Basler IP Camera

API Description

Document Number: AW000662
Version: 06 Language: 000 (English)
Release Date: 16 July 2009

BASLER
VISION TECHNOLOGIES

Contacting Basler Support Worldwide

Europe:

Basler AG
An der Strusbek 60 - 62
22926 Ahrensburg
Germany

bc.support.ip@baslerweb.com

Americas:

Basler, Inc.
855 Springdale Drive, Suite 160
Exton, PA 19341
U.S.A.

bc.support.ip@baslerweb.com

Asia:

Basler Asia Pte. Ltd
8 Boon Lay Way
03 - 03 Tradehub 21
Singapore 609964

bc.support.ip@baslerweb.com

www.basler-ipcam.com

All material in this publication is subject to change without notice and is copyright Basler Vision Technologies.

Table of Contents

1 Introduction

This document describes the HTTP/CGI-based application programming interface (API) on the Basler IPCam. The interface provides the ability to set internal camera parameters and to request images. All requests are handled by a web server that is built into the camera.

A user application communicates with the IPCam through the use of an HTTP client. The HTTP client sends requests to the camera using the standard HTTP "post" or "get" methods. The requests consist of remote procedure calls to the camera's parameters. The remote procedure calls are coded as a simple string representation (see the next section for details).

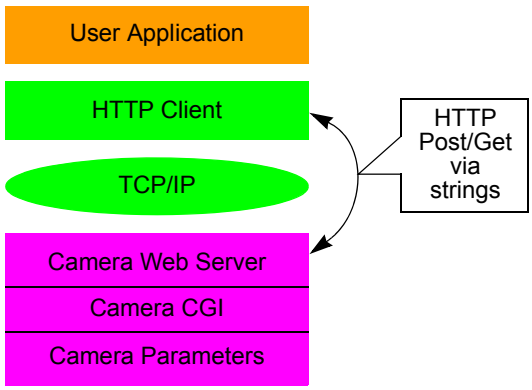

Fig. 1: Interface Diagram

1.1 Setting the IPCam's Parameter Values

Each camera parameter is configured by setting the value of the parameter. Access to the parameter values is implemented with HTTP/CGI.

Each parameter belongs to a parameter group. Section 4 lists the parameter groups and details which parameters are included in each group.

The value for each parameter is of a particular data type such as Integer, String, Enumeration, etc. The value for the Gain parameter, for example, is of the Integer data type.

A set of methods is associated with each data type. The methods associated with a data type can be performed on any parameter value with that data type. For example, two of the methods associated with the Integer data type are GetValue and SetValue. Since the value for the Gain parameter is of the Integer data type, the GetValue and SetValue methods can be performed on the Gain parameter value. Section 2 lists the data types and shows the methods available for each data type. The tables in Section 4 list the camera parameters and indicate the data type for each parameter value.

Requests to set a parameter value or to get a parameter value are issued to the camera via HTTP/CGI. A simplified textual protocol is available for getting the current value for any camera parameters and for setting the value of the parameters.

All requests to get a parameter's current value are formatted as follows:

```
http://<camera>/cgi-bin/param_if.cgi?NumActions=<N>
&Action_<i>=<ParameterGroup>.<Parameter>.<Method>
```

Where:

<*camera*> = the camera from which you want to get the parameter value.
This can be entered as an IP Address:Port Number.
If your network has a properly configured domain name server, it can also be entered as a user assigned host name.

<*N*> = the number of method calls (actions) included in the request

<*i*> = an integer value that increments for each method call (action) included in the request. The starting value for i = 0.

<*ParameterGroup*> = the group name for the parameter you want to work with. For example, the ExposureMode parameter belongs to the ImageControls parameter group.

<*Parameter*> = the name of the parameter you want to work with, for example, ExposureMode or Gamma.

<*Method*> = the method that you want to perform on the parameter value, for example, the GetValue method. Remember that the method you apply to a particular parameter value must be appropriate for the parameter value's data type.

For parameters of the integer, string, or enumeration data type, all requests to set a parameter value are formatted as follows:

```
http://<camera>/cgi-bin/param_if.cgi?NumActions=<N>
&Action_<i>=<ParameterGroup>.<Parameter>.<Method>&Parameter_<i>_<j>=<Value>
```

Where:

<*camera*> = the camera on which you want to set the parameter value.
This can be entered as an IP Address:Port Number.
If your network has a properly configured domain name server, it can also be entered as a user assigned host name.

<*N*> = the number of method calls (actions) included in the request

<*i*> = an integer value that increments for each method call (action) included in the request. The starting value for i = 0.

<*j*> = an integer that increments for each value required by a parameter. The starting value for j = 0.

<*ParameterGroup*> = the group name for the parameter you want to work with. For example, the ExposureMode parameter belongs to the ImageControls parameter group.

<*Parameter*> = the name of the parameter you want to work with, for example, ExposureMode or Gamma.

<*Method*> = the method that you want to perform on the parameter value, for example, the SetValue method. Remember that the method you apply to a particular parameter value must be appropriate for the parameter value's data type.

<*Value*> = the setting for the parameter value.

For parameters of the command data type, all requests to set a parameter value are formatted as follows:

```
http://<camera>/cgi-bin/param_if.cgi?NumActions=<N>
&Action_<i>=<ParameterGroup>.<Parameter>.<Method>
```

Where:

<*camera*> = the camera on which you want to set the parameter value.
This can be entered as an IP Address:Port Number.
If your network has a properly configured domain name server, it can also be entered as a user assigned host name.

<*N*> = the number of method calls (actions) included in the request

<*i*> = an integer value that increments for each method call (action) included in the request. The starting value for i = 0.

<*j*> = an integer that increments for each value required by a parameter. The starting value for j = 0.

<*ParameterGroup*> = the group name for the parameter you want to work with. For example, the ExposureMode parameter belongs to the ImageControls parameter group.

<*Parameter*> = the name of the parameter you want to work with, for example, ExposureMode or Gamma.

<*Method*> = the method that you want to perform on the parameter value, for example, the SetValue method. Remember that the method you apply to a particular parameter value must be appropriate for the parameter value's data type.

Response to a Request

The response to a request will be a line starting with "Return_<*i*>" that shows the return value of the method call. For methods that do not return a value, the return will contain no data. If an exception is thrown to signal an error condition, Return_<*i*> is replaced by "Exception_<*i*>" and contains the exception data.

Requests can contain multiple actions (see Examples 4 and 5 on the following pages).

Requests can also be issued as Post requests.

Examples

Example 1 - A request to set the value of the ExposureOffset parameter to +20 on a camera that has been named "IPCam_1". The ExpsoureOffset parameter is part of the ImageControls parameter group. The parameter value is of the Integer data type and is a signed integer.

Request:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=ImageControls.ExposureOffset.SetValue&Parameter_0_0=+20
```

Return:

```
Return_0=
```

Example 2 - A request to set the value of the TestImageMode parameter to TestImage_1 on a camera that has been named "IPCam_1". This is followed by a request to get the current parameter value. The TestImageMode parameter is part of the Sensor parameter group and the parameter value is of the Enumeration data type.

Request:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=Sensor.TestImageMode.SetValue&Parameter_0_0=TestImage_1
```

Return:

```
Return_0=
```

Request:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=Sensor.TestImageMode.GetValue
```

Return:

```
Return_0=TestImage_1
```

Example 3 - A request to set the value of the sensor AOIWidth parameter to 2500 on a camera that has been named "IPCam_1". The sensor AOIWidth parameter is part of the Sensor parameter group.

Since 2500, exceeds the maximum allowed value for the sensor AOIWidth parameter on this particular camera model, the request will not be successful.

Request:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=Sensor.AOIWidth.SetValue&Parameter_0_0=2500
```

Return:

```
Exception_0=validation error, value out of range
```

Example 4 - A request containing multiple actions including: set the value of the TestImageMode parameter to Off, get the current value for the TestImageMode parameter, and get the maximum allowed value for the sensor AOIWidth parameter, on a camera that has been named "IPCam_1". The parameters are part of the Sensor parameter group.

Request:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=3
&Action_0=Sensor.TestImageMode.SetValue&Parameter_0_0=Off
&Action_1=Sensor.TestImageMode.GetValue
&Action_2=Sensor.AOIWidth.GetMax
```

Return:

```
Return_0=
Return_1=Off
Return_2=640
```

Example 5 - A request to place the camera in configuration mode, i.e., set the OperationMode parameter to Configure, on a camera that has been named "IPCam_1". This is followed by a request containing multiple actions including: set the value of the sensor AOIWidth parameter to 128, the sensor AOIHeight parameter to 80, the sensor AOILeft parameter to 20, and the sensor AOITop parameter to 10. And finally, a request is issued to set the OperationMode to Normal.

The OperationMode parameter is part of the Global parameter group and the other parameters are part of the Sensor group.

Request:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=Global.OperationMode.SetValue&Parameter_0_0=Configure
```

Return:

```
Return_0=
```

Request:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=4
&Action_0=Sensor.AOIWidth.SetValue&Parameter_0_0=128
&Action_1=Sensor.AOIHeight.SetValue&Parameter_1_0=80
&Action_2=Sensor.AOILeft.SetValue&Parameter_2_0=20
&Action_3=Sensor.AOITop.SetValue&Parameter_3_0=10
```

Return:

```
Return_0=
Return_1=
Return_2=
Return_3=
```

Request:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=Global.OperationMode.SetValue&Parameter_0_0=Normal
```

Return:

```
Return_0=
```

Example 6 - A request to Reboot the camera on a camera that has been named "IPCam_1".

The Reboot parameter is part of the System parameter group. The parameter is of the Command data type.

Request:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=System.Reboot.Execute
```

Return:

```
Return_0=
```

2 Data Types and Methods

2.1 Integer Data Type

A parameter value of the integer data type implements a 64 bit integer type.

Available Methods

GetValue - returns the current value.

SetValue - sets the value.

GetMin - returns the minimum allowed value.

GetMax - returns the maximum allowed value.

GetInc - returns the allowed increment for setting the parameter value.

2.2 String Data Type

A parameter value of the string data type implements a string with a fixed maximum size.

Available Methods

GetValue - returns the current value.

SetValue - sets the value.

GetMaxLength - gets maximum allowed length of the string

Note

Strings included in a request must be in URL encoded format.

2.3 Enumeration Data Type

A parameter value of the Enumeration data type can take a value from a finite set of values.

Available Methods

GetEntries - returns a list of enumeration entries for the enumeration. The list will include string values and an index value for each string.

For example, if you issued this request to list the enumeration values for the TestImage Mode parameter on a camera that has been named "IPCam_1":

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=Sensor.TestImageMode.GetEntries
```

You would get this return:

```
Return_0=(0,"Off"),(1,"TestImage_1"),(2,"TestImage_2"),
(3,"TestImage_3"),(4,"TestImage_4"),(5,"TestImage_5"),
(6,"TestImage_6"),(7,"TestImage_7")
```

The return tells you that for the TestImageMode parameter, there is an "Off" string value with an index value of 0, a "TestImage_1" string value with an index value of 1, etc.

GetStringValue or Get Value - returns the current enumeration value as a string.

SetStringValue or SetValue - sets the enumeration value as a string.

GetIntValue - returns the index value that corresponds to the current enumeration value.

SetIntValue - sets the index value that corresponds to the desired enumeration value.

2.4 Command Data Type

A parameter of the Command data type can be used to trigger the execution of a specific action inside of the camera.

Available Methods

Execute - submits the command for execution.

IsDone - returns "true" if the command has been executed and "false" if it is still in the process of being executed.

Stop - attempts to stop a command that is in progress.

Example

A request to execute the reboot command on a camera that has been named "IPCam_1".

Request:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=System.Reboot.Execute
```

Return:

```
Return_0=
```

3 Masks

Several of the camera parameters described in the next section are based on an image mask. On these cameras, the mask divides the captured images into a 32 block wide by 24 block high array. A value for each block within the array can be set to 1 (= active) or to 0 (= inactive).

A string is used to set the values of the blocks within the array. Such a string might look like this (hexadecimal):

FFFFFFFF, FFFFFFFF

Each group of 8 hexadecimal digits in the string represents one row of blocks in the array. There are 24 groups of digits to represent the 24 rows of blocks in the array. The first group of 8 digits represents the first (top) row in the array, the second group represents the second row in the array, and so on.

The binary representation of the 8 hex digits within a single group determines the active/inactive status for the blocks in the row. For example, if the first group of hex digits in the string was "89ABCDEF", this would translate to the following binary digits:

1000 1001 1010 1011 1100 1101 1110 1111

And this would mean that for the top row of blocks, the first block would be active, the second block would be inactive, the third block would be inactive, the fourth block would be inactive, the fifth block would be active, and so on.

As an example, consider the PrivacyMask parameter in the ImageControls group. Assume that we would like to set all of the blocks in the bottom three rows of the privacy mask array to active and all of the other blocks in the array to inactive. The request would look like this:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=AnalogControls.PrivacyMask.SetValue&Parameter_0_0=00000000,
00000000,00000000,00000000,00000000,00000000,00000000,00000000,
00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,
00000000,00000000,00000000,00000000,00000000,00000000,FFFFFFFF,
FFFFFFFF,FFFFFFFF
```

4 Parameter Groups

4.1 Global Group

The parameters in this group affect the operation of the camera as a whole.

Parameter:	OperationMode
Data Type:	Enum
Valid Values:	Normal = the normal streaming mode. The camera delivers video streams as they are configured by the camera's current parameter settings. Some camera parameters can not be changed when the camera is in normal mode. See for Table 1 for details. Configure = the camera outputs a full size MJPEG image on stream 0. All camera parameters can be changed. Changes will take effect when the camera is switched back to the Normal mode.
Comments:	The camera should usually be set to the normal mode.

Parameter:	Temperature
Data Type:	Signed Integer
Comments:	Read Only Returns the current reading from the camera's internal temperature sensor in °C.

Parameters Than Can Not Be Changed in the Normal Operation Mode (But Can Be Changed in the Configure Operation Mode)	
Sensor AOIWidth	Stream LiveBufferSize
Sensor AOIHeight	Stream AlarmBufferSize
Stream AOIWidth	Stream Post Alarm Buffer Size
Stream AOIHeight	Stream OutputSize
Stream EncoderType	Stream OutputScaling
	Stream FrameRateScaling

Table 1: Parameters That Can Not Be Changed in the Normal Operation Mode

4.2 Sensor Group

The parameters in this group set the basic characteristics of the image area that will be captured by the camera's sensor.

Many of the parameters in this group are used to set the imaging sensor's "area of interest" (AOI). The AOI settings let you define the area on the sensor that will actually be used when the camera is capturing images. You can set the AOI settings so that the full sensor is used to capture images or so that just a portion of the sensor is used.

