Basler IP Cameras

API Description for Cameras with Version 3.x Firmware

Document Number: AW000973
Version: 11 Language: 000 (English)
Release Date: 14 September 2016

BASLER
the power of sight

Contacting Basler Support Worldwide

Europe, Middle East, Africa

Basler AG
An der Strusbek 60–62
22926 Ahrensburg
Germany

Tel. +49 4102 463 515
Fax +49 4102 463 599

support.europe@baslerweb.com

The Americas

Basler, Inc.
855 Springdale Drive, Suite 203
Exton, PA 19341
USA

Tel. +1 610 280 0171
Fax +1 610 280 7608

support.usa@baslerweb.com

Asia-Pacific

Basler Asia Pte. Ltd.
35 Marsiling Industrial Estate Road 3
#05–06
Singapore 739257

Tel. +65 6367 1355
Fax +65 6367 1255

support.asia@baslerweb.com

www.baslerweb.com

Table of Contents

1 Introduction

Document Applicability

This document applies to cameras with firmware version 3.15.0 or higher. Features on cameras with a different firmware version may not operate exactly as described in this document.

If you have an existing application written against the API for cameras with version 2.x firmware and you want to extend the application to work with the API for cameras with version 3.15.0 firmware, refer to the migration guide in Appendix A on page 137.

The latest version of camera firmware is available for download on the Basler website: www.baslerweb.com/firmware

	Version 3.15.0 firmware is only compatible with cameras that have BIP2 as part of the model name (e.g., BIP2-640c). Version 2.x firmware is not compatible with cameras that have BIP2 as part of the model name.

Overview

This document describes the HTTP/CGI-based application programming interface (API) on the Basler IP camera. 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 Basler IP camera 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).

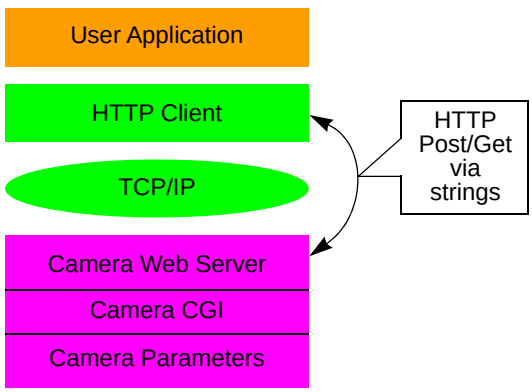


Fig. 1: Interface Diagram

1.1 Setting a Basler IP Camera'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 on page 8 lists the data types and shows the methods available for each data type. The tables in Section 4 on page 12 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.

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 On 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.TestIma
geMode.SetValue&Parameter_0_0=On
```

Return:

```
Return_0=
```

Request:

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

Return:

```
Return_0=On
```

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.

Assuming that 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. Lastly, 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.

IsEnabled - returns "true" if the parameter is available, otherwise returns "false".

IsReadonly - returns "true" if the parameter is read-only, otherwise returns "false".

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.

IsEnabled - returns "true" if the parameter is available, otherwise returns "false".

IsReadonly - returns "true" if the parameter is read-only, otherwise returns "false".

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 IrisMode parameter in the ImageControls group on a camera that has been named "IPCam_1":

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

You would get this return:

```
Return_0=(0,"Auto"),(1,"Open"),(2,"Closed"),(4,"PrioOpen"),(5,"PrioClosed")
```

The return tells you that for the IrisMode parameter, there is an "Auto" string value with an index value of 0, an "Open" string value with an index value of 1, etc.

GetStringValue or GetValue - 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.

IsEnabled - returns "true" if the parameter is available, otherwise returns "false".

IsReadonly - returns "true" if the parameter is read-only, otherwise returns "false".

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.

IsEnabled - returns "true" if the parameter is available, otherwise returns "false".

IsReadonly - returns "true" if the parameter is read-only, otherwise returns "false".

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 a camera with a sensor aspect ratio of 4:3, the mask divides the captured image into a 32 block wide by 24 block high array. On a camera with a sensor aspect ratio of 16:9, the array is 32 block wide by 18 block high.

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. For a 32 x 24 block sized 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 (or 18 groups of digits for a camera with a sensor aspect ratio of 16:9) to represent the 24 (or 18) 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

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. Consider a camera with a sensor aspect ratio of 16:9 and assume that all of the blocks in the bottom three rows of the privacy mask array are set to active and all of the other blocks in the array are set to inactive. The request would look like this:

```
http://IPCam_1/cgi-bin/param_if.cgi?NumActions=1&Action_0=ImageControls.
PrivacyMask.SetValue&Parameter_0_0=00000000,00000000,00000000,00000000,0
0000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,0
0000000,00000000,00000000,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 determined by the camera's current parameter settings. Some camera parameters cannot be changed while the camera is in the normal operation mode; they can only be changed when the camera is in the configure operation mode (see Table 1).
	Configure =	all camera parameters can be changed while the camera is in the configure operation mode. While in the configure mode, the camera delivers video streams as they were set when the camera entered the mode. Any parameter changes will take effect when the operation mode is set back to normal.
Comments:	The camera should usually be set to the "Normal" mode.	

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

Parameter:	FanSpeed
Data Type:	Integer
Comments:	Read-only. Returns the current speed of the camera's cooling fan in RPM (revolutions per minute).

Parameters That Cannot Be Changed in the Normal Operation Mode (But Can Be Changed in the Configure Operation Mode)		
Sensor.AOIWidth	Stream.AOIWidth	Stream.AlarmBufferSize
Sensor.AOIHeight	Stream.AOIHeight	Stream.LiveBufferSize
Stream.EncoderType	Stream.OutputScaling	Stream.AudioEncoderType
Stream.ChainMode	Stream.OutputSize	

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

When the camera is switched to Configure mode:

- All enabled streams will continue streaming at the frame rate and AOI size that was in place when the Configuration Mode was entered.
- If you change the LiveBufferSize parameter setting or the AlarmBufferSize parameter setting for any stream, the setting changes will be applied when you switch back to the Normal mode and **all streams** will be stopped and restarted.
- If you change the EncoderType parameter setting, the OutputScaling parameter setting, the AOIWidth parameter setting, or the AOIHeight parameter setting for a stream, the setting changes will be applied when you switch back to Normal mode and that stream will be stopped and restarted.

  (Keep in mind that if you change the sensor AOIWidth and AOI Height settings to values that are smaller than the current stream AOIWidth and AOI Height settings, the camera will automatically make the stream AOIWidth and AOIHeight settings smaller.)

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.

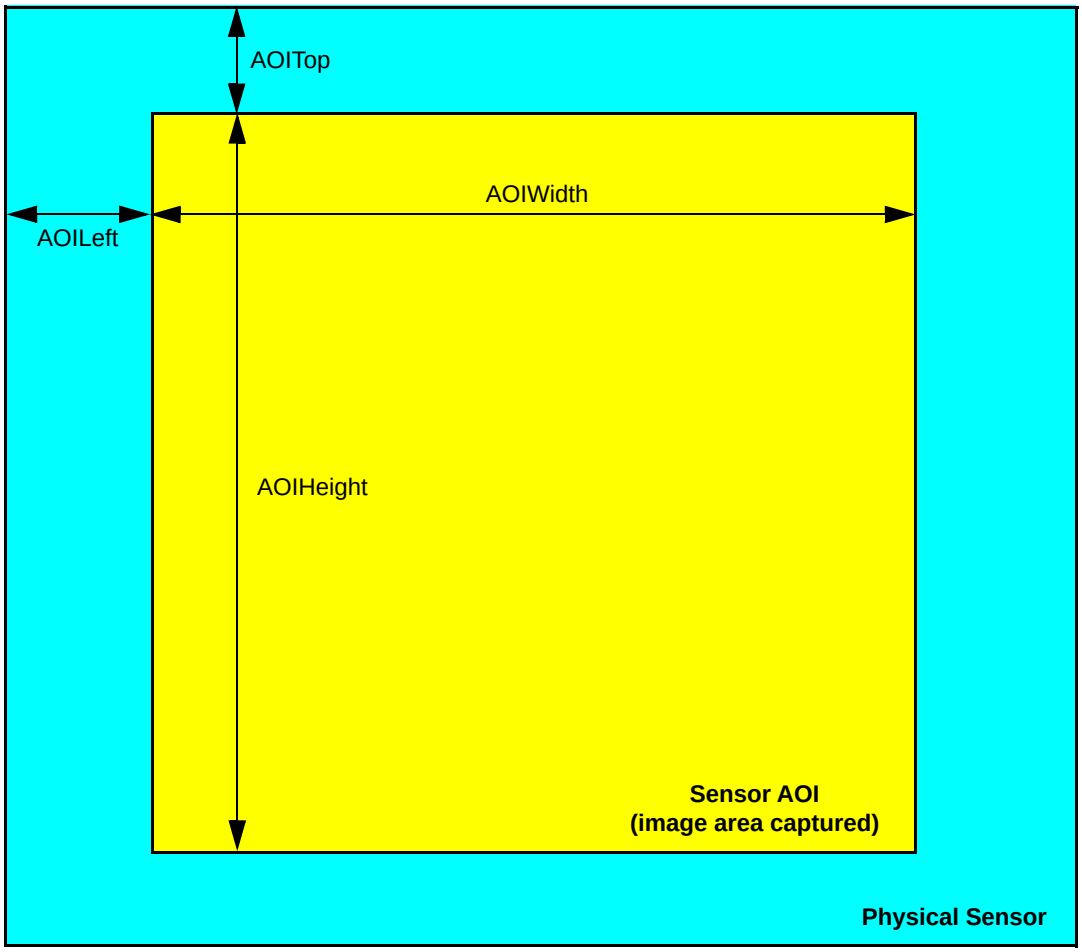


Fig. 2: Sensor AOI

Parameter:	AOIWidth
Data Type:	Integer
Valid Values:	Use the GetMin and GetMax methods to determine the minimum and maximum. Use the GetInc method to determine the increment.
Comments:	As shown in Figure 2, sets the width (in pixels) of the sensor AOI. **The sensor 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 12).

Parameter:	AOIHeight
Data Type:	Integer
Valid Values:	Use the GetMin and GetMax methods to determine the minimum and maximum. Use the GetInc method to determine the increment.
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 12).

Parameter:	AOILeft
Data Type:	Integer
Valid Values:	Use the GetMin and GetMax methods to determine the minimum and maximum. Use the GetInc method to determine the increment. 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.
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:	Integer
Valid Values:	Use the GetMin and GetMax methods to determine the minimum and maximum. Use the GetInc method to determine the increment. 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.
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:	Integer
Comments:	Read-only. Indicates the max allowed width (in pixels) of the sensor AOI.

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

Parameter:	SetMaxAOI
Data Type:	Command
Comments:	Sets the sensor AOI to the largest possible size.

Parameter:	ImageMirror
Data Type:	Enum
Valid Values:	Off = image mirroring is disabled. Mirror_H = horizontal image mirroring (flip left to right) is enabled. Mirror_V = vertical image mirroring (flip top to bottom) is enabled. Mirror_HV = horizonal and vertical mirroring are both enabled.
Comments:	Sets the degrees of rotation for the images captured by the sensor.

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. On = test image function enabled. The camera will stream a moving diagonal red/green/blue gradient test pattern.
Comments:	Enables or disables the camera's test image feature. When the test image mode 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 enabled streams. The test images will include diagonal red, green, and blue gradients, and the gradients will appear to move as the test images are received. Test images are useful for troubleshooting the camera's basic functionality and the network connection.

Parameter:	FrameRateMode
Data Type:	Enum
Valid Values:	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. Example of 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_7_5 = the camera will capture 7.5 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_3_75 = the camera will capture 3.75 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**. Use this parameter to set the expected frame rate when the Function parameter in the IO group is set to "ExternalTrigger" (see page 83). Note that this setting will represent the absolute maximum frame rate that can be achieved by any stream.

Parameter:	FramePeriod_us
Data Type:	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 (see above) 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 page 24). Automatic exposure time control is enabled. ManualExposureTime = automatic exposure time control is disabled, and you must manually set the exposure time by using the ExposureTime parameter (see page 21). 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 ExposureTime parameters (see page 21 and page 24).
Comments:	Sets the functionality of the camera's automatic exposure and gain controls. If desired, the ExposureTimeLimit and the GainLimit_db parameters (see page 19) can be used to set limits on the auto controls.

Parameter:	AutoControlsMask
Data Type:	String
Valid Values:	See Section 3 on page 11
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:	ExposureOffset
Data Type:	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:	BacklightCompensation
Data Type:	Integer
Valid Values:	0 = Backlight compensation disabled. 1 = Backlight compensation enabled.
Comments:	This setting is used to enable the camera's backlight compensation feature. This feature automatically compensates when the main lighting comes from behind the image subject.

Parameter:	ExposureTimeLimit
Data Type:	Enum
Valid Values:	Time_1_30000_s = the exposure time is 1/30000 second Time_1_25000_s = the exposure time is 1/25000 second Time_1_20000_s = the exposure time is 1/20000 second Time_1_16000_s = the exposure time is 1/16000 second Time_1_12000_s = the exposure time is 1/12000 second 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 (continued on the next page)

	(continued from the previous page) 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 1/1.6 second Time_1_1p3_s = the exposure time is 1/1.3 second Time_1_s = the exposure time is 1 second Off = there is no exposure time limit
Comments:	Whenever automatic exposure control is enabled (see the ExposureMode parameter on page 18), 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. Note that if the Anti-Flicker feature is enabled (see page 24), the ExposureTimeLimit parameter setting must be 1/60s or greater.

Parameter:	GainLimit_dB 4
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 Gain_26_dB = the gain is 26 dB Gain_28_dB = the gain is 28 dB Gain_30_dB = the gain is 30 dB Off = there is no gain limit
Comments:	Whenever automatic gain control is enabled, (see the ExposureMode parameter on page 18) 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. Note that if the Anti-Flicker feature is enabled (see page 24), the GainLimit_db parameter setting must be 6dB or greater.

Parameter:	ExposureTime
Data Type:	Enum
Valid Values:	Time_1_30000_s = the exposure time is 1/30000 second Time_1_25000_s = the exposure time is 1/25000 second Time_1_20000_s = the exposure time is 1/20000 second Time_1_16000_s = the exposure time is 1/16000 second Time_1_12000_s = the exposure time is 1/12000 second 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 (continued on the next page)

(continued from the previous page)

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 1/1.6 second
Time_1_1p3_s = the exposure time is 1/1.3 second
Time_1_s = the exposure time is 1 second

Comments: Whenever automatic exposure control is disabled (see the ExposureMode parameter on page 18), the ExposureTime 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:	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 on page 18), 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 Gain_26_dB = the gain is 26 dB Gain_28_dB = the gain is 28 dB Gain_30_dB = the gain is 30 dB
Comments:	When automatic gain control is disabled (see the ExposureMode parameter on page 18), 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.

Parameter:	Gain
Data Type:	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 on page 18), the Gain parameter sets the gain as an 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. From version 3.11.0, this parameter is deprecated. Instead, it is recommended to use the Gain_dB parameter (see above) to set the gain in dB.

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. PrioOpen = the iris control will attempt to keep the iris as open as possible while still maintaining good image quality. PrioClosed = the iris control will attemt to keep the iris as closed as possible while still maintaining good image quality.
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. The PrioOpen and PrioClosed settings may improve image quality in certain situations. Trying these settings under actual field conditions is the best way to determine whether one of the settings is appropriate for a particular application.

Parameter:	AntiFlicker
Data Type:	Enum
Valid Values:	Off = the anti-flicker feature is disabled. AntiFlicker_50Hz = the anti-flicker feature is enabled and set for an environment where the lights flicker at a 50 Hz rate. AntiFlicker_60Hz = the anti-flicker feature is enabled and set for an environment where the lights flicker at a 60 Hz rate.
Comments:	Enables or disables the anti-flicker feature. If the camera is operating in an environment where the lighting flickers at a 50 Hz or a 60 Hz rate (such as incandescent or fluorescent lights), the flickering lights can cause significant changes in brightness from image to image. Enabling the anti-flicker feature may reduce the effect of the flickering in the captured images Note that the Anti-Flicker feature will only be available when the Exposure Mode parameter (see page 18) is set to "PrioNone", "PrioFramerate", or "PrioQuality".

Parameter:	RegulationDelay_ds
Data Type:	Integer
Valid Values:	1 to 1800
Comments:	This setting is used to customize the operation of the camera's automatic controls. The RegulationDelay_ds parameter determines the delay time in tenths of a second (0.1 s) that should pass before a detected change in the image starts the adjustment by the automatic exposure, gain, and iris controls as well as the automatic white balance control.

Parameter:	Sharpness
Data Type:	Integer
Valid Values:	0 to 100
Comments:	Sets the sharpness of the captured images. Higher settings produce sharper images.

Parameter:	Saturation
Data Type:	Integer
Valid Values:	0 to 200
Comments:	Sets the color saturation of the images transmitted by the camera. Higher settings produce more saturated (colorful) images.

