureGraphBuilder2** interface.

The first parameter to the *RenderStream* method specifies a pin category; for a preview graph use `PIN_CATEGORY_PREVIEW`. The second parameter specifies a media type, as a major type GUID. For video, use `MEDIATYPE_Video`. The third parameter is a pointer to the capture filter's `IBaseFilter` interface. The next two parameters are not needed in this example. They are used to specify additional filters that might be needed to render the stream.

Setting the last parameter to `NULL` causes the Capture Graph Builder to select a default renderer for the stream, based on the media type. For video, the Capture Graph Builder always uses the Video Renderer filter as the default renderer.

```
hr = pBuilder->RenderStream(&PIN_CATEGORY_PREVIEW, &MEDIATYPE_Video,  
pFilter, NULL, NULL);
```

To start/stop the graph, use the **IMediaControl** interface:

```
CComQIPtr <IMediaControl, &IID_IMediaControl> control = pGraph;
```

```
hr = control->Run();    // start the graph  
////  
//// .....  
hr=control->Stop();    //stop the graph
```

### 4.1.5 Capturing to AVI

To capture the video stream to an AVI file use the AVI MUX filter as follows.

```
IBaseFilter *pMux;
```

```
//specify the output media type, file name and get the resulting MUX  
hr = pBuild->SetOutputFileName(&MEDIASUBTYPE_Avi, L"C:\\test.avi",  
    &pMux, NULL);
```

```
// Render the capture pin to the multiplexer  
hr = pBuild->RenderStream(&PIN_CATEGORY_CAPTURE, &MEDIATYPE_Video,  
    pCap, NULL, pMux);
```

```
// Release the mux filter.  
pMux->Release();
```

You can encode the video stream by inserting an encoder filter between the capture filter and the AVI Mux filter. Use the System Device Enumerator or the Filter Mapper to select an encoder filter. Specify the encoder filter as the fourth parameter to *RenderStream*.

If you want to capture a video while you are previewing the video call the *RenderStream* method successively. The capture graph builder will automatically add a Smart Tee to split the capture stream from the preview stream.

### 4.1.6 Getting the Image Data

The Sample Grabber filter provides a way to retrieve samples as they pass through the filter graph. It is a transform filter with one input pin and one output pin. It passes all samples downstream unchanged, so you can insert it into a filter graph without altering the data stream. Your application can then retrieve individual samples from the filter by calling methods on the **ISampleGrabber** interface. If you want to retrieve samples without rendering the data, connect the Sample Grabber filter to the Null Renderer filter.

Before building the rest of the graph, you should set a media type for the Sample Grabber by calling the *SetMediaType* method. When the Sample Grabber connects, it will compare this media type against the media type offered by the other filter.

If you want to discard the samples after you are done with them connect the Sample Grabber to the Null Renderer Filter. The sample grabber operates either in buffering mode (makes a copy of each sample before delivering the sample downstream) or in callback mode (invokes an application-defined callback function on each sample).

*// Create a Sample Grabber.*

```
IBaseFilter *pGrabberF = NULL;
hr = CoCreateInstance(CLSID_SampleGrabber, NULL, CLSCTX_INPROC_SERVER,
    IID_IBaseFilter, (void**)&pGrabberF);
if (FAILED(hr))
{
    Error( "Failed to create a grabber" );
    return FALSE;
}
hr = pGraph->AddFilter(pGrabberF, L"Sample Grabber");
if (FAILED(hr))
{
    Error( "Failed to add the grabber to graph" );
    return FALSE;
}
```

*// Query the Sample Grabber for the ISampleGrabber interface.*

```
ISampleGrabber *pGrabber;
pGrabberF->QueryInterface(IID_ISampleGrabber, (void**)&pGrabber);
```

*//specify RGB24 uncompressed video*

```
AM_MEDIA_TYPE mt;
ZeroMemory(&mt, sizeof(AM_MEDIA_TYPE));
mt.majorType = MEDIATYPE_Video;
mt.subtype = MEDIASUBTYPE_RGB24;
hr = pGrabber->SetMediaType(&mt);
```

*// Activate buffering mode*

```
hr = pGrabber->SetBufferSamples(TRUE);
```

*// Run the graph.*

```
pControl->Run();
pEvent->WaitForCompletion(INFINITE, &evCode); // Wait till it's done.
```

*// Find the required buffer size.*

```
long cbBuffer = 0;
hr = pGrabber->GetCurrentBuffer(&cbBuffer, NULL);
```

*// Allocate the array and call the method a second time to copy the buffer:*

```
char *pBuffer = new char[cbBuffer];  
//get the image data  
hr = pGrabber->GetCurrentBuffer(&cbBuffer, (long*)pBuffer);
```

Note that images DirectShow are converted into standard video types (RGB8, RGB24, YUV411, YUV422, YUV444) . If your work with high-bit depth video formats and want to access original pixel values, use the [GetImageData](#) or [GetImagePointer](#) methods of [IActiveGige](#) interface.

### 4.1.7 Displaying Property Pages

*ActiveGige* Video Capture Filter provides a pass-through access to *ActiveGige* [Property Pages](#). The filter and its output pin expose [ISpecifyPropertyPages](#) interface for retrieving a list of CLSIDs which may then be passed to the *OLECreatePropertyFrame* API function for display and control. The following code shows how to display the filter's property pages:

```
//get the property page interface
ISpecifyPropertyPages *pProp;
HRESULT hr = pFilter->QueryInterface(IID_ISpecifyPropertyPages, (void **)&pProp);

//get the filter's name and Unknown pointer
if (SUCCEEDED(hr))
{
    IUnknown *pFilterUnk;
    pFilter->QueryInterface(IID_IUnknown, (void **)&pFilterUnk);

    //show the property pages
    CAUUID caGUID;
    pProp->GetPages(&caGUID);
    pProp->Release();
    OleCreatePropertyFrame(
        hwnd,                      // parent window handle
        0, 0, NULL,                // caption for the dialog box
        1,                         // number of objects
        &pFilterUnk,                // array of object pointers
        caGUID.cElems,             // number of property pages
        caGUID.pElems,             // array of property pages CLSIDs
        0, 0, NULL
    );

    //release objects
    pFilterUnk->Release();
    CoTaskMemFree(caGUID.pElems);
}
```

For an example on how to display the pin's property pages refer to [ISpecifyPropertyPages](#).

The filter's property pages provides an access to those camera properties that can be modified while the graph is running, while the pin's property pages control the camera settings that cannot be accepted dynamically such as video mode and format change. Therefore you must stop the graph before displaying the pin's property pages and rebuild it afterwards.