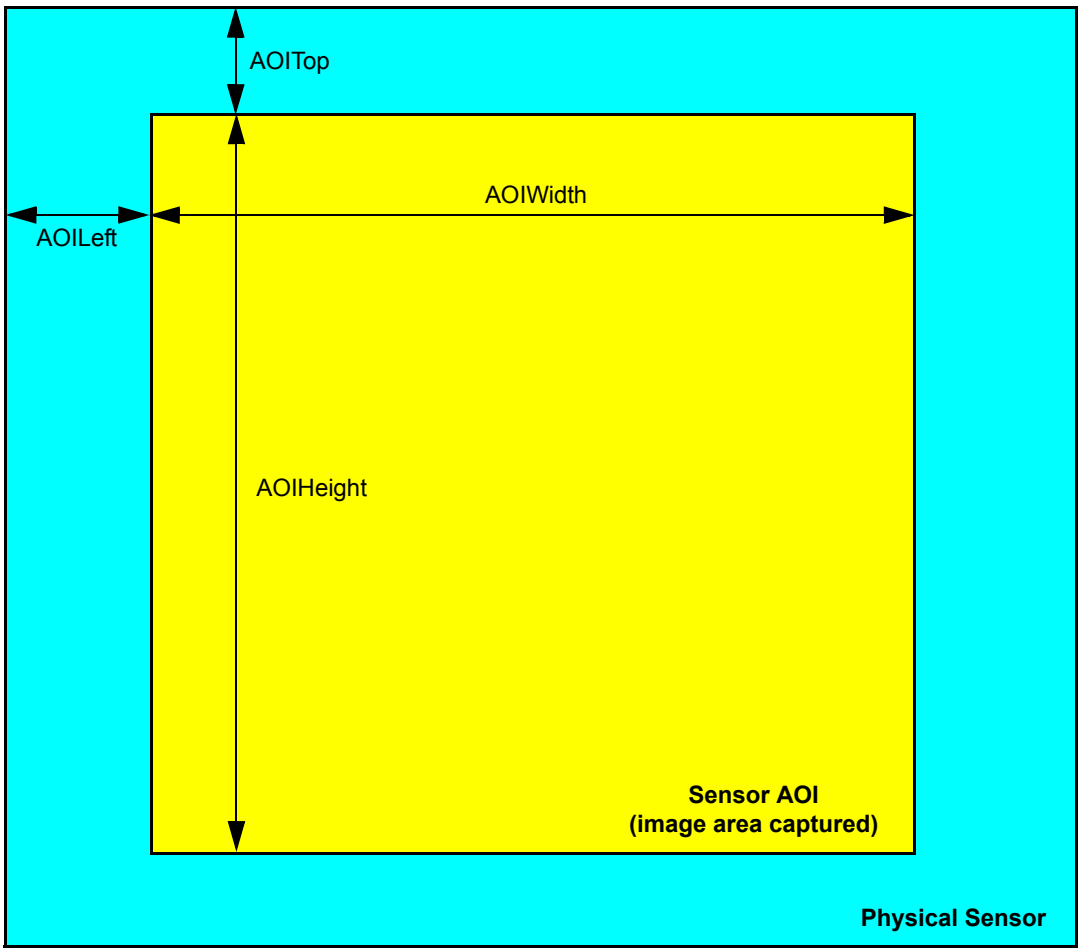

Fig. 2: Sensor AOI

Parameter:	AOIWidth
Data Type:	Unsigned integer
Valid Values:	Min = 64 Max = as indicated by the sensor AOIWidthMax parameter. Increment = 8
Comments:	As shown in Figure 2, sets the width (in pixels) of the sensor AOI. **The sensor AOI Width parameter value can not be changed when the camera is in normal operation mode.** It can only be changed when the camera is in configure operation mode (see Section 4.1 on page 15).

Parameter:	AOIHeight
Data Type:	Unsigned integer
Valid Values:	Min = 40 Max = as indicated by the sensor AOIHeightMax parameter. Increment = 8
Comments:	As shown in Figure 2, sets the height (in pixels) of the sensor AOI. **The sensor AOIHeight parameter value can not be changed when the camera is in normal operation mode.** It can only be changed when the camera is in configure operation mode (see Section 4.1 on page 15).

Parameter:	AOILeft
Data Type:	Unsigned integer
Valid Values:	Min = 0 The total of the sensor AOILeft parameter setting plus the sensor AOIWidth parameter setting must be less than or equal to the value indicated by the sensor AOIWidthMax parameter. Increment = 2
Comments:	As shown in Figure 2, sets the left offset (in pixels) for the sensor AOI, i.e., how far the sensor AOI will be offset from the left edge of the sensor.

Parameter:	AOITop
Data Type:	Unsigned Integer
Valid Values:	Min = 0 The total of the sensor AOITop parameter setting plus the sensor AOIHeight parameter setting must be less than or equal to the value indicated by the sensor AOIHeightMax parameter. Increment = 2
Comments:	As shown in Figure 2, sets the top offset (in pixels) for the sensor AOI, i.e., how far the sensor AOI will be offset from the top edge of the sensor.

Parameter:	AOIWidthMax
Data Type:	Unsigned Integer
Comments:	Read Only Indicates the max allowed width (in pixels) of the sensor AOI.

Parameter:	AOIHeightMax
Data Type:	Unsigned Integer
Comments:	Read Only Indicates the max allowed height (in pixels) of the sensor AOI.

Parameter:	ImageRotation
Data Type:	Enum
Valid Values:	ROT_0 = do not rotate the image. ROT_180 = rotate the image 180 degrees.
Comments:	Sets the degrees of rotation for the images captured by the sensor.

Parameter:	TestImageMode
Data Type:	Enum
Valid Values:	Off = test image function disabled. TestImage_1 = enables test image 1, a fixed diagonal gray scale gradient pattern. TestImage_2 = enables test image 2, a moving diagonal gray scale gradient pattern. TestImage_3 = enables test image 3, a moving diagonal gray scale gradient pattern. TestImage_4 = enables test image 4, a moving diagonal gray scale gradient pattern. TestImage_5 = enables test image 5, a moving diagonal gray scale gradient pattern. TestImage_6 = enables test image 6, a moving diagonal red/green/blue gradient pattern.
Comments:	Enables or disables the camera's test image feature. When one of the test image modes is enabled, the camera will generate test images using it's digital devices rather than the imaging sensor. The generated test images will be transmitted on all active streams. Test images are useful for troubleshooting the camera's basic functionality and the network connection.

Parameter:	FrameRateMode
Data Type:	Enum
Valid Values:	Fps_30 = the camera will capture 30 frames per second. Fps_25 = the camera will capture 25 frames per second. Fps_20 = the camera will capture 20 frames per second. Fps_15 = the camera will capture 15 frames per second. Fps_12_5 = the camera will capture 12.5 frames per second. Fps_10 = the camera will capture 10 frames per second. Fps_6_25 = the camera will capture 6.25 frames per second. Fps_5 = the camera will capture 5 frames per second. Fps_2_5 = the camera will capture 2.5 frames per second. Manual = the rate at which the camera captures frames will be determined by the setting of the FramePeriod_us parameter.
Comments:	Sets the rate at which frames (images) will be captured **by the camera's sensor**. The valid values will vary by camera model and may also vary depending on how some camera parameters (such as the sensor AOI parameters) are set. Use the GetEntries method to determine the current valid settings. Note that this setting will represent the absolute maximum frame rate that can be achieved by any stream.

Parameter:	FramePeriod_us
Data Type:	Unsigned integer
Valid Values:	Range varies depending on other camera settings such as the sensor AOI settings. Use the GetMin and GetMax methods to determine the currently allowed range.
Comments:	If the FrameRateMode parameter is set to Manual, the FramePeriod_us parameter sets the rate at which the camera's sensor will capture frames (images). With the FrameRateMode set to Manual: Frame Rate = 1000000 / Frame Period_us

4.3 ImageControls Group

The parameters in this group control the quality of the images captured by the camera's imaging sensor.

Parameter:	ExposureMode
Data Type:	Enum
Valid Values:	PrioNone = automatic exposure time control and gain control are both enabled. The camera will automatically adjust both the exposure time and the gain to maintain good overall image quality as lighting conditions change. Neither the exposure time adjustment nor the gain adjustment will have priority. PrioFramerate = automatic exposure time control and gain control are both enabled. The camera's automatic exposure and gain adjustments will be biased so that the frame rate is maintained at as high a level as possible. Maintaining image quality is given a lower priority. PrioQuality = automatic exposure time control and gain control are both enabled. The camera's automatic exposure and gain adjustments will be biased so that image quality is maintained at as high a level as possible. Maintaining the frame rate is given a lower priority. ManualGain = automatic gain control is disabled, and you must manually set the gain by using the Gain parameter (see below). Automatic exposure time control is enabled. ManualExposureTime = automatic exposure time control is disabled, and you must manually set the exposure time by using the Exposure Time parameter (see below). Automatic gain control is enabled. ManualGainAndExposureTime = automatic gain control and exposure time control are both disabled. You must manually set the gain and the exposure time by using the Gain and Exposure Time parameters (see below).
Comments:	Sets the functionality of the camera's automatic exposure and gain controls. If desired, the ExposureTimeLimit and the GainLimit_db parameters (see below) can be used to set limits on the auto controls. **Note:** The valid values listed above became active with version 1.4 firmware. On older versions of firmware, the valid values were Normal, FastShutter, LowNoise, LowLight, and Manual. Cameras with version 1.4 firmware and above will no longer accept the older valid values.

Parameter:	ExposureOffset
Data Type:	Signed integer
Valid Values:	-100 to +100
Comments:	This setting is used to customize the operation of the camera's automatic controls. Negative settings will bias the auto controls toward producing darker images. Positive settings will bias the auto controls toward producing lighter images.

Parameter:	ExposureTimeLimit
Data Type:	Enum
Valid Values:	Time_1_5000_s = the exposure time limit is 1/5000 second Time_1_2500_s = the exposure time limit is 1/2500 second Time_1_1000_s = the exposure time limit is 1/1000 second Time_1_500_s = the exposure time limit is 1/500 second Time_1_250_s = the exposure time limit is 1/250 second Time_1_125_s = the exposure time limit is 1/125 second Time_1_60_s = the exposure time limit is 1/60 second Time_1_30_s = the exposure time limit is 1/30 second Time_1_15_s = the exposure time limit is 1/15 second Time_1_8_s = the exposure time limit is 1/8 second Time_1_4_s = the exposure time limit is 1/4 second Time_1_2_s = the exposure time limit is 1/2 second Time_1_s = the exposure time limit is 1 second Off = there is no exposure time limit
Comments:	When automatic exposure control is enabled (see the ExposureMode parameter above), the ExposureTimeLimit parameter sets the maximum exposure time that the automatic exposure control can use. If the exposure time limit is set to "off", there will be no limit for the automatic exposure control.

Parameter:	GainLimit_dB
Data Type:	Enum
Valid Values:	Gain_6_dB = the gain limit is 6 dB Gain_12_dB = the gain limit is 12 dB Gain_18_dB = the gain limit is 18 dB Gain_24_dB = the gain limit is 24 dB Off = there is no gain limit
Comments:	When automatic gain control is enabled, (see the ExposureMode parameter above) the GainLimit_dB parameter sets the maximum gain that the automatic gain control can use. If the gain limit is set to "off", there will be no limit for the automatic gain control.

Parameter:	ShutterMode
Data Type:	Enum
Valid Values:	Auto = the exposure time for each captured image will be automatically controlled by the camera. Manual = the exposure time for each captured image will be determined by the setting of the ExposureTime_us parameter.
Comments:	Sets the functionality of the of the camera's automatic exposure control. **Note:** Starting with version 1.4 software, this parameter became redundant and should no longer be used. To maintain backward compatibility, the parameter will still be available.

Parameter:	GainMode
Data Type:	Enum
Valid Values:	Auto = the gain for each captured image will be automatically controlled by the camera. Manual = the gain for each captured image will be determined by the setting of the Gain parameter.
Comments:	Sets the functionality of the of the camera's automatic gain control. **Note:** Starting with version 1.4 software, this parameter became redundant and should no longer be used. To maintain backward compatibility, the parameter will still be available.

Parameter:	ExposureTime
Data Type:	Enum
Valid Values:	Time_1_10000_s = the exposure time is 1/10000 second Time_1_8000_s = the exposure time is 1/8000 second Time_1_6000_s = the exposure time is 1/6000 second Time_1_5000_s = the exposure time is 1/5000 second Time_1_4000_s = the exposure time is 1/4000 second Time_1_3000_s = the exposure time is 1/3000 second Time_1_2500_s = the exposure time is 1/2500 second Time_1_2000_s = the exposure time is 1/2000 second Time_1_1600_s = the exposure time is 1/1600 second Time_1_1200_s = the exposure time is 1/1200 second Time_1_1000_s = the exposure time is 1/1000 second Time_1_800_s = the exposure time is 1/800 second Time_1_640_s = the exposure time is 1/640 second Time_1_500_s = the exposure time is 1/500 second Time_1_400_s = the exposure time is 1/400 second Time_1_320_s = the exposure time is 1/320 second Time_1_250_s = the exposure time is 1/250 second Time_1_200_s = the exposure time is 1/200 second Time_1_160_s = the exposure time is 1/160 second Time_1_125_s = the exposure time is 1/125 second Time_1_100_s = the exposure time is 1/100 second Time_1_80_s = the exposure time is 1/80 second Time_1_60_s = the exposure time is 1/60 second Time_1_50_s = the exposure time is 1/50 second Time_1_40_s = the exposure time is 1/40 second Time_1_30_s = the exposure time is 1/30 second Time_1_25_s = the exposure time is 1/25 second Time_1_20_s = the exposure time is 1/20 second Time_1_15_s = the exposure time is 1/15 second Time_1_12p5_s = the exposure time is 1/12.5 second Time_1_10_s = the exposure time is 1/10 second Time_1_8_s = the exposure time is 1/8 second Time_1_6_s = the exposure time is 1/6 second Time_1_5_s = the exposure time is 1/5 second Time_1_4_s = the exposure time is 1/4 second Time_1_3_s = the exposure time is 1/3 second Time_1_2p5_s = the exposure time is 1/2.5 second Time_1_2_s = the exposure time is 1/2 second Time_1_1p6_s = the exposure time is 1/1.6second Time_1_1p3_s = the exposure time is 1/1.3 second Time_1_s = the exposure time is 1 second
Comments:	When automatic exposure control is disabled (see the ExposureMode parameter above), the Exposure Time parameter sets the exposure time in fractions of a second. If automatic exposure control is enabled, the ExposureTime parameter is read only and will indicate the current exposure time in fractions of a second as set by the automatic exposure control. Note that you can also use the ExposureTime_us parameter (see below) to set the exposure time in microseconds. Depending on your personal preference, you can set either the ExposureTime parameter or the ExposureTime_us parameter. When you set one of these parameters, the other is automatically set to an equivalent value.

Parameter:	ExposureTime_us
Data Type:	Unsigned integer
Valid Values:	Varies by camera model and depends on the current frame rate. Use the GetMin and GetMax methods to determine the range.
Comments:	When automatic exposure control is disabled (see the ExposureMode parameter above), the ExposureTime_us parameter sets the exposure time in microseconds. If automatic exposure control is enabled, the ExposureTime_us parameter is read only and will indicate the current exposure time in microseconds as set by the automatic exposure control. Note that you can also use the ExposureTime parameter (see above) to set the exposure time in fractions of a second. Depending on your personal preference, you can set either the ExposureTime parameter or the ExposureTime_us parameter. When you set one of these parameters, the other is automatically set to an equivalent value.