Parameter:	SaturationIRFilterOpen **New in V.3.15.0**
Data Type:	Integer
Valid Values:	0 to 100
Comments:	Lets you capture color images in night mode. A setting of 0 corresponds to normal night mode operation, where the camera uses the near-infrared light to capture black and white images. With settings above 0, the near-infrared light is still used but the captured images will contain gradually more color. Note that color accuracy is reduced in night mode because the IR-cut filter is not in front of the sensor.

Parameter:	Gamma
Data Type:	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:	PrivacyMask
Data Type:	String
Valid Values:	See Section 3 on page 11

Parameter:	Shadows **New in V 3.15.0**
Data Type:	Integer
Valid Values:	0 to 100
Comments:	Increases the contrast in dark areas of an image by selectively lightening the pixels in those areas alone. As a result, more detail is visible that would otherwise have been shades of black and gray. A setting of 0 means no pixels will be changed. Increasing the value, increases the lightening effect.

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 RedGain, and BlueGain parameters (see below).
Comments:	Sets the camera's white balance mode.

Parameter:	RedGain
Data Type:	Integer
Valid Values:	10 to 1000
Comments:	If the WhiteBalanceMode parameter (see above) is set to "Manual", then the RedGain parameter can be used to adjust the intensity of the red in the captured images. Decrease the setting to make the images less red, and increase the setting to make the images more red. Note that if the WhiteBalanceMode parameter is set to a value other than "Manual", the RedGain parameter will be read-only and will indicate the current red gain value as set by the automatic white balance control.

Parameter:	BlueGain
Data Type:	Integer
Valid Values:	10 to 1000
Comments:	If the WhiteBalanceMode parameter (see above) is set to "Manual", then the BlueGain parameter can be used to adjust the intensity of the blue in the captured images. Decrease the setting to make the images less blue, and increase the setting to make the images more blue. Note that if the WhiteBalanceMode parameter is set to a value other than "Manual", the BlueGain parameter will be read-only and will indicate the current blue gain value as set by the automatic white balance control.

Parameter:	WhiteBalanceMask
Data Type:	String
Valid Values:	See Section 3 on page 11
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:	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. InputControlled = the position of the IR-cut filter will be controlled by the state of an I/O port. For the InputControlled setting to work correctly, the I/O port that you want to use to control the position of the filter must be set to act as an input and the function for that I/O port must be set to "IRSwitch" (see the Section 4.18 on page 82 for information about setting the I/O ports). If theI/O port becomes active, the filter will be moved to the open position. If the I/O port becomes inactive, the filter will be moved to the closed position.
Comments:	Sets the functionality of the camera's IR-cut filter. Note that this parameter is only fully available on day/night cameras. On cameras with an external IR-cut filter where there is no automatic change mechanism for the IR-cut filter, the values "Open", "Closed", and "InputControlled" can be used to indicate the position of the filter. For detailed information about the IR-cut filter, see the user's manual of your camera.

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 user's manual of your camera.

Parameter:	IRFilterSwitchLevel
Data Type:	Integer
Valid Values:	-100 to +100
Comments:	When the IR Filter Mode parameter (see page 27) 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 IRFilterCurrentLevel 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 IRFilterCurrentLevel 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:	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:	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:	BlacklevelStrength
Data Type:	Integer
Valid Values:	0 to 1000
Comments:	Enables compensation of the black level of an image in order to achieve a higher contrast. This parameter can be used on its own or in conjunction with BlacklevelAutomatic (see below). If BlacklevelAutomatic is set to 0, the black level will be adjusted directly by changing the BlacklevelStrength alone. If BlacklevelAutomatic is set to a value from 1 to 1000, the black level will also be adjusted based on the internal black level of the image. In both cases, the higher the BlacklevelStrength, the higher the contrast. If both BlacklevelStrength and BlacklevelAutomatic are set to 0, no black level compensation will occur.

Parameter:	BlacklevelAutomatic
Data Type:	Integer
Valid Values:	0 to 1000
Comments:	Enables compensation of the black level of an image in order to achieve a higher contrast. If this is set to 1000, the camera will evaluate the image in order to determine its internal black level. The black level compensation can then be increased by adjusting the BlacklevelStrength (see above). If both BlacklevelStrength and BlacklevelAutomatic are set to 0, no black level compensation will occur.

4.4 Focus Group

The parameters in this group are used to configure the camera's autofocus functionality.

Note: Not all camera models are equipped with an autofocus motor.

Parameter:	OnePush
Data Type:	Command
Comments:	Starts the auto focus sequence.

Parameter:	Position
Data Type:	Integer
Valid Values:	0 to 100
Comments:	Read-only. Indicates the relative focus position.

Parameter:	Move
Data Type:	Enum
Valid Values:	NearCoarse = a coarse lens movement to focus near (N) objects. NearMedium = a medium lens movement to focus near (N) objects. NearFine = a fine lens movement to focus near (N) objects. Stop = stops lens movement. FarFine = a fine lens movement to focus far (F) objects. FarMedium = a medium lens movement to focus far (F) objects. FarCoarse = a coarse lens movement to focus far (F) objects.
Comments:	Adjusts the focus stepwise. The lens can be moved in fine, medium, or coarse steps to focus near (N) or far (F) objects.

4.5 ImageControlsRTT Group

The parameters in this group control the quality of images that are captured using real-time triggering.

Note: Not all camera models offer real-time triggering capabilities. To find out if your camera does, check the "Technical Specifications" chapter in the user's manual of your camera.

Parameter:	ControlMode	**New in V 3.15.0**
Data Type:	Enum	
Valid Values:	Manual = settings are controlled using the parameters in the ImageControlsRTT group. Stream = the same settings as for the ImageControls group will be used. ManualAdaptive = allows the camera to adapt to changes in lighting conditions. The camera detects these changes and is then able to intelligently switch from using the ImageControlsRTT group settings to using the ImageControls group settings if they are more appropriate. The following parameters are affected by this: ▪ ExposureTime ▪ Gain_dB ▪ WhiteBalanceMode	
Comments:	Determines how settings for real-time triggered images are controlled.	

Parameter:	ExposureTime	**New in V 3.15.0**
Data Type:	Enum	
Valid Values:	Time_1_25000_s = the exposure time is 1/25000 second Time_1_20000_s = the exposure time is 1/20000 second Time_1_16000_s = the exposure time is 1/16000 second Time_1_12000_s = the exposure time is 1/12000 second 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 (continued on the next page)	

(continued from the previous page)

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 1/1.6 second
Time_1_1p3_s = the exposure time is 1/1.3 second
Time_1_s = the exposure time is 1 second

Comments: Whenever the ControlMode parameter is set to "Manual" (see page 31), the ImageControlsRTT.ExposureTime parameter sets the exposure time in fractions of a second.

If the ControlMode parameter is set to "Stream", the ImageControlsRTT.ExposureTime parameter is read-only and the same settings as for the ImageControls.ExposureTime parameter (see page 21) are used.

If the ControlMode parameter is set to "ManualAdaptive", the camera first of all uses the ImageControlsRTT.ExposureTime parameter. If a change in lighting conditions is detected, the camera can intelligently switch to using the ImageControls.ExposureTime parameter.

Parameter:	Gain_dB	**New in V 3.15.0**
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 Gain_26_dB = the gain is 26 dB Gain_28_dB = the gain is 28 dB Gain_30_dB = the gain is 30 dB Gain_32_dB = the gain is 32 dB Gain_34_dB = the gain is 34 dB Gain_36_dB = the gain is 36 dB Gain_38_dB = the gain is 38 dB Gain_40_dB = the gain is 40 dB Gain_42_dB = the gain is 42 dB Gain_44_dB = the gain is 44 dB Gain_46_dB = the gain is 46 dB Gain_48_dB = the gain is 48 dB	
Comments:	Whenever the ControlMode parameter is set to "Manual" (see page 31), the ImageControlsRTT.Gain_dB parameter sets the gain in dB. If the ControlMode parameter is set to "Stream", the ImageControlsRTT.Gain_dB parameter is read-only and the same settings as for the ImageControls.Gain_dB parameter (see page 23) are used. If the ControlMode parameter is set to "ManualAdaptive", the camera first of all uses the ImageControlsRTT.Gain_dB parameter. If a change in lighting conditions is detected, the camera can intelligently switch to using the ImageControls.Gain_dB parameter.	

Parameter:	Sharpness	**New in V 3.15.0**
Data Type:	Integer	
Valid Values:	0 to 400	
Comments:	Whenever the ControlMode parameter is set to "Manual" (see page 31), the ImageControlsRTT.Sharpness parameter sets the sharpness of the captured images. Higher settings produce sharper images. If the ControlMode parameter is set to "Stream", the ImageControlsRTT.Sharpness parameter is read-only and the same settings as for the ImageControls.Sharpness parameter (see page 25) are used. The "ManualAdaptive" setting has no effect on this parameter. For more information about the "ManualAdaptive" setting, see page 31.	

Parameter:	Saturation	**New in V 3.15.0**
Data Type:	Integer	
Valid Values:	0 to 200	
Comments:	Whenever the ControlMode parameter is set to "Manual" (see page 31), the ImageControlsRTT.Saturation parameter sets the color saturation of the images transmitted by the camera. Higher settings produce more saturated (colorful) images. If the ControlMode parameter is set to "Stream", the ImageControlsRTT.Saturation parameter is read-only and the same settings as for the ImageControls.Saturation parameter (see page 25) are used. The "ManualAdaptive" setting has no effect on this parameter. For more information about the "ManualAdaptive" setting, see page 31.	

Parameter:	SaturationIRFilterOpen	**New in V 3.15.0**
Data Type:	Integer	
Valid Values:	0 to 100	
Comments:	Lets you capture color images in night mode on cameras with real-time trigger. A setting of 0 corresponds to normal night mode operation, where the camera uses the near-infrared light to capture black and white images. With settings above 0, the captured images will contain gradually more color. Note that color accuracy is reduced in night mode because the IR-cut filter is not in front of the sensor.	

Parameter:	Gamma	**New in V 3.15.0**
Data Type:	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. Whenever the ControlMode parameter is set to "Manual" (see page 31), the ImageControlsRTT.Gamma settings are used. If the ControlMode parameter is set to "Stream", the ImageControlsRTT.Gamma parameter is read-only and the same settings as for the ImageControls.Gamma parameter (see page 25) are used. The "ManualAdaptive" setting has no effect on this parameter. For more information about the "ManualAdaptive" setting, see page 31.	

Parameter:	Blacklevel	**New in V 3.15.0**
Data Type:	Integer	
Valid Values:	0 to 1000	
Comments:	Sets a black level compensation value in order to achieve a higher contrast. A setting of 0 means no black level compensation will take place. At higher settings, the black level will be adjusted based on the internal black level of the image. The higher the setting, the higher the contrast. Whenever the ControlMode parameter is set to "Manual" (see page 31), the ImageControlsRTT.Blacklevel settings are used. If the ControlMode parameter is set to "Stream", the ImageControlsRTT.Blacklevel parameter is read-only and the same settings as for the ImageControls.BlacklevelStrength and ImageControls.BlacklevelAutomatic parameters (see page 28) are used. The "ManualAdaptive" setting has no effect on this parameter. For more information about the "ManualAdaptive" setting, see page 31.	

Parameter:	Shadows	**New in V 3.15.0**
Data Type:	Integer	
Valid Values:	0 to 100	
Comments:	Increases the contrast in dark areas of an image by selectively lightening the pixels in those areas alone. As a result, more detail is visible that would otherwise have been shades of black and gray. A setting of 0 means no pixels will be changed. Increasing the value, increases the lightening effect. Whenever the ControlMode parameter is set to "Manual" (see page 31), the ImageControlsRTT.Shadows settings are used. If the ControlMode parameter is set to "Stream", the ImageControlsRTT.Shadows parameter is read-only and the same settings as for the ImageControls.Shadows parameter (see page 25) are used. The "ManualAdaptive" setting has no effect on this parameter. For more information about the "ManualAdaptive" setting, see page 31.	

Parameter:	WhiteBalanceMode	**New in V 3.15.0**
Data Type:	Enum	
Valid Values:	Default = the same setting as for the ImageControls.WhiteBalanceMode parameter (see page 16) will be used. Manual = white balance will be determined by the values for the RedGain and BlueGain parameters (see below).	
Comments:	Whenever the ControlMode parameter is set to "Manual" (see page 31), the ImageControlsRTT.WhiteBalanceMode parameter controls the camera's white balance mode. If the ControlMode parameter is set to "Stream", the ImageControlsRTT.WhiteBalanceMode parameter is read-only and the same settings as for the ImageControls.WhiteBalanceMode parameter (see page 25) are used. If the ControlMode parameter is set to "ManualAdaptive", the camera first of all uses the ImageControlsRTT.WhiteBalanceMode parameter. If a change in lighting conditions is detected, the camera can intelligently switch to using the ImageControls.WhiteBalanceMode parameter.	

Parameter:	RedGain	**New in V 3.15.0**
Data Type:	Integer	
Valid Values:	10 to 1000	
Comments:	If the ImageControlsRTT.WhiteBalanceMode parameter (see above) is set to "Manual", the RedGain parameter can be used to adjust the intensity of the red in the captured images. Decrease the setting to make the images less red, and increase the setting to make the images more red. If the ImageControlsRTT.WhiteBalanceMode parameter is set to "Default", the ImageControlsRTT.RedGain parameter will be read-only and will indicate the current red gain value as determined by the settings for the ImageControls.WhiteBalanceMode parameter (see page 26). If the ControlMode parameter is set to "Stream", the ImageControlsRTT.RedGain parameter is read-only and will indicate the current red gain value determined by the settings for the ImageControls.RedGain parameter. If the ControlMode parameter is set to "ManualAdaptive", the camera first of all uses the ImageControlsRTT.RedGain parameter. If a change in lighting conditions is detected, the camera can intelligently switch to using the ImageControls.RedGain parameter.	

Parameter:	BlueGain	**New in V 3.15.0**
Data Type:	Integer	
Valid Values:	10 to 1000	
Comments:	If the ImageControlsRTT.WhiteBalanceMode parameter (see above) is set to "Manual", the BlueGain parameter can be used to adjust the intensity of the blue in the captured images. Decrease the setting to make the images less blue, and increase the setting to make the images more blue. If the ImageControlsRTT.WhiteBalanceMode parameter is set to "Default", the ImageControlsRTT.BlueGain parameter will be read-only and will indicate the current blue gain value as determined by the settings for the ImageControls.WhiteBalanceMode parameter (see page 26). If the ControlMode parameter is set to "Stream", the ImageControlsRTT.BlueGain parameter is read-only and will indicate the current red gain value determined by the settings for the ImageControls.BlueGain parameter. If the ControlMode parameter is set to "ManualAdaptive", the camera first of all uses the ImageControlsRTT.BlueGain parameter. If a change in lighting conditions is detected, the camera can intelligently switch to using the ImageControls.BlueGain parameter.	

4.6 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.