## 4.2 Interfaces

The *ActiveGige Video Capture Filter* provides a programmatic access to video acquisition and camera properties via the following COM-interfaces:

### Video Capture Filter

<a href="#"><u>IAMCameraControl</u></a>	Provides programmatic control over shutter speed, iris, pan, tilt, zoom, focus, optical filter
<a href="#"><u>IAMVideoProcAmp</u></a>	Provides programmatic control over brightness, gain, gamma, sharpness, auto exposure, hue, saturation, white balance
<a href="#"><u>IAMVideoControl</u></a>	Provides programmatic control over image flipping and camera triggering
<a href="#"><u>IActiveGige</u></a>	Provides a pass-through interface to <i>ActiveGige</i> COM object
<a href="#"><u>ISpecifyPropertyPages</u></a>	Returns a list of <i>ActiveGige</i> property pages supported by the Filter

### Video Capture Pin

<a href="#"><u>IAMStreamConfig</u></a>	Provides an ability to retrieve and set video modes
<a href="#"><u>ISpecifyPropertyPages</u></a>	Provides a list of <i>ActiveGige</i> property pages supported by the Pin
<a href="#"><u>IAMVideoCompression</u></a>	Provides the name and serial number of a camera
<a href="#"><u>IAMDroppedFrames</u></a>	Provides information about dropped and non-dropped frames



## 4.2.1 IAMCameraControl

### Description

Provides programmatic control over the camera exposure.

### Methods

**HRESULT Get ( long *Property*, long *\*pValue*, long *\*Flags* )**

Retrieves the value of a specific camera control property.

**HRESULT Set ( long *Property*, long *lValue*, long *Flags* );**

Sets the value of a specific camera control property.

**HRESULT GetRange ( long *Property*, long *\*pMin*, long *\*pMax*, long *\*pSteppingDelta*, long *\*pDefault*, long *\*pFlags* );**

Retrieves values associated with the range of a specified camera property.

### Parameters

*Property*

[in] Value that specifies the camera property. Use the following value:  
*CameraControl\_Exposure* - to control the camera's shutter.

*pValue / lValue*

[in/out] New value or pointer to the value of the specified property.

*pValue*

[in] New value of the specified property.

*pMin*

[out] Pointer to the minimum range for the specified property.

*pMax*

[out] Pointer to the maximum range for the specified property.

*pSteppingDelta*

[out] Pointer to the step size for the specified property.

*pDefault*

[out] Pointer to the default value of the specified property.

*Flags / pFlags*

[in/out] Value or pointer to the value indicating whether the camera property is automatic or manual:

*CameraControl\_Flags\_Auto* - sets the property's automatic flag.

*CameraControl\_Flags\_Manual* - sets the property's manual flag.

### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid argument.

### Example

This C++ code request a pointer to **IAMCameraControl** interface from the video capture filter and sets the shutter value to 512:

```
IAMCameraControl *pCameraControl;  
HRESULT hr;  
hr = pFilter->QueryInterface(IID_IAMCameraControl, (void **)&pCameraControl);  
if(hr==S_OK)  
{  
    hr=pCameraControl->Set(CameraControl_Exposure,512,Camera Control_Flags_Manual);  
}
```

**Remarks**

**IAMCameraControl** is a standard DirectShow interface. Refer to *Microsoft DirectX SDK* documentation for more details.

---

## 4.2.2 IAMVideoProcAmp

### Description

Provides programmatic control over the following camera properties: black level, gain, gamma, white balance.

### Methods

**HRESULT Get ( long *Property*, long *\*pValue*, long *\*Flags* )**

Retrieves the value of a specific camera control property.

**HRESULT Set ( long *Property*, long *IValue*, long *Flags* );**

Sets the value of a specific camera control property.

**HRESULT GetRange ( long *Property*, long *\*pMin*, long *\*pMax*, long *\*pSteppingDelta*, long *\*pDefault*, long *\*pFlags* );**

Retrieves values associated with the range of a specified camera property.

### Parameters

*Property*

[in] Value that specifies the camera property. Use one of the following values:

*VideoProcAmp\_Brightness* - to set and retrieve the camera's black level setting.

*VideoProcAmp\_Gain* - to control the camera's gain.

*VideoProcAmp\_Gamma* - to control the camera's gamma setting.

*VideoProcAmp\_WhiteBalance* - to control the camera's white balance.

*pValue / IValue*

[in/out] New value or pointer to the value of the specified property.

*pValue*

[in] New value of the specified property.

*pMin*

[out] Pointer to the minimum range for the specified property.

*pMax*

[out] Pointer to the maximum range for the specified property.

*pSteppingDelta*

[out] Pointer to the step size for the specified property.

*pDefault*

[out] Pointer to the default value of the specified property.

*Flags / pFlags*

[in/out] Value or pointer to the value indicating whether the camera property is automatic or manual:

*VideoProcAmp\_Flags\_Auto* - sets the property's automatic flag.

*VideoProcAmp\_Flags\_Manual* - sets the property's manual flag.

### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid argument.

**Example**

This C++ code request a pointer to **IAMVideoProcAmp** interface from the video capture filter, initiates the automatic gain control and sets the black level value to 64:

```
IAMVideoProcAmp *pVideoProcAmp;  
HRESULT hr;  
hr = pFilter->QueryInterface(IID_IAMVideoProcAmp, (void **)&pVideoProcAmp);  
if(hr==S_OK)  
{  
    hr=pVideoProcAmp->Set(VideoProcAmp_Gain,0,VideoProcAmp_Flags_Auto);  
    hr=pVideoProcAmp->Set(VideoProcAmp_Brightness,64,VideoProcAmp_Flags_Manual);  
}
```

**Remarks**

**IAMVideoProcAmp** is a standard DirectShow interface. Refer to *Microsoft DirectX SDK* documentation for more details.

---

### 4.2.3 IAMVideoControl

#### Description

Provides an ability to flip a picture horizontally and/or vertically, set up a trigger mode, issue the software trigger, and list the available frame rates.

#### Methods

**HRESULT GetMode ( IPin \*pPin, long \*pFlags )**

Retrieves the video control properties.

**HRESULT SetMode ( IPin \*pPin, long Flags )**

Sets the video control properties.

**HRESULT GetCaps ( IPin \*pPin, long \*pFlags )**

Requests availability of the video control properties.

**HRESULT GetCurrentActualFrameRate ( IPin \*pPin, LONGLONG \*pFrameRate )**

Retrieves the actual frame rate, expressed as a frame duration in 100-nanosecond units.

**HRESULT GetFrameRateList ( IPin \*pPin, long iIndex, SIZE Dimensions, long \*ListSize, LONGLONG \*\*FrameRates )**

Retrieves a list of available frame rates for the specified video mode.