Parameter:	Gain_dB
Data Type:	Enum
Valid Values:	Gain_0_dB = the gain is 0 dB Gain_2_dB = the gain is 2 dB Gain_4_dB = the gain is 4 dB Gain_6_dB = the gain is 6 dB Gain_8_dB = the gain is 8 dB Gain_10_dB = the gain is 10 dB Gain_12_dB = the gain is 12 dB Gain_14_dB = the gain is 14 dB Gain_16_dB = the gain is 16 dB Gain_18_dB = the gain is 18 dB Gain_20_dB = the gain is 20 dB Gain_22_dB = the gain is 22 dB Gain_24_dB = the gain is 24 dB
Comments:	When automatic gain control is disabled (see the ExposureMode parameter above), the Gain_dB parameter sets the gain in dB. If automatic gain control is enabled, the Gain_dB parameter is read only and will indicate the current gain in dB as set by the automatic gain control. Note that you can also use the Gain parameter (see below) to set the gain as an integer value. Depending on your personal preference, you can set either the Gain_dB parameter or the Gain parameter. When you set one of these parameters, the other is automatically set to an equivalent value.

Parameter:	Gain
Data Type:	Unsigned integer
Valid Values:	Varies by camera model. Use the GetMin and GetMax methods to determine the range.
Comments:	When automatic gain control is disabled (see the ExposureMode parameter above), the Gain parameter sets the gain as in integer value. The higher you set the integer value, the higher the gain. If automatic gain control is enabled, the Gain parameter is read only and will indicate the current gain value as set by the automatic gain control. Note that you can also use the Gain_dB parameter (see above) to set the gain in dB. Depending on your personal preference, you can set either the Gain parameter or the Gain_dB parameter. When you set one of these parameters, the other is automatically set to an equivalent value.

Parameter:	Sharpness
Data Type:	Signed integer
Valid Values:	-40 to 100
Comments:	Sets the sharpness of the captured images. Higher settings produce sharper images.

Parameter:	Saturation
Data Type:	Unsigned integer
Valid Values:	0 to 100
Comments:	Sets the color saturation of the images transmitted by the camera. Higher settings produce more saturated (colorful) images.

Parameter:	Gamma
Data Type:	Unsigned integer
Valid Values:	50 to 150
Comments:	Sets the degree of gamma correction applied to captured images. Gamma corrects the captured images for non-linearities in the human eye's perception of brightness. A setting of 50 represents a gamma correction of 0.5. A setting of 100 represents a gamma correction of 1. And a setting of 150 represents a gamma correction of 1.5.

Parameter:	IrisMode
Data Type:	Enum
Valid Values:	Auto = the iris will be automatically controlled by the camera. Open = the iris is fully open. Closed = the iris is fully closed.
Comments:	Sets the iris functionality if the camera is equipped with a DC iris. The Open and Closed settings can be used to test the functionality of an iris mechanism.

Parameter:	BacklightCompensation
Data Type:	Unsigned Integer
Valid Values:	0 = compensation is disabled. 1 = compensation is enabled.
Comments:	Enables or disables the camera's backlight compensation feature. This feature automatically compensates when the main lighting comes from behind the image subject.

Parameter:	AutoControlsMask
Data Type:	String
Valid Values:	See Section 3 on page 13
Comments:	Determines which areas of the images captured by the sensor will be used by the automatic exposure, gain, and iris controls when the controls are calculating how to adjust the camera. Active blocks within the mask will be used. Inactive blocks will not be used. Note that if the privacy mask overlays the auto controls mask, the area of the auto controls mask under the privacy mask will still be used for auto control. The camera will not use any part of the auto controls mask that is outside of the sensor AOI. If the entire auto controls mask is outside of the sensor AOI, the automatic controls will not work.

Parameter:	WhiteBalanceMode
Data Type:	Enum
Valid Values:	Auto = white balance will be set for normal lighting conditions. Auto_2 = the camera will attempt to identify the type of lighting present (i.e., daylight, incandescent, fluorescent, etc.) and then will automatically adjust the white balance based on the lighting type detected. This selection works best when the lighting conditions are uniform. Daylight = white balance will be set for outdoor lighting. Incandescent = white balance will be set for incandescent lighting. Fluorescent_1 = white balance will be set for normal fluorescent lighting. Fluorescent_2 = white balance will be set for bright fluorescent lighting. Manual = white balance will be determined by the values for the WhitePointX, and WhitePointY parameters.
Comments:	Sets the camera's white balance mode.

Parameter:	WhitePointX
Data Type:	Unsigned integer
Valid Values:	1 to 999
Comments:	If the WhiteBalanceMode parameter (see above) is set to Manual, the WhitePointX parameter sets the red/cyan balance in the captured images. Decreasing the setting makes the images more red, and increasing the setting makes the images more cyan. If the WhiteBalanceMode parameter is set to value other than manual, the WhitepointX parameter will be read only and will indicate the current whitepoint X value as set by the automatic white balance control.

Parameter:	WhitePointY
Data Type:	Unsigned integer
Valid Values:	1 to 999
Comments:	If the WhiteBalanceMode parameter (see above) is set to Manual, the WhitePointY parameter sets the green/purple balance in the captured images. Decreasing the setting makes the images more green, and increasing the setting makes the images more purple. If the WhiteBalanceMode parameter is set to value other than manual, the WhitepointY parameter will be read only and will indicate the current whitepoint Y value as set by the automatic white balance control.

Parameter:	WhiteBalanceMask
Data Type:	String
Valid Values:	See Section 3 on page 13
Comments:	Determines which areas of the captured images will be used to control white balancing. Active blocks within the mask will be used. Inactive blocks will not be used. Note that if the privacy mask overlays the white balance mask, the areas of the white balance mask under the privacy mask will still be used for auto control. The camera will not use any part of the white balance mask that is outside of the sensor AOI. If the entire white balance mask is outside of the sensor AOI, automatic white balancing will not work.

Parameter:	PrivacyMask
Data Type:	String
Valid Values:	See Section 3 on page 13
Comments:	Determines which areas of the images captured by the sensor will be blacked out to maintain privacy. Active blocks within the mask will be blacked out. Inactive blocks will not be blacked out.

Parameter:	IRFilterMode
Data Type:	Enum
Valid Values:	Auto = the camera automatically senses the change from night to day or from day to night and sets the position of the camera's IR-cut filter accordingly. Open = the IR-cut filter is moved to the open position (filter is not in front of the camera's sensor) and kept there. Closed = the IR-cut filter is moved to the closed position (filter is in front of the camera's sensor) and kept there. InputPin0Controlled = the position of the IR-cut filter will be controlled by the state of input pin 0. If input pin 0 is active, the filter will be in the open position. If input pin 0 is inactive, the filter will be in the closed position.
Comments:	Sets the functionality of the camera's IR-cut filter. Note that this parameter is only available on day/night cameras. For detailed information about the IR-cut filter, see the camera User's Manual.

Parameter:	IRFilterState
Data Type:	Enum
Valid Values:	Unknown = the position of the IR-cut filter is unknown. This state is typically indicated for a short period of time immediately after bootup and lasts until the camera has placed the filter into one of the defined positions (i.e., either open or closed). Open = the IR-cut filter is in the open position (filter is not in front of the camera's sensor). Closed = the IR-cut filter is in the closed position (filter is in front of the camera's sensor).
Comments:	Read only. Indicates the current position of the of the camera's IR-cut filter. Note that this parameter is only available on day/night cameras. For detailed information about the IR-cut filter, see the camera User's Manual.

Parameter:	IRFilterSwitchLevel
Data Type:	Signed integer
Valid Values:	-100 to +100
Comments:	When the IR Filter Mode parameter (see above) is set to auto, the IR Filter Switch Level setting is mainly used to adjust when the camera will switch from day mode to night mode. The higher the IR Filter Switch Level setting, the darker it must be before the camera will make the switch. Setting the switch level to a higher value typically means that the camera will switch from day mode to night mode later in the day, i.e., when it is darker. If the current level of darkness as indicated by the IR Filter Current Level parameter (see below) becomes greater than the switch level setting and remains there for a time period longer than the IR Filter Wait Time (see below), the camera will switch from day mode to night mode. If the current level of darkness as indicated by the IR Filter Current Level parameter (see below) becomes less than the switch level setting and remains there for a time period longer than the IR Filter Wait Time (see below), the camera will switch from night mode to day mode.

Parameter:	IRFilterCurrentLevel
Data Type:	Signed integer
Comments:	Read only. Indicates the current level of darkness as measured by the camera's auto controls. As the area being viewed by the camera gets darker, the value of the IR Filter Current Level will rise (a high positive value indicates that the area being viewed is very dark). As the area being viewed by the camera becomes brighter, the value of the IR Filter Current Level will fall (a large negative value indicates that the area being viewed is very bright).

Parameter:	IRFilterWaitTime
Data Type:	Unsigned integer
Valid Values:	0 to 3600
Comments:	Sets the amount of time in seconds that the value of the IRFilterCurrentLevel (see above) must continuously remain above the IRFilterSwitchLevel before the camera will switch from day mode to night mode or the amount of time that the IR Filter Current Level must continuously remain below the IR Filter Switch Level before the camera will switch from night mode to day mode.

Parameter:	IRFilterAnnounceMode
Data Type:	Enum
Valid Values:	Off = the IR-cut filter announce feature is disabled. OutputPin0 = the camera will announce the current position of the IR-cut filter by setting the state of output pin 0. When the IR-cut filter is in the open position (filter is not in front of the sensor), output pin 0 will be set to active. And when the IR-cut filter is in the closed position (filter is in front of the sensor), output pin 0 will be set to inactive.
Comments:	Sets the mode of the camera's IR-cut filter announce feature. Note that this parameter is only available on day/night cameras. For detailed information about the IR-cut filter, see the camera User's Manual.

4.4 Stream Group

The parameters in this group are used to configure the camera's image streams.

Some of the parameters in this group are used to set the "area of interest" (AOI) for each video stream. The AOI settings let you define an area within each captured image and only the pixel data from the defined area will be encoded and streamed. You can set the stream AOI settings so that the entire captured image is encoded and streamed or so that just a portion of the captured image is encoded and streamed.

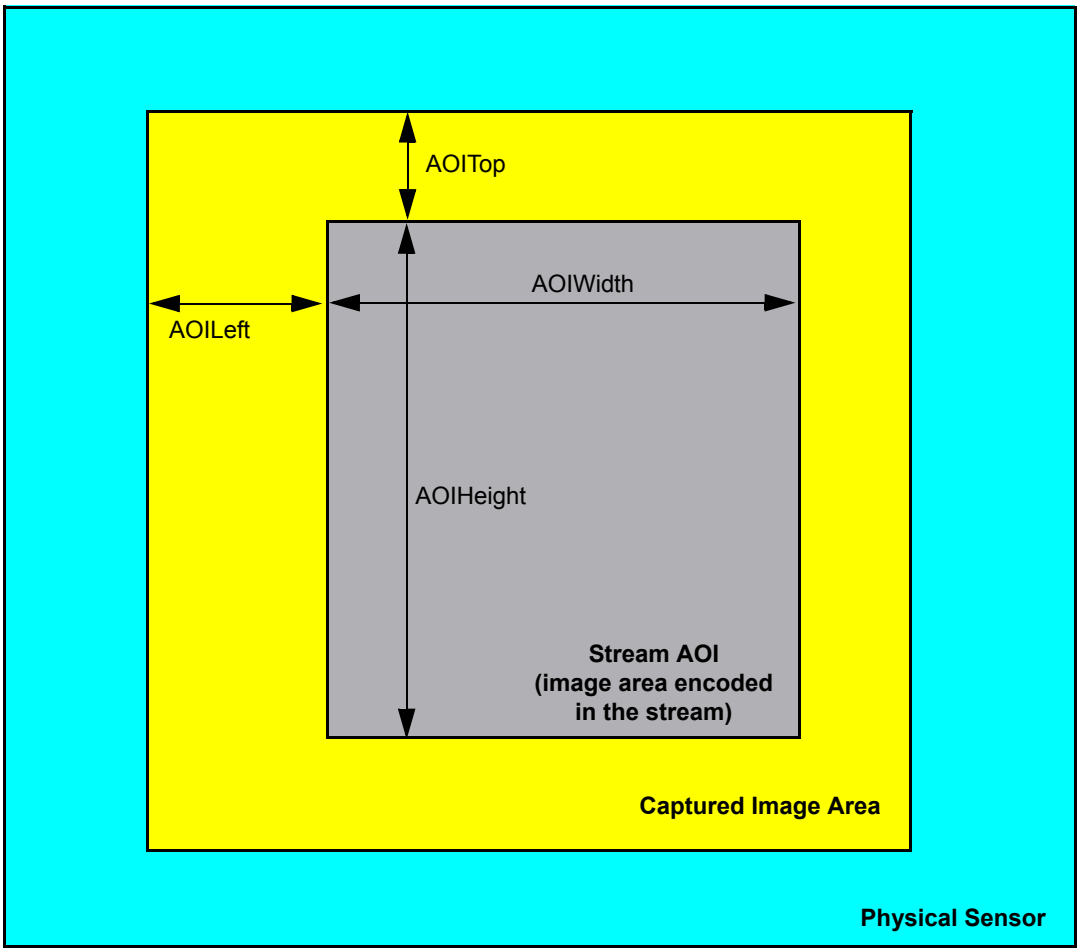

Fig. 3: Stream AOI

Parameter:	StreamSelector
Data Type:	Unsigned integer
Valid Values:	0 = stream 0. 1 = stream 1. 2 = stream 2.
Comments:	Selects an image stream to work with. **All changes made to the other parameters in this group will affect the stream that has been selected by the stream selector parameter.**

Parameter:	EncoderType
Data Type:	Enum
Valid Values:	Off = stream disabled. JPEG = stream enabled and set for motion JPEG encoding. MPEG4 = stream enabled and set for MPEG4 encoding. H_264 = stream enabled and set for H.264 encoding.
Comments:	Enables or disables the selected stream and sets the encoder type for the stream. Stream 0 can be set for JPEG, MPEG4, or H.264 encoding. All other streams can only be set for JPEG encoding. Stream 1 must be enabled before you can enable stream 2. Stream 2 can not be enabled if stream 1 is disabled. If stream 1 and 2 are both enabled and you subsequently disable stream 1, stream 2 will automatically be disabled. **The EncoderType parameter value can not be changed when the camera is in normal operation mode.** It can only be changed when the camera is in configure operation mode (see Section 4.1 on page 15).

Parameter:	EncoderMode
Data Type:	Enum
Valid Values:	CBR = constant bit rate. The encoder attempts to maintain a constant output bit rate by adjusting image quality as necessary. The bit rate will be determined by the setting of the Bitrate parameter. VBR = variable bit rate. The encoder attempts to maintain constant image quality by adjusting the output bit rate as necessary. The quality level will be determined by the setting of the Quality parameter.
Comments:	Sets the encoder mode for the selected stream.

Parameter:	Bitrate
Data Type:	Unsigned integer
Valid Values:	Use the GetMin and GetMax methods to determine the range.
Comments:	If the EncoderMode parameter is set to CBR, the Bitrate parameter sets the bit rate for the selected stream.

Parameter:	Quality
Data Type:	Unsigned integer
Valid Values:	Use the GetMin and GetMax methods to determine the range.
Comments:	If the EncoderMode parameter is set to VBR, the Quality parameter sets the quality level for the selected stream. Higher values equal better quality. Note that the effect of the quality setting is not precisely equivalent for each encoder type. For example, a quality setting of 50 will have a slightly different effect when the Encoder Type parameter is set to JPEG than it will have when it is set to MPEG4 or to H.264.