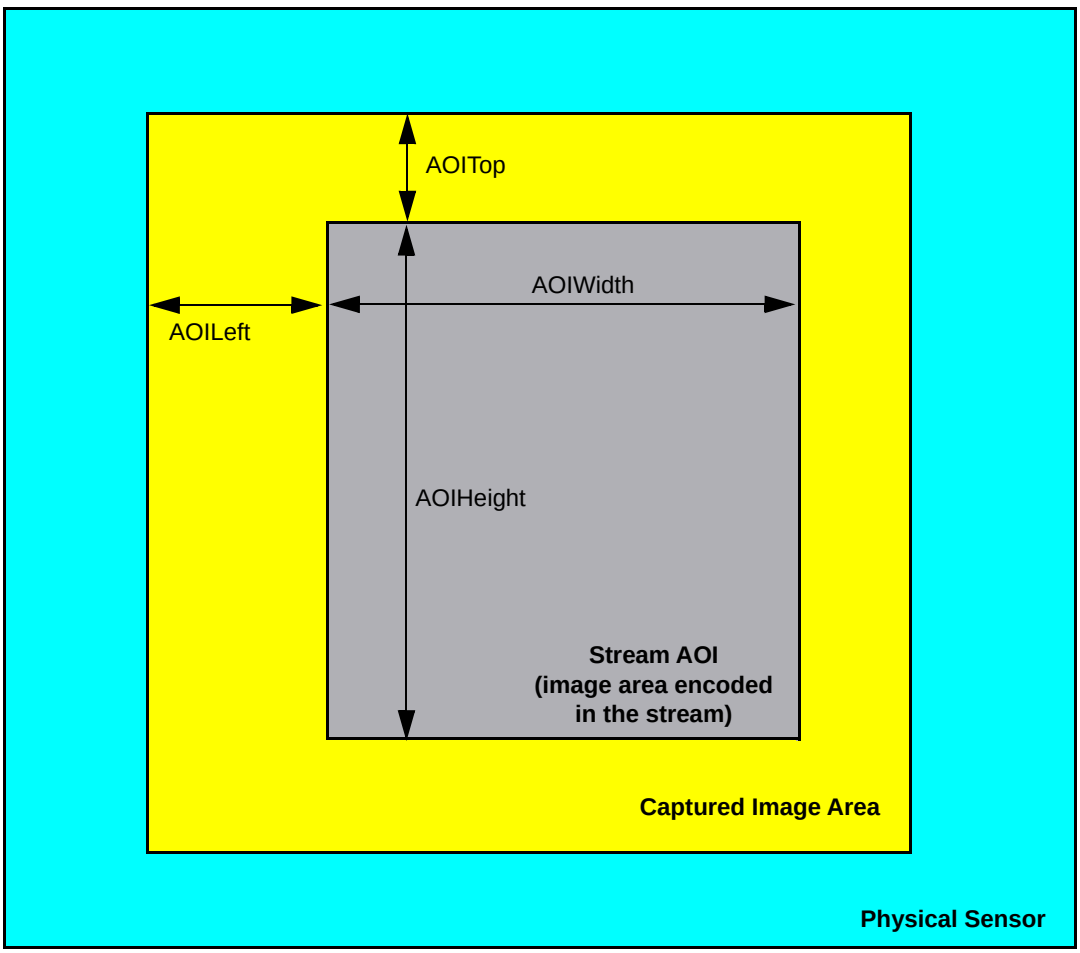


Fig. 3: Stream AOI

Parameter:	StreamSelector
Data Type:	Integer
Valid Values:	0 = stream 0. 1 = stream 1. 2 = stream 2. 3 = stream 3.
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.** Note that the web application only makes three streams (0, 1, and 2) available to the user. Stream 3 is only accessible via the API. Some camera models may only support a limited number of streams. Use the GetMax method to determine the maximum value for the StreamSelector 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 MPEG-4 encoding. H_264 = stream enabled and set for H.264 base profile encoding. H_264_HIGH = stream enabled and set for H.264 high profile encoding. JPEG_TRIGGERED = stream will be enabled and will only include images that were acquired by the application of a real-time trigger (see the Function parameter on page 83 and also see the camera user's manual for more details). When the EncoderType parameter is set to "JPEG_TRIGGERED", the EncoderMode parameter (see below) is fixed to "VBR" and the FrameRateScaling parameter (see page 44) is fixed to "FpsScale_1_1". YUV_TRIGGERED = stream enabled and set for I420 color format. The stream will only include images that were acquired by the application of a real-time trigger (see the Function parameter on page 83 and also the camera user's manual for more details). No compression is used for YUV images. Note that the large size of the YUV images may have an impact on the camera's overall performance. For more information about the YUV encoder type, see the *Lossless YUV Images* application note (AW001261). You can download this document from the Basler website: www.baslerweb.com
Comments:	Enables or disables the selected stream (see StreamSelector above) and sets the encoder type for the stream. Stream 0 cannot be set to "Off". If the sensor AOI (see Section 4.2 on page 14) is set to a size greater than 1920 x 1920, MPEG-4 cannot be used as the encoder type. This restriction only applies to camera models with a sensor large enough to allow a sensor AOI greater than 1920 x 1920 (see the specifications in the camera user's manual for information about the sensor size in each model). **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 12).

Parameter:	EncoderMode
Data Type:	Enum
Valid Values:	CBR = constant bitrate. The encoder attempts to maintain a constant output bitrate by adjusting image quality as necessary. The bitrate will be determined by the setting of the Bitrate parameter. VBR = variable bitrate. The encoder attempts to maintain constant image quality by adjusting the output bitrate as necessary. The quality level will be determined by the setting of the Quality parameter. CVBR = constant-variable bitrate. As long as there is no motion in the scene, the encoder attempts to maintain a constant output bitrate by adjusting image quality as necessary. The bitrate will be determined by the setting of the Bitrate parameter. When the scene includes motion, the camera will increase the output bitrate (up to a maximum determined by the setting of the MaxBitrate parameter) for an extended period of time. When that time ends, the camera will reduce the bitrate as needed to achieve a long term average bitrate equal to the setting of the Bitrate parameter. CAPPED_VBR = Capped variable bitrate. This is the default mode when communication with the camera occurs via ONVIF. By setting the Quality and MaxBitrate parameters, you can ensure that the encoder attempts to maintain the desired quality without exceeding a certain bitrate at which the performance of the camera would be negatively affected.
Comments:	Sets the encoder mode for the selected stream (see StreamSelector on page 39). If the EncoderType parameter (see page 39) is set to "JPEG_TRIGGERED" the EncoderMode parameter is fixed to "VBR". The CVBR setting is only valid when the EncoderType parameter is set to H_264 or H_264_HIGH. The CAPPED_VBR setting is only valid when the EncoderType parameter is set to H_264 or JPEG.

Parameter:	ChainMode
Data Type:	Enum
Valid Values:	Chained = the P-frame is predicted from the I-frame or P-frame directly preceding it. Unchained = the P-frame is predicted from the preceding I-frame.
Comments:	The Chained setting is only valid, if the EncoderType parameter is set to H_264 or H_264_HIGH. In an H.264 encoded 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. Since P-frames are predicted frames, they require information from the previous I-frame or P-frame. **The ChainMode 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 12).

Parameter:	SetSynchronizationPoint
Data Type:	Command
Comments:	Triggers the creation of an I-frame in the currently selected H.264 or MPEG-4 encoded stream.

Parameter:	Bitrate
Data Type:	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 bitrate for the selected stream (see StreamSelector on page 39). If the EncoderMode parameter is set to "CVBR", the Bitrate parameter sets the bitrate for the selected stream when there is no motion in the scene.

Parameter:	MaxBitrate
Data Type:	Integer
Valid Values:	Use the GetMin and GetMax methods to determine the range.
Comments:	If the EncoderMode parameter is set to "CVBR", the MaxBitrate parameter sets the maximum bitrate for the selected stream when there is motion in the scene. If the EncoderMode parameter is set to "CAPPED_VBR", the MaxBitrate parameter sets the maximum bitrate for the stream that can’t be exceeded while trying to achieve the desired quality.

Parameter:	Quality
Data Type:	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 (see StreamSelector on page 39). Higher values equal better quality. If the EncoderMode parameter is set to "CAPPED_VBR", the Quality parameter sets the quality level for the selected stream. 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 EncoderType parameter is set to "JPEG" than it will have when it is set to "MPEG4".

Parameter:	GopLength_ms
Data Type:	Integer
Valid Values:	Use the GetMin and GetMax methods to determine the range.
Comments:	If the EncoderType parameter is set to "MPEG4", "H_264" or "H_264_HIGH", then the GOPLength_ms parameter sets the time between I-frames in milliseconds for the selected stream (see StreamSelector on page 39). In an MPEG-4 or an H.264 encoded 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 GOPLength 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:	Integer
Valid Values:	Must be less than or equal to the current setting for the **sensor** AOIWidth parameter. Use the GetMin and GetMax methods to determine the minimum and maximum. Use the GetInc method to determine the increment.
Comments:	As shown in Figure 3 on page 38, sets the width of the image area (in pixels) that will be encoded in the selected stream (see StreamSelector on page 39). **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 12).

Parameter:	AOIHeight
Data Type:	Integer
Valid Values:	Must be less than or equal to the current setting for the **sensor** AOIHeight parameter. Use the GetMin and GetMax methods to determine the minimum and maximum. Use the GetInc method to determine the increment.
Comments:	As shown in Figure 3 on page 38, sets the height of the image area (in pixels) that will be encoded in the selected stream (see StreamSelector on page 39). **The 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 12).

Parameter:	AOILeft
Data Type:	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. Use the GetMin and GetMax methods to determine the minimum and maximum. Use the GetInc method to determine the increment.
Comments:	As shown in Figure 3 on page 38, 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 (see StreamSelector on page 39).

Parameter:	AOITop
Data Type:	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. Use the GetMin and GetMax methods to determine the minimum and maximum. Use the GetInc method to determine the increment.
Comments:	As shown in Figure 3 on page 38, 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 (see StreamSelector on page 39).

Parameter:	SetMaxAOI
Data Type:	Command
Comments:	Sets the stream AOI to the largest possible size.

Parameter:	OutputSize
Data Type:	Enum
Valid Values:	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. Auto = the image size is non-standard. (Whenever you manually change the stream AOIWidth parameter or the stream AOIHeight parameter to a value that will not result in standard size image output, the value of the OutputSize parameter will change to "Auto"). Size_2560x1920 = the size of the images in the stream will be 2560 x 1920. Size_1920x1080 = the size of the images in the stream will be 1920 x 1080 (Full HD). Size_1600x1200 = the size of the images in the stream will be 1600 x 1200 (UXGA). Size_1408x1152 = the size of the images in the stream will be 1408 x 1152 (16CIF). Size_1280x1024 = the size of the images in the stream will be 1280 x 1024 (SXGA). Size_1280x960 = the size of the images in the stream will be 1280 x 960. Size_1280x720 = the size of the images in the stream will be 1280 x 720 (HD 720). Size_1056x864 = the size of the images in the stream will be 1056 x 864 (9CIF). 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 (SVGA). 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_640x480 = 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 480 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 (see StreamSelector on page 39) 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 12).

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 (see StreamSelector on page 39) 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 12).

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 (see StreamSelector on page 39). If the EncoderType parameter (see page 39) is set to "JPEG_TRIGGERED" the FrameRateScaling parameter is fixed to "FpsScale_1_1".

Parameter:	OverlayText
Data Type:	String
Valid Values:	The string can include text and also the following variables: $date$ = display current date/time (see page 78 to set the format). $timestamp$ = display timestamp (sec:µsec 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. cpu = display the load on the camera's CPU. $SysInfo.ModelName$ = 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. $image$ = display an overlay image in the current stream (see Section 8 on page 111).
Comments:	Sets the text that will appear as an overlay on the selected stream (see StreamSelector on page 39). It is possible to define 2 separate text overlays.

Parameter:	OverlayText2
Data Type:	String
Valid Values:	See OverlayText parameter above.
Comments:	See OverlayText parameter above.

Parameter:	OverlayPosition
Data Type:	Enum
Valid Values:	Top = the text overlay will appear at the top left of the streamed images. Bottom = the text overlay will appear at the bottom left of the streamed images. TopRight = the text overlay will appear at the top right of the streamed images. BottomRight = the text overlay will appear at the bottom right of the streamed images.
Comments:	Sets the position of the text overlay for the selected stream (see StreamSelector on page 39).

Parameter:	OverlayPosition2
Data Type:	Enum
Valid Values:	See OverlayPosition parameter above.
Comments:	See OverlayPosition parameter above.

Parameter:	OverlaySize
Data Type:	Enum
Valid Values:	Small = the text in the overlay will be in 10 point type. Medium = the text in the overlay will be in 15 point type. Large = the text in the overlay will be in 20 point type. Scaled = the text in the overlay will be scaled to 1/15 of the stream AOI height.
Comments:	Sets the size of the text in the overlay for the selected stream (see StreamSelector on page 39). Note that if the text overlay is longer than the image width, the trailing end of the overlay will be truncated.

Parameter:	OverlaySize2
Data Type:	Enum
Valid Values:	See OverlaySize parameter above.
Comments:	See OverlaySize parameter above.

Parameter:	OverlayStyle
Data Type:	Enum
Valid Values:	Normal = the overlay will display black text on a white background bar. Inverted = the overlay will display white text on a black background bar. White = the overlay will display white text with no background bar. Black = the overlay will display black text with no background bar.
Comments:	Sets the style of the text overlay for the selected stream (see StreamSelector on page 39). This parameter applies across both overlays. If you choose "Black" for overlay 1, the style for overlay 2 will also be "Black".

Parameter:	LiveRecording
Data Type:	Enum
Valid Values:	Off = live recording is turned off. The live stream of the enabled stream will not be saved to the SD card/FTP server. Always = live recording is always turned on. The live stream of the enabled stream will permanently be saved to the SD card/FTP server. OnAlarm = live recording will be started for the enabled stream if an alarm condition has been declared and if you have enabled the LiveRecording alarm action (see page 64). When the alarm condition is declared, the LiveRecording parameter automatically changes its value from OnAlarm to AlarmRecording. Note that the beginning and the behavior of recording the live stream is determined by the LiveRecordingMode parameter (see page 67). AlarmRecording = live recording will be started immediately for the enabled stream (regardless of whether the LiveRecording alarm action has been enabled or not). When an alarm condition ends and the LiveRecording alarm action has been enabled, the LiveRecording parameter automatically changes its value from AlarmRecording to OnAlarm. Note that the behavior of recording the live stream is determined by the LiveRecordingMode parameter (see page 67).
Comments:	Sets the mode for live recording. You should be aware that if the load on the camera's microprocessor is already large during normal operation, saving live streams to the SD card can influence the overall performance of the camera. Also take this fact into consideration if you want to save several streams to the SD card simultaneously. When saving live stream files to the SD card, note the settings of the LiveMemoryAllocation parameter (see page 74) and the OverwriteLiveRecordings parameter (see page 74). (Not all camera models are equipped with an SD card slot. See Section 6 on page 105 for information about accessing data saved to the SD card.)

Parameter:	LiveRecordingStorageLocation
Data Type:	Enum
Valid Values:	SDCard = The live stream will be saved to an SD card. FTP = The live stream will be saved to an FTP server.
Comments:	(Not all camera models are equipped with an SD card slot. See Section 6 on page 105 for information about accessing data saved to the SD card.)

Parameter:	LiveRecordingVideoFileLength
Data Type:	Integer
Valid Values:	0 to 60
Comments:	Sets the maximum length in seconds of an individual live stream video segment saved to the SD card or the FTP server. The live stream(s) is/are saved in the form of many individual live stream video segments in a raw image format. Each video segment has a defined length in seconds. For JPEG encoded streams, a parameter value of 0 causes the camera to store single images. (Not all camera models are equipped with an SD card slot. See Section 6 on page 105 for information about accessing data saved to the SD card.)

Parameter:	LiveBufferSize
Data Type:	Integer
Valid Values:	Min = 2048 KB Max = see note on page 50.
Comments:	Sets the size of the "live image" buffer for the selected stream (see StreamSelector on page 39). The live image buffer is a ring 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 89. **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 12).

Parameter:	AlarmBufferSize
Data Type:	Integer
Valid Values:	Min = 0 KB Max = see note on page 50.
Comments:	Sets the size of the alarm buffer (in KB) for the selected stream (see StreamSelector on page 39). 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 ring 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 set 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 89. **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 12).

Parameter:	PostAlarmBufferSize
Data Type:	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.

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 (see StreamSelector on page 39).

Parameter:	AlarmBufferDisable
Data Type:	Command
Comments:	Disables the alarm buffer for the selected stream (see StreamSelector on page 39).

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.)

Parameter:	ExcludeRttImages	**New in V. 3.15.0**
Data Type:	Enum	
Valid Values:	Off = Real-time triggered images will be included in the image stream. On = If the Stream.EncoderType parameter (see page 39) is set to a non-RTT type, real-time triggered images will be excluded from the live stream.	
Comments:	Determines whether real-time triggered images will be included in the live stream.	

Parameter:	AudioEncoderType
Data Type:	Enum
Valid Values:	Off = No audio encoder. G711_ALaw = The selected stream is A-law encoded. The content type is "audio/PCMA". G711_ULaw = The selected stream is µ-law encoded. The content type is "audio/PCMU". PCM_RAW = The selected stream uses PCM RAW encoding. Audio data are sent as 16-bit PCM data sampled with 16 kHz in network (big-endian) order. The content type is "audio/raw".
Comments:	Enables or disables the audio functionality for the selected stream and sets the audio encoder type for the stream. The audio data can be streamed via HTTP or RTSP together with MJPEG, MPEG-4, and H.264 encoded streams. How to access audio streams via HTTP or RTSP, see Chapter 5 on page 89. **The AudioEncoderType 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 12).

Note

The live buffer for each stream must be a minimum of 2048 KB. The alarm buffer for each stream can 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. 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.

4.7 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:	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:	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", areas where motion has been detected in the images will be highlighted with blocks of green pixels.

Parameter:	RegionSelector
Data Type:	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 11
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:	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:	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:	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:	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:	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.

4.8 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:	Integer
Valid Values:	0 = RTP streaming disabled. 1 = RTP streaming enabled.
Comments:	Enables or disables RTP streaming.

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

Parameter:	Multicast
Data Type:	Integer
Valid Values:	0 = multicast streaming disabled. 1 = multicast streaming enabled.
Comments:	Enables or disables multicast streaming. Multicast streaming is only valid for stream 0, and for multicasting to operate correctly, the EncoderType parameter (see page 39) for stream 0 must be set to "MPEG4", "H_264", or "H_264_HIGH".

Parameter:	MulticastOnDemand
Data Type:	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 page 54) 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:	MulticastStream
Data Type:	Integer
Valid Values:	0 to *n*
Comments:	Sets the number of the stream that is used for multicast streaming.

Parameter:	MulticastMetainfo
Data Type:	Integer
Valid Values:	0 = no SEI NAL header is being added before all frames. 1 = SEI NAL header is being added before all frames.
Comments:	Sets whether the SEI NAL header is added before all frames when using H.264 multicast streaming.