**HRESULT GetMaxAvailableFrameRate( IPin \*pPin, long iIndex, SIZE Dimensions, LONGLONG \*pFrameRate )**

Retrieves the maximum frame rate available for the specified video mode.

#### Parameters

*pPin*

[in] Pointer to the video capture pin.

*pFlags / Flags*

[in/out] New value or pointer to the value specifying a combination of the video control flags:

*VideoControlFlag\_FlipHorizontal* - specifies that the picture is flipped horizontally.

*VideoControlFlag\_FlipVertical* - specifies that the picture is flipped vertically.

*VideoControlFlag\_ExternalTriggerEnable* - specifies that the camera is set to the trigger mode.

*VideoControlFlag\_Trigger* - issues a software trigger signal.

*pFrameRate*

[in/out] Pointer to the frame rate. The frame rate is expressed as frame duration in 100-nanosecond units.

*iIndex*

[in] Index of the video mode to query for the frame rates.

*Dimensions*

[in] Frame image size (width and height) in pixels

*ListSize*

[out] Pointer to the number of elements in the list of frame rates.

*FrameRates*

[in] Address of a pointer to an array of frame rates in 100-nanosecond units.

#### Return Values

S\_OK

Success  
E\_FAIL  
Failure.  
E\_INVALIDARG  
Invalid argument.

### Example

This C++ code request a pointer to **IAMVideoControl** interface from the video capture filter, and sets the camera into the trigger mode with the horizontal image flipping:

```
IAMVideoProcAmp *pVideoControl;  
HRESULT hr;  
hr = pFilter->QueryInterface(IID_IAMVideoControl, (void **)&pVideoControl);  
if(hr==S_OK)  
{  
    hr=pVideoControl->Set(VideoProcAmp_Gain,0,VideoProcAmp_Flags_Auto);  
    hr=pVideoControl->Set(VideoProcAmp_Brightness,64,VideoProcAmp_Flags_Manual);  
}
```

### Remarks

**IAMVideoControl** is a standard DirectShow interface. Refer to *Microsoft DirectX SDK* documentation for more details.

---

## 4.2.4 IActiveGige

### Description

Provides a pass-through access to an *ActiveGige* COM object associated with the video capture filter.

### Methods

For a detailed description of methods refer to [Properties](#) and [Methods](#) in the *ActiveX Reference* chapter.

### Example

This C++ code request a pointer to **IActiveGige** interface from the video capture filter and sets up a bilinear Bayer interpolation:

```
IActiveGige *pActiveGige;  
HRESULT hr;  
hr = pFilter->QueryInterface(IID_IActiveGige, (void **)&pCameraControl);  
if(hr==S_OK)  
{  
    hr=pActiveGige->put_Bayer(2);  
}
```

### Remarks

Use **IActiveGige** interface to access the camera properties and methods not available via standard DirectShow interfaces. Since DirectShow does not support 16 bits per channel media types, you might want to use *IActiveGige*'s image access methods for retrieving high-depth pixel data.

Note that for YUV video modes the [Display](#) property must be set to TRUE in order to use image access and image analysis functions such as [GetImageData](#), [GetComponentData](#), [GetImageLine](#), [GetImageWindow](#), [GetHistogram](#), [GetImageStat](#) etc. These functions require internal RGB conversion which by default is turned off by DirectShow filter to reduce the CPU load.

Refer to [ActiveX Reference](#) chapter for more details.

## 4.2.5 IAMStreamConfig

### Description

Provides an ability to set stream formats and to find out what types of formats the Capture Pin can be connected to.

### Methods

**HRESULT GetFormat( AM\_MEDIA\_TYPE \*\*pmt )**

Retrieves the video stream's format.

**HRESULT SetFormat( AM\_MEDIA\_TYPE \*pmt )**

Sets the video stream's format.

**HRESULT GetNumberOfCapabilities( int \*piCount, int \*piSize )**

Retrieves the number of stream capabilities (video modes) available for the current camera.

**HRESULT GetStreamCaps( int iIndex, AM\_MEDIA\_TYPE \*\*pmt, BYTE \*pSCC )**

Obtains video capabilities of a stream.

### Parameters

*pmt*  
[in/out] Pointer to a AM\_MEDIA\_TYPE structure.

*piCount*  
[out] Pointer to the number of VIDEO\_STREAM\_CONFIG\_CAPS (video modes) supported.

*piSize*  
[out] Pointer to the size of the VIDEO\_STREAM\_CONFIG\_CAPS structure.

*iIndex*  
[in] Index to the desired media type and capability pair.

*pSCC*  
[out] Pointer to a stream configuration structure. The structure is defined as follows:

```
typedef struct _VIDEO_STREAM_CONFIG_CAPS
{
    GUID guid;                                // set to MEDIATYPE_Video
    ULONG VideoStandard;                       // set to AnalogVideo_None
    SIZE InputSize;                           // maximum image size
    SIZE MinCroppingSize;                     // minimum ROI size (Format 7)
    SIZE MaxCroppingSize;                     // maximum ROI size (Format 7)
    int CropGranularityX;                     // horizontal size granularity (Format 7)
    int CropGranularityY;                     // vertical size granularity (Format 7)
    int CropAlignX;                           // horizontal offset granularity (Format 7)
    int CropAlignY;                           // vertical offset granularity (Format 7)
    SIZE MinOutputSize;                       // same as MinCroppingSize
    SIZE MaxOutputSize;                       // same as MaxCroppingSize
    int OutputGranularityX;                   // set to zero
    int OutputGranularityY;                   // set to zero
    int StretchTapsX;                         // set to zero
    int StretchTapsY;                         // set to zero
    int ShrinkTapsX;                         // set to zero
    int ShrinkTapsY;                         // set to zero
    LONGLONG MinFrameInterval;                // minimum frame interval in 100 nanoseconds
    LONGLONG MaxFrameInterval;                // maximum frame interval in 100 nanoseconds
    LONG MinBitsPerSecond;                   // minimum bandwidth
    LONG MaxBitsPerSecond;                   // maximum bandwidth
} VIDEO_STREAM_CONFIG_CAPS;
```



## Return Values

S\_OK  
Success

E\_FAIL  
Failure.

E\_INVALIDARG  
Invalid argument.

