



RVMedia

© 2025 TRichView.com

Table of Contents

Part I Overview 12

1..About IP cameras and USB cameras	12
2..Features	13
3..Comparison	15
4..How it works: cameras	16
5..How it works: video chat without a server	18
6..How it works: video chat with a server	20
7..Modes of connections	21
8..Third-party libraries	24
9..Additional information	25
10..Version history	26

Part II Components 38

1..Video	39
TRVCamControl	39
Properties	40
TRVCamControl's colors	41
TRVCamControl.ShowFocus	42
TRVCamera	42
Properties	43
TRVCamera.Agent	44
TRVCamera.Bitrates	45
TRVCamera.Brightness, Contrast, Hue, Saturation, Sharpness	45
TRVCamera.CameraControl	46
TRVCamera.CameraHost, CameraPort	46
TRVCamera.CameraSearchTimeOut	46
TRVCamera.CommandMode	47
TRVCamera.CycleVideoImages, VideoImagesURLs	47
TRVCamera.DesktopMode, DesktopRect,	47
TRVCamera.DeviceType	49
TRVCamera.FFMpegProperty	50
TRVCamera.FileName, ToFile	50
TRVCamera.FlipHorizontally, FlipVertically	51
TRVCamera.FocusType, FocusDistance	51
TRVCamera.GStreamerProperty	52
TRVCamera.IPCameraTypes	52
TRVCamera.JpegIntegrity	52
TRVCamera.Language	52
TRVCamera.Latency	53
TRVCamera.LoginPrompt	53
TRVCamera.MaxCameraSearchThreadCount	53
TRVCamera.Parameters	54
TRVCamera.ProxyProperty	55
TRVCamera.Quality	55
TRVCamera.Rotation	55
TRVCamera.Searching	55
TRVCamera.SmoothImage	56

TRVCamera.SourceFileName.....	56
TRVCamera.URL	57
TRVCamera.UserAccess	57
TRVCamera.UserName, UserPassword	57
TRVCamera.Users	58
TRVCamera.VideoDeviceIndex, VideoDeviceCount, VideoDeviceList, VideoDeviceIdList.....	58
TRVCamera.VideoFormat.....	59
TRVCamera.VideoMode	60
TRVCamera.VideoResolution	61
Methods	61
TRVCamera.AddUser, ModifyUser.....	62
TRVCamera.FillVideoDeviceList.....	63
TRVCamera.Get- SetCamVideoMode, etc.....	63
TRVCamera.Get- SetDesktopVideoMode, etc.....	63
TRVCamera.GetAvailableCamProperties, GetAvailableCamMethods, GetAccessibleCamProperties, GetAccessibleCamMethods	64
TRVCamera.GetColorControlPropertyRange.....	65
TRVCamera.GetSnapshot.....	65
TRVCamera.IsSupportedFFMPEG.....	65
TRVCamera.IsSupportedGStreamer	66
TRVCamera.LockCommands, UnlockCommands.....	66
TRVCamera.Move***	66
TRVCamera.PlayVideoStream, PlayVideoFile.....	67
TRVCamera.ResetImageSetting.....	68
TRVCamera.SearchCamera.....	68
TRVCamera.SwitchLEDOOn, SwitchLEDOff.....	69
TRVCamera.WaitForVideoStream, WaitForVideoFile, WaitForVideo, WaitForSearch.....	69
Events	69
TRVCamera.OnDownload, OnDownloaded.....	70
TRVCamera.OnEndVideoFile, OnEndVideoStream.....	70
TRVCamera.OnError	70
TRVCamera.OnGetImage	71
TRVCamera.OnGetVideoStreamIndex.....	71
TRVCamera.OnLogin	71
TRVCamera.OnLoginFailed.....	72
TRVCamera.OnMoved	72
TRVCamera.OnNewImage.....	72
TRVCamera.OnPrepared	72
TRVCamera.OnProgress	73
TRVCamera.OnSearchComplete.....	73
TRVCamera.OnSetProperty.....	73
TRVCamera.OnStartVideoFile, OnStartVideoStream.....	73
TRVCamera.OnVideoStart.....	74
TRVCamMultiView	74
Properties	75
TRVCamMultiView.AllowFullScreen.....	76
TRVCamMultiView.AudioSource.....	76
TRVCamMultiView.CameraControl.....	76
TRVCamMultiView.CamMoveMode.....	77
TRVCamMultiView.CaptionColor, CaptionFont, CaptionHeight.....	77
TRVCamMultiView.Color, ViewerColor	78
TRVCamMultiView.CurRenderMode, RenderMode.....	78
TRVCamMultiView.FocusLineColor.....	78
TRVCamMultiView.Font, ParentFont.....	79
TRVCamMultiView.FullScreen.....	79
TRVCamMultiView.FullScreenMultiView.....	79
TRVCamMultiView.HoverLineColor.....	80
TRVCamMultiView.IconStyle.....	80