Parameter:	MulticastIP
Data Type:	String
Valid Values:	An IP address in the range from 224.0.1.0 to 239.255.255.255.
Comments:	Sets the IP address for multicast streaming.

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

Parameter:	MulticastTTL
Data Type:	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.

4.9 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:	ZeroConfiguration
Data Type:	Integer
Valid Values:	0 = Acquisition of a "zero configuration" IP address is disabled. 1 = Acquisition of a "zero configuration" IP address is enabled.
Comments:	Enables or disables the acquisition of a dynamic IP address in the 169.254.0.0/16 IP subnet, which is reserved for networking within the local network. By default, the camera acquires a "zero configuration" IP address. For more information about zero configuration, see Section 11.2 on page 134.

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

Parameter:	HostNameSource
Data Type:	Enum
Valid Values:	DHCP = The camera receives the HostName parameter setting from a DHCP server response, if available. Manual = The value for the HostName parameter must be set manually.
Comments:	Determines the way the HostName parameter will be set.

Parameter:	NameServerSource
Data Type:	Enum
Valid Values:	DHCP = The camera receives the NameServer parameter setting from a DHCP server response, if available. Manual = The value for the NameServer parameter must be set manually.
Comments:	Determines the way the NameServer parameter will be set.

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:	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:	HostName
Data Type:	String
Valid Values:	Only letters, digits, and dashes (up to 64 characters) are allowed.
Comments:	Assigns a host name to the camera if DHCP is disabled or the HostNameSource parameter is set to "Manual".

Parameter:	NameServer
Data Type:	String
Valid Values:	A space separated list of valid IP addresses.
Comments:	Sets the name server the camera will use if DHCP is disabled or the NameServerSource parameter is set to "Manual". Multiple name servers can be defined.

Parameter:	SearchDomain
Data Type:	String
Valid Values:	A space separated list of valid domain names.
Comments:	Sets the domain names the camera will use for resolving non-fully qualified host names via DNS lookups.

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

Parameter:	RxTraffic
Data Type:	Integer
Comments:	Read-only. Indicates the amount of incoming network traffic in kilobits per second.

Parameter:	TxTraffic
Data Type:	Integer
Comments:	Read-only. Indicates the amount of outgoing network traffic in kilobits per second.

4.10 QoS Group

The parameters in this group are used to set network traffic prioritization settings, commonly known as QoS or Quality of Service settings. Note that these settings will only have an effect on networks where all network switches and routers support QoS.

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:	HTTPDSCP
Data Type:	Integer
Valid Values:	0 to 63
Comments:	Sets the HTTP DSCP (differentiated services code point) value.

Parameter:	RTSPDSCP
Data Type:	Integer
Valid Values:	0 to 63
Comments:	Sets the RTSP DSCP (differentiated services code point) value.

Parameter:	AlarmDSCP
Data Type:	Integer
Valid Values:	0 to 63
Comments:	Sets the alarm DSCP (differentiated services code point) value.

4.11 SNMP Group

Basler IP cameras support the Simple Network Management Protocol (SNMPv2c). The parameter in this group is used to enable the camera's SNMP agent.

Parameter:	Enable
Data Type:	Integer
Valid Values:	0 = SNMP agent disabled. 1 = SNMP agent enabled.
Comments:	Enables or disables the SNMP agent. Note that the SNMP agent has to be enabled for using the heartbeat missing alarm source (see page 63).

Heartbeat Functionality via SNMP

The aim of the heartbeat functionality is to enable an external server telling the camera via SNMP that it is still available. For this purpose, the SNMP server sends a heartbeat timeout value ("kick"), which has to be greater than or equal to 0, to the SNMP agent in the camera.

If a missing heartbeat is detected by the camera, an alarm condition can be declared. This missing heartbeat is detected by an expired timeout. To use the heartbeat functionality for declaring an alarm condition, the heartbeat missing alarm source has to be selected (see page 63).

The SNMP agent provides a Basler Enterprise MIB with the OID ".1.3.6.1.4.1.39093". The variables necessary for using the heartbeat functionality are described in the following table.

Variable	OID	Description
basler.heartbeat.kick	.1.3.6.1.4.1.39093.1.2	the timeout value in seconds of the heartbeat. The default value is -1. A timeout value greater than or equal to 0 sets and starts the timeout period. To restart the timeout period, the timeout value has to be set again. A timeout value of -1 effectively disables the heartbeat functionality.
basler.heartbeat.remaining	.1.3.6.1.4.1.39093.1.3	the remaining time in seconds within this timeout period. This value is set automatically to the timeout value when the "kick" is received. The value will be decremented down to 0 updating every second.
basler.heartbeat.expired	.1.3.6.1.4.1.39093.1.4	the time in seconds since the heartbeat has expired. This value will be incremented by 1 updating every second.

For read access, the SNMP server must use the community string "public". For read-write access (i.e. setting the "kick" timeout value), the SNMP server must use the community string "private".

4.12 ONVIF Group

All Basler IP cameras offer ONVIF support based on core specification version 1.02. For more information about ONVIF, see Section 7 on page 107.

Parameter:	Enable
Data Type:	Integer
Valid Values:	0 = Disables the camera for ONVIF support. 1 = Enables the camera for ONVIF support.
Comments:	By default, the ONVIF support is disabled.

Parameter:	NumStreams
Data Type:	Integer
Valid Values:	1 = Makes stream 1 accessible via ONVIF. 2 = Makes stream 1 and 2 accessible via ONVIF. 3 = Makes stream 1, 2, and 3 accessible via ONVIF.
Comments:	By default, this parameter is set to 1. Changing the value while ONVIF is enabled, will cause an internal restart of the ONVIF daemon. Existing connections will be interrupted. If the number of streams is reduced, the parameters of the streams that are not used anymore will keep their current values.

4.13 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 = an alarm condition can be declared via the UserTrigger parameter (see page 64). PIO = an alarm condition will be declared when an active signal becomes present on a camera I/O port. For the PIO setting to work correctly, the I/O port that you want to use to declare an alarm condition must be set to act as an input and the function for that I/O port must be set to "AlarmTrigger" or to "Real-timeTrigger" (see Section 4.18 on page 82 for more information about setting the I/O ports). MotionDetection = an alarm will be declared when motion is detected. Motion detection must be enabled (see Section 4.7 on page 51). Cyclic = a recurrent alarm condition can be declared. The alarm will be declared each time when a configurable time has expired (see the CyclicPeriod parameter on page 64). HeartbeatMissing = an alarm condition will be declared when the heartbeat timeout expires. Note that the SNMP agent has to be enabled for using the heartbeat missing alarm source. (See Section 4.11 on page 61 for more information about the heartbeat functionality via SNMP.)
Comments:	Selects an alarm source to enable or disable (see the next parameter). The alarm source is used to declare an alarm condition.

Parameter:	SourceEnable
Data Type:	Integer
Valid Values:	0 = the selected alarm source will be disabled 1 = the selected alarm source will be enabled
Comments:	Enables or disables the selected alarm source.

Parameter:	ActionSelector
Data Type:	Enum
Valid Values:	PIO = when an alarm condition is declared, an active signal will become present on a camera I/O port. For the PIO setting to work correctly, the I/O port that you want to use to declare an alarm condition must be set to act as an output and the function for that I/O port must be set to "AlarmAnnounce" (see the Section 4.18 on page 82 for more information about setting the I/O ports). The amount of time that the port will remain active after an alarm is declared will be determined by the setting of the PIOHoldTime parameter (see page 65). Email = when an alarm condition is declared, send an Email to the recipient specified by the Email parameter (see page 69). See page 71 for a sample email. HTTP = when an alarm condition is declared, send an HTTP request to the URL specified by the HTTPURL parameter. FTP = when an alarm condition is declared, send a text file upload to the FTP server specified by the FTPServer parameter (see page 70). See page 71 for a sample text file. SDCard = save a text file to the SD card. See page 71 for a sample text file. LiveRecording = when an alarm condition is declared, save the live stream to the SD card. You can save several streams to the SD card simultaneously. The behavior of saving the live stream to the SD card is determined by the LiveRecordingMode parameter (see page 67).
Comments:	Selects an alarm action to work with. (Not all camera models are equipped with an SD card slot. See Section 6 on page 105 for information about accessing data saved to the SD card.)

Parameter:	ActionEnable
Data Type:	Integer
Valid Values:	0 = the selected alarm action will be disabled. 1 = the selected alarm action will be enabled.
Comments:	Enables or disables the selected alarm action (see the ActionSelector parameter on page 64). When an alarm is declared, all enabled actions will be executed.

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

Parameter:	CyclicPeriod
Data Type:	Integer
Valid Values:	0 to 86400000
Comments:	If the Cyclic alarm source has been enabled (see the SourceSelector parameter on page 63), the CyclicPeriod parameter sets the time in seconds between two alarm conditions that must expire before the recurrent alarm is declared.

Parameter:	PIOHoldTime
Data Type:	Integer
Valid Values:	0 to 3600000
Comments:	If the PIO action has been enabled (see the ActionSelector parameter page 64), the PIOHoldTime parameter sets the amount of time in milliseconds that the output port will remain active after an alarm is declared. Note that if the PIOHoldTime is set to 0, the output port will remain active only as long as the alarm condition remains active.

Parameter:	ActionIncludeImg
Data Type:	Integer
Valid Values:	0 = do not include an image with the selected alarm action. 1 = include an image with the selected alarm action.
Comments:	If you have selected the Email or FTP, or SDCard alarm action (see the AlarmSelector parameter above), the ActionIncludeImg parameter enables or disables the inclusion of an image along with the selected alarm action. A JPG image will be included for each enabled stream that is set for JPEG encoding. No image will be included for MPEG-4 or H.264 encoded streams. (Not all camera models are equipped with an SD card slot. See Section 6 on page 105 for information about accessing data saved to the SD card.)

Parameter:	ActionIncludeStream
Data Type:	Integer
Valid Values:	0 = do not include alarm buffers with the selected alarm action. 1 = include alarm buffers with the selected alarm action.
Comments:	If you have selected the SD card alarm action (see the AlarmSelector parameter above), the ActionIncludeStream parameter enables or disables the saving of alarm buffers to the SD card when an alarm condition is declared. When an alarm condition is declared, any alarm buffer that is in the "armed" state will transition to the "active" state and begin to accumulate post alarm images. When the post alarm portion of the buffer(s) is full, the buffer(s) will transition from the "active" state to the "done" state. If saving the alarm buffers to the SD card is enabled, the pre and post alarm images in the buffer(s) will be saved to the SD card when the buffer(s) transition from the "active" state to the "done" state. The saved file for each active alarm buffer on an MJPEG encoded stream will have a .mjpeg file extension. The saved file for each active alarm buffer on an MPEG-4 encoded stream will have a .m4v file extension. The saved file for each active alarm buffer on an H.264 encoded stream will have a .h264 file extension. (Not all camera models are equipped with an SD card slot. See Section 6 on page 105 for information about accessing data saved to the SD card.)

Parameter:	SDCardRearmAlarmBuffer
Data Type:	Integer
Valid Values:	0 = do not rearm alarm buffer(s). 1 = rearm alarm buffer(s).
Comments:	Specifies whether alarm buffers should automatically be rearmed after they have been successfully saved to the SD card. (Not all camera models are equipped with an SD card slot.)

Parameter:	SDCardOverwrite
Data Type:	Integer
Valid Values:	0 = discard any new saved files if the allocated memory space on the SD card is full. 1 = overwrite the existing data if the allocated memory space on the SD card is full.
Comments:	Specifies what the system should do with new saved files, if the portion of the memory space that is currently allocated for alarm data storage is full. This portion is determined by the setting of the LiveMemoryAllocation parameter described on page 74. (Not all camera models are equipped with an SD card slot.)

Parameter:	LiveRecordingPreAlarmTimePeriod
Data Type:	Integer
Valid Values:	0 to 60
Comments:	Sets the amount of time in seconds that will be used for saving the data of a live stream before an alarm is declared. The storage requirement in the live buffer for saving pre-alarm data depends on the bitrate of the encoder. To make sure that the live buffer is large enough for saving the determined time of pre-alarm data, the settings of the live buffer size and the alarm buffer size for the selected stream must be carefully tuned. Higher settings of the LiveRecordingPreAlarmTimePeriod parameter require that the live buffer size is also set to a higher value, which thus reduces the remaining space available for the alarm buffer. (For information about the live buffer size and the alarm buffer size, see page 48.)

Parameter:	LiveRecordingPostAlarmTimePeriod
Data Type:	Integer
Valid Values:	0 to 60
Comments:	Sets the amount of time in seconds that will be used for saving the data of a live stream after an alarm has been declared. The determined time of post-alarm data has only a minor impact on the storage requirement in the live buffer. Unlike the pre-alarm data, the post-alarm data does not need to be kept temporarily in the live buffer (see the LiveRecordingPreAlarmTimePeriod parameter). When an alarm is declared, storing post alarm data only requires as much memory as is needed for buffering the encoder output before it will be saved to the SD card/FTP server.

Parameter:	LiveRecordingMode
Data Type:	Enum
Valid Values:	DuringAlarm = saving the live stream to the SD card will be started if an alarm condition has been declared and if you have selected the LiveRecording alarm action (see page 64). Saving the live stream will be stopped as soon as the alarm condition ends. StartOnly = saving the live stream to the SD card will be started if an alarm condition has been declared and if you have selected the LiveRecording alarm action (see page 64). It will, however, not be stopped when the alarm condition ends. StopOnly = saving the live stream to the SD card will be stopped if an alarm condition ends and if you have selected the LiveRecording alarm action (see page 64). It will, however, not be started automatically when the alarm condition is declared.
Comments:	Specifies the behavior of the LiveRecording alarm action. This parameter will only have an effect if the LiveRecording parameter has one of the values "OnAlarm" or "AlarmRecording" (see page 47). (Not all camera models are equipped with an SD card slot.)

Parameter:	HTTPURL
Data Type:	String
Valid Values:	Any valid URL.
Comments:	If the HTTP action has been enabled, the HTTPURL parameter sets the URL that will be requested when an alarm is declared. You could, for example, enter this URL: http://MyServer/cgi-bin/alarm.cgi The transmitted string will be URL encoded. The camera will automatically add the following parameters to the end of the URL: host=<hostname> = the camera's host name, typically something like this: "Basler-20809681" date=<date/time> = the date and time when the alarm has been declared, e.g.: "2012-01-03 16:30:41 CEST". How to set the date and time format, see page 78, how to set the time zone, see page 81). ts=<timestamp> = time in sec:µsec since 1 Jan 1970, 0:00:00 GMT, when the alarm has been declared, e.g. something like this: "1330342430.625201" nr=<alarm number> = the number of the alarm, e.g. "2". Source_User=<boolean> = if the user is the source that has declared the alarm, the value is set to "1", otherwise to "0". Source_PIO=<boolean> = if the camera I/O ports are enabled as a source for declaring an alarm condition and an active signal becomes present on a camera I/O port, the value is set to "1", otherwise to "0". Source_Motion=<boolean> = if motion detection is enabled as an alarm source and motion is detected, the value is set to "1", otherwise to "0". Source_Cyclic=<boolean> = if the cyclic alarm source has been enabled to declare a recurrent alarm, the value is set to "1", otherwise to "0". Source_Heartbeat=<boolean> = if the heartbeat missing alarm source has been enabled and a heartbeat timeout occurs, the value is set to "1", otherwise to "0". If motion detection is enabled as an alarm source and motion is detected, the camera will automatically add the following parameters to the end of the above mentioned parameters. MotionDetected=<boolean> = if motion has been detected, the value is set to "1". MotionRegion_0=<boolean> = if motion has been detected in motion detection region 0, the value is set to 1", otherwise to "0". MotionLevel_0=<motion level> = current motion level (number of changed pixels) in region 0. The parameter descriptions for the remaining parameters MotionRegion_1/MotionLevel_1 to MotionRegion_4/MotionLevel_4, are identical to the MotionRegion_0/MotionLevel_0 parameter descriptions.

Parameter:	HTTPURLAlarmEnd
Data Type:	String
Valid Values:	Any valid URL.
Comments:	If the HTTP action has been enabled, the HTTPURLAlarmEnd parameter sets the URL that will be requested when an alarm has ended. You could, for example, enter this URL: http://MyServer/cgi-bin/alarm.cgi The transmitted string will be URL encoded. The camera will automatically add the following parameters to the end of the URL: host=<hostname> = the camera's host name, typically something like this: "Basler-20809681" date=<date/time> = the date and time when the alarm has been declared, e.g.: "2012-01-03 16:30:41 CEST". How to set the date and time format, see page 78, how to set the time zone, see page 81). ts=<timestamp> = time in sec:µsec since 1 Jan 1970, 0:00:00 GMT, when the alarm has been declared, e.g. something like this: "1330342430.625201" nr=<alarm number> = the number of the alarm, e.g. "2".