## Example

This C++ code request a pointer to **IAMStreamConfig** interface, collects available video modes from the video capture filter, finds a Format 7 mode with a horizontal resolution of 1024, sets ROI to 600x400 and switches the camera to this mode.

```
IAMStreamConfig* pSC;
if( pFilter->QueryInterface( IID_IAMStreamConfig, (void **)&pSC ) == S_OK )
{
    int iCount, iSize;
    VIDEO_STREAM_CONFIG_CAPS caps;
    pSC->GetNumberOfCapabilities(&iCount, &iSize);
    for(int i=0;i<iCount;i++)
    {
        AM_MEDIA_TYPE *pmt = NULL;
        if( pSC->GetStreamCaps(i, &pmt, (BYTE*)&caps) == S_OK )
        {
            if( caps.MaxOutputSize.cx == 1024 && caps.MaxOutputSize.cx!=caps.MinOutputSize)
            {
                ((VIDEOINFOHEADER *)(pmt->pbFormat))->rcSource.left=0;
                ((VIDEOINFOHEADER *)(pmt->pbFormat))->rcSource.top=0;
                ((VIDEOINFOHEADER *)(pmt->pbFormat))->rcSource.right=600;
                ((VIDEOINFOHEADER *)(pmt->pbFormat))->rcSource.bottom=400;
                pConfig->SetFormat(pmt);
            }
            DeleteMediaType (pmt);
            break;
        }
        DeleteMediaType (pmt);
    }
}
```

## Remarks

Use the **GetNumberOfCapabilities** and **GetStreamCaps** methods to collect the information about available video modes. The information in VIDEO\_STREAM\_CONFIG\_CAPS structures is particularly useful for specifying a Region of Interest (ROI) in partial scan modes (Format 7). In this case, *MinCroppingSize* and *MaxCroppingSize* fields define the range of possible ROI sizes, while *CropGranularity* and *CropAlign* define the ROI size and offset granularity. Modify the *rcSource* and *AvgTimePerFrame* fields of the VIDEOINFOHEADER block of a AM\_MEDIA\_TYPE structure in the **SetFormat** method to set up a specific ROI and frame rate. Refer to *Microsoft DirectX SDK* documentation for more details.

## 4.2.6 IAMVideoCompression

### Description

Provides the name and serial number of the currently selected camera.

### Methods

**HRESULT** GetInfo ( *WCHAR \*pszVersion*, *int \*pcbVersion*, *LPWSTR pszDescription*, *int \*pcbDescription*, *NULL*, *NULL*, *NULL*, *NULL* )

### Parameters

*pszVersion*

[out] Pointer to a version string, such as "Version 2.1.0".

*pcbVersion*

[in, out] Size needed for a version string. Pointer to the number of bytes in the Unicode™ string, not the number of characters, so it must be twice the number of characters the string can hold. Call with this set to NULL to retrieve the current size.

*pszDescription*

[out] Pointer to a camera description.

*pcbDescription*

[in, out] Size needed for a description string. Pointer to the number of bytes in the Unicode string, not the number of characters, so it must be twice the number of characters the string can hold. Call with this set to NULL to retrieve the current size.

### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid argument.

### Example

This C++ code request a pointer to **IAMVideoCompression** interface from the video capture filter and retrieves the information about the current camera:

```
IAMVideoCompression *pVideoCompression;
HRESULT hr;
int verSize = VER_SIZE;
int descSize = DESC_SIZE;
WCHAR wachVer[VER_SIZE]={0}, wachDesc[DESC_SIZE]={0};
TCHAR camName[VER_SIZE + DESC_SIZE + 5]={0};
hr = pBuilder->FindInterface(&PIN_CATEGORY_CAPTURE,
    &MEDIATYPE_Video, pFilter,
    IID_IAMVideoCompression, (void **)&pVideoCompression);
if(hr==S_OK)
{
    hr = pVideoCompression->GetInfo(wachVer, &verSize, wachDesc, &descSize, NULL, NULL, NULL, NULL);
    if(hr==S_OK)
        if(wcslen(wachDesc) && wcslen(wachVer))
            wprintf(camName, TEXT("%s - %s\\0"), W2T(wachDesc), W2T(wachVer));
}
```

### Remarks

**IAMVideoCompression** is a standard DirectShow interface. Refer to *Microsoft DirectX SDK* documentation for

more details.

## 4.2.7 IAMDroppedFrames

### Description

Provides information about frames that the filter dropped (that is, did not send), the frame rate achieved, and the data rate achieved.

.

### Methods

**HRESULT GetNumDropped(long \*pDropped)**

Retrieves the total number of frames that the pin dropped since it last started streaming.

**HRESULT GetNumNotDropped( long \*pNotDropped)**

Retrieves the total number of frames that the pin delivered downstream (did not drop).

**HRESULT GetAverageFrameSize( long \*pAverageSize)**

Retrieves the average size of frames that the pin dropped.

### Parameters

*pDropped*

[out] Pointer to the total number of dropped frames.

*pNotDropped*

[out] Pointer to the total number of frames that were not dropped

*pAverageSize*

[out, retval] Pointer to the average size of frames sent out by the pin since the pin started streaming, in bytes.

### Return Values

S\_OK

Success

E\_FAIL

Failure.

E\_INVALIDARG

Invalid argument.

### Example

This C++ code request a pointer to **IAMDroppedFrame** interface from the video capture pin and retrieves the number of dropped and non-dropped frames:

```
IAMDroppedFrames *pDroppedFrames;
HRESULT hr;
long nDropped, nNonDropped;
hr = pBuilder->FindInterface(&PIN_CATEGORY_CAPTURE,
    &MEDIATYPE_Video, pFilter,
    IID_IAMDroppedFrames, (void **)&pDroppedFrames);
if(hr)
{
    pDroppedFrames->GetNumDropped(&nDropped);
    pDroppedFrames->GetNumNotDropped(&nNotDropped);
}
```

### Remarks

**IAMDroppedFrames** is a standard DirectShow interface. Refer to *Microsoft DirectX SDK* documentation for more details.

## 4.2.8 ISpecifyPropertyPages

### Description

Provides a list of *ActiveGige* property pages supported by the Video Capture Filter and Video Capture Pin. Property pages provide a built-in camera control GUI to DirectShow compatible applications.

### Methods

**HRESULT** GetPages( CAUUID \*pPages )

Fills an array of GUID values where each GUID specifies the CLSID of a corresponding *ActiveGige* property page.

### Parameters

*pPages*

[out] Pointer to a caller-allocated **CAUUID** structure that must be initialized and filled before returning. The *pElems* field in the **CAUUID** structure is allocated by the callee with **CoTaskMemAlloc** and freed by the caller with **CoTaskMemFree**.

### Return Values

S\_OK

Success

E\_POINTER

The address in *pPages* is not valid.