Parameter:	GopLength_ms
Data Type:	Unsigned integer
Valid Values:	Use the GetMin and GetMax methods to determine the range.
Comments:	If the Encoder Type parameter is set to MPEG4 or H.264, then the GOPLength_ms parameter sets the time between I-frames in milliseconds for the selected stream. In an MPEG4 or an H.264 stream, the camera transmits periodic I-frames and transmits several P-frames between each I-frame. I-frames carry complete information for a captured image. P-frames only carry information about the areas of the image that have changed since the last I-frame was transmitted. The time between the transmission of I-frames is set by the GOP Length parameter. Increasing the time between I-frames (i.e., the GOP length) will increase the efficiency of the encoder. But be aware that increasing the GOP length also increases the latency when you start an image stream because the decoder in the receiving device must wait longer for the initial I-frames.

Parameter:	AOIWidth
Data Type:	Unsigned integer
Valid Values:	Must be less than or equal to the current setting for the **sensor** AOIWidth parameter.
Comments:	As shown in Figure 3 on page 30, sets the width of the image area (in pixels) that will be encoded in the selected stream. **The AOIWidth parameter value can not be changed when the camera is in normal operation mode.** It can only be changed when the camera is in configure operation mode (see Section 4.1 on page 15).

Parameter:	AOIHeight
Data Type:	Unsigned integer
Valid Values:	Must be less than or equal to the current setting for the **sensor** AOIHeight parameter.
Comments:	As shown in Figure 3 on page 30, sets the height of the image area (in pixels) that will be encoded in the selected stream. **The AOI Height parameter value can not be changed when the camera is in normal operation mode.** It can only be changed when the camera is in configure operation mode (see Section 4.1 on page 15).

Parameter:	AOILeft
Data Type:	Unsigned integer
Valid Values:	The total of the stream AOIWidth parameter setting plus the stream AOILeft parameter setting must be less than or equal to the current setting of the **sensor** AOIWidth parameter.
Comments:	As shown in Figure 3 on page 30, sets the distance (in pixels) between the left side of the captured image and the left side of the area that will be encoded in the selected stream.

Parameter:	AOITop
Data Type:	Unsigned integer
Valid Values:	The total of the stream AOIHeight parameter setting plus the stream AOITop parameter setting must be less than or equal to the current setting of the **sensor** AOIHeight parameter.
Comments:	As shown in Figure 3 on page 30, sets the distance (in pixels) between the top of the captured image and the top of the area that will be encoded in the selected stream.

Parameter:	OutputSize
Data Type:	Enum
Valid Values:	Size_1024x768 = the size of the images in the stream will be 1024 x 768 (XGA) Size_800x600 = the size of the images in the stream will be 800 x 600 (SVA) Size_720x576 = the size of the images in the stream will be 720 x 576 (D1 PAL) Size_720x480 = the size of the images in the stream will be 720 x 480 (D1 NTSC) Size_704x576 = the size of the images in the stream will be 704 x 576 (4CIF) Size_640x470 = the size of the images in the stream will be 640 x 480 (VGA) Size_480x360 = the size of the images in the stream will be 470 x 360 Size_352x288 = the size of the images in the stream will be 352 x 288 (CIF) Size_320x240 = the size of the images in the stream will be 320 x 240 (QVGA) Size_176x144 = the size of the images in the stream will be 176 x 144 (QCIF) Size_160x120 = the size of the images in the stream will be 160 x 120 (QQVGA)
Comments:	Sets the images output in the selected stream to a standard size. When you select a size, the camera checks the setting of the OutputScaling parameter. It then automatically sets the stream AOILeft, stream AOITop, stream AOIWidth, and stream AOIHeight parameters so that the AOI will be centered on the sensor and set to the right width and height to result in output images of the size you selected. The valid values will vary by camera model and will vary depending on how the OutputScaling parameter is set. Use the GetEntries method to determine the current valid settings. **The OutputSize parameter value can not be changed when the camera is in normal operation mode.** It can only be changed when the camera is in configure operation mode (see Section 4.1 on page 15).

Parameter:	OutputScaling
Data Type:	Enum
Valid Values:	Scale_1_1 = do not rescale images. Scale_1_2 = rescale images to 1/2 size. Scale_1_4 = rescale images to 1/4 size. Scale_1_8 = rescale images to 1/8 size.
Comments:	Sets the amount that encoded images in the selected stream will be rescaled before they are transmitted in the selected stream. **The OutputScaling parameter value can not be changed when the camera is in normal operation mode.** It can only be changed when the camera is in configure operation mode (see Section 4.1 on page 15).

Parameter:	FrameRateScaling
Data Type:	Enum
Valid Values:	FpsScale_1_1 = every image captured will be encoded and streamed. FpsScale_1_2 = every second image captured will be coded and streamed. FpsScale_1_4 = every fourth image captured will be coded and streamed. FpsScale_1_8 = every eighth image captured will be coded and streamed.
Comments:	Sets the ratio of captured to encoded images for the selected stream. **The FrameRateScaling parameter value can not be changed when the camera is in normal operation mode.** It can only be changed when the camera is in configure operation mode (see Section 4.1 on page 15)

Parameter:	OverlayText
Data Type:	String
Valid Values:	The string can include text and also the following variables: $date$ = display current date/time (see Section 4.10 on page 51 to set the format). $timestamp$ = display timestamp (sec:usec since 1970). $counter$ = display frame counter. $motion$ = display motion/no motion (no motion = blank space, motion = *). $motion_n$ = display motion/no motion (no motion = blank space, motion = *) in region n. $motion_level$ = display current motion level (number of changed pixels) in region 0. $motion_levels$ = display current motion levels (number of changed pixels) in all regions. $motion_level_n$ = display current motion level (number of changed pixels) in region n. $frame_size$ = display the width and height of the sensor AOI. $frame_position$ = display the left offset and top offset for the sensor AOI. $alarm$ = display if an alarm condition has been declared (no alarm = blank space, alarm = *). $alarm_nr$ = display alarm number (if any). fps = display the current frame rate for this stream. $SysInfo.ModelName$ = display the camera's model name. $SysInfo.FirmwareVersion$ = display the camera's firmware version info. $SysInfo.ManName$ = display the camera vendor's name. $SysInfo.Serial$ = display the camera's serial number. $SysInfo.MACAddress$ = display the camera's MAC address. $System.DateTimeFormat$ = display the current date/time format setting. $Network.RxTraffic$ = display the current incoming network traffic level in Kbits/s. $Network.TxTraffic$ = display the current outgoing network traffic level in Kbits/s. $Network.HostName$ = display the camera's host name.
Comments:	Sets the text that will appear as an overlay on the selected stream. Note that the $Network.RxTraffic$ and $Network.TxTraffic$ are returned in Kbits/s starting with version 1.4 firmware. On cameras with older firmware, these two values are returned in KBytes/s.

Parameter:	OverlayPosition
Data Type:	Enum
Valid Values:	Top = the text overlay bar will appear at the top of the streamed images. Bottom = the text overlay bar will appear at the bottom of the streamed images.
Commenst:	Sets the position of the text overlay for the selected stream.

Parameter:	LiveBufferSize
Data Type:	Unsigned integer
Valid Values:	Min = 2048 KB Max = see note on page 37.
Comments:	Sets the size of the "live image" buffer for the selected stream. The live image buffer is a FIFO buffer that stores the last N captured images for the selected stream. (N depends on the size of the images being encoded and the size of the buffer.) Each enabled stream has a live buffer and the live buffer must be set to a minimum of 2048 KB. Each enabled stream also has an alarm buffer, but the alarm buffer can be disabled. Alarms have no effect on the operation of a live buffer. For information about viewing images in the live buffer, see Section 5 on page 55. **The LiveBufferSize parameter value can not be changed when the camera is in normal operation mode.** It can only be changed when the camera is in configure operation mode (see Section 4.1 on page 15).

Parameter:	AlarmBufferSize
Data Type:	Unsigned integer
Valid Values:	Min = 0 KB Max = see note on page 37.
Comments:	Sets the size of the alarm buffer (in KB) for the selected stream. When an alarm buffer is enabled, it operates in the following manner: If the camera is operating normally and no alarm condition has been declared, the alarm buffer simply works as a FIFO buffer that stores captured images. When an alarm is declared, the alarm buffer will continue to buffer images until the portion of the buffer that is allocated for post alarm image storage is full (see the next parameter). At that point, buffering will stop. The stored images will be held in the buffer until a new AlarmBufferArm command is issued. For information about viewing images in the alarm buffer, see Section 5 on page 55. **The AlarmBufferSize parameter value can not be changed when the camera is in normal operation mode.** It can only be changed when the camera is in configure operation mode (see Section 4.1 on page 15).

Parameter:	PostAlarmBufferSize
Data Type:	Unsigned integer
Valid Values:	Min = 0 KB Max = current setting for the AlarmBufferSize parameter.
Comments:	Sets the portion of the alarm buffer (in KB) that will be used for "post alarm" image storage. For example, if the AlarmBufferSize is set to 2048 KB and the PostAlarmBufferSize is set to 1228 KB, then 1228 KB (i.e., 60%) of the alarm buffer will be allocated for holding post alarm images. **The PostAlarmBufferSize parameter value can not be changed when the camera is in normal operation mode.** It can only be changed when the camera is in configure operation mode (see Section 4.1 on page 15).

Note

The live buffer for each stream must be a minimum of 2048 KB. The alarm buffer for each stream can be either be set to 0, or to a value greater than or equal to 2048 KB. There is no fixed maximum size for any buffer, however, the sum of the buffer sizes cannot exceed a certain maximum. This maximum can vary and you can determine the current maximum by placing the camera in configure mode and requesting a GetValue for the LiveBufferSize parameter, i.e.:

http://IPCam_1/cgi-bin/
param_if.cgi?NumActions=1&Action_0=Global.OperationMode.SetValue
&Parameter_0_0=Configure

http://IPCam_1/cgi-bin/
param_if.cgi?NumActions=1&Action_0=Stream.LiveBufferSize.GetMax

This will return the maximum allowed **total** buffer size (in KB), not just the maximum for the live buffers as you may guess. For example, assume that you issued these two requests and the return for the GetValue was 50000. In this case, the total size of live buffer stream 0 + live buffer stream 1 + live buffer stream 2 + alarm buffer stream 0 + alarm buffer stream 1 + alarm buffer stream 2 must be a maximum of 50000 KB.

Parameter:	AlarmBufferState
Data Type:	Enum
Valid Values:	Off = the alarm buffer is disabled. Arming = the alarm buffer is in the process of being armed. Armed = the alarm buffer is armed and ready to react to an alarm. Active = the alarm buffer is now buffering post alarm images. Done = the alarm buffer has finished buffering post alarm images and has stopped buffering.
Comments:	Read only. Indicates the current condition of the alarm buffer for the selected stream.

Parameter:	AlarmBufferDisable
Data Type:	Command
Comments:	Disables the alarm buffer for the selected stream.

Parameter:	AlarmBufferArm
Data Type:	Command
Comments:	Arms the alarm buffer and makes it begin buffering. (If the alarm buffer is disabled, this command will also enable the buffer.)

4.5 Motion Group

The parameters in this group are used to set the behavior of the camera's motion detection functionality.

Parameter:	MotionDetectionMode
Data Type:	Enum
Valid Values:	Off = motion detection is disabled. On = motion detection is enabled.
Comments:	Enables motion detection. Motion detection uses the History Reference Difference method. The camera detects motion by calculating the difference between the pixels in the current frame and the pixels in a "history" frame that is an average of the last several images.

Parameter:	HistoryImageFrames
Data Type:	Unsigned integer
Valid Values:	1 to 5
Comments:	Determines the number of frames that will be averaged to make the "history" image. This setting represents a power of 2. For example, if the value is set to 3, then the past 2^3 frames (i.e., 8 frames) will be used.

Parameter:	Granularity
Data Type:	Unsigned integer
Valid Values:	1 to 16
Comments:	Determines which pixels from each captured image will be used to determine the difference between the current image and the history image. A setting of 1 means all pixels will be used. A setting of 2 means that every second pixel in each row and every second pixel in each column of the image pixels will be used. A setting of 3 means that every third pixel in each row and every third pixel in each column will be used. Etc.

Parameter:	ShowMotion
Data Type:	Enum
Valid Values:	Off = motion display is disabled. On = motion display is enabled.
Comments:	When ShowMotion is set to on, pixels in the areas where motion has been detected will be displayed in green.

Parameter:	RegionSelector
Data Type:	Unsigned integer
Valid Values:	0 to 4
Comments:	Up to 5 separate motion detection regions can be defined. The region selector is used to select a region to work with. **All of the following parameters will apply to the selected region.**

Parameter:	Mask
Data Type:	String
Valid Values:	See Section 3 on page 13
Comments:	Determines which areas of the captured images will be included in the selected motion detection region. Active blocks within the mask will be included. Inactive blocks will not be used.

Parameter:	Sensitivity
Data Type:	Unsigned integer
Valid Values:	1 to 100
Comments:	Sets the degree of difference that must be present between a pixel in the motion detection region of the current image and the corresponding pixel in the history image for a change in the pixel to be detected. Higher settings make motion detection for the region less sensitive.

Parameter:	MotionThreshold
Data Type:	Unsigned integer
Valid Values:	Varies by camera model. Use the GetMin and GetMax methods to determine the range.
Comments:	Sets a threshold for motion detection. If the number of changed pixels in the motion detection region is above the threshold and below the limit (see next parameter), motion will be detected. The units for this parameter are 1/100000 of the number of pixels in the camera's sensor. For example, if your camera has a 1024 x 768 pixel sensor, then the units would be 1024 x 768 x 1/1000000 = 7.9 pixels (round up to 8). So in this case, if you set the MotionThreshold parameter to 1, the threshold would be 8, and more than 8 pixels in the region must change for the threshold to be exceeded. If you set it to 2, the threshold would be 16, and more than 16 pixels must change for the threshold to be exceeded. And so on.

Parameter:	MotionLimit
Data Type:	Unsigned integer
Valid Values:	Varies by camera model. Use the GetMin and GetMax methods to determine the range.
Comments:	Sets a limit for motion detection. If the number of changed pixels in the motion detection region is above the threshold (see previous parameter) and below the limit, motion will be detected. The units for this parameter are 1/100000 of the number of pixels in the camera's sensor. For example, if your camera has a 1024 x 768 pixel sensor, then the units would be 1024 x 768 x 1/1000000 = 7.9 pixels (round up to 8). So in this case, if you set the MotionLimit parameter to 1000, the limit would be 8000, and less than 8000 pixels in the region must change to be below the limit. If you set it to 2000, the limit would be 16000, and less than 16000 pixels must change to be below the limit. And so on.

Parameter:	AlarmOnDelay
Data Type:	Unsigned integer
Valid Values:	0 to 86400000
Comments:	Sets the amount of time (in milliseconds) that continuous motion must be detected in order for an alarm to be declared.

Parameter:	AlarmOffDelay
Data Type:	Unsigned Integer
Valid Values:	0 to 86400000
Comments:	Sets the amount of time (in milliseconds) that no motion must be detected in order for a declared alarm to be ended.