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 JPG image captured at the time of the alarm. The JPG image will be included for each enabled stream that is set for JPEG encoding. No image will be included for MPEG-4 or H.264 encoded streams.

Parameter:	EmailFrom
Data Type:	String
Valid Values:	Any valid email address.
Comments:	Specifies the "From" address that will appear in the email sent by the camera. The variable $hostname$ can be used in the string and will be replaced by the camera's actual host name.

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:	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 digits 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 JPG image captured at the time of the alarm. A JPG image will be included for each enabled stream that is set for JPEG encoding. No image will be included for MPEG-4 or H.264 encoded streams.

Parameter:	FTPRemoteDir
Data Type:	String
Valid Values:	All standard keyboard characters are valid.
Comments:	Sets the path to a target subdirectory for the FTP upload. (If no path is specified, the root directory will be used.)

Parameter:	FTPPort
Data Type:	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 digits 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.

Parameter:	ImageOverlayText	Deleted in V 3.0.1
Comments:	This parameter is no longer valid.	

Parameter:	ImageOverlayPosition	Deleted in V 3.0.1
Comments:	This parameter is no longer valid.	

Sample Email / Text File

If the camera is set so that an email or a text file will be sent when an alarm is declared, the content of the email of the text file will be similar to the following two examples:

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

Triggered by Cyclic: 0

Triggered by Heartbeat: 0

An alarm has been triggered by the camera named: Basler-21140068

Time: Mon Nov 28 16:17:54 UTC 2011

This is alarm number: 145

Triggered by User: 0

Triggered by PIO: 0

Triggered by Motion: 1

Triggered by Cyclic: 0

Triggered by Heartbeat: 0

Motion Status: 1

- in Region 0: 1 (Level: 1948)
- in Region 1: 0 (Level: 0)
- in Region 2: 0 (Level: 0)
- in Region 3: 0 (Level: 0)
- in Region 4: 0 (Level: 0)

4.14 LiveRecording Group

The parameters in this group are used to configure the settings for saving live recording data to an FTP server.

Parameter:	FTPServer
Data Type:	String
Valid Values:	Any valid FTP server IP address.
Comments:	Sets the FTP server that will be used for storing live recording data via FTP.

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

Parameter:	FTPRemoteDir
Data Type:	String
Valid Values:	All standard keyboard characters are valid.
Comments:	Sets the path to a target subdirectory for the FTP upload. (If no path is specified, the root directory will be used.)

Parameter:	FTPUserName
Data Type:	String
Valid Values:	Up to 15 digits 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.

4.15 SDCard Group

The parameters in this group are used to configure the camera's SD card.

Not all camera models are equipped with an SD card slot.

The following file systems are supported:

- FAT16 (VFAT)
- FAT32 (VFAT)

Parameter:	Present
Data Type:	Integer
Valid Values:	0 = SD card not present/mounted. 1 = SD card present/mounted.
Comments:	Read-only. Indicates the presence of an SD card in the camera. SD cards formatted with an unsupported file system will not be recognized.

Parameter:	Erase
Data Type:	Command
Comments:	Erases the contents of the SD card. If the SD card cannot be accessed at the file system level, it will be reformatted. If the SD card is formatted with an unsupported file system, this parameter can be used to format the SD card with a supported file system.

Parameter:	Size
Data Type:	Integer
Comments:	Read-only. Indicates the total size of the SD card in kilobytes.

Parameter:	Avail
Data Type:	Integer
Comments:	Read-only. Indicates the amount of free space available on the SD card in kilobytes.

Parameter:	LiveMemoryAllocation
Data Type:	Integer
Valid Values:	0 to 100
Comments:	Sets the percentage of free space available on the SD card that should be allocated for live stream data storage. The remaining memory space is used for alarm data storage. Default = 50 The setting can only be changed in fixed increments of 10. Note that if the value is set to 0, recording of live streams to the SD card is effectively turned off. If the value is set to 100, saving of alarm data to the SD card is effectively turned off.

Parameter:	OverwriteLiveRecordings
Data Type:	Integer
Valid Values:	0 = discard any new saved live stream files if the allocated memory space on the SD card is full. 1 = overwrite the oldest live stream files if the allocated memory space on the SD card is full.
Comments:	Sets whether the oldest live stream files saved to the SD card should be overwritten automatically, if the portion of the memory space that is currently allocated for live stream data storage is full. This portion is determined by the setting of the LiveMemoryAllocation parameter described on page 74.

4.16 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:	Integer
Valid Values:	0 = forwarding disabled. 1 = forwarding enabled.
Comments:	Enables or disables serial port forwarding via TCP/IP.

Parameter:	Mode
Data Type:	Enum
Valid Values:	S_TELNET = the network connection behaves like a normal telnet connection using the byte "0xFF" as an escape character for the input of commands. S_RAW = the telnet protocol is disabled. Only the data from the serial port is transmitted over the network without any interpretation.
Comments:	Sets the mode for serial port forwarding.

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:	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:	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 digits 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.

4.17 System Group

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

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

Parameter:	UserData
Data Type:	String
Valid Values:	Up to 4096 characters. All standard keyboard characters are valid.
Comments:	Provides storage for user defined data.

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. If the value ends with UTC, the date and time will be interpreted as UTC instead of local time, which is the default.

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 $f = milliseconds 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 (For an example of using these variables to format the date and time, see the request example on page 78.)

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.
	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=$F $T`

Parameter:	DateTimeStatus
Data Type:	Enum
Valid Values:	Invalid = the date and time information in the camera is invalid. Valid = the date and time information in the camera is valid.
Comments:	If camera power is lost, the camera includes a capacitive device that can maintain the date and time information for several days at least. If the camera loses power for a short period of time, it will maintain its internal date and time information and this parameter will indicate "Valid" once power is reapplied and the camera finishes booting up. If the camera loses power for an extended period of time, it will lose its internal date and time information and this parameter will indicate "Invalid" once power is reapplied and the camera finishes booting up.

Parameter:	ExtClockSynchronization
Data Type:	Enum
Valid Values:	None = Clock synchronization is disabled. NTP = NTP synchronization is enabled. PTP = PTP synchronization is enabled
Comments:	Enables or disables clock synchronization and sets the synchronization method. The BIP2 camera supports clock synchronization using IEEE 1588-2008 (Version 2) PTP specification as well as synchronization with an NTP server. If this parameter is set to "NTP", the synchronization period has to be set using the NTPSyncPeriod parameter (see page 80).

Parameter:	NTP
Data Type:	Enum
Valid Values:	Off = NTP synchronization is disabled. M_1 = NTP synchronization occurs once every 1 minute. M_2 = NTP synchronization occurs once every 2 minutes. M_4 = NTP synchronization occurs once every 4 minutes. M_8 = NTP synchronization occurs once every 8 minutes. M_15 = NTP synchronization occurs once every 15 minutes. M_30 = NTP synchronization occurs once every 30 minutes. H_1 = NTP synchronization occurs once every 1 hour. H_2 = NTP synchronization occurs once every 2 hours. H_4 = NTP synchronization occurs once every 4 hours. H_12 = NTP synchronization occurs once every 12 hours. H_24 = NTP synchronization occurs once every 24 hours. Weekly = NTP synchronization occurs once per week.
Comments:	Enables or disables clock synchronization with an NTP server. From version 3.10.0, you can achieve the same effect using the ExtClockSynchronization parameter (see above). The NTP parameter still behaves as before but is deprecated now.

Parameter:	NTPSyncPeriod
Data Type:	Enum
Valid Values:	M_1 = NTP synchronization occurs once every 1 minute. M_2 = NTP synchronization occurs once every 2 minutes. M_4 = NTP synchronization occurs once every 4 minutes. M_8 = NTP synchronization occurs once every 8 minutes. M_15 = NTP synchronization occurs once every 15 minutes. M_30 = NTP synchronization occurs once every 30 minutes. H_1 = NTP synchronization occurs once every 1 hour. H_2 = NTP synchronization occurs once every 2 hours. H_4 = NTP synchronization occurs once every 4 hours. H_12 = NTP synchronization occurs once every 12 hours. H_24 = NTP synchronization occurs once every 24 hours. Weekly = NTP synchronization occurs once per week.
Comments:	If the ExtClockSynchronization parameter (see page 79) is set to "NTP", NTPSyncPeriod is used to set the synchronization period.

Parameter:	NTPServerSource
Data Type:	Enum
Valid Values:	DHCP = The camera receives the NTPServer parameter setting from a DHCP server response, if available. Manual = The value for the NTPServer parameter must be set manually
Comments:	Determines the way the NTPServer parameter will be set.

Parameter:	NTPServer
Data Type:	String
Valid Values:	A space separated list of valid IP addresses or server names.
Comments:	IP address or name of the NTP server to use (NTP must be enabled) if DHCP is disabled (see the DHCP parameter on page 57) or the NTPServerSource parameter is set to "Manual". Multiple NTP servers can be defined.

Parameter:	TimeZoneDesc		
Data Type:	String		
Valid Values:	Any valid time zone code as specified by POSIX.		
	The following entries are also valid:		
	CET	Europe/Oslo	Israel
	EET	Europe/Paris	Japan
	MET	Europe/Prague	Poland
	EST	Europe/Stockholm	Portugal
	CST6CDT	Europe/Warsaw	PRC
	MST	Europe/Zagreb	Singapore
	PST8PDT	Europe/Zurich	Turkey
	US/Central	Africa/Cairo	GMT
	US/Eastern	Africa/Johannesburg	Etc/GMT-1
	US/Hawaii	Asia/Almaty	Etc/GMT+1
	US/Mountain	Asia/Bangkok	Etc/GMT-2
	US/Pacific	Asia/Dubai	Etc/GMT+2
	America/Denver	Asia/Hong_Kong	Etc/GMT-3
	America/Detroit	Asia/Istanbul	Etc/GMT+3
	America/Los_Angeles	Asia/Jerusalem	Etc/GMT-4
	America/Montreal	Asia/Kuala_Lumpur	Etc/GMT+4
	America/New_York	Asia/Kuwait	Etc/GMT-5
	America/Vancouver	Asia/Novosibirsk	Etc/GMT+5
	Europe/Amsterdam	Asia/Qatar	Etc/GMT-6
	Europe/Athens	Asia/Shanghai	Etc/GMT+6
	Europe/Belfast	Asia/Singapore	Etc/GMT-7
	Europe/Berlin	Asia/Taipei	Etc/GMT+7
	Europe/Bratislava	Asia/Tehran	Etc/GMT-8
	Europe/Copenhagen	Asia/Tokyo	Etc/GMT+8
	Europe/Dublin	Asia/Vladivostok	Etc/GMT-9
	Europe/Helsinki	Atlantic/Reykjavik	Etc/GMT+9
	Europe/Kaliningrad	Australia/Adelaide	Etc/GMT-10
	Europe/Kiev	Australia/Brisbane	Etc/GMT+10
	Europe/Lisbon	Australia/Melbourne	Etc/GMT-11
	Europe/London	Australia/Sydney	Etc/GMT+11
	Europe/Luxembourg	Egypt	Etc/GMT-12
	Europe/Madrid	Eire	Etc/GMT+12
	Europe/Malta	Hongkong	Etc/GMT-13
	Europe/Minsk	Iceland	Etc/GMT-14
	Europe/Moscow	Iran	
Comments:	Sets the time zone.		

4.18 IO Group

Basler IP Fixed Box Cameras includes three I/O ports: I/O-0, I/O-1, and I/O-2. The ports are accessed via pins in the camera's terminal connector.

Basler IP Fixed Dome Cameras includes two I/O ports: I/O-0 and I/O-1. The ports are accessed via pins in the camera's main terminal block.

Each I/O port can be set to act as either an input or an output. Once a port has been set as an input or an output, the functionality of the port can be set in several different ways.

The parameters in this group are used to set each port as an input or an output and to set the functionality of the configured ports.

Parameter:	IOSelector
Data Type:	Integer
Valid Values:	0 = I/O port 0. 1 = I/O port 1. 2 = I/O port 2.
Comments:	Selects the I/O port to work with. Changes made to the Direction, Function, State, and Invert parameters will affect the port that has been selected by the I/O selector parameter.

Parameter:	Direction
Data Type:	Enum
Valid Values:	Input = the port will be set as an input. Output = the port will be set as an output.
Comments:	Determines whether the selected port will be set as an input or an output.

Parameter:	Function
Data Type:	Enum
Valid Values:	**For a port with the Direction parameter (see page 82) set to Input:** Monitor = The state of the input port can be monitored only (using the State parameter described on page 84). The camera will not react to any changes in the state of the port. IRSwitch = On day/night camera models, the input port will be used to set the position of the camera's IR-cut filter. When the port becomes active, the filter will move to the open position (filter not in front of the sensor). When the port becomes inactive, the filter will move to the closed position (filter in front of the sensor). For the IRSwitch setting to work correctly, the IRFilterMode parameter (see page 27) must be set to "InputControlled". AlarmTrigger = When the input port becomes active, an alarm condition will be declared. For the AlarmTrigger setting to work correctly, the parameter in the Alarm group (see page 63) must be set to "PIO", and the SourceEnable parameter in the Alarm group must be set to 1. RealtimeTrigger = The real-time trigger function is used to force acquisition of an image at a specific point in time. When the input port becomes active, any images in the process of being acquired by the camera will be aborted. When the abort process is complete, the camera will begin exposure of a new image. The camera will then resume capturing images as it normally does. (See the camera user's manual for more details.) If the SourceSelector parameter in the Alarm group (see page 63) is set to "PIO" and the SourceEnable parameter in the Alarm group is set to 1, an alarm condition will be declared each time the input port becomes active. Camera streams can be set so that they only include images that were captured as a result of the real-time trigger (see the EncoderType parameter on page 39). ExternalTrigger = The external trigger function is used to control the exposure start without the necessity to use any other parameters. If the Function parameter of an input pin is set to "ExternalTrigger" and no external trigger is applied, the camera immediately stops capturing images. If this function is used and the EncoderMode parameter is set to "CBR" or "CVBR" (see page 40), the camera's sensor frame rate has to be set manually via the FrameRateMode parameter in the Sensor group (see page 17). The reason is that for those two encoder modes, the encoder needs a constant bit rate as input. As this is difficult to achieve with external triggers, which can be applied at irregular intervals, and because the camera does not measure the current frame rate, you have to set a frame rate manually. For the encoder to be able to calculate the desired bitrate for CBR and CVBR correctly, the FrameRateMode should be set to the same frame rate as the frequency of the externally applied triggers. (continued on the next page)

(continued from the previous page)

For a port with the Direction parameter (see page 82) set to Output:

UserOutput = You can use the State parameter (see page 84) to set the state of the output port.

IRFilterAnnounce = On day/night camera models, the state of the output port will announce the position of the camera's IR-cut filter. When the filter is in the open position (filter not in front of the sensor), the port will be set to active. When the filter is in the closed position (filter in front of the sensor), the port will be set to inactive.

Strobe = The camera will output a pulse on the output port that is synchronized to the start of each image capture. This signal is useful for controlling such things as a strobe exposure lamp. The StrobeDelay and StrobeDuration parameters (see page 85) are used to adjust the characteristics of the pulse.

AlarmAnnounce = The state of the output port will announce the camera's alarm condition. The output port will become active when an alarm condition is declared. The period of time that the port will remain active is determined by the PIOHoldTime parameter (see page 65).
For the AlarmAnnounce setting to work correctly, the ActionSelector parameter in the Alarm group (see page 64) must be set to "PIO", and the ActionEnable parameter in the Alarm group must be set to 1.

Comments: Sets the functionality of the port.

Parameter:	State
Data Type:	Integer
Valid Values:	0 = the state of the input port is inactive. 1 = the state of the input port is active.
Comments:	If the Direction parameter (see page 82) for the port is set to "Input", the State parameter will indicate the current state of the port and is read-only. If the Direction parameter for a port is set to "Output" and the Function parameter (see page 83) is set to "IRFilterAnnounce", "Strobe", or "AlarmAnnounce", the State parameter will indicate the current state of the port and is read-only. If the Direction parameter for a port is set to "Output" and the Function parameter is set to "UserOutput", the State parameter can be used to set the state of the port.

Parameter:	Invert
Data Type:	Integer
Valid Values:	0 = the port will operate normally. 1 = the operation of the port will be inverted.
Comments:	Sets whether the behavior of the selected port with regard to being active or inactive is normal or is inverted. (See the Terminal Connector section of the camera user's manual for detailed information about the meaning of active and inactive with regard to the ports.)

Parameter:	StrobeMode	**New in V.3.15.0**
Data Type:	Enum	
Valid Values:	FlashWindow = lets you control the length of the strobe pulse using the StrobeDelay and StrobeDuration parameters (see below) on cameras with rolling shutter. ExposureActive = lets you control the length of the strobe pulse using the StrobeDelay and StrobeDuration parameters (see below) on cameras with global shutter. On cameras with rolling shutter, StrobeDelay and StrobeDuration are fixed and can't be changed. RTTExposureActive = The strobe signal will only be active for real-time triggered images. You can control the length of the strobe pulse using the StrobeDelay and StrobeDuration parameters.	
Comments:	If the Function parameter (see page 83) is set to "Strobe", the StrobeMode parameter allows you to choose between different ways of controlling strobe lighting because depending on whether your camera has a global or a rolling shutter or whether it offers real-time triggering, the strobe requirements differ. Check the specification tables in your user's manual to find out which option applies to your camera.	