### Example

This C++ code displays built-in property pages for the Video Capture pin:

```
ISpecifyPropertyPages *pSpec;
CAUUID cauuid;
IAMStreamConfig *pSC;
hr = pBuilder->FindInterface(&PIN_CATEGORY_CAPTURE, &MEDIATYPE_Video,
    pFilter, IID_IAMStreamConfig, (void **)&pSC);
if (FAILED(hr))
{
    Error( "Capture pin not found" );
    return FALSE;
}
hr = pFilter->QueryInterface(IID_ISpecifyPropertyPages, (void **)&pSpec);
if(hr == S_OK)
{
    hr = pSpec->GetPages(&cauuid);
    hr = OleCreatePropertyFrame(ghwndApp, 30, 30, NULL, 1,
        (IUnknown **)&pSC, cauuid.cElems,
        (GUID *)cauuid.pElems, 0, 0, NULL);
    CoTaskMemFree(cauuid.pElems);
    pSpec->Release();
}
```

**Remarks**

Video Capture Filter supports the [Exposure](#) and [Color](#) property pages. Video Capture Pin supports the [Source](#), [Format](#) and [Advanced](#) property pages. Refer to [ActiveX Reference](#) chapter for more information.

## 5 TWAIN

*ActiveGige* TWAIN driver allows Windows users to view and capture images in TWAIN-compliant graphic editing and scientific imaging applications such as Photoshop, Image Pro and others. Depending on the selected video mode, the TWAIN driver can capture images in 8- or 16-bit monochrome and 24- or 48-bit RGB format.

The driver is automatically installed on your system during *ActiveGige* [Installation](#). To use GigE Vision™ cameras via a TWAIN compliant imaging application (e.g. Photoshop, Image Pro etc.), you must first set **GigE Vision Compliant Camera** as the default TWAIN source.

- Open the TWAIN compliant application (e.g. Photoshop, Image Pro etc.)
  - Specify the TWAIN source that you want to interface with. The procedure for this step will vary depending on the software you are using, but you should see a list of TWAIN devices.
  - Chose **GigE Vision Compliant Camera**.
  - Open the TWAIN interface. This step also varies depending on the software. The GigE Vision TWAIN window will appear as shown below.
-



The user interface of the TWAIN driver includes the following buttons and controls:

**Capture**

Captures the current video frame into the application.

**Live**

Switches the camera into the continuous acquisition mode and initiates live preview.

**Freeze**

Stops the acquisition from the camera and freezes the last video frame in the image window.

**Zoom in**

Click this button to increase the magnification of the image.

**Zoom out**

Click this button to decrease the magnification of the image.

**Fit to screen**

Click this button to change the magnification factor of the image so that it fits to the image window.

**Settings**

Click this button to display ActiveGige [Property Pages](#).

**Format**

Use this option to select the desired pixel format from the list of available formats.

## 6 Samples

The ActiveGige distribution package includes the following sample applications:



Programming language	Project name	Description
Visual Basic 6.0	GcamProfile	Live image preview, real-time line profile, drawing text, rectangles and ellipses
	MultiGcam	Dual camera viewer with custom exposure controls and pixel value display
	VBProcess	Live image processing with direct pixel manipulation in the frame buffer
	GcamStat	Real-time image statistics over an adjustable luminance range
	GcamEnhance	Real-time frame averaging and integration, software gain control, hot pixel correction
	GcamAlpha	Animation effects in the alpha-plane over the live video, full screen mode
	GcamLUT	Selection among several user-defined look-up tables, color matrix correction
	GcamBarcode	1D and 2D barcode decoding
	GcamLensCorrect	Real-time lens distortion correction
	GcamByRef	Live image in PictureBox object, pixel values at cursor coordinates, fps display, using ActiveGige by reference
	GcamChannel1 GcamChannel2	Video streaming on two stream channels to different destinations (multi-channel camera is required)
	GcamAVI (DVR-version)	Video-capture and playback to/from AVI files with the sound recording
	GcamSequence (DVR-version)	Video capture and playback to/from memory sequence
	GcamReplay (DVR-version)	Live video with a delayed replay using memory loop recording
	GcamStreamer (DVR-version)	Live video with simultaneous web streaming to a specified ip address
Visual Basic 6.0, DirectShow	VBCap	Live image preview, built-in property pages. Based on ActiveGige Video Capture Filter and DirectShow programming interface.

Visual C++, MFC	ActiveDemo	Live image preview, real-time line profile, histogram and image statistics over ROI, saving image into file, continuous capture into multiple frames.
Visual C++, Win32 API	GcamConsole	Console application showing how to use ActiveGigE API to retrieve the list of connected cameras and video formats, capture a series of frames and modify camera properties
	GcamWin	Live image preview using DIB rendering, built-in and custom camera controls, saving image into file. Written in C with Win32 API (no MFC used)
Visual C++, DirectShow	GcamCap	Live image preview, built in and custom camera controls. Based on ActiveGigE Video Capture Filter and Microsoft DirectX SDK.
VB.NET	GcamOverlay	Flipped video preview, pseudo-coloring, pixel window extraction, overlay animation
	GcamCapture	Time-lapse video capture to AVI files or series of images, overlaid frame counter
Visual C#	FilterSharp	Direct access to the frame buffer using pointers, real-time Emboss filter, non-destructive color overlay, interactive drawing
	GcamSharp	Two-channel Image viewer with custom camera controls, pixel access and mouse event handling
	GcamChunk	Live image preview with real-time chunk data values (camera must support chunk features)
	GcamDynamic	Live image in PictureBox object, pixel values in real time using ActiveGigE by reference
Delphi	GcamHist	Live video with real time 3-color histogram overlay

MATLAB	SingleCamera	Single camera application with custom exposure control.
	MultiCamera	Two-camera application with built-in property controls.
	MatlabViewer	Live image preview with real-time pixel values, line-profile and 3D-plot, mouse event handling
Python	ActiveGige.py	Obtaining the camera and format list, modifying the gain value, retrieving pixel values
	ActiveGigeComTypes.py	Same as above, but with the use of ComTypes module. Retrieving image window array.
	ActiveGigeLive.py	Live image window with camera properties dialog
LabView	ActiveGige.vi	Live video with a dial knob for gain control, real-time pixel value indicator
	ActiveGigePtr.vi	Using ActiveGigE by reference, displaying camera list, accessing image data via a pointer
Adobe Flash	GcamFlash.swf	A mirrored live image preview, color picker, transparent bitmaps over the video
HTML	TryActiveGige.html	Web page with an integrated viewer

*Note that VB.NET and Visual C# samples utilize .NET Framework. Make sure it is installed on your system before using these samples. In addition GcamOverlay and GcamSharp samples assume the presence of MSCOMM32.ocx on the system.*

All the samples include complete source code and executables which can be run with any GigE Vision™ camera out of the box. Simply connect one or more cameras to your system, start an application of your choice and see how *ActiveGige* performs in real world!