4.6 Streaming Group

The parameters in this group are used to configure the camera's network services.

Parameter:	Commit
Data Type:	Command
Comments:	Changes to the values for the parameters in this group will become active when the Commit command is issued.

Parameter:	Revert
Data Type:	Command
Comments:	Reverts the values of the parameters in this group to what they were when the last Commit command was issued.

Parameter:	Enabled
Data Type:	Unsigned Integer
Valid Values:	0 = RTP streaming disabled. 1 = RTP streaming enabled.
Comments:	Enables or disables RTP streaming.

Parameter:	RTSPPort
Data Type:	Unsigned integer
Valid Values:	0 to 65534
Comments:	Sets the camera's RTSP port number. Default = 554.

Parameter:	Multicast
Data Type:	Unsigned Integer
Valid Values:	0 = multicast streaming disabled. 1 = multicast streaming enabled.
Comments:	Enables or disables multicast streaming. Note that multicast streaming can only be used for a stream where the EncoderType parameter (see page 31) is set to MPEG4 or H.264. (Since only stream 0 can be set to MPEG4 or H.264, this limits multicasting to stream 0.)

Parameter:	MulticastOnDemand
Data Type:	Unsigned Integer
Valid Values:	0 = on-demand multicast streaming disabled. 1 = on-demand multicast streaming enabled.
Comments:	Enables or disables on-demand multicast streaming. Note that on-demand multicast streaming can only be used when multicast streaming is enabled, i.e., the Multicast parameter (see the previous page) is set to 1. If multicast streaming is enabled and on-demand multicast streaming is disabled, the camera begins streaming to the multicast IP address as soon as multicast streaming is enabled and continues to stream to the multicast address until multicast streaming is disabled. If multicast streaming and on-demand multicast streaming are both enabled, the camera begins streaming to the multicast IP address when the first client issues an RTSP "PLAY" request. The camera continues streaming until: - the last client closes its session via an RTSP "TEARDOWN" request, or - the keep-alive check for the last client comes into effect (more specifically: until the RTSP server in the camera does not see an RTSP or RTCP packet within a client session for 65 seconds).

Parameter:	MulticastIP
Data Type:	String
Valid Values:	Valid IP address.
Comments:	Sets the IP address for multicast streaming.

Parameter:	MulticastPort
Data Type:	Unsigned Integer
Valid Values:	0 to 65534
Comments:	Sets the UDP port for multicast streaming.

Parameter:	MulticastTTL
Data Type:	Unsigned Integer
Valid Values:	0 to 255
Comments:	Sets the multicast Time-To-Live (TTL). The multicast Time-To-Live (TTL) value specifies the number of routers (hops) that multicast traffic is permitted to pass through before expiring on the network.

4.7 Network Group

The parameters in this group are used to configure the camera's IP configuration.

Parameter:	Commit
Data Type:	Command
Comments:	Changes to the values for the parameters in this group will become active when the Commit command is issued.

Parameter:	Revert
Data Type:	Command
Comments:	Reverts the values of the parameters in this group to what they were when the last Commit command was issued.

Parameter:	DHCP
Data Type:	Unsigned integer
Valid Values:	0 = DHCP disabled. 1 = DHCP enabled.
Comments:	Enables or disables camera IP addressing via a DHCP server.

Parameter:	HostName
Data Type:	String
Valid Values:	Only letters, digits, and dashes are allowed.
Comments:	Assigns a host name to the camera.

Parameter:	IPAddress
Data Type:	String
Valid Values:	Any valid IP address.
Comments:	Assigns an IP address to the camera that will be used if DHCP is disabled.

Parameter:	HTTPPort
Data Type:	Unsigned integer
Valid Values:	1 to 65535
Comments:	Sets the HTTP port for the camera.

Parameter:	NetPrefix
Data Type:	Integer
Valid Values:	1 to 32
Comments:	If DHCP is disabled, specifies the number of bits which represent the netmask for your network. For example: 16 = 255.255.0.0 24 = 255.255.255.0

Parameter:	Gateway
Data Type:	String
Valid Values:	Any valid IP address
Comments:	Sets the gateway the camera will use if DHCP is disabled.

Parameter:	NameServer
Data Type:	String
Valid Values:	Any valid IP address
Comments:	Sets the name server the camera will use if DHCP is disabled.

Parameter:	RxTraffic
Data Type:	Integer
Comments:	Read only. Indicates the amount of incoming network traffic in kilobits per second. **Note:** The amount of traffic is indicated in kilobits per second on cameras with version 1.4 firmware and up. On cameras with earlier firmware, the traffic is indicated in kilobytes per second.

Parameter:	TxTraffic
Data Type:	Integer
Comments:	Read only. Indicates the amount of outgoing network traffic in kilobits per second. **Note:** The amount of traffic is indicated in kilobits per second on cameras with version 1.4 firmware and up. On cameras with earlier firmware, the traffic is indicated in kilobytes per second.

4.8 Alarm Group

The parameters in this group set the behavior of the camera's alarm state functionality.

Parameter:	SourceSelector
Data Type:	Enum
Valid Values:	User = alarm declared via the UserTrigger command. PIO = alarm declared via an active signal on the camera's input pin. MotionDetection = alarm declared via motion detection.
Comments:	Selects an alarm source to enable or disable (see the next parameter).

Parameter:	SourceEnable
Data Type:	Unsigned integer
Valid Values:	0 = the selected alarm source is disabled 1 = the selected alarm source is enabled
Comments:	Enables or disables the selected alarm source.

Parameter:	ActionSelector
Data Type:	Enum
Valid Values:	PIO = make the signal on the camera's output pin active. Email = send an Email to the recipient specified by the Email parameter. HTTP = send an HTTP request to the URL specified by the HTTPURL parameter. FTP = send an upload will to the FTP server specified by the FTPServer parameter.
Comments:	Selects an alarm action to enable or disable (see the next two parameters).

Parameter:	ActionEnable
Data Type:	Unsigned integer
Valid Values:	0 = selected alarm action is disabled. 1 = selected alarm action is enabled.
Comments:	Enables or disables the selected alarm action. When an alarm is declared, all enabled actions will be executed.

Parameter:	ActionIncludeImg
Data Type:	Unsigned integer
Valid Values:	0 = do not include an image. 1 = include an image.
Comments:	Enables or disables the inclusion of an image within the selected alarm action (if the selected action is capable of including an image).

Parameter:	PIOHoldTime
Data Type:	Unsigned integer
Valid Values:	0 to 3600000
Comments:	If the PIO action is enabled, the PIOHoldTime parameter sets the time that the camera's output pin will remain active after an alarm is declared. Note that if the PIOHoldTime is set to 0, the output pin will remain active only as long as the alarm condition remains active.

Parameter:	UserTrigger
Data Type:	Command
Comments:	If "User" has been enabled as an alarm source, issuing the UserTrigger command will declare an alarm.

Parameter:	HTTPURL
Data Type:	String
Valid Values:	Any valid URL request.
Comments:	If the HTTP action has been enabled, the HTTPURL parameter sets the URL request that will be issued when an alarm is declared. You could, for example, enter this URL request: http://MyServer/cgi-bin/alarm.cgi You should be aware that the camera will automatically add the following two parameters to the end of the request: ?host=<hostname>&date=<date/time> where the hostname is the camera's host name and is typically something like this: "Basler-20809681" and the date/time is formatted like this: "2008-07-03 16:30:41 CEST". The string will be transmitted URL encoded

Parameter:	Email
Data Type:	String
Valid Values:	Any valid email address.
Comments:	If the Email action has been enabled, the Email parameter sets the recipient for the email that will be sent when an alarm is declared. A sample of the email text appears on the next page. If the ActionIncludeImg parameter is enabled, the email will include a JPEG image captured at the time of the alarm.

Parameter:	EmailServer
Data Type:	String
Valid Values:	Any valid SMTP server address.
Comments:	Sets the SMTP server to use to send the email.

Parameter:	EmailPort
Data Type:	Unsigned integer
Valid Values:	1 to 65535
Comments:	Sets the port to use on the target email server.

Parameter:	EmailUserName
Data Type:	String
Valid Values:	Up to 15 numbers and/or letters (upper or lower case).
Comments:	Specifies a user name for authentication on the SMTP server.

Parameter:	EmailPassword
Data Type:	String
Valid Values:	Up to 29 characters. All standard keyboard characters are valid.
Comments:	Specifies a password for authentication on the SMTP server.

Parameter:	FTPServer
Data Type:	String
Valid Values:	Any valid FTP server IP address.
Comments:	If the FTP action has been enabled, the FTP parameter sets the recipient for the upload that will be sent when an alarm is declared. The upload will be a text file whose contents are similar to the sample email text shown below. If the ActionIncludeImg parameter is enabled, a second file will be send that includes a JPEG image captured at the time of the alarm.

Parameter:	FTPPort
Data Type:	Unsigned integer
Valid Values:	1 to 65535
Comments:	Sets the port to use on the target FTP server.

Parameter:	FTPUserName
Data Type:	String
Valid Values:	Up to 15 numbers and/or letters (upper or lower case).
Comments:	Specifies a user name for authentication on the FTP server.

Parameter:	FTPPassword
Data Type:	String
Valid Values:	Up to 29 characters. All standard keyboard characters are valid
Comments:	Specifies a password for authentication on the FTP server.

Sample Email

An Alarm has been triggered by the camera named: Basler-20802071

Time: Wed Jun 25 22:28:15 UTC 2008

This is alarm number: 39

Triggered by User: 0

Triggered by PIO: 1

Triggered by Motion: 0

Motion Status: 0

- in Region 0: 0 (Level: 645)
- in Region 1: 0 (Level: 0)
- in Region 2: 0 (Level: 0)
- in Region 3: 0 (Level: 0)
- in Region 4: 0 (Level: 0)

4.9 Serial Group

The parameters in this group are used to configure the camera's serial port forwarding.

Parameter:	Commit
Data Type:	Command
Comments:	Changes to the values for the parameters in this group will become active when the Commit command is issued.

Parameter:	Revert
Data Type:	Command
Comments:	Reverts the values of the parameters in this group to what they were when the last Commit command was issued.

Parameter:	Forwarding
Data Type:	Unsigned Integer
Valid Values:	0 = forwarding disabled. 1 = forwarding enabled.
Comments:	Enables or disabled serial port forwarding via TCP/IP.

Parameter:	BaudRate
Data Type:	Enum
Valid Values:	B_1200 = baud rate set to 1200 bps. B_2400 = baud rate set to 2400 bps. B_4800 = baud rate set to 4800 bps. B_9600 = baud rate set to 9600 bps. B_19200 = baud rate set to 19200 bps. B_38400 = baud rate set to 38400 bps. B_57600 = baud rate set to 57600 bps. B_115200 = baud rate set to 115200 bps.
Comments:	Sets the baud rate for the serial port.

Parameter:	LineConfig
Data Type:	Enum
Valid Values:	S_8N1 = 8 bit data, no parity, 1 stop bit. S_8E1 = 8 bit data, even parity, 1 stop bit. S_8O1 = 8 bit data, odd parity, 1 stop bit. S_8N2 = 8 bit data, no parity, 2 stop bit. S_8E2 = 8 bit data, even parity, 2 stop bit. S_8O2 = 8 bit data, odd parity, 2 stop bit. S_7N1 = 7 bit data, no parity, 1 stop bit. S_7E1 = 7 bit data, even parity, 1 stop bit. S_7O1 = 7 bit data, odd parity, 1 stop bit. S_7N2 = 7 bit data, no parity, 2 stop bit. S_7E2 = 7 bit data, even parity, 2 stop bit. S_7O2 = 7 bit data, odd parity, 2 stop bit.
Comments:	Sets the serial port configuration.

Parameter:	Port
Data Type:	Unsigned Integer
Valid Values:	1 to 65535
Comments:	Sets the port to listen to for incoming TCP connection and forward all traffic to the serial console.

Parameter:	Auth
Data Type:	Unsigned Integer
Valid Values:	0 = authentication not required. 1 = authentication required.
Comments:	Enables or disables the need for a login to access the serial port.

Parameter:	UserName
Data Type:	String
Valid Values:	Up to 15 numbers and/or letters (upper or lower case).
Comments:	If authentication is enabled, sets the user name to access the port.

Parameter:	Password
Data Type:	String
Valid Values:	Up to 29 characters. All standard keyboard characters are valid
Comments:	If authentication is enabled, sets the password to access the port.

4.10 System Group

The parameters in this group provide access to the camera's system settings.

Parameter:	SetDateTime
Data Type:	String
Comments:	Sets the date and time in a numerical format where the numbers represent the date and time based on a 24 hour clock. For example, entering 042216362008.11 would set the date to 22-April-2008 and the time to 16:36:11.

Parameter:	CurDateTime
Data Type:	String
Comments:	Read only. Gets the current date and time in a human readable format.

Parameter:	DateTimeFormat
Data Type:	String
Valid Values:	Some of the variables that can be used in the string are: %d = day of the month as a decimal number (range 01 to 31) %D = same as %m/%d/%y %F = same as %Y-%m-%d %h = the abbreviated month name %H = hour as a decimal number using a 24-hour clock (range 00 to 23) %I = hour as a decimal number using a 12-hour clock (range 01 to 12) %m = month as a decimal number (range 01 to 12) %M = minute as a decimal number %r = time in a.m. and p.m. notation %R = time in 24 hour notation %S = seconds as a decimal number %T = current time, equal to %H:%M:%S %y = year without the century as a decimal number %Y = year as a decimal number %Z = display time zone code if available (Remember that the escape character for % is %25. For an example of using these variables to format the date and time, see the request example on the next page.)
Comments:	Sets the date and time format. If the text overlay for a stream is configured to display the date and time. This setting will determine the date and time format used in the display.

Parameter:	TimeZoneDesc
Data Type:	String
Valid Values:	Valid time zone code as specified by POSIX.
Comments:	Sets the time zone

Parameter:	NTP
Data Type:	Enum
Valid Values:	Off = NTP synchronization is disabled. H_1 = NTP synchronization is enabled and updates occur once every 1 hour. H_2 = NTP synchronization is enabled and updates occur once every 2 hours. H_4 = NTP synchronization is enabled and updates occur once every 4 hour.s H_12 = NTP synchronization is enabled and updates occur once every 12 hours. H_24 = NTP synchronization is enabled and updates occur once every 24 hours. Weekly = NTP synchronization is enabled and updates occur once per week.
Comments:	Enables or disables clock synchronization with an NTP server.

Parameter:	NTPServer
Data Type:	String
Valid Values:	Valid IP address or server name.
Comments:	IP address or name of the NTP server to use when NTP is enabled.

Parameter:	Reboot
Data Type:	Command
Comments:	Initiates a camera reboot.

Example Request to Set the Date and Time Format

```
// Set the date format to yyyy-mm-dd format and
// the time format to hh:mm:ss format

http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=System.DateTimeFormat.SetValue&Parameter_0_0=%25F%20%25T
```

(%25 is the escape character for the %sign and %20 is the escape character for a space.)

4.11 IOPins Group

The parameters in this group are used to read the state of the camera's input pin and to read or set the state of the camera's output pin.

Parameter:	InputPin0
Data Type:	Unsigned Integer
Valid Values:	0 = the input pin is inactive. 1 = the input pin is active.
Comments:	Read only.