Parameter:	StrobeDelay	**New in V.3.15.0**
Data Type:	Integer	
Valid Values:	0 to 10000	
Comments:	For any port where the Direction parameter (see page 82) is set to "Output" and the Function parameter is set to "Strobe" (see page 83), the StrobeDelay parameter sets a delay time (in µs) between when image capture starts and when the strobe pulse becomes active. The StrobeDelay setting applys to all ports set for "Output" and "Strobe".	

Parameter:	StrobeDuration
Data Type:	Integer
Valid Values:	10 to 20000
Comments:	For any port where the Direction parameter (see page 82) is set to "Output" and the Function parameter is set to "Strobe" (see page 83), the StrobeDuration parameter sets the length of time (in µs) that the strobe pulse will remain active. The StrobeDuration setting applys to all ports set for "Output" and "Strobe".

Parameter:	MonitorAll
Data Type:	8 Bit Bitfield
Comments:	This is a read-only 8 bit bitfield. Bit 0 indicates the current state of I/O-0. Bit 1 indicates the current state of I/O-1. Bit 2 indicates the current state of I/O-3. Bits 3 through 7 are reserved.

4.19 Audio Group

The parameters in this group are used to configure the camera's audio functionality.

For information about how to enable or disable the audio functionality for the selected stream and how to set the audio encoder type for the stream, see page 50.

Note: Not all camera models are equipped with an audio connector.

Parameter:	MicPowerEnable
Data Type:	Enum
Valid Values:	Off = switches the microphone (phantom) power off. On = switches the microphone (phantom) power on.
Comments:	Switches the microphone power on or off.

Parameter:	InputLevelSelect
Data Type:	Enum
Valid Values:	MicIn = the input device is a microphone. LineIn = the input device is a line-in device.
Comments:	Selects the type of the audio input device. If the input device is a microphone (MicIn), an additional gain of 20 dB is applied to the input. This is commonly used to amplify the input from unpowered microphones.

Parameter:	InputGain
Data Type:	Enum
Valid Values:	Gain_Auto = the gain will automatically be adjusted to obtain an appropriate level for a range of input signal levels. Weak signals will be amplified, strong signals will be attenuated. Gain_0_dB = no additional gain is applied to the input. Gain_6_dB/Gain_12_dB/Gain_18_dB/Gain_24_dB = an additional gain of 6/12/18/24 dB is applied to the input.
Comments:	Determines the level of the additional input gain.

4.20 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.

4.21 IOPins Group

The parameters in this group have been deleted and their functionality has been replaced by the parameters in the I/O group (see Section 4.18 on page 82).

For more information about handling legacy applications, see Appendix A on page 137.

Parameter:	InputPin0	**Deleted in V 3.0.1**
Comments:	This parameter should not be used.	

Parameter:	InputPin0Mode	**Deleted in V 3.0.1**
Comments:	This parameter should not be used.	

Parameter:	OutputPin0Function	**Deleted in V 3.0.1**
Comments:	This parameter should not be used.	
Parameter:	OutputPin0	**Deleted in V 3.0.1**
Comments:	This parameter should not be used.	

Parameter:	OutputPin0Mode	**Deleted in V 3.0.1**
Comments:	This parameter should not be used.	

Parameter:	StrobeDelay_us	**Deleted in V 3.0.1**
Comments:	This parameter should not be used.	

Parameter:	StrobeDuration_us	**Deleted in V 3.0.1**
Comments:	This parameter should not be used.	

5 Accessing Video and Audio Streams

This section describes how to access video and audio streams of a Basler IP camera including its buffers.

Only the dome camera models support an audio functionality and can deliver audio information along with the video stream. The audio stream (A-law, µ-law or PCM RAW encoded) is delivered as part of the video stream (MJPEG, MPEG-4, H.264, or YUV encoding). You can access the stream via HTTP or RTSP/RTP.

5.1 MJPEG Encoded Streams

To use the information in this section effectively, you must understand how the buffers of the camera operate. You can refer to the text on page 48 through page 50 for some basic information about the buffers. For more detailed information, you can read the application note called *Using Basler IP Camera Live Buffers and Alarm Buffers*. You can download the application note from the Basler website: www.baslerweb.com/manuals

5.1.1 Accessing MJPEG Streams

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

`rtsp://<camera>/jpeg`

The RTP payload format for motion JPEG encoded streams restricts the width and height of the stream to 2040x2040 pixels.

To access the stream via HTTP, 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 | triggered]

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 the last available image in the live or alarm buffer. 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.

triggered - gets the images from a stream that is set to only include images that are acquired by using the cameras real-time trigger feature.

stream = N

Number of the stream you want to access.

buffer = N

Number of the buffer you want to access.

seek = N

Seek back N milliseconds in time into the buffer of 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 = timeshift, 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.

audio = [true | false]

true - the audio data will be embedded into the video stream.

false - no audio data is delivered.

Note that the audio parameter is only used in HTTP requests and not in RTSP requests.

If the audio data is delivered via HTTP, an audio content type HTML header is used. The content type for a G.711 A-law encoder is "audio/PCMA", the content type for a G.711 µ-law encoder is "audio/PCMU", and the content type for the PCM_RAW encoder is "audio/raw" (see also the AudioEncoderType parameter on page 50).

If the audio data is delivered via RTSP/RTP and is available for the requested stream, the camera offers an additional audio track in the RTSP DESCRIBE response. The track can be selected in addition to the video track and will be streamed through the PLAY command.

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

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

5.1.2 MJPEG Stream Access Examples

There are two common techniques for accessing the MJPEG streams: the stream-oriented 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, that stream 0 is MJPEG encoded, and that you simply want to access the most current images from stream 0 including its audio data. Via HTTP, you would issue this request:

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

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

The same request via RTSP would look like this:

```
rtsp://IPCam_1/jpeg?stream=0&mode=live
```

If audio data is available for the requested stream, an additional audio track is delivered in the RTSP DESCRIBE response. Streaming the audio data can be started via the PLAY command.

Now assume that you want to view the contents of the alarm buffer on stream 0. First you must determine whether the alarm buffer for the stream is in an "armed", "active" or "done" state by issuing the following two requests to select stream 0 and check the alarm buffer state:

```
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 stream 0 alarm buffer is in an "armed" state, it means that the buffer is continuously buffering live images. **Streaming from an alarm buffer that is in an armed state is not recommended.**

If the return indicates that the stream 0 alarm buffer is in an "active" state, it means that an alarm condition has been declared, and that the buffer is currently buffering post alarm images.

If the return indicates that the stream 0 alarm buffer is in a "done" state, it means that the buffer is full (of pre and post alarm images) and is no longer buffering images.

If the buffer is in either an "active" or a "done" state, you can stream the stored content from the buffer by issuing a request with the mode parameter set to "replay" like this:

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

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

The fps parameter determines 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.

You cannot stream from an alarm buffer that is disabled.

Finally, assume that you have your camera set so that stream 1 is only encoding and streaming images that were triggered by the camera's real-time trigger function (i.e., stream 1 is set for the "JPEG_TRIGGERED" encoder type). To access the images from stream 1 via HTTP, you would issue this request:

```
http://IPCam_1/cgi-bin/mjpeg?stream=1&mode=triggered
```

(For more information about the camera's real-time trigger feature, see the camera user's manual.)

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,audio/PCMA)
buffer_1=(0,"Stream 0",image/jpeg,ALARM,1024x768,audio/PCMA)
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 including audio data, an alarm stream 0 including audio data, 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 including its audio data, you would issue this request via HTTP:

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

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

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

The same request via RTSP would look like this:

```
rtsp://IPCam_1/jpeg?buffer=2&mode=live
```

If audio data is available for the requested stream, an additional audio track is delivered in the RTSP DESCRIBE response. Streaming the audio data can be started via the PLAY command.

Now assume that you want to view the contents of the alarm buffer on stream 0. First you must determine whether the alarm buffer for the stream is in an "armed", "active" or "done" state by issuing the following two requests to select stream 0 and check the alarm buffer state:

```
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 stream 0 alarm buffer is in an "armed" state, it means that the buffer is continuously buffering live images. **Streaming from an alarm buffer that is in an armed state is not recommended.**

If the return indicates that the stream 0 alarm buffer is in an "active" state, it means that an alarm condition has been declared, and that the buffer is currently buffering post alarm images.

If the return indicates that the stream 0 alarm buffer is in a "done" state, it means that the buffer is full (of pre and post alarm images) and is no longer buffering images.

If the buffer is in either an "active" or a "done" state, you can stream the stored content from the buffer by issuing a request with the mode parameter set to "replay" like this:

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

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

The fps parameter determines 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.

You cannot stream from an alarm buffer that is disabled.

Finally, assume that you have your camera set so that stream 0 is encoding images as motion JPEG and stream 1 is only encoding and streaming images that were triggered by the camera's real-time trigger function (i.e., stream 0 is set for the "JPEG" encoder type and stream 1 is set for the "JPEG_TRIGGERED" encoder type). You would access the images from stream 1, in this manner:

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,1280x960,audio/PCMA)
buffer_1=(0,"stream 0",image/jpeg,ALARM,1280x960,audio/PCMA)
buffer_2=(1,"stream 1",image/jpeg,TRIGGERED,1280x960)
```

To access the images including the audio data from stream 1 via HTTP, you would issue this request:

```
http://IPCam_1/cgi-bin/mjpeg?buffer=2&mode=triggered&audio=true
```

(For more information about the camera's real-time trigger feature, see the camera user's manual.)

5.1.3 EXIF Information

Each image in a motion JPEG encoded stream from the camera includes an EXIF header that provides some basic information about the camera and the image. The header includes the following standard fields:

Field:	Content:
Manufacturer	Basler
Camera	Camera model
DateTime	Date and time of image capture in Yr:Mo:Day Hr:Min:Sec format
SubsecTime	Subseconds in microseconds
Image Description	Data text (see below)

An example of the data text included in the Image Description field is:

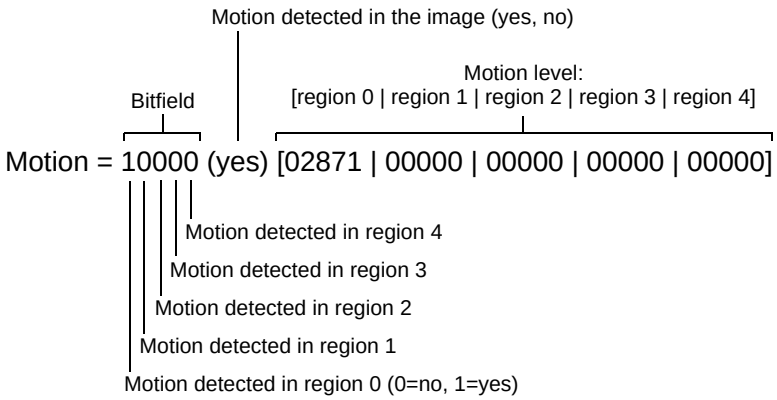
```
FrameNR=0000012703
AOI=(0800x0592)@(0400,0308)/(1600x1200)
Motion=10000 (yes) [02871 | 00000 | 00000 | 00000 | 00000]
Alarm=00100 (yes)
IO=000
RtTrigger=0
```

The FrameNR line indicates the value of the camera's frame counter when the image was captured.

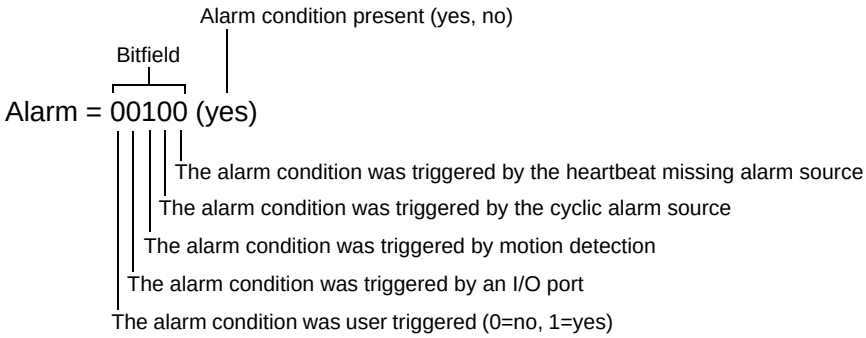
The information in the AOI line is interpreted as follows:

AOI=(AOIWidth x AOIHeight)@(AOILeft, AOITop)/(SensorWidth x SensorHeight)

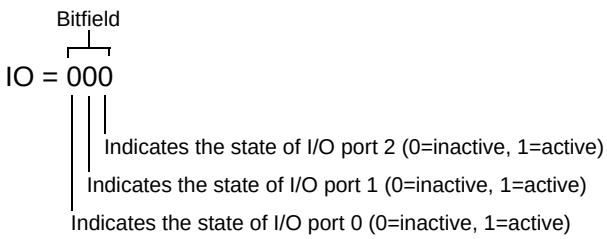
The information in the motion line is interpreted as follows:



The information in the alarm line is interpreted as follows:



The information in the IO line is interpreted as follows:



The RtTrigger line indicates whether the image was acquired as the result of a real-time trigger signal. A value of 0 indicates that the image acquisition was not the result of a real-time trigger. A value of 1 indicates that the image acquisition was the result of a real-time trigger. (For more details about the real-time trigger feature, see the camera user's manual.)

5.2 MPEG-4 or H.264 Encoded Streams

Note

To use the information in this section effectively, you must understand how the buffers of the camera operate. You can refer to page 48 and page 50 for some basic information about the buffers. For more detailed information, you can read the application note called *Using Basler IP Camera Live Buffers and Alarm Buffers*. You can download the application note from the Basler website: www.baslerweb.com/manuals

If audio data is available for the requested stream, the camera offers an additional audio track in the RTSP DESCRIBE response. The track can be selected in addition to the video track and will be streamed through the PLAY command.

Multicasting can only be enabled when stream 0 is set for MPEG-4 or H.264 encoding.

- If multicasting is disabled, the camera will only transmit a unicast stream 0.
- When multicasting is enabled, the camera transmits both a unicast stream and a multicast stream for stream 0.

5.2.1 Accessing Unicast H.264 or MPEG-4 Streams

You can access H.264 or MPEG-4 encoded **unicast** streams by using a request in one of the following forms:

`rtsp://<camera>/h264` (for an H.264 encoded stream)

`rtsp://<camera>/mpeg4` (for an MPEG-4 encoded stream)

Alternately, the streams can also be requested without specifying the encoder type. Only the stream number has to be specified. The encoder type that has been set for the stream will be used in this case.

`rtsp://<camera>/<stream>`

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 these requests:

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.

stream = N

Number of the stream you want to access.

Alternately, you can enter the stream as follows:

`rtsp://<camera>/<stream>`

In this case, you do not have to specify the encoder type that has been set for the stream.

buffer = N

Number of the buffer you want to access.

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.

metainfo = [sei | sei-i]

sei - the camera will transmit meta-information in a SEI NAL unit before each i-frame and each p-frame.

sei-i - the camera will transmit meta-information in a SEI NAL unit before each i-frame only.

(Note that SEI NAL units only apply to streams that are H.264 encoded. See the page 101 for information about the format of the SEI NAL unit.)

follow = [0 | 1]

0 - RTSP streaming will not be continued if the AOI size is changed.

1 - RTSP streaming will be continued if the AOI size is changed.

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

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

5.2.2 Unicast Stream Access Examples

There are two common techniques for accessing unicast H.264 or MPEG-4 streams: the stream-oriented 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, that stream 0 is H.264 encoded, and that you want to access the most current images from stream 0. You would issue this request:

```
rtsp://IPCam_1/h264?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.

Now assume that you want to view the contents of the alarm buffer on stream 0. First you must determine whether the alarm buffer for the stream is in an "armed", "active" or "done" state by issuing the following two requests to select stream 0 and check the alarm buffer state:

```
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 stream 0 alarm buffer is in an "armed" state, it means that the buffer is continuously buffering live images. **Streaming from an alarm buffer that is in an armed state is not recommended.**

If the return indicates that the stream 0 alarm buffer is in an "active" state, it means that an alarm condition has been declared, and that the buffer is currently buffering post alarm images.

If the return indicates that the stream 0 alarm buffer is in a "done" state, it means that the buffer is full (of pre and post alarm images) and is no longer buffering images.

If the buffer is in either an "active" or a "done" state, you can stream the stored content from the buffer by issuing a request with the mode parameter set to "replay" like this:

```
rtsp://IPCam_1/h264?stream=0&mode=replay&fps=5.0
```

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

The fps parameter determines 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.

You cannot stream from an alarm buffer that is disabled.

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/h264,LIVE,1024x768)
buffer_1=(0,"Stream 0",image/h264,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.) Stream 0 is H.264 encoded. Streams 1 and 2 are MJPEG encoded.

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

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

Now assume that you want to view the contents of the alarm buffer on stream 0. First you must determine whether the alarm buffer for the stream is in an "armed", "active" or "done" state by issuing the following two requests to select stream 0 and check the alarm buffer state:

```
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 stream 0 alarm buffer is in an "armed" state, it means that the buffer is continuously buffering live images. **Streaming from an alarm buffer that is in an armed state is not recommended.**

If the return indicates that the stream 0 alarm buffer is in an "active" state, it means that an alarm condition has been declared, and that the buffer is currently buffering post alarm images.