TRVCamMultiView.Language.....	80
TRVCamMultiView.RefreshRate.....	80
TRVCamMultiView.RememberLastFrame.....	81
TRVCamMultiView.ScaleViewers.....	81
TRVCamMultiView.SearchPanelColor, SearchPanelTextColor.....	81
TRVCamMultiView.ViewerIndex.....	82
TRVCamMultiView.Viewers.....	82
TRVCamMultiView.WaitAnimationDelay.....	84
Events.....	84
TRVCamMultiView.OnFullScreen.....	85
TRVCamMultiView.OnSelectViewer.....	85
TRVCamMultiView.OnViewerPaint.....	85
TRVCamRecorder.....	85
Properties.....	86
TRVCamRecorder.Active.....	87
TRVCamRecorder.Audio*.....	87
TRVCamRecorder.AudioCodec, VideoCodec.....	88
TRVCamRecorder.AudioCodecName, VideoCodecName.....	88
TRVCamRecorder.AudioSource, UseAudio.....	89
TRVCamRecorder.OutputFileName.....	89
TRVCamRecorder.Paused.....	89
TRVCamRecorder.SourceAudioGUID, SourceAudioIndex.....	90
TRVCamRecorder.SourceVideoGUID, SourceVideoIndex.....	90
TRVCamRecorder.Video*.....	91
TRVCamRecorder.VideoEncodingParameters.....	91
TRVCamRecorder.VideoSource, UseVideo.....	92
Events.....	92
TRVCamRecorder.OnError.....	92
TRVCamRecorder.OnFirstFrame.....	92
TRVCamRecorder.OnGetImage.....	93
TRVCamView.....	93
Properties.....	94
TRVCamView.AllowFullScreen.....	95
TRVCamView.AutoSize.....	95
TRVCamView.CamMoveMode.....	95
TRVCamView.Color.....	96
TRVCamView.CurRenderMode, RenderMode.....	96
TRVCamView.FocusLineColor.....	96
TRVCamView.Font, ParentFont.....	97
TRVCamView.FrameScaleQuality.....	97
TRVCamView.FullScreen.....	97
TRVCamView.FullScreenView.....	98
TRVCamView.GUIDFrom, IndexFrom.....	98
TRVCamView.HoverLineColor.....	98
TRVCamView.IconStyle.....	99
TRVCamView.Language.....	99
TRVCamView.RememberLastFrame.....	99
TRVCamView.SearchPanelColor, SearchPanelTextColor.....	99
TRVCamView.ShowCameraSearch.....	100
TRVCamView.ShowCaption, CaptionParts, Title, CaptionColor, CaptionFont, CaptionHeight.....	101
TRVCamView.UseOptimalVideoResolution.....	102
TRVCamView.VideoSource, Camera, Receiver.....	103
TRVCamView.ViewMode.....	103
TRVCamView.WaitAnimationDelay.....	105
Events.....	106
TRVCamView.OnBeginMove, OnEndMove.....	107
TRVCamView.OnFullScreen.....	107
TRVCamView.OnMouseEnter, OnMouseLeave.....	107