Parameter:	InputPin0Mode
Data Type:	Enum
Valid Values:	Normal = The input pin will operate normally. Inverted = The operation of the input pin will be inverted.
Comments:	Sets the behavior of the camera's input pin, i.e., what input level will make the camera see the input pin as active and what input level will make the camera see the input pin as inactive. For details about the operation of the input pin, see the camera Installation Guide or User's Manual.

Parameter:	OutputPin0
Data Type:	Unsigned Integer
Valid Values:	0 = the output pin is inactive. 1 = the output pin is active.
Comments:	The state of the output pin can be read or can be set with the OutputPin0 parameter.

Parameter:	OutputPin0Mode
Data Type:	Enum
Valid Values:	Normal = The output pin will operate normally. Inverted = The operation of the output pin will be inverted.
Comments:	Sets the behavior of the camera's output pin, i.e., how the output pin will operate when it is set to active and how it will operate when it is set to inactive. For details about the operation of the output pin, see the camera Installation Guide or User's Manual. **Note:** If output pin 0 has been set to the inverted mode and you restart the camera or you power it off and back on, output pin 0 will be in the normal mode during the camera bootup process and will return to the inverted mode once the bootup process is complete.

4.12 SysInfo Group

The parameters in this group provide access to a variety of basic information about the camera such as the vendor name and the firmware versions.

Parameter:	ModelName
Data Type:	String
Comments:	Read only. Indicates the model name of the camera.

Parameter:	DeviceVersion
Data Type:	String
Comments:	Read only. Indicates a manufacturer specific ID specifying the revision of the camera.

Parameter:	ManName
Data Type:	String
Comments:	Read only. Indicates the camera manufacturer’s name.

Parameter:	ManSpecInfo
Data Type:	String
Comments:	Read only. Indicates any special information unique to the camera.

Parameter:	FirmwareVersion
Data Type:	String
Comments:	Read only. Indicates the camera’s firmware version.

Parameter:	Serial
Data Type:	String
Comments:	Read only. Indicates the camera’s serial number.

Parameter:	MACAddress
Data Type:	String
Comments:	Read only. Indicates the camera’s MAC address.

5 Accessing Video Streams and Buffers

5.1 MJPEG Encoded Streams

Note

To use the information in this section effectively, you must understand how the buffers on the camera operate. See page 36 and page 37 for detailed information about the buffers.

To access a motion JPEG encoded stream from the camera, use a request in the following form:

`http://<camera>/cgi-bin/mjpeg?`

Where:

<*camera*> = the camera from which you want to get the stream.
This can be entered as an IP Address:Port Number.
If your network has a properly configured domain name server, it can also be entered as a user assigned host name.

The following optional parameters can be used with this request:

mode = [live | timeshift | replay | single]

live - shows the newest images from the selected stream. This parameter can be used with a live stream. It can also be used with an alarm stream if the alarm buffer is in the "armed" state.

timeshift - plays a stream timeshifted into the stream's live buffer. This option can be used with a live stream. It can also be used with an alarm stream if the alarm buffer is in the "armed" state. When using mode = timeshift, the seek parameter should also be used.

replay - replays the contents of an alarm buffer when the alarm buffer is in the "done" state. The fps parameter can also be used to control the replay speed.

single - gets a single, most current image. This parameter can be used with a live stream. It can also be used with an alarm stream if the alarm buffer is in the "armed" state.

stream = N

Number for the stream you wish to access.

buffer = N

Buffer number for the stream you wish to access.

seek = N

Seek back N milliseconds in time into the buffer on the stream.

Only valid for mode=replay or mode=timeshift.

N=-1 means seek back as far as possible.

fps = N

If used with mode = replay, the buffer contents will be streamed at the specified rate.

If used with mode = live or mode = time shift, the camera will attempt to deliver images at the specified rate. If the camera is capturing images at a higher rate than specified, it will stream images at the specified rate by dropping some of the captured images. If the camera is capturing images at a lower rate than specified, it will stream images at the lower rate.

You can request a list of all available streams by using the following request:

`http://<camera>/cgi-bin/stream?list`

Examples

There are two common techniques for accessing the MJEG streams, the stream oriented approach and the buffer oriented approach. The stream oriented approach is less complicated, but also less flexible. The buffer oriented approach is more complicated, but more flexible.

Stream Oriented Approach

With the stream oriented approach, you use the stream parameter within your request to access stream 0, stream 1, or stream 2.

For example, assume that you are working with a camera named IPCam_1 and that you simply want to access the most current images from stream 0. You would issue this request:

```
http://IPCam_1/cgi-bin/mjpeg?stream=0&mode=live
```

This would access the most current images from the buffer on live stream 0. Note that if you do not include the mode parameter, the mode will default to live.

If you want to seek back into the live buffer on stream 0 and you want to look at images that were captured 2 seconds in the past, you would issue a request that includes the mode parameter set to "timeshift" and also includes the seek parameter like this:

```
http://IPCam_1/cgi-bin/mjpeg?stream=0&mode=timeshift&seek=2000
```

Now assume that you want to view the contents of the alarm buffer on stream 0. You would issue a request with the mode parameter set to "replay" like this:

```
http://IPCam_1/cgi-bin/mjpeg?stream=0&mode=replay&fps=5.0
```

When this request is issued, the camera will begin streaming the contents of the alarm buffer on stream 0 at the specified frame rate.

If the alarm buffer is in the "armed" state, it means that alarm stream 0 is live and is actively buffering images. So the above request will stream a live stream from the buffer.

If the stream 0 alarm buffer is in the "done" state, it means that an alarm condition has been declared, the alarm buffer has filled, and alarm stream 0 is no longer live. So the above request will stream the stored content in the buffer and then will stop streaming.

(Refer to the Buffer Oriented Approach description for more information about how you can tell when an alarm buffer is armed or done.)

Buffer Oriented Approach

With the buffer oriented approach, you first obtain detailed information about the current state of the live buffer and the alarm buffer on each stream. You then use specific requests to access a particular buffer.

Assume that you are working with a camera named IPCam_1. Begin the process by requesting a list of available streams:

```
http://IPCam_1/cgi-bin/stream?list
```

Assume that you receive the following return

```
buffer_0=(0,"Stream 0",image/jpeg,LIVE,1024x768)
buffer_1=(0,"Stream 0",image/jpeg,ALARM,1024x768)
buffer_2=(1,"Stream 1",image/jpeg,LIVE,512x384)
buffer_3=(2,"Stream 2",image/jpeg,LIVE,512x384)
```

This means that four streams are available, a live stream 0, an alarm stream 0, a live stream 1, and a live stream 2. (The alarm buffer has been enabled on stream 0, so this makes two streams available - a "live" stream 0 and an "alarm" stream 0.)

To access the most current images from live stream 0, you would issue this request:

```
http://IPCam_1/cgi-bin/mjpeg?buffer=0&mode=live
```

To access the most current images from live stream 1, you would issue this request:

```
http://IPCam_1/cgi-bin/mjpeg?buffer=2&mode=live
```

If you want to seek back into the buffer on live stream 0 and you want to look at images that were captured 2 seconds in the past, you would issue this request:

```
http://IPCam_1/cgi-bin/mjpeg?buffer=0&mode=timeshift&seek=2000
```

Now assume that you want to look at alarm stream 0. First you must determine whether the alarm buffer for the stream is in the "armed" state or the "done" state by issuing the following requests:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=Stream.StreamSelector.SetValue&Parameter_0_0=0
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=Stream.AlarmBufferState.GetValue
```

If the return indicates that the alarm buffer state is "armed", it means that alarm stream 0 is live and is actively buffering images. So you can access a live stream from the buffer using this request:

```
http://IPCam_1/cgi-bin/mjpeg?buffer=1&mode=live
```

If the return indicates that the alarm buffer state is "done", it means that an alarm condition has been declared, the alarm buffer has filled, and alarm stream 0 is no longer live. In this case you can stream the stored content from the buffer by using this request:

```
http://IPCam_1/cgi-bin/mjpeg?buffer=1&mode=replay&fps=5.0
```

When all of the stored content has been streamed, streaming will stop.

The fps parameter will determine the rate at which the camera will stream the contents of the alarm buffer. If the specified rate is greater than the rate at which the buffered images were captured, then the replay will appear to be in fast motion. If the specified rate is less than the capture rate, then the playback will appear to be in slow motion. The fps parameter is optional - if it is left out, the camera will simply stream the buffer contents at the fastest rate possible.

5.2 MPEG4 or H.264 Encoded Streams

Note

To use the information in this section effectively, you must understand how the buffers on the camera operate. See page 36 and page 37 for detailed information about the buffers.

Multicasting can only enabled when stream 0 is set for MPEG4 or H.264 encoding.

When multicasting on stream 0 is disabled, the camera only transmits a unicast stream 0.

When multicasting on stream 0 is enabled, the camera transmits both a unicast stream and a multicast stream for stream 0.

When MPEG4 or H.264 encoding is enabled on stream 0, you can access the camera's **unicast** stream by using a request in one of the following forms:

`rtsp://<camera>/mpeg4`

`rtsp://<camera>/h264`

When MPEG4 or H.264 is encoding enabled on stream 0 and **multicasting is enabled**, you can access the camera's multicast stream by using a request in one of the following forms:

`rtsp://<camera>/mpeg4?multicast`

`rtsp://<camera>/h264?multicast`

Where:

<*camera*> = the camera from which you want to get the stream.
This can be entered as an IP Address:Port Number.
If your network has a properly configured domain name server, it can also be entered as a user assigned host name.

When you access a unicast stream, the following optional parameters can be used with your request:

mode = [live | timeshift | replay]

live - shows the newest images from the selected stream. This parameter can be used with a live stream. It can also be used with an alarm stream if the alarm buffer is in the "armed" state. (default)

timeshift - plays a stream timeshifted into the stream's live buffer. This option can be used with a live stream. It can also be used with an alarm stream if the alarm buffer is in the "armed" state. When using mode = timeshift, the seek parameter should also be used.

replay - replays the contents of an alarm buffer when the alarm buffer is in the "done" state. The fps parameter can also be used to control the replay speed.

buffer = [0 | 1]

0 = live buffer (default)

1 = alarm buffer

seek = N

Seek back N milliseconds in time into the buffer on the stream.

N=-1 means seek back as far as possible.

Only valid for mode=replay or mode=timeshift.

fps = N

Only valid for mode = replay.

When used with mode = replay, the buffer contents will be streamed at the specified rate.

You can request a list of all available streams by using the following request:

`http://<camera>/cgi-bin/stream?list`

Note

Early production cameras used an older request format to access MPEG4 and H.264 cameras. On these cameras, requests took these forms:

`rtsp://<camera>/Live`

`rtsp://<camera>/AlarmReplay`

This format is now outdated. But to maintain backward compatibility, newer cameras will accept requests in the old format. Because the current request format described in Section 5.2 is more flexible, we strongly recommend that you use it.

If you need more information about the older request format, please contact Basler technical support.

Example 1 - Accessing a Unicast Stream

Assume that you are working with a camera named IPCam_1. Also assume that you want to access the camera's **unicast stream**. Begin the process by requesting a list of available streams:

```
http://IPCam_1/cgi-bin/stream?list
```

Assume that you receive the following return

```
buffer_0=(0,"Stream 0",video/mp4v-es,LIVE,1024x768)
buffer_1=(0,"Stream 0",video/mp4v-es,ALARM,1024x768)
```

This means that two streams are available, a live stream 0 and an alarm stream 0. (The alarm buffer has been enabled on stream 0, so this makes two streams available - a "live" stream and an "alarm" stream.) The stream 0 live stream and stream 0 alarm stream are MPEG4 encoded.

To access the most current images from live stream 0, you would issue this request:

```
rtsp://IPCam_1/mpeg4?buffer=0&mode=live
```

If you want to seek back into the buffer on live stream 0 and you want to look at images that were captured 2 seconds in the past, you would issue this request:

```
rtsp://IPCam_1/mpeg4?buffer=0&mode=timeshift&seek=2000
```

Now assume that you want to look at alarm stream 0. First you must determine whether the alarm buffer for the stream is in the "armed" state or the "done" state by issuing the following requests:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=Stream.StreamSelector.SetValue&Parameter_0_0=0
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=Stream.AlarmBufferState.GetValue
```

If the return indicates that the alarm buffer state is "armed", this means that alarm stream 0 is live and is actively buffering images. So you can access a live stream from the buffer using this request:

```
rtsp://IPCam_1/mpeg4?buffer=1&mode=live
```

If the return indicates that the alarm buffer state is "done", this means that an alarm condition has been declared, the alarm buffer has filled, and alarm stream 0 is no longer live. In this case you can stream the stored content from the buffer by using this request:

```
rtsp://IPCam_1/mpeg4?buffer=1&mode=replay&fps=5.0
```

When all of the stored content has been streamed, streaming will stop.

The replay parameter will determine the rate at which the camera will stream the contents of the alarm buffer. If the specified rate is greater than the rate at which the buffered images were captured, then the replay will appear to be in fast motion. If the specified rate is less than the capture rate, then the playback will appear to be in slow motion. The replay parameter is optional - if it is left out, the camera will simply stream the buffer contents at the fastest rate possible.

Example 2 - Accessing a Multicast Stream

Assume that you are working with a camera named IPCam_1. Also assume that multicasting is enabled and that you want to access the camera's **multicast stream**. Begin the process by requesting a list of available streams:

```
http://IPCam_1/cgi-bin/stream?list
```

Assume that you receive the following return

```
buffer_0=(0,"Stream 0",video/mp4v-es,LIVE,1024x768)
buffer_1=(0,"Stream 0",video/mp4v-es,ALARM,1024x768)
```

This means that two streams are available, a live stream 0 and an alarm stream 0. (The alarm buffer has been enabled on stream 0, so this makes two streams available - a "live" stream and an "alarm" stream.) The stream 0 live stream and stream 0 alarm stream are MPEG4 encoded.

With a multicast request, you can only access live stream 0.

To access the most current images from live stream 0, you would issue this request:

```
rtsp://IPCam_1/mpeg4?multicast
```

Note that when a camera is set for multicasting, it also issues a unicast stream. You can access the unicast stream via the unicast type of requests.

6 User Authentication

When user authentication is enabled on the camera, each CGI based request to the camera is checked for valid authentication. This includes all of the param_if.cgi form of requests described in Section 1 of this document and the stream requests described in Section 5. Note that enabling user authentication **does not** mean that the traffic will be encrypted.

When user authentication is enabled on an IP Camera, there are two approaches for issuing requests to the camera: basic access authentication and session based authentication.

Basic access authentication is a simple username/password based approach that is easy to use, but not very secure. With this approach, a valid user name and password is simply added to each param_if.cgi form of request or stream request that you issue to the camera. Basic access authorization access is explained in detail in Section 6.1 on page 65.

With session based authentication, you must first log into the camera with a valid user name and password to obtain a "session_ID". This ID must then be added to each param_if.cgi form of request or stream request that you issue to the camera. This approach is somewhat more secure and a bit more complex. Session based authorization is explained in detail in Section 6.2 on page 66.

Enabling and disabling authentication, user management, and logging in or out of the camera (for session based authorization) are all handled through a special authentication API. The authentication API is described in detail in Section 6.3 on page 67.