## 7 Camera list

*ActiveGige* is designed to work with any GigE Vision™ compatible camera. The following devices have been tested with *ActiveGige*:

Make	Models
A&B Software	GigESim camera simulator
Allied Vision Technologies	Prosilica, Manta, Mako, Goldeye, Pearleye, Bigeye series
AOS Technologies	H-EM 501
Automation Technology	C4-GigE series
Basler Vision Technologies	Ace, Aviator, Pilot, Runner and Scout series
Baumer Optronics	TXG, HXG, SXG, MXG, EXG series
e2V	AViiVA EM1, DiViINA LM1
Emergent Vision Tech	HS 10-GigE series
FLIR Systems	A5, A15, A35, A65, A320G, A315, A615
Gardasoft	TR-RC-120 lighting controller
GeviCam	GD, GF and GP series
Gigalinx	GigE-Connect products
Hitachi	KP-F**GV and HV-F**GV series
IMI Technology	Amazon and Amazon2 series
Imaging Source	DMK and DFK GigE series
Imperx	Lynx IPX and Bobcat IGV GigE series
ISG	LightWise Allegro GigE series
JAI Pulnix	All Gige Vision models
Kappa	PS, DS and Zelos GigE series
Leutron Vision	PicSight GigE series
NET GmbH	GimaGo and GigEPro series
Matrix Vision	mvBlueCOUGAR-S and mvBlueCOUGAR-X series
Microtron	Gigabit Ethernet series
Nippon	XCM GigE Series
Ophir Photonics	PyroCam III and PyroCam IV series
PhotonFocus	MV1-D1312-40-GB, MV1-D1312I-40-GB
Pixelink	PL-***G Gigabit Ethernet series
Pleora Technologies	iPORT digital and analog IP Engines
Princeton Instruments	Megaplug series
Point Grey Research	BackFly, Flea3, Grasshopper, Zebra 2
RICOH	EV-G series
Sensor to Image	CANCam-GigE series
Sentech	EHS and POE GigE series

Smartek	Giganetix series
Sony	XCG Series
SVS-VISTEK	SVCam-GigE series
Tattile	LARA and TAG series
Teledyne DALSA	Genie, Genie TS, Spyder3 series
Toshiba Teli	CSG and BG series
Xenics	Bobcat, Gobi, Onca series
Xilinx	GigEVCORE FPGA

## 8 Troubleshooting

Below is the list of the most frequently encountered issues and remedies for their resolutions:

Problem description	Cause	Resolution
The camera is not recognized by <i>ActiveGige</i> . "No Give Vision camera found" message appears in the control window.	The camera and network adapter are assigned non-matching IP-addresses.	Make sure that the camera and corresponding network adapter have IP addresses assigned to the same subnet (typically 169.254.x.x). Refer to <a href="#">Network Setup</a> or manufacturer's documentation for details.
	Firewall is enabled and the camera does not support the firewall traversing.	Disable the Windows Firewall or any additional third party firewall on the NIC connected to the camera.
	A non-compatible system driver is used with the network adapter.	<i>ActiveGige</i> will not work with "high-performance" drivers provided by certain camera manufacturers. Make sure the manufacturer's default driver is installed for your network adapter.
	The camera is not GigE Vision™ compliant.	<i>ActiveGige</i> only works with GigE Vision™ compliant cameras. Non-compliant Ethernet cameras (without GigE Vision™ logo) will not be recognized by <i>ActiveGige</i> . Use the manufacturer's provided software.
Live video would not start or occasionally freezes.	Firewall is enabled.	Disable the Windows Firewall or any additional third party firewall on the NIC connected to the camera.
	1.0 Gbps speed is not established.	Check the connection speed in the status dialog of your network adapter. If it is less than 1 Gbps, verify the settings and specifications of the NIC and network switch.
	UDP packet size exceeds the maximum value allowed by your network configuration.	Configure your network adapter for Jumbo frames. If Jumbo option is not available for your NIC, set the packet size in the <a href="#">Format</a> to 1500.
	The computer/network card/cabling has an intermittent hardware issue.	Replace the cable or try to run the camera/software on a different system.
The video is corrupted (frames are broken into parts, or synchronization is lost). Some video formats and frame rates do not work.	UDP packet size is incorrect or too small.	See the solution above.
	1.0 Gbps speed is not established.	See the solution above
	GCAM filter driver is not installed.	Follow the installation instructions for the <a href="#">Filter Driver</a> .

Problem description	Cause	Resolution
I am getting a message about mismatching camera's and network card IP addresses even though they match each other.	<p>You are using a camera that responds to the discovery message in a non-standard way (e.g. Toshiba Teli or Hitachi cameras).</p> <p>You do not have the right permissions due to User Account Control (UAC).</p>	<p>Run the IP Configuration utility and select "Use multiple sockets" option in the GigE panel.</p> <p>Run your application or developmnet environment "as administrator" using the right-click context menu.</p>
I work with multiple cameras, and they get randomly disconnected from <i>ActiveGige</i> .	Excessive activity on the network prevents discovery messages from being received in time.	Run the IP Configuration utility and set the Broadcast Channel Timeout to a larger value. If you do not expect the cameras to be plugged/unplugged on the fly, set the Broadcast Channel Period to 1000000.
When I run a live video application from within the Visual Studio, the application shows "Camera in use by another process" message.	Visual Studio does not close the design view while starting the application.	The design window of the IDE maintains control over the camera blocking your application from initiating the acquisition. Make sure to close the design view before running your application.
My compiled application does not work while GcamViewer and ActiveGigE sample applications work well.	You do not have the right permissions due to User Account Control (UAC).	Run your application or developmnet environment (i.e. Visual Studio or Matlab) "as administrator" using the right-click context menu.
When I try to run Gcam Viewer or any other ActiveGigE-based application from a non-administrative account, they do not work or crash.	Windows policy setting do not allow regular users to create global memory objects.	Ask your system administrator to add your user account to the "Create global objects" policy. This is done by running "gpedit.msc" and selecting Computer Configuration->Windows Settings->Security Settings->Local Policies->User Rights Assignment
When I try to compile sample projects from their default directories in C:\Program Files, I get an error message "Could not create an output directory"	You do not have the right permissions due to User Account Control (UAC).	<p>Copy the Samples folder from C:\Program Files\ActiveGige\ to one of your user directories (e.g. Desktop) and open sample projects from there.</p> <p>Alternatively, ask your system administrator to provide you with writing permissions for the ActiveGige folder.</p>