TRVCamView.OnPaint	107
TRVWebCamDialog	107
Properties	109
TRVWebCamDialog.Camera	109
TRVWebCamDialog.Language	110
Methods	110
TRVWebCamDialog.Execute	110
2. Audio	110
TRVAudioPlayer	110
Properties	112
TRVAudioPlayer.Encode*	112
TRVAudioPlayer.OutputFileName	113
TRVAudioPlayer.Recording	113
TRVAudioPlayer.UseFFmpeg	114
Events	114
TRVAudioPlayer.OnStopRecording	114
TRVAudioPlayer.OnError	114
TRVCamSound	115
Properties	116
TRVCamSound.Camera	116
TRVCamSound.GUID	116
Events	116
TRVCamSound.OnGetAudioStreamIndex	116
TRVMicrophone	117
TRVMicrophoneView	117
Properties	118
TRVMicrophoneView.Orientation	118
TRVMicrophoneView.Style	119
Events	119
TRVMicrophoneView.OnPaint	119
3. Network	120
TRVCamReceiver	120
Properties	121
TRVCamReceiver.Active	122
TRVCamReceiver.AudioLatency, VideoLatency	122
TRVCamReceiver.BufferDuration	122
TRVCamReceiver.BufferSize	122
TRVCamReceiver.Color	122
TRVCamReceiver.ConnectionProperties	123
TRVCamReceiver.FilterSystemCmd	123
TRVCamReceiver.GUIDMy	123
TRVCamReceiver.IgnoreCorruptedFrames	123
TRVCamReceiver.JpegIntegrity	124
TRVCamReceiver.Mute	124
TRVCamReceiver.Port	124
TRVCamReceiver.Protocol	124
TRVCamReceiver.ProxyProperty	125
TRVCamReceiver.ReceiveMediaTypes	125
TRVCamReceiver.RetryCount	125
TRVCamReceiver.Senders	125
TRVCamReceiver.SessionKey, SessionKey2	126
TRVCamReceiver.SmoothImage	126
TRVCamReceiver.State	127
TRVCamReceiver.SynchronizedReceiveUserData	127
TRVCamReceiver.TCPConnectionType	128
TRVCamReceiver.UseTempFiles	128
TRVCamReceiver.Volume	128

Methods	128
TRVCamReceiver.GetOpenChannelCount, GetMaxChannelCount.....	129
Events	129
TRVCamReceiver.OnConnected, OnConnecting, OnDisconnect, OnConnectError.....	130
TRVCamReceiver.OnDecodeAudio.....	130
TRVCamReceiver.OnDecodeVideo.....	131
TRVCamReceiver.OnGetGroupInfo.....	131
TRVCamReceiver.OnRequestJoinGroup.....	132
TRVCamReceiver.OnGetAllGroups	133
TRVCamReceiver.OnGetAllUsers, OnGetAllOnlineUsers	133
TRVCamReceiver.OnGetGroupUsers	134
TRVCamReceiver.OnGetImage.....	134
TRVCamReceiver.OnOpenChannel, OnCloseChannel.....	135
TRVCamReceiver.OnReceiveCmdData	135
TRVCamReceiver.OnReceiveFileData	136
TRVCamReceiver.OnReceiveUserData.....	137
TRVCamReceiver.OnSessionConnected, OnSessionDisconnected.....	137
TRVCamReceiver.OnUserEnter, OnUserExit.....	138
TRVCamReceiver.OnUserJoinsGroup, OnUserLeavesGroup.....	138
TRVCamReceiver.OnMediaAccessRequest, OnMediaAccessCancelRequest.....	139
TRVCamSender	139
Properties	142
TRVCamSender.Active	143
TRVCamSender.AudioSource.....	143
TRVCamSender.BufferSize.....	144
TRVCamSender.ChangedAreaProcessingMode.....	144
TRVCamSender.CompressionOptions	144
TRVCamSender.CompressionQuality.....	145
TRVCamSender.ConnectionProperties	145
TRVCamSender.Encoding.....	145
TRVCamSender.ExtraMediaSources.....	146
TRVCamSender.FilterBlur.....	146
TRVCamSender.FrameDifferenceInterval.....	146
TRVCamSender.FullFrameInterval	147
TRVCamSender.GUIIDFrom, UseGUID.....	147
TRVCamSender.GUIIDTo, GUIDGroup.....	147
TRVCamSender.MinChangeAreaSize, PixelColorThreshold.....	148
TRVCamSender.Protocol	149
TRVCamSender.ProxyProperty.....	149
TRVCamSender.ReceiverHost, ReceiverPort.....	150
TRVCamSender.SenderPort.....	150
TRVCamSender.SendMediaTypes	150
TRVCamSender.SendOptions.....	151
TRVCamSender.SessionKey.....	151
TRVCamSender.ShowCmd.....	151
TRVCamSender.SourceAudioIndex.....	151
TRVCamSender.SourceVideoIndex.....	152
TRVCamSender.TCPConnectionType.....	153
TRVCamSender.TestMode.....	154
TRVCamSender.VideoResolution.....	154
TRVCamSender.VideoSendType.....	154
TRVCamSender.VideoSource, SourceGUID.....	155
Methods	155
TRVCamSender.AddAllowedSender and others	156
TRVCamSender.AddDefaultReceiver and others.....	157
TRVCamSender.AllowMediaAccess, CancelMediaAccess	158
TRVCamSender.GetAllUsers, GetAllOnlineUsers.....	158
TRVCamSender.JoinGroup, LeaveGroup, etc.....	158