By default, user authentication is disabled.

Default User Name and Password

The default user name is: admin

The default password is: admin

The default user is an administrator level user.

User Name and Password Limitations

When user authentication is enabled, each user must be assigned a user name and password. User names and passwords can include ASCII characters (upper and lower case), digits, and the underscore (_).

User names and passwords are case sensitive!

User Levels

When user authentication is enabled, each user must be assigned a user level. The table below describes the available user levels.

At least one user must be assigned to the administrator level.

User Level	Description
0	**Administrator** - Can change all camera parameters. Can add or delete users. Can change the level or password of all existing users.
9	**Viewer** - Can view images in the Surveillance Web Client. Can change his or her own password.

Table 2: User Level Descriptions

6.1 Basic Access Authentication

Basic authentication is a standard authentication scheme used by simple web pages and some devices. With basic access authorization, whenever a server (such as the one in your camera) feels the need to check the authentication of a user, it responds to an HTTP request with a special error code 401 and the "WWW-Authenticate:..." header. If the request was issued via a web browser, a Username/Password window will open and after the user enters the user name and password, the browser will resend the request with the values included. If the server is satisfied, it responds with the normal HTTP 200 OK. From then on, the browser will resend the username/password, essentially in clear text, with every request to the server. This explains why basic authorization is not very secure.

You can include the basic authorization information directly in the URL of a request by forming your request in this fashion: http://<username>:<password>@<ip>...

When authentication is enabled on your camera and you choose to use basic access authentication to access the camera, there is no specific procedure for logging onto the camera. Instead, all of the CGI requests described in Section 1 of this document and the stream requests described in Section 5 should be preceded with the user name and password in this fashion: http://<*username*>:<*password*>@<camera>/cgi-bin/...

For a camera named "IPCam_1", and a user named "ABrown" with a password of "Abc12", some examples of correctly formatted requests when authentication is enabled and you are using basic access authorization would be:

```
http://ABrown:Abc12@IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=Sensor.TestImageMode.SetValue&Parameter_0_0=TestImage_1
```

```
http://ABrown:Abc12@IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=System.Reboot.Execute
```

```
http://ABrown:Abc12@IPCam_1/cgi-bin/mjpeg?buffer=0&mode=live
```

The example basic access authorization samples shown above and throughout the rest of Section 6 all include *<username>:<password>* at the beginning of each request. Many of the tools used to issue requests to the camera will store the *<username>:<password>* combination the first time that you use it and will automatically add it to the beginning of each subsequent request. If you are using a tool that does this, you only need to include the user name and password with the first request.

Enabling and disabling authentication on a camera and user management tasks such as adding users and changing passwords are all handled via a special authentication API. The authentication API is described in detail in Section 6.3 on page 67.

6.2 Session Based Authentication

The session based approach to user authentication does not include the user name and password with every request, which makes it more secure that basic authentication.

The session based approach works in this manner:

1. The client sends a request to "login" to the server (in the camera), including a user name and password (in plaintext).
2. The server checks the user name and password. If they are correct, the server creates a "session" and sends back an identifier (a long string) called the "session_id" to the client.
3. The client now sends requests to the server as it normally would, but always appends the session_id it got from the server during login.
4. As long as the session_id is valid, the server will act on a request. It will also adjust a "last access time" for the session.
5. A session_id becomes invalid if there hasn't been a request with that session_id for a period of time (60 minutes in our case).
6. The client can make the session_id invalid by requesting a "logout" from the server. The session_id becomes invalid immediately after the request and can no longer be used.

When authentication is enabled on the camera and you are using session based authentication, all of the CGI requests described in Section 1 of this document and the stream requests described in Section 5 must be appended with a valid session ID in this fashion:
http://<camera>/cgi-bin/... &session_id=<*id*>

For example, assume that you are working with a camera named "IPCam_1" that has authentication enabled. Also assume that you have logged into the camera and received a session id of "569435e299659a912f56b425acdafbe7". Some examples of correctly formatted requests would be:

```
http://Cam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=Sensor.TestImageMode.SetValue&Parameter_0_0=TestImage_1
&session_id=569435e299659a912f56b425acdafbe7
```

```
http://Smith:12345@IPCam_1/cgi-bin/param_if.cgi?NumActions=1
&Action_0=System.Reboot.Execute&session_id=569435e299659a912f56b425acdafbe7
```

```
http://Smith:12345@IPCam_1/cgi-bin/mjpeg?buffer=0
&mode=live&session_id=569435e299659a912f56b425acdafbe7
```

When you log into a camera, the return will include a valid session ID.

Requests to log in or log out of a camera or for performing user management tasks such as adding users and changing passwords are performed via a special authentication API. Section 6.3 on page 67 describes the authentication API in detail.

6.3 The Authentication API

The authentication API is used to enable or disable authentication, to log into and out of the camera, and to handle user management tasks.

The authentication API is implemented using a CGI known as the cgi-bin/auth_if.cgi. The cgi-bin/auth_if.cgi includes a variety of authentication and user management related requests. The basic format of requests to the authentication API is as follows:

`http://<camera>/cgi-bin/auth_if.cgi?<Method>&<Parameter>`

Where:

<*camera*> = the camera on which you want to set the parameter value.
This can be entered as an IP Address:Port Number.
If your network has a properly configured domain name server, it can also be entered as a user assigned host name.

<*Method*> = one of the supported methods listed in Section 6.3.1 on page 68.

<*Parameter*> = a parameter associated with the method. A method may have more that one parameter.

If you are using basic access authentication, requests using the authentication API must typically be preceded with the user name and password.

If you are using session based authentication, requests typically include the session ID as a parameter.

The method descriptions in the next section include sample requests and returns.

6.3.1 Authentication API Methods

?Status

The "?Status" method can be used by any client at any time. You do not need to log in before making a status request and no username/password or session ID is needed as part of the request. This request will simply tell you if user authentication is enabled and returns some basic configuration information.

Below is a sample status request on a camera named "IPCam_1":

```
http://<IPCam_1>/cgi-bin/auth_if.cgi?Status
```

The return from a status request typically looks like this:

```
{method: 'Status', success: true, errorcode: 0, reason: 'Success',
enabled: true, realm: 'Basler-20802071', json_valid: true }
```

Where

method: = indicates the method used in the request.

success: = indicates the success or failure of the request. The value can be either "true" or "false".

errorcode: = a code that indicates the success of the request or indicates the reason for failure of the request (see Table 3 on page 79).

reason: = text that indicates the success of the request or indicates the reason for failure of the request.

enabled: = indicates whether authentication is enabled or not. The value can be either "true" or "false".

realm: = the vendor name followed by the serial number.

json_valid: = reserved for future use.

?Enable

The "?Enable" method is used to enable user authentication. You do not need to log in before making an enable request and no username/password or session ID is needed as part of the request.

Below is a sample enable request on a camera named "IPCam_1":

```
http://<IPCam_1>/cgi-bin/auth_if.cgi?Enable
```

The return from an enable request typically looks like this:

```
{method: 'Enable', success: true, errorcode: 0, reason: 'Success',
json_valid: true }
```

Where

method: = indicates the method used in the request.

success: = indicates the success or failure of the request. The value can be either "true" or "false".

errorcode: = a code that indicates the success of the request or indicates the reason for failure of the request (see Table 3 on page 79).

reason: = text that indicates the success of the request or indicates the reason for failure of the request.

json_valid: = reserved for future use.

?Login

The "?Login" method is used with session based authentication to log into the camera and obtain a session ID. The Login method takes the following parameters:

username=<username of an existing user>

password =<password for the existing user>

User names and passwords are case sensitive. The return from the login request will include a valid session ID.

Below is a sample login request on a camera named "IPCam_1", and a user named "ABrown" with a password of "Abc12":

```
http://<IPCam_1>/cgi-bin/auth_if.cgi?Login&username=ABrown&password=Abc12
```

The return from a login request typically looks like this:

```
{method: 'Login', success: true, errorcode: 0, reason: 'Success',
id: 1001, username: 'ABrown', adminstate: true, level: 0,
session_id: '5be8d6e71c2e09cea25c237767489e1a', json_valid: true }
```

Where

method: = indicates the method used in the request.

success: = indicates the success or failure of the request. The value can be either "true" or "false".

errorcode: = a code that indicates the success of the request or indicates the reason for failure of the request (see Table 3 on page 79).

reason: = text that indicates the success of the request or indicates the reason for failure of the request.

id: = the user ID number currently assigned by the system to the user who just logged in.

username: = indicates the user name under which you logged in.

adminstate: = indicates if the logged in user is an administrator level user (true or false).

level: = indicates the user's level (see Table 2 on page 64).

session_id: = indicates a valid session ID that can be used when making other requests.

json_valid: = reserved for future use.

?ListUser

The "?ListUser" method returns a list of currently existing users. The method can be used with either basic access authentication or with session based authentication.

When you are using session based authentication, the method takes the following parameter:

session_id=<a currently valid session id obtained during login>

Below is a sample list user request for a camera named "IPCam_1" when basic access authentication is being used and the user's name is "ABrown" with a password of "Abc12":

```
http://ABrown:Abc12@<IPCam_1>/cgi-bin/auth_if.cgi?ListUser
```

Below is a sample list user request on a camera named "IPCam_1" when session based authentication is being used and the session ID is "5be8d6e71c2e09cea25c237767489e1a":

```
http://<IPCam_1>/cgi-bin/auth_if.cgi?ListUser
&session_id=5be8d6e71c2e09cea25c237767489e1a
```

The return from a list user request typically looks like this:

```
{ method: 'ListUser', success: true, errorcode: 0, reason: 'Success', users:
[ { id: 1000, username: 'admin', level: 0 }, { id: 1001, username: 'ABrown',
level: 0 }, { id: 1002, username: 'BGreen', level: 0 }, { id: 1003, username:
'CWhite', level: 9 } ], json_valid: true }
```

Where

method: = indicates the method used in the request.

success: = indicates the success or failure of the request. The value can be either "true" or "false".

errorcode: = a code that indicates the success of the request or indicates the reason for failure of the request (see Table 3 on page 79).

reason: = text that indicates the success of the request or indicates the reason for failure of the request.

users: = list of currently existing users with the following information for each user:

id = the user ID number currently assigned to the user by the system
username = the user's name
level = the user's level (see Table 2 on page 64)

json_valid: = reserved for future use.

?AddUser

The "?AddUser" method is used to add a new user. The method can be used with either basic access authentication or with session based authentication.The method takes the following parameters when you are using either basic access or session based authentication:

username=<the username for the new user>

password=<the password for the new user>

level=<the new user's level>

(see Table 2 on page 64 for a description of user levels)

The method takes an additional parameter when you are using session based authentication:

session_id=<a currently valid session id>

Below is a sample add user request on a camera named "IPCam_1" when basic access authentication is being used. An admin level user named "ABrown" with a password of "Abc12" will add the user. The new user will be "DBlack" who will have a password of "Jkl78" and view-only rights:

```
http://ABrown:Abc12@<IPCam_1>/cgi-bin/auth_if.cgi?AddUser
&username=DBlack&password=Jkl78&level=9
```

Below is a sample of a similar add user request when session based authentication is being used and the session ID is "5be8d6e71c2e09cea25c237767489e1a":

```
http://<IPCam_1>/cgi-bin/auth_if.cgi?AddUser
&username=DBlack&password=Jkl78&level=9
&session_id=5be8d6e71c2e09cea25c237767489e1a
```

The return from an add user request typically looks like this:

```
{method: 'AddUser', success: true, errorcode: 0, reason: 'Success',
json_valid: true }
```

Where

method: = indicates the method used in the request.

success: = indicates the success or failure of the request. The value can be either "true" or "false".

errorcode: = a code that indicates the success of the request or indicates the reason for failure of the request (see Table 3 on page 79).

reason: = text that indicates the success of the request or indicates the reason for failure of the request.

json_valid: = reserved for future use.

?ChangeLevel

The "?ChangeLevel" method is used to change the user level an existing user. The method can be used with either basic access authentication or with session based authentication.The method takes the following parameters when you are using either basic access or session based authentication:

id=<the user's ID as returned by the ?ListUser method>

username=<the user name for an existing user>

level=<level>

Where <level> is an integer value from Table 2 on page 64 (e.g., 0 or 9) or is a string value indicating the level (e.g., "Administrator" or "Viewer").

Note: Use either the id parameter or the username parameter with the method, not both.

The method takes an additional parameter when you are using session based authentication:

session_id=<a currently valid session ID>

Below is a sample change level request on a camera named "IPCam_1" when basic access authentication is being used. An admin level user named "ABrown" with a password of "Abc12" will change the level. An existing user named "CWhite" is the user to be changed. A list user request has returned the information that user CWhite has an ID of 1003 and a current user level of 9. The user level will be changed to 0:

```
http://ABrown:Abc12@<IPCam_1>/cgi-bin/auth_if.cgi?ChangeLevel
&id=1003&level=0
```

Below is a sample of a similar change level request when session based authentication is being used and the session ID is "5be8d6e71c2e09cea25c237767489e1a":

```
http://<IPCam_1>/cgi-bin/auth_if.cgi?ChangeLevel&id=1003&level=0
&session_id=5be8d6e71c2e09cea25c237767489e1a
```

The return from a change user request typically looks like this:

```
{method: 'ChangeLevel', success: true, errorcode: 0, reason: 'Success',
json_valid: true }
```

Where

method: = indicates the method used in the request.

success: = indicates the success or failure of the request. The value can be either "true" or "false".

errorcode: = a code that indicates the success of the request or indicates the reason for failure of the request (see Table 3 on page 79).

reason: = text that indicates the success of the request or indicates the reason for failure of the request.

json_valid: = reserved for future use.

?ChangePasswd

The "?ChangePasswd" method is used to change the password an existing user. The method can be used with either basic access authentication or with session based authentication.The method takes the following parameters when you are using either basic access or session based authentication:

id=<the user's ID as returned by the ?ListUser method>

username=<the user name for an existing user>

password=<the new password>

Notes: Use either the id parameter or the username parameter with the method, not both.
If no id or username parameter is included, the password for the currently authenticated user will be changed.

The method takes an additional parameter when you are using session based authentication:

session_id=<a currently valid session ID>

Below is a sample password request on a camera named "IPCam_1" when basic access authentication is being used. An admin level user named "ABrown" with a password of "Abc12" will change the password. An existing user "CWhite" is the user to be changed. The return from a ListUser request returns the information the user CWhite has an ID of 1003. CWhite's password will be changed to "Mno90":

```
http://ABrown:Abc12@<IPCam_1>/cgi-bin/auth_if.cgi?ChangePasswd
&id=1003&password=Mno90
```

Below is a sample of a similar password request when session based authentication is being used and the session ID is "5be8d6e71c2e09cea25c237767489e1a":

```
http://<IPCam_1>/cgi-bin/auth_if.cgi?ChangePasswd&id=1003&password=Mno90
&session_id=5be8d6e71c2e09cea25c237767489e1a
```

The return from a password request typically looks like this:

```
{method: 'ChangePasswd', success: true, errorcode: 0, reason: 'Success',
json_valid: true }
```

Where

method: = indicates the method used in the request.

success: = indicates the success or failure of the request. The value can be either "true" or "false".

errorcode: = a code that indicates the success of the request or indicates the reason for failure of the request (see Table 3 on page 79).

reason: = text that indicates the success of the request or indicates the reason for failure of the request.

json_valid: = reserved for future use.