Problem description	Cause	Resolution
I am trying to do real-time image processing in .NET, and I experience a significant drop in the frame rate.	<p>VB.NET and C# have performance issues when working with large arrays.</p> <p><a href="#">FrameAcquired</a> event has an overhead that might affect the performance.</p> <p>Multi-core processing is disabled in ActiveGigE.</p>	<p>For performance boost, use unsafe code and pointers to directly access <i>ActiveGigE</i> image buffer. A pointer to the image buffer is provided by <a href="#">GetImagePointer</a>.</p> <p>For VB.NET, C# and C++ applications use <a href="#">FrameAcquiredX</a> event.</p> <p>Enable multi-core processing to distribute the CPU load among several cores.</p>
AVI files are not recorded in real-time, many frames are being dropped.	Your system does not provide enough throughput or CPU power to keep up with the camera frame rate.	Switch to a lower resolution mode or reduce the frame rate. If you are using a Bayer camera, consider recording the monochrome video instead of color one. Alternatively, upgrade your system to a faster hard drive (such as an SSD) and/or faster CPU.
VB.NET and C# sample applications do not work.	.NET framework is not installed on the system	Install the latest .NET framework from Microsoft: <a href="http://msdn.microsoft.com/netframework/default.aspx">http://msdn.microsoft.com/netframework/default.aspx</a>
<i>ActiveGigE</i> installation fails with the following error: "ActiveGigE.dll failed to register, HRESULT - 2147024770"	Runtimes components of Visual C++ 2005 are not installed on your system.	<p>Install Microsoft's <a href="#">VC++ 2005 Redistributable Package</a>, then run <i>ActiveGigE</i> installation again.</p> <p>For the 64-bit OS you may also need to install <a href="#">VC++ 2005 Redistributable Package 64-bit</a></p>
<i>ActiveGigE</i> installation fails with the following error: "ActiveGigE.dll failed to register, HRESULT - 2147220473"	This error is usually caused by a corrupted registration of atl.dll system file.	<ol style="list-style-type: none"> <li>1. Locate atl.dll, generally found in "C:\WINNT\system32" or "C:\WINDOWS\system32".</li> <li>2. Open the command-line prompt <ul style="list-style-type: none"> <li>• o Start Menu/Run</li> <li>• o type "cmd"</li> <li>• o press "ENTER"</li> </ul> </li> <li>3. Using the atl.dll filename and path, call regsvr32, i.e. at the command-line prompt, type: <ul style="list-style-type: none"> <li>• o regsvr32 "C:\WINNT\system32\atl.dll"</li> <li>• o press "ENTER"</li> </ul> </li> <li>4. You should see a regsvr32 window, confirming success: Close it.</li> <li>5. Repeat <i>ActiveGigE</i> installation.</li> </ol>



# Index

## - 3 -

3D look 115

## - A -

Accumulate 148, 149  
 Acquire 66  
 Acquisition 66, 68, 73  
 AcquisitionFrameCount 68  
 AcquisitionFrameRate 70  
 AcquisitionFrameRateAbs 71  
 AcquisitionFrameRateRaw 72  
 AcquisitionMode 73  
 Action 248, 405, 408, 423  
 Affinity 75  
 Alpha 76  
 Analog property page 513  
 Anti-tearing 78  
 API 53  
 Average 148, 149  
 AVI 225, 228, 252, 253, 254, 277, 278, 279, 370, 371, 372, 373, 382, 390, 410, 411, 413, 415, 448, 449, 451, 453, 456, 457, 458, 463, 465, 469, 471, 472, 473, 483, 493, 494, 504

## - B -

Background color 80  
 BalanceRatioAbs 81, 83  
 BalanceRatioRaw 85  
 BalanceRatioSelector 87  
 BalanceWhiteAuto 89  
 Bayer filter 91  
 BinningtX 93  
 BinningtY 94  
 BkgCorrect 95  
 BkgName 97  
 Black level 98, 100, 102, 104, 106, 261, 262  
 BlackLevel 98  
 BlackLevelAbs 100  
 BlackLevelAuto 102  
 BlackLevelRaw 104

BlackLevelSelector 106  
 bmp 378, 399  
 Building the Graph 528

## - C -

C# 44  
 C++ 37  
 Camera 108  
 Camera list 266, 268, 271  
 CameraPlugged event 481  
 CameraUnplugged event 482  
 CaptureComplete 403  
 CaptureCompleted event 483  
 Capturing to AVI 530  
 CloseVideo 225  
 ColorCorrect 110  
 Compatibility 554  
 CreateSequence 226  
 CreateVideo 228

## - D -

DecimationX 111  
 DecimationY 112  
 Digital I/O 152, 154, 156, 158, 160, 206, 208  
 DirectShow 522, 525, 526, 528, 529, 530, 531, 533, 535, 537, 539, 541, 542, 544, 546, 547  
 DirectShow Interfaces 534  
 DirectShow Quick Reference Guide 523  
 Display 113  
 Displaying Property Pages 533  
 Displaying the Preview 529  
 Distributing 59  
 Draw 230  
 DrawAlphaClear 231  
 DrawAlphaEllipse 232  
 DrawAlphaLine 234  
 DrawAlphaPixel 236  
 DrawAlphaRectangle 237  
 DrawAlphaText 239  
 DrawEllipse 241  
 DrawLine 243  
 DrawPixel 244  
 DrawRectangle 245  
 DrawText 247  
 Driver 16

## - E -

Edge 115  
EventDataMessage event 486  
EventMessage event 484  
Events 479  
Exposure 116, 118, 120, 122, 124, 288, 289  
ExposureAuto 116  
ExposureMode 118  
ExposureTime 120  
ExposureTimeAbs 122  
ExposureTimeRaw 124

## - F -

Feature access 290, 292, 296, 302, 304, 306, 309, 320, 427, 429, 431, 433  
Feature information 286, 299, 311, 312  
Feature list 300  
File Access 315, 316, 318, 394  
FileAccess 476  
Filter 16  
FilterConfig 525  
Flip 126  
Font 128  
Format list 322  
FormatChanged event 488  
FPS 70, 71, 72, 250, 251  
Frame rate 324, 325  
FrameAcquired event 489  
FrameAcquiredX event 490  
FrameDropped event 492  
FrameLoaded event 493  
FrameReady event 495  
FrameRecorded event 494  
Fromat 129

## - G -

Gain 132, 134, 136, 138, 140, 326, 327  
GainAbs 134  
GainAuto 136  
GainRaw 138  
GainSelector 140  
Gamma 142  
GcamViewer 30