TRVCamSender.NeedSendFullFrame.....	160
TRVCamSender.Reconnect.....	160
TRVCamSender.RestartServer.....	160
TRVCamSender.SendCmd and others.....	160
TRVCamSender.SendFile, SendUseData.....	161
TRVCamSender.SendMediaAccessRequest, SendMediaAccessCancelRequest.....	162
Events	163
TRVCamSender.OnConnected, OnConnecting, OnDisconnect, OnConnectError.....	163
TRVCamSender.OnEncodeAudio.....	164
TRVCamSender.OnEncodeVideo.....	164
TRVCamSender.OnSendCmd, OnSentCmd.....	165
TRVMediaServer	165
Properties	167
TRVMediaServer.BufferOptions, TempFolder	167
TRVMediaServer.CmdOptions.....	168
TRVMediaServer.FilterUserCmd	168
TRVMediaServer.GUIDMy	169
TRVMediaServer.HTTPActive, HTTPPort.....	169
TRVMediaServer.KeepClientInfoMode.....	169
TRVMediaServer.MaxGroupCount.....	170
TRVMediaServer.SenderConnectionProperties, ReceiverConnectionProperties	170
TRVMediaServer.SessionKey.....	170
TRVMediaServer.UDPActive, UDPPort.....	170
Methods	171
TRVMediaServer.SendCmdToGroup.....	171
TRVMediaServer.SendCmdToUser.....	171
TRVMediaServer.SendCommandToGUID.....	172
Events	172
TRVMediaServer.OnDataRead	172
TRVMediaServer.OnPacketProcessing, OnConnectionCountChanged.....	173
TRVMediaServer.OnServerCmd.....	173
TRVMediaServer.OnStart, OnStop, OnError.....	174
TRVMediaServer.OnUserConnect, OnUserDisconnect.....	174
TRVTrafficMeter	175
Properties	176
TRVTrafficMeter.Camera.....	176
TRVTrafficMeter.Language.....	176
TRVTrafficMeter.Receiver	176
TRVTrafficMeter.Sender	176
4. Ancestor classes	176
TCustomRVMicrophone	177
Properties	178
TCustomRVMicrophone.AudioInputDeviceIndex, AudioInputDeviceCount, AudioInputDeviceList.....	179
TCustomRVMicrophone.BitsPerSample, SamplesPerSec.....	179
TCustomRVMicrophone.BufferDuration.....	179
TCustomRVMicrophone.Mute.....	180
TCustomRVMicrophone.NoiseReduction, NoiseReductionLevel, UseRNNoise.....	180
TCustomRVMicrophone.Pitch.....	181
TCustomRVMicrophone.SoundIgnoreInterval.....	181
TCustomRVMicrophone.SoundMinLevel.....	181
TCustomRVMicrophone.SourceType.....	182
TCustomRVMicrophone.VolumeMultiplier.....	182
TCustomRVMicrophone.WAVFileName.....	183
TCustomRVMicrophone.WAVUseOptions	183
Events	183
TCustomRVMicrophone.OnOpenWavFile, OnReadWavFile, OnCloseWavFile.....	183
TCustomRVReceiver	184