?DeleteUser

The "?DeleteUser" method is used to delete a user. The method can be used with either basic access authentication or with session based authentication.The method takes the following parameter when you are using either basic access or session based authentication:

id=<the user's ID as returned by the ?ListUser method>

username=<the user name for an existing user>

Note: Use either the id parameter or the username parameter with the method, not both.

The method takes an additional parameter when you are using session based authentication:

session_id=<a currently valid session ID>

Below is a sample delete user request on a camera named "IPCam_1" when basic access authentication is being used. An admin level user named "ABrown" with a password of "Abc12" will delete the user. "BGreen" is the exisitng user to be deleted. A list user request has returned the information that user BGreen has an ID of 1002:

```
http://WSmith:12345@<IPCam_1>/cgi-bin/auth_if.cgi?DeleteUser&id=1002
```

Below is a sample of a similar delete user request when session based authentication is being used and the session ID is "5be8d6e71c2e09cea25c237767489e1a":

```
http://<IPCam_1>/cgi-bin/auth_if.cgi?DeleteUser&id=1002
&session_id=5be8d6e71c2e09cea25c237767489e1a
```

The return from a delete user request typically looks like this:

```
{method: 'DeleteUser', success: true, errorcode: 0, reason: 'Success',
json_valid: true }
```

Where

method: = indicates the method used in the request.

success: = indicates the success or failure of the request. The value can be either "true" or "false".

errorcode: = a code that indicates the success of the request or indicates the reason for failure of the request (see Table 3 on page 79).

reason: = text that indicates the success of the request or indicates the reason for failure of the request.

json_valid: = reserved for future use.

There must be at least one administrator level user. When there is only one administrator level user, the camera will not allow you to delete that user.

?CheckSession

The "?CheckSession" method is used with session based authentication. The method is used to determine if a session ID is still valid and to get the user name and level of the user who owns the session. The session ID obtained during login (see page 70) is required.

The method takes the following parameter:

session_id=<a currently valid session id obtained during login>

Below is a sample check session request on a camera named "IPCam_1" when session based authentication is being used and the session ID is "5be8d6e71c2e09cea25c237767489e1a":

```
http://<IPCam_1>/cgi-bin/
auth_if.cgi?CheckSession&session_id=5be8d6e71c2e09cea25c237767489e1a
```

The return from a check session request typically looks like this:

```
{method: 'CheckSession', success: true, errorcode: 0, reason: 'Success',
id: 1001, username: 'ABrown', adminstate: true, level: 9,
session_id: '5be8d6e71c2e09cea25c237767489e1a', json_valid: true }
```

Where

method: = indicates the method used in the request.

success: = indicates the success or failure of the request. The value can be either "true" or "false".

errorcode: = a code that indicates the success of the request or indicates the reason for failure of the request (see Table 3 on page 79).

reason: = text that indicates the success of the request or indicates the reason for failure of the request.

id: = the user ID number currently assigned by the system to the session ID owner.

username: = indicates the user name of the session ID owner.

adminstate: = indicates if the logged in user is an administrator level user (true or false).

level: = indicates the user's level (see Table 2 on page 64).

session_id: = indicates the session ID.

json_valid: = reserved for future use.

?Logout

The "?Logout" method is used with session based authentication to logout of the camera and invalidate the current session ID. The method takes the following parameter:

session_id=<a currently valid session id obtained during login>

Logging out renders the current session ID invalid.

Below is a sample logout request on a camera named "IPCam_1" when session based authentication is being used and the session ID is "5be8d6e71c2e09cea25c237767489e1a":

```
http://<IPCam_1>/cgi-bin/
auth_if.cgi?Logout&session_id=5be8d6e71c2e09cea25c237767489e1a
```

The return from a logout request typically looks like this:

```
{method: 'Logout', success: true, errorcode: 0, reason: 'Success',
json_valid: true }
```

Where

method: = indicates the method used in the request.

success: = indicates the success or failure of the request. The value can be either "true" or "false".

errorcode: = a code that indicates the success of the request or indicates the reason for failure of the request (see Table 3 on page 79).

reason: = text that indicates the success of the request or indicates the reason for failure of the request.

json_valid: = reserved for future use.

?Disable

The "?Disable" method disables user authentication. The method can be used with either basic access authentication or with session based authentication.

When you are using session based authentication, the method takes the following parameter:

session_id=<a currently valid session id obtained during login>

To use the ?Disable method with session based authentication, you must be currently logged in as an administrator level user.

Below is a sample disable request on a camera named "IPCam_1" when basic access authentication is being used. An admin level user named "ABrown" with a password of "Abc12" will disable authentication.

```
http://ABrown:Abc12@<IPCam_1>/cgi-bin/auth_if.cgi?Disable
```

Below is a sample disable request when session based authentication is being used and the session ID is "5be8d6e71c2e09cea25c237767489e1a":

```
http://<IPCam_1>/cgi-bin/auth_if.cgi?Disable
&session_id=5be8d6e71c2e09cea25c237767489e1a
```

The return from a password request typically looks like this:

```
{method: 'Disable', success: true, errorcode: 0, reason: 'Success',
json_valid: true }
```

Where

method: = indicates the method used in the request.

success: = indicates the success or failure of the request. The value can be either "true" or "false".

errorcode: = a code that indicates the success of the request or indicates the reason for failure of the request (see Table 3 on page 79).

reason: = text that indicates the success of the request or indicates the reason for failure of the request.

json_valid: = reserved for future use.

6.3.1.1 Error Code Definitions for Authentication Returns

Table 3 lists the codes that can appear in the error code field of a return.

Error Code	Meaning
0	The operation worked correctly
1	The operation is not permitted. User level is insufficient to perform the operation.
13	Permission denied, the user could not be authenticated. (Incorrect user name or password or incorrect or invalid session ID.)
16	Internal error in the camera. Try again later.
22	Invalid argument. A generic code delivered if an argument is incorrect, such as the username/ password combination.
110	Session timed out. (The session ID is no longer valid because the timeout has been reached.)

Table 3: Authentication Return Error Codes

7 The ActiveX Control

The Basler IP Camera ActiveX Control provides a few properties to configure the control, mainly to set the camera's video stream URL and to enable video resizing.

Property IBaslerIPCameraControl::URL

Declaration: Property Get/Put URL As String

Used to set or retrieve the camera's MJPEG stream URL in the following format:

```
http://[username:password@]ip_adresse[:port]/path/cgi_file[?parameters]
```

This is a simple example URL:

```
http://192.168.178.37/cgi-bin/mjpeg
```

Setting the URL starts the video stream immediately. Set an empty string in order to stop the video stream.

C Prototype:

```
HRESULT put_URL(BSTR newVal);
HRESULT get_URL(BSTR *pVal);
```

Property IBaslerIPCameraControl::EnableResize

Declaration: Property Get/Put EnableResize As Boolean

If enabled, the video is resized to the parent window dimensions. If disabled, the unscaled video is displayed. If the unscaled video doesn't fit into the window, the center of the video is displayed.

C Prototype:

```
HRESULT put_EnableResize(VARIANT_BOOL newVal);
HRESULT get_EnableResize(VARIANT_BOOL *pVal);
```

Property IBaslerIPCameraControl::BackColor

Declaration: Property Get/Put BackColor As Long

Used to change the control's background color that is visible while the video stream is stopped.

C Prototype:

```
HRESULT put_BackColor(LONG newVal);
HRESULT get_BackColor(LONG *pVal);
```

Property IBaslerIPCameraControl::Caption

Declaration: Property Get/Put Caption As String

Used to change the control's caption that is displayed on the background while the stream is stopped. (default = Basler IP Control)

C Prototype:

```
HRESULT put_Caption(BSTR newVal);
HRESULT get_Caption(BSTR *pVal);
```

Property IBaslerIPCameraControl::Enabled

Declaration: Property Get/Put Enabled As Boolean

Enables or disables the ActiveX control.

C Prototype:

```
HRESULT put_Enabled(VARIANT_BOOL newVal);
HRESULT get_Enabled(VARIANT_BOOL *pVal);
```

Property IBaslerIPCameraControl::HWND

Declaration: Property Get HWND As Long

Used to retrieve the handle of the control's parent window.

C Prototype:

```
HRESULT get_HWND(LONG *pVal);
```

8 Zero Configuration Networking Information

8.1 Using a Program to Locate a Basler IP Camera on Your Network

Basler IP Cameras use well-documented, well-known standards to announce their presence in a local network. To locate the cameras from within your own software, you must be familiar with the following technologies:

- Zeroconf / Dynamic Configuration of IPv4 Link-Local Addresses
- Multicast DNS
- DNS-SD - DNS based Service Discovery.

These technologies are implemented in the mDNSResponder Library from Apple (TM), which is available for the MacOS X and Windows and is marketed under the "Bonjour" brand name. These technologies are also implemented in the "Avahi" Library, the defacto standard for Linux systems.

8.2 Zero Configuration

The Basler IP Camera always acquires a dynamic IP-Address in the 169.254.0.0/16 IP-Subnet, which is reserved for networking within the local network. This makes it possible to communicate with the camera even if the network configuration of the camera does not match the configuration of the network to which it is connected. The methods that a camera uses to pick its address are implemented according to the IETF RFC 3927, "Dynamic Configuration of IPv4 Link-Local Addresses".

Note that to communicate with the camera, your workstation does not need to have an IP-Address in the same range. It is sufficient that a route for 169.254.0.0/16 is pointing to the correct network.

8.3 Multicast DNS

The Service Discovery is based on Multicast DNS, where DNS-Queries and Answers are sent to the multicast address 224.0.0.251, Port 5353. Please note that the specification for Multicast DNS requires certain caching behavior to ease the load on the local network. We recommend using existing libraries that have a caching infrastructure in place.

8.4 DNS based Service Discovery

On startup, a Basler IP Camera broadcasts a set of Multicast-DNS records that describe the type of its service, along with other meta information. Since the structure of these records is mostly mandated by the DNS-SD specification (and the existing libraries have a convenient API to access this information), we limit ourselves to a high level description of records for which you must query in order to discover the camera.

- Query for a PTR record named "_http._tcp.local.". You will get a list of HTTP services in your local network as a result. Lets assume that your camera advertises an HTTP service with the name "Basler-12345678._http._tcp.local." Note that you cannot identify the camera by this name alone.
- Iterate over the returned service names and query for SRV- and TXT-records for each of these names.

  The SRV record will point to a link local hostname (in our example "Basler-12345678.local.") and the port on which the web server in the camera is listening (80 by default).

  The TXT-record contains key/value pairs with more information about this service:

  - "path": the path to the web interface (as mandated by the standard)
  - "model.baslerweb.com": the model of the camera
  - "serial.baslerweb.com": the serial number of the camera
  - "vendor.baslerweb.com": the vendor of the camera.
- Using the information in the TXT record you can identify the Basler cameras on the local network. Note that all of the keys ending in "baslerweb.com" are specific to Basler cameras.
- Querying for A-records for the link-local hostname as returned by the SRV record will yield up to two answers, specifying the two IP addresses of the Basler IP camera.

Notes:

Since modern operating systems frequently include a name resolution service for the .local-Domain, it may not be necessary to query for the A-records explicitly. If this is not the case, you can install the third party software mentioned above to resolve these names.

You will note that the TXT record contains two additional entries:

- "ipv4-net.baslerweb.com": the configured network address of the camera
- "ipv4-zc.baslerweb.com": the zeroconf network address of the camera

Because they are not defined in the DNS-SD standard and might change with future firmware releases, we recommend that you do not use these entries to establish network connectivity. They are intended as a backup and/or for troubleshooting purposes if it is not possible to reliably establish a connection to the camera due to configuration issues.

8.5 Recommended Reading

http://www.zeroconf.org/

http://www.multicastdns.org/

http://www.dns-sd.org/

http://developer.apple.com/opensource/internet/bonjour.html

Daniel Steinberg, Stuart Cheshire:

Zero Configuration Networking: The Definitive Guide

ISBN 0596101007

Revision History

Doc. ID Number	Date	Changes
AW00066201000	23 Apr 2008	Preliminary version.
AW00066202000	7 Jul 2008	Initial release for production cameras.
AW00066203000	19 Dec 2008	Added information to Section 1.1 on page 1 describing how to work with a parameter value that has a data type of command. Added an example to Section 2.4 on page 11. Added Table 1 on page 15. Added more prominent warnings to the parameter values that can not be change in normal operation mode. Added a notation to the description of the FrameRateMode parameter on page 19 indicating that the available values may change. Added the Sharpness parameter description to page 25. Added the WhitePointX and WhitePointY parameter descriptions to page 26. Added the H_264 value to the EncoderType parameter description on page 31. Added a note to the description of the Quality parameter on page 32. Added the GopLength_ms parameter description to page 32. Added the OutputSize parameter description to page 33. Added a note to the Multicast parameter description on page 41. Corrected the example request on page 52. Added a note in Section 5.1 on page 55 indicating that an understanding of buffers is necessary and also corrected the description of "list" and "mode". Updated and expanded Section 5.2 on page 59.
AW00066204000	24 Mar 2009	Added text to the EncoderType parameter description on page 31 indicating that stream 1 must be enabled in order to enable stream 2. Added descriptions of the new MulticastOnDemand and MulticastTTL parameters to page 42. Added a description of the new HTTPPort parameter to page 43. Added a description of the new EmailPort parameter to page 46. Added a description of the new FTPPort parameter to page 47. Added a description of the new InputPin0Mode and OutputPin0Mode parameters to page 53. Added the descriptions of the stream oriented approach and buffer oriented approach to accessing MJPEG streams on page 56 and page 57. Added the new User Authentication section starting on page 63.
AW00066205000	8 Apr 2009	Added a description of the new Saturation parameter to page 25. Added descriptions of the new IRFilterMode, IRFilterState, and IRFilterAnnounceMode parameters to page 27.

Doc. ID Number	Date	Changes
AW00066206000	16 Jul 2008	Changed the valid values and added a note regarding versions to the Exposure Mode parameter description to page 20. Added a description of the new ExposureTimeLimit parameter to page 21. Added a description of the new GainLimit_dB parameter to page 21. Added a note to the ShutterMode parameter description on page 22 indicating that it is now obsolete. Added a note to the GainMode parameter description on page 22 indicating that it is now obsolete. Added a description of the new ExposureTime parameter to page 23. Added a note to the ExposureTime_us parameter on page 24 indicating how it can be used. Added a description of the new Gain_dB parameter to page 24. Added a note to the Gain parameter description on page 25 indicating how it can be used. Added the descriptions of the IRFilterSwitchLevel, IRFilterCurrentLevel, and IRFilterWaitTime parameters to page 28. Added a new valid value and a note to the OverlayText parameter description on page 35. Added notes to the RxTraffic and TxTraffic parameters on page 44 indicating that the display units have changed. Added a note to the PIOHoldTime parameter description on page 46 indicating the effect of using a 0 value. Added the Zero Configuration section starting on page 83.

Feedback

Your feedback will help us improve our documentation. Please click the link below to access an online feedback form. Your input is greatly appreciated.

http://www.baslerweb.com/umfrage/survey.html

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W