If the return indicates that the stream 0 alarm buffer is in a "done" state, it means that the buffer is full (of pre and post alarm images) and is no longer buffering images.

If the buffer is in either an "active" or a "done" state, 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.

	You cannot stream from an alarm buffer that is disabled.

5.2.3 Accessing an H.264 or MPEG-4 Multicast Stream

When MPEG-4 or H.264 encoding is 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` (for an MPEG-4 encoded stream)

`rtsp://<camera>/h264?multicast` (for an H.264 encoded stream)

	Multicast streaming is only available for stream 0. No optional parameters can be used with a request to access a multicast stream.

5.2.4 Multicast Stream Access Examples

Assume that you are working with a camera named IPCam_1. Also assume that stream 0 is set for MPEG-4 encoding, that multicasting is enabled on stream 0, and that you want to access the stream 0 **multicast stream**.

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

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

When stream 0 is set for multicasting, the camera issues both a unicast stream and a muticast stream for stream 0. You can access the stream 0 unicast stream via a unicast type of requests.

5.2.5 SEI NAL Unit Format

If you include the metainfo parameter when accessing an H.264 encoded stream, the format of the SEI NAL unit included with the images is as follows:

Payload Type: user_data_unregistered (5)

UUID: 628a4ec0-b429-4348-976c-ddd5bfb15438

user_data_payload:

Version: 0x02 (1 byte)

Data: ASCII text

Example of the data text:

FrameNR=0000012703

AOI=(0800x0592)@(0400,0308)/(1600x1200)

Motion=10000 (yes) [02871 | 00000 | 00000 | 00000 | 00000]

Alarm=00100 (yes)

IO=000

DateTime=2009:07:14 12:59:48

SubsecTime=055788

RtTrigger=0

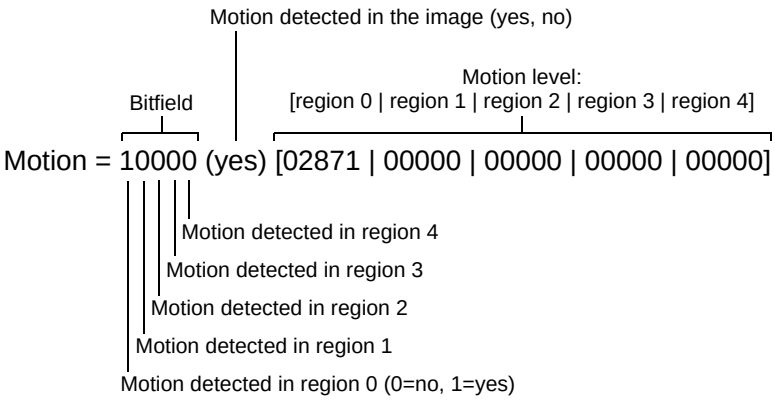
Each Key=Value pair is followed by a new line character (0x0a).

The FrameNR line indicates the value of the camera's frame counter when the image was captured.

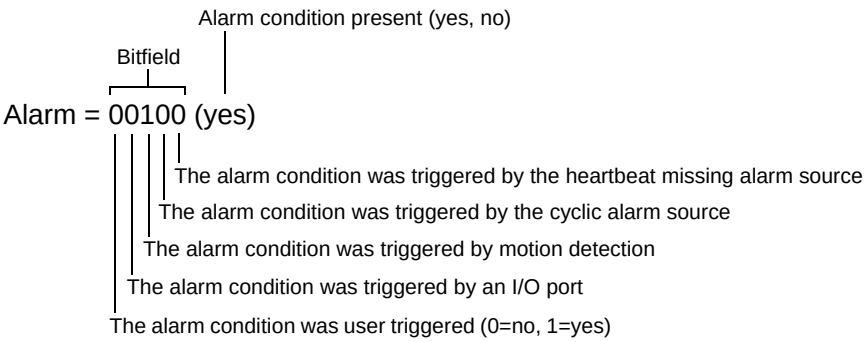
The information in the AOI line is interpreted as follows:

AOI=(AOIWidth x AOIHeight)@(AOILeft, AOITop)/(SensorWidth x SensorHeight)

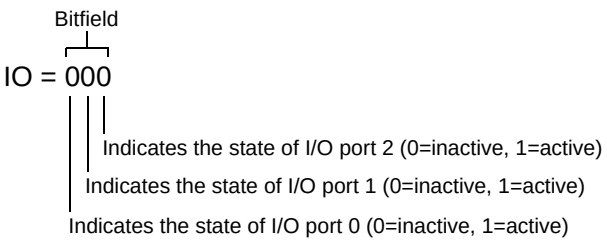
The information in the motion line is interpreted as follows:



The information in the alarm line is interpreted as follows:



The information in the IO line is interpreted as follows:



The DateTime line indicates the date and time of image capture in Yr:Mo:Day Hr:Min:Sec format.

The SubsecTime line indicates the subseconds in microseconds.

The RtTrigger line indicates whether the image was acquired as the result of a real-time trigger signal. A value of 0 indicates that the image acquisition was not the result of a real-time trigger. A value of 1 indicates that the image acquisition was the result of a real-time trigger. (For more details about the real-time trigger feature, see the camera user's manual.)

5.3 YUV-Encoded Streams

The output of a YUV-encoded stream is not a stream as such but a series of individual images that are triggered by a real-time trigger. Use the following HTTP request to receive the YUV images:

`http://<camera>/cgi-bin/stream.cgi?stream=<N>&type=yuv&mode=triggered`

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.

stream = the number of the stream you want to access.

type = the encoder type of the stream you want to access.

mode = the mode of the stream you want to access. This has to be set to "triggered" in order to receive YUV images.

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

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

5.4 RTSP/RTP over HTTP Tunneling

The Basler IP camera supports RTSP/RTP over HTTP streaming.

To request an RTSP/RTP stream over an HTTP tunnel, use the following base address:

`http://<camera>/cgi-bin/rtsptunnel.cgi`

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 base address substitutes the RTSP prefix `rtsp://<camera>` in the RTSP request. For example, the request

`rtsp://<camera>/h264?stream=0&mode=live`

tunnelled through HTTP would look like this:

`http://<camera>/cgi-bin/rtsptunnel.cgi/h264?stream=0&mode=live`

As described in Section 5.2.1 on page 96, you do not have to specify the encoder type. In this case, the request would look like this:

`http://<camera>/cgi-bin/rtsptunnel.cgi/0?mode=live`

The base address `http://<camera>/cgi-bin/rtsptunnel.cgi` can be followed by the same parameters as used for normal RTSP streaming.

See Section 5.4.1 on page 104 for an example of RTSP/RTP over HTTP tunneling.

5.4.1 RTSP/RTP over HTTP Tunneling Example

For example, assume that you are working with a camera named IPCam_1, that stream 0 is H.264 encoded, and that you want to view the live images from stream 0 via HTTP tunneling.

The following example describes how to proceed when using the VLC media player:

1. Open the VLC media player.
2. Open the **Tools** menu and select the **Preferences** menu item.
   The **Preferences** menu opens.
3. In the **Show settings** group, select **All**.
4. In the structure tree, select **Input / Codecs** -> **Demuxers** -> **RTP/RTSP**.
   The **RTP/RTSP/SDP demuxer (using Live555)** menu opens.
5. Check the **Tunnel RTSP and RTP over HTTP** check box.
6. Make sure that the **HTTP tunnel port** setting matches the camera's HTTP port setting.
7. Click **Save**.
8. In the VLC media player open the **Media** menu and select the **Open Network Stream...** menu item.
   The **Open Media** menu opens with the selected **Network** tab.
9. In the **Network Protocol** group enter the following network URL:
   `rtsp://IPCam_1/cgi-bin/rtsptunnel.cgi/h264?stream=0&mode=live`
10. Click **Play**.

6 Accessing the SD Card Data

Not all camera models are equipped with an SD card slot.

The following file systems are supported:

- FAT16 (VFAT)
- FAT32 (VFAT)

An FTP server is available on the camera and this server allows access to data stored on the SD card. Access to the SD card data is controlled by the user authentication functionality in the camera.

If user authentication is enabled:

- FTP user = any user currently existing in the camera's user authentication system
- FTP password = the actual password for the user

If user authentication is disabled:

- FTP user = "ftp" or "anonymous"
- FTP password = any

Once you are connected to the camera via an FTP client, the content of the SD card can be found in the "data" directory. The alarm data is saved in the "data/alarms" directory. The live stream data is saved in the "data/live" directory.

A corresponding example structure tree looks as follows:

```
ftp://172.16.51.105/
+-data/
  +-alarms/
  | +-YYYY-MM-DD_hh/
  | | +-YYYY-MM-DD_hh-mm-ss_SSS.<N>.<EXT>
  | | +-...
  | +-...
  +-live/
    +-YYYY-MM-DD_hh/
    | +-YYYY-MM-DD_hh-mm-ss_SSS.<N>.<EXT>
    | +-...
    +-...
```

The alarm and live stream data is written to individual files with the following name pattern:
YYYY-MM-DD_hh-mm-ss_SSS.<N>.<EXT>

YYYY-MM-DD_hh-mm-ss_SSS = the timestamp when the file was saved

Where:

- YYYY-MM-DD = the date
- hh-mm-ss_SSS = the time of day (24-hour time) with millisecond precision

Example:

A file that was saved at 2:23:24.03 PM on 2 April 2011 will be named as follows:

2011-04-02_14-23-24_030.<N>.<EXT>

The file name ends with the suffix:
<N>.<EXT>

Where:

- <N> = the number of the stream
- <EXT> = the file name extension that depends on the file type:
  - jpeg = the alarm image
  - txt = the alarm description file
  - mjpeg = motion JPEG stream
  - m4v = MPEG-4 stream
  - h264 = H.264 stream

The individual files are saved in sub-directories with the following name pattern:
YYYY-MM-DD_hh

Where:

- YYYY-MM-DD_hh = the prefix of the files contained in the directory

Example:

The file 2011-04-02_14-23-24_030.2.mjpeg will be saved to the following directory:

data/alarms/2011-04-02_14/2011-04-02_14-23-24_030.2.mjpeg

	The saved alarm buffer files on the SD card can be viewed via FTP using the VLC media player. Saved H.264 files include an SEI NAL unit with meta information before each I-frame and each P-frame similar to what you would see if you request an H.264 stream via RTSP and use the metainfo=sei parameter.

7 ONVIF Support

Basler IP cameras offer ONVIF support based on core specification version 1.02. The ONVIF (Open Network Video Interface Forum) standard specifies how IP products within video surveillance and other physical security areas can communicate with each other. It defines a network video communication framework based on relevant IETF and Web Services standards. For more information about the ONVIF standard, visit http://www.onvif.org/.

7.1 ONVIF Functions

Currently, Basler cameras support the following ONVIF functions:

- Device Discovery
- Device Management
- Media Configuration
- Real-Time Streaming
- Event Handling

7.1.1 Device Discovery

ONVIF devices support WS-Discovery, a mechanism for locating ONVIF capable devices in a network. Devices that come online can send `hello` messages to announce their presence, and clients can send `probe` messages to find other devices and services.

7.1.2 Device Management

Device management functions are handled through the device service.

The capability commands allow an NVC (Network Video Client) to ask for the services provided by an NVT (Network Video Transmitter = the camera) and to determine which ONVIF services are offered by the NVT. The capabilities are structured as the different NVT services and are further divided into subcategories (where applicable) as follows:

- Device
  - Capabilities
  - Network
  - System
  - Security
- Event
- Media

The capabilities for the different categories indicate those commands and parameter settings that are available for the particular service or service subcategory.

7.1.2.1 Network

The following set of network commands allows standardized management of functions:

- Get and set hostname.
- Get and set DNS configurations.
- Get and set NTP configurations.
- Get and set network interface configurations.
- Get and set default gateway.
- Get and set zero configuration.

7.1.2.2 System

The system commands are used to manage the following NVT system settings:

- Get device information.
- Get and set system date and time.
- Factory default reset.
- Reboot.

7.1.2.3 Security

The following security operations are used to manage the NVT security configurations:

- Get and set access security policy.
- Handle user credentials and settings.

7.1.3 Media Configuration

Media configurations are handled through the media service. Media configurations are used to determine the streaming properties of requested media streams as defined in the ONVIF specification. The NVT provides media configuration through the media service.

Media Profiles

Real-time video streaming configurations are controlled using *media profiles*.

A profile consists of a set of interconnected *configuration entities*. A configuration entity is one of the following:

- Video source configuration
- Video encoder configuration
- Audio source configuration
- Audio encoder configuration

Figure 4 shows a media profile.

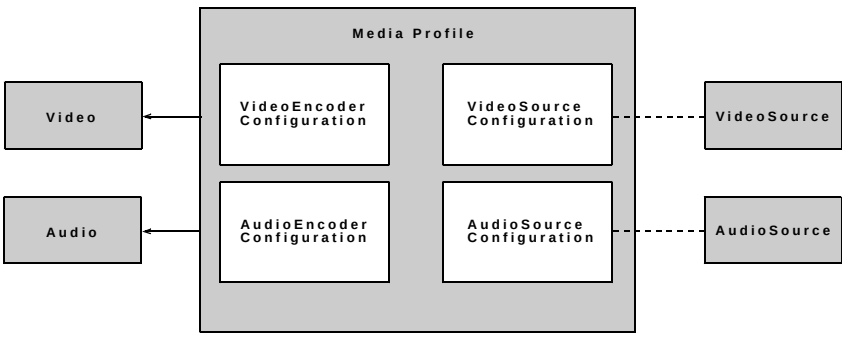


Fig. 4: Media Profile

A complete media profile defines how and what to present to the NVC in a media stream.

The following commands list existing sources:

- Get all existing video sources in the device.
- Get all existing audio sources in the device.

The following commands manage media profiles:

- Create a new media profile.
- Get all existing media profiles.
- Get a specific media profile.
- Delete a specific media profile.

The following commands manage configuration entities:

- Get the valid property values for a specific configuration entity.
- Set a configuration entity configuration.
- Get all existing configuration entities of the type.
- Get a specific configuration entity.
- Get all configuration entities compatible with a specific media profile.

The following commands initiate and manipulate a video/audio stream:

- Request a valid RTSP URI for a specific media profile and protocol.
- Insert a synchronization point (I-frame etc.) in active streams.

7.1.4 Real-Time Streaming

The ONVIF specification defines different media streaming options and formats. The following video codecs are supported on Basler cameras:

- JPEG (over RTP)
- H.264, Baseline

7.1.5 Event Handling

Event handling is based on the OASIS WS-BaseNotification and WS-Topic specifications. The following events are reported for Basler cameras:

- Temperature
- Media profile change

8 Using Images in Stream Overlays

You can customize stream overlays by uploading images of your choice, eg. company logos, and displaying them in your streams. The images you upload have to meet the following criteria:

- The file format is PNG.
- The maximum file size is 200 kByte. The maximum PNG compression level can be used.
- The image size must not exceed 4 MByte after the image has been unpacked in the camera's memory.

You can display one image per overlay, however, you can only upload one image per stream to the camera. This means that you can display the same image in both overlays of a stream, but not one image in overlay 1 and a different image in overlay 2.

You can upload different image types. The PNG format supports color or grayscale images with or without a mask as well as black and white images. Please note the following about using images in text overlays:

- If an image has a mask applied, this is always going to be used, even if the OverlayStyle is Normal or Inverted.
- Color and grayscale images are always displayed the same, regardless of the OverlayStyle. They are never displayed Inverted.
- Color images can contain a transparent layer.
- In high-resolution streams, color overlay images when combined with a large text size, can have a negative impact on the frame rate.
- Black and white images are interpreted as characters which means they can be inverted and are integrated into the background.

8.1 API Methods

There are three API methods for working with overlay images. For all of them it is necessary to specify the stream number.

Attempting to use any other method than the ones described below, will result in the following error message:

`ERROR: Unknown action '`*`method`*`'`

Upload

The "Upload" method is used for uploading an image to the camera via an HTTP post request.

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

```
http://<IPCam_1>/cgi-bin/overlay_image.cgi?upload&stream=0
```

The return from an upload request typically looks like this:

```
{success: true, message: “PNG file uploaded successfully.” }
```

Where

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

message: indicates the result of the request. If the request was not successful (`success: false`), an error message will be displayed.