Properties	184
TCustomRVReceiver.AudioOutput.....	185
Events	185
TCustomRVReceiver.OnDataRead	185
TCustomRVSender	185
TRVAudioSource	185
Properties	186
TRVAudioSource.Active	186
TRVAudioSource.Volume.....	186
Events	187
TRVAudioSource.OnGetAudio	187
TRVAudioSourceWithOutput	187
Properties	187
TRVAudioSourceWithOutput.AudioOutput.....	188
TRVAudioViewer	188
Properties	188
TRVAudioViewer.AudioSource.....	189
TRVAudioViewer.ReceiverSource, GUIDFrom.....	189
TRVMediaSource	189
TRVVideoSource	190
Properties	190
TRVVideoSource.Aborting.....	190
TRVVideoSource.FramePerSec	190
Methods	191
TRVVideoSource.Abort	191
TRVVideoSource.GetOptimalVideoResolution.....	191
TCustomRVAudioOutput	191
Properties	192
TCustomRVAudioOutput.Active.....	192
TCustomRVAudioOutput.Volume.....	192
Events	192
TCustomRVAudioOutput.OnGetAudio	192
TCustomRVAudioPlayer	192
Properties	193
TCustomRVAudioPlayer.AudioInputDeviceIndex, AudioInputDeviceCount, AudioInputDeviceList.....	194
TCustomRVAudioPlayer.BufferDuration.....	194
TCustomRVAudioPlayer.Mute.....	194
TCustomRVAudioPlayer.NoiseReduction.....	194
TCustomRVAudioPlayer.Pitch.....	195
TCustomRVAudioPlayer.VolumeMultiplier.....	195

Part III Classes 195

1..TRVBufferOptions	196
2..TRVCmd	197
3..TRVCmdParamCollection	198
4..TRVCmdParamItem	198
5..TRVCompressionOptions	199
6..TRVConnectionProperties	200
7..TRVFFMpegProperty	201
8..TRVFFMpegRemuxProperty	203
9..TRVGStreamerProperty	204
10..TRVImageWrapper	206
11..TRVMBitmap	206

12	TRVMediaSourceCollection	207
13	TRVMediaSourceItem	208
14	TRVMotionDetector	209
	Properties	209
	TRVMotionDetector.ChangedAreaProcessingMode	209
	TRVMotionDetector.TestMode	210
	TRVMotionDetector.MinChangeAreaSize, PixelColorThreshold	210
	Methods	210
	TRVMotionDetector.DetectChanges	211
	TRVMotionDetector.GetRect, IsRectValid	211
15	TRVProxyProperty	211
16	TRVSenderCollection	212
17	TRVSenderCollectionEx	213
18	TRVSenderItem	213
19	TRVSenderItemEx	214
20	TRVSendOptions	214
Part IV Global variables and constants		215
1.	*_DATA	215
2.	glRVInetMaxConnect	216
Part V Global procedures		216
1.	MRVCore Unit	217
	RVMGetWindowRect	217
2.	MRVDesktop Unit	218
	GetVisibleWindowsHandles	218
	GetWindowTitleByHandle	218
3.	MRVFFmpeg Unit	219
	LoadFFmpegLibraries	219
	IsSupportedFFmpeg	219
4.	MRVFFMpegLists Unit	220
	GetListOfAvailable-Audio/Video-Decoders/Encoders	220
	GetListOfAvailableFFmpegAudioCodecs	221
	GetListOfAvailableFFmpegFileFormats	222
	GetListOfAvailableFFmpegVideoCodecs	222
	GetListOfAvailableSampleFormats	223
	GetListOfAvailableSampleRates	223
	GetListOfVideoInputFormats	223
5.	MRVFormatInfo Unit	224
	GetAudioCodecFileExt, GetVideoCodecFileExts	224
	GetAudioCodecName, GetVideoCodecName	225
	GetSampleFormatName	225
6.	MRVGStreamer Unit	225
	LoadGStreamerLibraries	225
	IsSupportedGStreamer	226
7.	MRVRNNoise Unit	227
	LoadRNNoise	227
	IsRNNoiseLoaded	227
8.	MRVWebCamFuncs Unit	228
	DescribeVideoMode	228