GenICam property page 517  
GetAcquisitionFrameRateMax 250  
GetAcquisitionFrameRateMin 251  
GetActionAcknowledgeInfo 248  
GetAudioLevel 252  
GetAudioList 253  
GetAudioSource 254  
GetBalanceRatioMax 255  
GetBalanceRatioMin 256  
GetBarcode 257  
GetBitsPerChannel 259  
GetBlackLevelMax 261  
GetBlackLevelMin 262  
GetBlockId 263  
GetBytesPerPixel 264  
GetCameraIP 265  
GetCameraIPList 266  
GetCameraList 268  
GetCameraMAC 270  
GetCameraMACList 271  
GetChunkPointer 273  
GetChunkSize 275  
GetCodec 277  
GetCodecList 278  
GetCodecProperties 279  
GetComponentData 280  
GetComponentLine 282  
GetDIB 284  
GetEnumList 286  
GetExposureTimeMax 288  
GetExposureTimeMin 289  
GetFeature 290  
GetFeature64 292  
GetFeatureAccess 294  
GetFeatureArray 296  
GetFeatureDependents 298  
GetFeatureDescription 299  
GetFeatureIncrement 302  
GetFeatureList 300  
GetFeatureMax 306  
GetFeatureMin 304  
GetFeatureRepresentation 308  
GetFeatureString 309  
GetFeatureTip 311  
GetFeatureType 312  
GetFeatureVisibility 314  
GetFileAccessMode 315  
GetFileList 316

GetFileSize 318  
GetFileTransferProgress 320  
GetFormatList 322  
GetFPS 325  
GetFPSAcquired 324  
GetGainMax 326  
GetGainMin 327  
GetHeight 328  
GetHeightMax 330  
GetHistogram 331  
GetImageData 333, 425  
GetImageLine 335  
GetImagePointer 337  
GetImageStat 339  
GetImageWindow 341  
GetLevels 343  
GetLUT 344  
GetOptimalPacketSize 345  
GetPicture 347  
GetPixel 349  
GetRawData 351  
GetRGBPixel 353  
GetROI 355  
GetSequenceFrameCount 356  
GetSequencePicture 357  
GetSequencePixel 358  
GetSequencePointer 360  
GetSequenceRawData 362  
GetSequenceTimestamp 364  
GetSequenceWindow 365  
GetTimestamp 367  
Getting the Image Data 531  
GetTriggerDelayMax 368  
GetTriggerDelayMin 369  
GetVideoFPS 370  
GetVideoFrameCount 371  
GetVideoPosition 372  
GetVideoVolume 373  
GetWidthMax 374  
Grab 375  
Guide 34

## - H -

Heartbeat 144  
Height 185  
Histogram 331  
Horizontal binning 93

Horizontal decimation 111  
Horizontal offset 187  
HotPixelCorrect 146  
HotPixelLevel 147

## - I -

IActiveGige 541  
IAmCameraControl 535  
IAMDroppedFrames 546  
IAMStreamConfig 542  
IAMVideoCompression 544  
IAMVideoControl 539  
IAMVideoProcAmp 537  
Input/Output property page 515  
Installation 13  
Integrate 148  
IntegrateWnd 149  
Introduction 8  
IP configuration utility 25  
IsFeatureAvailable 376  
IsMasterApplication 377  
ISpecifyPropertyPages 547

## - J -

jpeg 378, 399

## - L -

LensCorrect 150  
License 8, 10  
LineFormat 152  
LineInverter 154  
LineMode 156  
LineSelector 158  
LineSource 160  
LoadImage 378  
LoadSequence 381  
LoadSettings 379  
Lookup table 162, 344, 435, 442  
LUTMode 162

## - M -

Magnification 163  
Matlab 48

Methods 213  
 MonitorSync 165  
 MouseDbClick event 497  
 MouseDown event 498  
 MouseDownRight event 499  
 MouseMove event 501  
 MouseUp event 502  
 MouseUpRight event 503  
 Multicast 167  
 Multiple cameras 57

## - N -

Network setup 21

## - O -

OffsetX 187  
 OffsetY 188  
 OpenVideo 382  
 Overlay 169  
 OverlayClear 384  
 OverlayColor 170  
 OverlayEllipse 385  
 OverlayFont 171  
 OverlayLine 386  
 OverlayPixel 387  
 OverlayRectangle 388  
 OverlayText 389  
 Overview 8

## - P -

PacketSize 172  
 Palette 174  
 PalyVideo 390  
 Pixel depth 259, 264  
 PlayCompleted event 504  
 Privilege 176  
 Properties 60  
 Property pages 460, 508  
 Pseudo colors 174  
 Python 52

## - R -

RawFrameAcquired event 505

Read block 392  
 Read register 396  
 ReadFile 394  
 Reference 59  
 Registration 20  
 Requirements 12  
 Retrieving the Filter 526  
 ROI 355, 444  
 Rotate 178

## - S -

Samples 550  
 SaveBkg 397  
 SaveImage 399  
 SaveSequence 403  
 SaveSettings 401  
 Scroll event 506  
 ScrollBars 180  
 ScrollX 182  
 ScrollY 183  
 SendActionCommand 405  
 SendActionConditions 408  
 Sequence 226, 356, 357, 358, 360, 362, 364, 365, 370, 372, 381, 382, 390, 403, 448, 449, 451, 466, 468, 472, 473, 483, 493, 494, 504  
 SetAudioLevel 410  
 SetAudioSource 411  
 SetCodec 413  
 SetCodecProperties 415  
 SetColorMatrix 417  
 SetControlKey 419  
 SetDestinationIP 421  
 SetDeviceKey 423  
 SetFeature 427  
 SetFeature64 429  
 SetFeatureArray 431  
 SetFeatureString 433  
 SetGains 435  
 SetLensDistortion 437  
 SetLevels 439  
 SetLUT 442  
 SetROI 444  
 SetStreamChannel 446  
 SetVideoFPS 448  
 SetVideoPosition 449  
 SetVideoSync 451  
 SetVideoVolume 453

SetWebStreamer 454  
ShowAudioDlg 456  
ShowCodecDlg 457  
ShowCompressionDlg 458  
ShowProperties 460  
SizeX 184  
SizeY 185  
SoftTrigger 462  
Software trigger 462  
Source property page 509, 519  
StartCapture 463  
StartSequenceCapture 466  
StartVideoCapture 469  
StopCapture 465  
StopSequenceCapture 468  
StopVideo 472  
StopVideoCapture 471

## - T -

TestImageSelector 190  
tiff 378, 399  
Timeout 189, 507  
Trigger 192  
TriggerActivation 194  
TriggerDelay 196  
TriggerDelayAbs 198  
TriggerDelayRaw 200  
Triggering 194, 196, 198, 200, 202, 204, 368, 369  
TriggerSelector 202  
TriggerSource 204  
TriggerVideo 473  
Troubleshooting 555  
TWAIN 548

## - U -

UserOutputSelector 206  
UserOutputValue 208  
UserSetSelector 209

## - V -

VB 35  
VB.NET 41  
Vertical binning 94  
Vertical decimation 112

Vertical offset 188  
Video Format 511  
Visual Basic 35  
Visual C# 44  
Visual C++ 37

## - W -

Web streaming 211  
White balance 81, 83, 85, 87, 89, 255, 256  
Width 184  
Window/Level 343, 439  
Write block 474  
Write register 478  
WriteFile 476