Download

The "Download" method is used for downloading the image that is currently loaded in the camera.

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

```
http://<IPCam_1>/cgi-bin/overlay_image.cgi/download?stream=0
```

The return from a download request is the image that the user has uploaded. The image will be displayed in the web browser.

Reset

The "Reset" method is used for deleting the image that is currently loaded in the camera.

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

```
http://<IPCam_1>/cgi-bin/overlay_image.cgi/reset?stream=0
```

The return from a reset request always looks like this:

SUCCESS.

9 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 a Basler 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 is explained in detail in Section 9.1 on page 115.

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 9.2 on page 116.

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 9.3 on page 117.

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 IP Camera Web Client. Can change his or her own password.

Table 2: User Level Descriptions

9.1 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 9 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 9.3 on page 117.

9.2 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 than 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 this 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 9.3 on page 117 describes the authentication API in detail.

9.3 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 9.3.1 on page 117.

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

9.3.1 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, anonymous_viewer: 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 the reason for failure of the request (see Table 3 on page 130).

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

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

anonymous_viewer: = indicates whether an anonymous access to streams (via HTTP, RTSP) is possible without user authentication. 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 the reason for failure of the request (see Table 3 on page 130).

reason: = text that indicates the success of the request or 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 the reason for failure of the request (see Table 3 on page 130).

reason: = text that indicates the success of the request or 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 114).

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 the reason for failure of the request (see Table 3 on page 130).

reason: = text that indicates the success of the request or 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 114)

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 114 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=DBlac
k&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=Jk
l78&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 the reason for failure of the request (see Table 3 on page 130).

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

json_valid: = reserved for future use.

ChangeLevel

The "ChangeLevel" method is used to change the user level of 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 114 (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 the reason for failure of the request (see Table 3 on page 130).

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

json_valid: = reserved for future use.

ChangePasswd

The "ChangePasswd" method is used to change the password of 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 change 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 list user request returns the information that 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 change 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 change 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 the reason for failure of the request (see Table 3 on page 130).

reason: = text that indicates the success of the request or 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 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>

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=5be8d
6e71c2e09cea25c237767489e1a
```

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 the reason for failure of the request (see Table 3 on page 130).

reason: = text that indicates the success of the request or 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 119) 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 the reason for failure of the request (see Table 3 on page 130).

reason: = text that indicates the success of the request or 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 114).

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=5be8d6e71c2e09cea
25c237767489e1a
```

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 the reason for failure of the request (see Table 3 on page 130).

reason: = text that indicates the success of the request or 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 the reason for failure of the request (see Table 3 on page 130).

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

json_valid: = reserved for future use.

EnableAnonymousViewer

The "EnableAnonymousViewer" method is used to enable an anonymous access to streams (via HTTP, RTSP) without requiring the user to provide a valid user name and password prior to access. 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 EnableAnonymousViewer method with session-based authentication, you must be currently logged in as an administrator level user.

Below is a sample enable anonymous viewer 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 enable access to streams for anonymous viewers.

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

Below is a sample enable anonymous viewer request when session-based authentication is being used and the session ID is "5be8d6e71c2e09cea25c237767489e1a":

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

The return from an enable anonymous viewer request typically looks like this:

```
{method: 'EnableAnonymousViewer', 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 the reason for failure of the request (see Table 3 on page 130).

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

json_valid: = reserved for future use.

DisableAnonymousViewer

The "DisableAnonymousViewer" method is used to disable an anonymous access to streams (via HTTP, RTSP) without requiring the user to provide a valid user name and password prior to access. 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 DisableAnonymousViewer method with session-based authentication, you must be currently logged in as an administrator level user.

Below is a sample disable anonymous viewer 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 access to streams for anonymous viewers.

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

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

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

The return from an disable anonymous viewer request typically looks like this:

```
{method: 'DisableAnonymousViewer', 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 the reason for failure of the request (see Table 3 on page 130).

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

json_valid: = reserved for future use.

9.3.2 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

10 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 IBaslerVideoControl::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 IBaslerVideoControl::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 does not fit into the window, the center of the video is displayed. If disabled, you can also change the zoom factor and the image detail.

C Prototype:

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

Property IBaslerVideoControl::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 IBaslerVideoControl::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 = Video Control)

C Prototype:

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

Property IBaslerVideoControl::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);
```

Property IBaslerVideoControl::Version

Declaration: Property Get Version As String

Used to retrieve the file version of the Basler Video Control, e.g. 0.1.2.230.

C Prototype:

```
HRESULT get_Version(BSTR *pVal);
```

Property IBaslerVideoControl::Formats

Declaration: Property Get Formats As String

Used to retrieve a list of the supported encoder types.

C Prototype:

```
HRESULT get_Formats(BSTR *pVal);
```

Property IBaslerVideoControl::RTPProtocol

Declaration: Property Get/Put RTPProtocol As String

Used to set the network protocol for streaming via RTSP: "udp", "tcp", "auto"

C Prototype:

```
HRESULT put_RTPProtocol(BSTR newVal);
HRESULT get_RTPProtocol(BSTR *pVal);
```

Property IBaslerVideoControl::Zoom

Declaration: Property Get/Put Zoom As Double

Used to set the zoom factor for the stream display.

C Prototype:

```
HRESULT put_Zoom(DOUBLE newVal);
HRESULT get_Zoom(DOUBLE *pVal);
```

Property IBaslerVideoControl::SlideX

Declaration: Property Get/Put SlideX As Double

Used to set the X slider position for the stream display. Possible values: 0 to 1

C Prototype:

```
HRESULT put_SlideX(DOUBLE newVal);
HRESULT get_SlideX(DOUBLE *pVal);
```

Property IBaslerVideoControl::SlideY

Declaration: Property Get/Put SlideY As Double

Used to set the Y slider position for the stream display. Possible values: 0 to 1

C Prototype:

```
HRESULT put_SlideY(DOUBLE newVal);
HRESULT get_SlideY(DOUBLE *pVal);
```

11 Zero Configuration Networking Information

11.1 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.

11.2 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.

11.3 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. Basler recommends using existing libraries that have a caching infrastructure in place.

11.4 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), this is only 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. Let's 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 this 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, Basler recommends 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.

11.5 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

Appendix A V2.x to V 3.X API Migration Guide

The API for version 3.x software was designed to be as compatible as possible with cameras that use version 2.x firmware. However, if you have an existing application written against the API for cameras with version 2.x firmware and you want to extend the application to work with the API for cameras with version 3.x firmware, or if you are writing an application against the API for cameras with version 3.x firmware and you also want to use that application with cameras that have version 2.x firmware, there are some changes that you must keep in mind.

The latest version of camera firmware is available for download on the Basler website: www.baslerweb.com/firmware

	Version 3.x firmware is only compatible with cameras that have BIP2 as part of the model name (e.g., BIP2-640c). Version 2.x firmware is not compatible with cameras that have BIP2 as part of the model name.

The tables below list the changes that you may notice when using the V3.x API with cameras that have V2.x firmware. For each change, the table describes the legacy support that is built into the V3.x API.

Change:	The InputPin0 enumeration value has been deleted from the IRFilterMode parameter in the ImageControls group.
Legacy Support:	Set the IRFilterMode parameter to the "InputPin0Controlled" value (version 3.x will still accept this value). The IR-cut filter position will be controlled by a signal input into I/O port 1.

Change:	The read-only InputPin0 parameter in the IOPins group has been has been deleted.
Legacy Support:	IOPins.InputPin0 can still be used. Indicates the state of I/O port 1 regardless of whether the port is set to act as an input or as an output.

Change:	The InputPin0Mode parameter in the IOPins group has been has been deleted.
Legacy Support:	IOPins.InputPin0Mode can still be used with the following enumeration values: Nornal Inverted Write: sets the behavior of I/O port 1 to normal or to inverted. Read: indicates the current setting for I/O port 1

Change:	The OutputPin0Mode parameter in the IOPins group has been has been deleted.
Legacy Support:	IOPins.OutputPin0Mode can still be used with to write the following enumeration values: Nornal Inverted Write: sets the behavior of I/O port 0 to normal or to inverted. Read: indicates the current setting for I/O port 0.

Change:	The OutputPin0 parameter in the IOPins group has been has been deleted.
Legacy Support:	IOPins.OnputPin0 can still be used. Write: With I/O port 0 set for a direction of "Output" and a function of "User Output", use IOPins.Output0 to set the state of the port to 0 (inactive) or 1 (active). Read: Indicates the state of I/O port 0, regardless of whether the port is set to act as an input or an output.

Change:	The OutputPin0Function parameter in the IOPins group has been has been deleted.
Legacy Support:	IOPins.OutputPin0Function can still be used. With I/O port 0 set for a direction of "Output": IOPins.OutputPin0Function can be used to set the function of the port by writing one of the following enumeration values: UserOutput IRFilterAnnounce Strobe AlarmAnnounce With I/O port 0 set for a direction of "Input": IOPins.OutputPin0Function can be used to read the function of the port and will yield one of the following enumeration value: Monitor IRSwitch AlarmTrigger

Change:	The StrobeDelay_us parameter in the IOPins group has been deleted.
Legacy Support:	IOPins.StrobeDelay_us can still be used. The functionality will be the same as the current StrobeDelay parameter in the IO group.

Change:	The StrobeDuration_us parameter in the IOPins group has been has been deleted.
Legacy Support:	IOPins.StrobeDuration_us can still be used. The functionality will be the same as the current StrobeDuration parameter in the IO group.

Change:	The content of the EXIF header (see Section 5.1.3 on page 94) included with the images in MJPEG encoded streams has changed. On cameras with version 2.x firmware, each EXIF header includes an "InputPins=0" line and an "OutputPins=0" line. On cameras with version 3.x firmware, these lines have been deleted and replaced with a "IO=000" line.
Legacy Support:	The EXIF headers in MJPEG images from cameras with version 2.x firmware will continue to include an "InputPins=0" line and an "OutputPins=0" line. The EXIF headers in MJPEG images from cameras with version 3.x firmware will only include the "IO=000" line.

Change:	The content of the SEI NAL units (see Section 5.2.1 on page 96) that can be included with MPEG-4 or H.264 streams has changed. On cameras with version 2.x firmware, each SEI NAL unit includes an "InputPins=0" line and an "OutputPins=0" line. On cameras with version 3.x firmware, these lines have been deleted and replaced with a "IO=000" line.
Legacy Support:	The SEI NAL units from cameras with version 2.x firmware will continue to include an "InputPins=0" line and an "OutputPins=0" line. The SEI NAL units from cameras with version 3.x firmware will only include the "IO=000" line.

Revision History

Doc. ID Number	Date	Changes
AW00097301000	6 Dec 2010	Initial release of this document.
AW00097302000	23 Feb 2011	Added information about the new image mirror (see page 16) and real-time trigger (see page 83) features.
AW00097303000	16 Jun 2011	Added comments to the StreamSelector parameter description (see page 39) regarding the number of streams available. Added comments to the EncoderType parameter description (see page 39) regarding MPEG-4 streaming limitations on cameras with large sensors. Added the CVBR mode to the EncoderMode parameter description (see page 40). Added the new MaxBitrate parameter description (see page 41). Added the new OverlaySize and OverlayStyle parameter descriptions (see page 46). Updated the file name format shown in Section 6 on page 105. Added all parameter descriptions associated with the SD card functionality on the dome camera models.
AW00097304000	27 Mar 2012	Added the IsEnabled and IsReadonly methods to the data type sections (see Section 2 on page 8). Updated the section on image masks on page 11 regarding cameras with a sensor aspect ratio of 16:9. Updated the list of parameters that cannot be changed in the normal operation mode (see Table 1 on page 12). Added the new focus group description with the new parameters OnePush, Position, and Move (see page 30). Added the new ChainMode parameter description (see page 40). Added valid values to the OutputSize parameter description (see page 43). Added the new LiveRecording parameter description (see page 47). Added the new SNMP group description with the new Enable parameter (see page 61). Added the Cyclic and the HeartbeatMissing values to the SourceSelector parameter description (see page 63). Added the new LiveRecording value to the ActionSelector parameter description (see page 64). Added the new CyclicPeriod parameter description (see page 64). Added the new LiveRecordingMode parameter description (see page 67). Updated the HTTPURL parameter description (see page 68). Added the new HTTPURLAlarmEnd parameter description (see page 69). Updated sample email / text file that will be sent when an alarm is declared (see page 71). (continued on the next page)

Doc. ID Number	Date	Changes
		(continued from the previous page) Added the new LiveMemoryAllocation parameter description (see page 74). Added the new OverwriteLiveRecordings parameter description (see page 74). Added the new Mode parameter description for serial port forwarding (see page 75). Added valid values to the NTP parameter description (see page 79). Updated Section 6 on page 105 and added a live stream data example. Added the new authentication methods EnableAnonymousViewer and DisableAnonymousViewer (see page 128 and page 129). Updated Section 10 on page 131 with new properties.
AW00097305000	31 Jul 2012	Added the new RegulationDelay_ds parameter description (see page 25). Added the new SetSynchronizationPoint parameter description (see page 40). Added the new parameters LiveRecordingStorageLocation and LiveRecordingVideoFileLength (see page 47). Added the new AudioEncoderType parameter description (see page 50). Added the descriptions of the new parameters ZeroConfiguration, HostNameSource, NameServerSource, and SearchDomain (see page 57). Updated the NameServer parameter description (see page 58). Added the ONVIF group description with the new Enable parameter (see page 62). Added the description of the LiveRecordingPreAlarmTimePeriod and LiveRecordingPostAlarmTimePeriod parameters (see page 66). Added the LiveRecording group description with the new parameters FTPServer, FTPPort, FTPRemoteDir, FTPUserName, and FTPPassword (see page 72). Updated the SetDateTime parameter description (see page 77). Added the description of the new NTPServerSource parameter (see page 80). Updated the NTPServer parameter description (see page 80). Added the Audio group description with the new parameters MicPowerEnable, InputLevelSelect, and InputGain (see page 86). Updated Section 5 on page 89 with information on how to access audio streams. Added the description about accessing MJPEG encoded streams via RTSP/RTP to Section 5.1.2 on page 91. Added the description about accessing RTSP streams without specifying the encoder type in the request URL (see page 96 and page 97). Added the "follow" parameter description in chapter 5.2.1 on page 97. Added the new Section 5.4 RTSP/RTP over HTTP tunneling on page 103.

Doc. ID Number	Date	Changes
AW00097306000	13 Dec 2012	Added new values for the FrameRateMode parameter (see page 17). Added the descriptions of the new BlacklevelStrength and BlacklevelAutomatic parameters (see page 28). Updated the descriptions and values of the OverlayText, OverlayPosition, OverlaySize, and OverlayStyle parameters (see page 45 to page 46). Added the descriptions of the OverlayText2, OverlayPosition2, and OverlaySize2 parameters (see page 45 to page 46). Added the new NumStreams parameter description (see page 62). Added the new ExtClockSynchronization parameter description (see page 79). Added the new NTPSyncPeriod parameter description (see page 80). Added the new ExternalTrigger value for the IO.Function parameter (see page 83). Added Section 7 on page 107 about ONVIF. Added Section 8 on page 111 about using overlay images.
AW00097307000	3 Apr 2014	Updated the Support contact information in the Contacting Basler Support Worldwide section (see page 2). Added the description of the new SetMaxAOI parameter in the Sensor group (see page 16). Added the new YUV_TRIGGERED value for the EncoderType parameter (see page 39). Added the CAPPED_VBR value to the EncoderMode parameter (see page 40). Updated the MaxBitrate parameter (see page 41) to reflect its use with the CAPPED_VBR encoder mode. Updated the Quality parameter (see page 41) to reflect its use with the CAPPED_VBR encoder mode. Added the description of the new SetMaxAOI parameter in the Stream group (see page 43). Added missing values to the OutputSize parameter (see page 43). Added the new PCM_Raw value for AudioEncoderType parameter (see page 50). Added the description of the new MulticastStream parameter (see page 55). Added the description of the new MulticastMetainfo parameter (see page 55). Added Section 5.3 on page 103 about YUV-encoded streams. Added ONVIF audio profile information in Section 7 on page 107.

Doc. ID Number	Date	Changes
AW00097308000	28 Oct 2015	Minor updates throughout the manual. Updated the Support contact information in the Contacting Basler Support Worldwide section (see page 2). Added the description of the new SaturationIRFilterOpen parameter (see page 25). Added the description of the new Shadows parameter (see page 26). Updated the description of the IRFilterMode parameter (see page 27). Added the description of the new ExcludeRttImages parameter (see page 49). Added information about supported file systems (see page 73). Added the description of the new StrobeMode parameter (see page 85). Updated the description of the single value of the mode parameter (see page 90). Added information about supported file systems (see page 105). Updated information about using images in stream overlays (see page 111).
AW00097309000	12 Nov 2015	Removed section.
AW00097310000	13 Jan 2016	Added Section 4.5 on page 31 about the ImageControlsRTT parameter group.
AW00097311000	14 Sep 2016	Removed manual line breaks in code samples throughout the manual.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Z