DescribeVideoModePixelFormat	228
------------------------------------	-----

Part VI Examples 228

1..Example 1: playing a camera video (wait mode)	228
2..Example 2: playing a camera video (no wait mode)	229
3..Example 3: playing a video stream	230

Part VII Types 230

1..Events	231
TRVAudioEvent	231
TRVCamDoneEvent	232
TRVCamErrorEvent	232
TRVCmdEvent	233
TRVDataReadEvent	234
TRVFullScreenEvent	234
TRVGetMediaStreamIndexEvent	234
TRVImageEvent	235
TRVSocketEvent	235
2..TRVAudioCodec	236
3..TRVBitsPerSample	237
4..TRVBoundsTestMode	237
5..TRVCameraType, TRVCameraTypes	238
6..TRVCamMoveMode	238
7..TRVCamVideoMode, TRVColorModel	239
8..TRVChangedAreaProcessingMode	240
9..TRVColorControlProperty	240
10..TRVCompressionType	241
11..TRVDesktopVideoMode	241
12..TRVEncodingType	242
13..TRVFFMpegFilter	243
14..TRVJpegIntegrity	243
15..TRVMAnsiString	244
16..TRVMColor	244
17..TRVMediaType	244
18..TRVMIconStyle	245
19..TRVMLanguage	245
20..TRVMRect, TRVMPoint, TRVUnitSize	246
21..TRVMRenderMode	246
22..TRVMWindowHandle	247
23..TRVParamType	248
24..TRVProtocol	248
25..TRVProtocolEx, TRVProtocolsEx	249
26..TRVRTSPFlag, TRVRTSPFlags	249
27..TRVSampleFormat	249
28..TRVSamplesPerSec	250

29..TRVSessionKey 250

30..TRVSocket 251

31..TRVTCPConnectionType 251

32..TRVVideoCodec 251

33..TRVVideoResolution 252

Index 254

1 Overview

RVMedia is a set components for working with local webcams and IP-cameras, for transferring video via the IP network, for organizing video chats, for recording audio and video files.

Supported frameworks:

- Delphi and C++Builder VCL
- Delphi and C++Builder FireMonkey (for Windows, Linux, and macOS)
- Lazarus (for Windows and Linux)

Requirements

For VCL, minimum required versions are: Delphi 7, C++Builder 2009.

For FireMonkey Windows platform, minimum required versions are: Delphi and C++Builder XE6.

For FireMonkey Linux platform, minimum required versions are: Delphi 10.3. FMXLinux is required.

For FireMonkey macOS 64-bit (Intel) platform, minimum required versions are: Delphi 10.3.

For FireMonkey macOS 64-bit (ARM) platform, minimum required versions are: Delphi 11.

Overview

- About IP cameras and USB cameras ⁽¹²⁾
- Features of RVMedia components ⁽¹³⁾
- Comparison ⁽¹⁵⁾
- How it works: cameras ⁽¹⁶⁾
- How it works: video chat without a server ⁽¹⁸⁾
- How it works: video chat with a server ⁽²⁰⁾
- Third-party libraries used by RVMedia ⁽²⁴⁾
- Additional information ⁽²⁵⁾
- Version history ⁽²⁶⁾

See also

- Components ⁽³⁸⁾

1.1 About IP cameras and USB cameras

USB webcams vs IP webcams

A **USB camera** is connected to a computer using a USB cable. Not the camera itself, but the computer can broadcast a video from this camera to other computers. A USB webcam cannot work if not connected to a computer. Additional functions depend on the software developed by the manufacturer.

An Internet protocol camera, or **IP camera**, can send and receive data via a computer network and the Internet. An IP camera has a microprocessor and a network interface (Ethernet and/or WiFi), so it is ready to be connected to a network. IP cameras include a web server, they can be connected directly to LAN/WAN/Internet, they often include motion detectors, can send e-mails, can use external sensors, etc.

Why using IP cameras may be difficult

- If a software created by the original camera manufacturer is used, the camera features are limited with the functions of this software (such as switching to the camera when a motion is detected, face recognition, focusing on the most important part of the view, etc.).
- Applications developed by different manufacturers are not compatible. IP cameras by different manufactures have different programming interface.
- It's hard to use different cameras in a network, because a detection and an administration of each camera require different software.
- Software developers waste their time implementing a support of different cameras instead of focusing on the main functions (video processing, pattern recognition, usability). As a result, applications become more expensive.

There are standard interfaces for capturing video from USB webcams. However, if you need to implement a kind of a video conference software, you have to write a code for sending and receiving video streams through a network.

All these problems are solved by our components.

1.2 Features



Features of RVCamera component

This version of RVCamera ⁴² can configure cameras by the following manufacturers:

- Foscam
- Axis
- Panasonic
- D-Link

Planned: Samsung, Planet, Arecont Vision, Trendnet, Smartec, ACTi, Beward, Apexis, Genius, AVIOSYS, Vivitek.

RVCamera recognizes a camera model at the specified address (CameraHost:CameraPort). After that, the camera properties and available commands can be read from the Parameters property (both a list of commands available for the camera, and a list of commands accessible for the specified user (defined in UserName:UserPassword) can be read).

Also, RVCamera can receive an MJPEG video stream from the specified address (URL property).

Additionally, RVCamera can work with USB webcams. It use the same programming interface to access to USB webcams and IP cameras.

The chart of supported camera models

Camera model	Camera detection	Video capture	Control		
			Movement	Video settings	Administration
Axis	●	●	●	●	●
Planet	●	●			
D-Link	●	●		●	
Panasonic	●	●	●	●	●
ArcVision	●	●			
TRENDnet	●	●			
Smartec	●	●			
ACTi	●	●			
Beward	●	●			
Foscam	●	●	●	●	●
Genius	●	●			
AVIOSYS	●	●			
VIVOTEC	●	●			
Samsung	●	●			
Mobotix	●	●	●	●	

Note 1: many cameras of other manufactures are clones of the cameras listed in this list, or they use the same protocol. They are also supported by RVMedia.

Note 2: if cameras of other developers support protocols Axis, D-Link, Panasonic, or Foscam cameras, they can be controlled by RVMedia like these cameras. For example, RVMedia can control movement of Samsung cameras supporting the Axis protocol.



Video conferences

Using our components, you can implement a video conferencing software. USB webcam, IP camera, or a video stream from the Internet can be used as a video source. Then you can broadcast this video using the RVCamSender⁽¹³⁹⁾ component, and receive it (from a single sender or multiple senders) using the RVCamReceiver⁽¹²⁰⁾ component.

1.3 Comparison

Disclaimer: we believe that the information in this topic is correct (at the moment of creation of this topic). If you find errors in this topic, please inform us.

The most widely used libraries

DSPack, Delphi DirectX wrapper. It is useful for processing video received from USB webcam (in future, we may implement video displaying using DSPack).

TVideoGrabber, an analog of DSPack. This is not only a DirectX wrapper, it has a more convenient interface to configure a video and a sound. It's rather expensive. TVideoGrabber was not designed to work with IP cameras, it cannot detect camera models or work with specific camera functions. It is targeted to play and to process audio/video data.

VisioForge Video Capture contains no functions for IP cameras.

AVICAP32.DLL, the standard library for webcams. It does not support IP cameras.

VideoLab, **iConf ActiveX Video Conferencing**, **SMInternet Component Suite** do not provide functions for IP cameras.

All the libraries above can only play/process a video from the specified IP address. No programming library known to us contains functions specific to IP cameras: they cannot detect a camera model, cannot configure it, cannot read its properties, cannot control its motion.

Comparison chart

Feature \ Library	DSPack	TVideo Grabber	AVICAP 32	iConf	SMInternet	VideoLab	VisioForge	RVMedia
IP camera detection								●
IP camera administration								●
IP camera movement control								●
optimization of video settings when receiving video from multiple cameras	○	○				○		●
IP camera video settings								●

receiving videos using "exotic" streaming options (such reading frames from a range of files)								●
multiple camera view	◉	◉		●		◉	●	●
reading IP camera information (manufacturer, properties)								●
single interface for different IP camera models								●
USB webcams	●	●	● (obsolete technology)	●	●	●	●	●
network video broadcast				●	●			●
price (approximate)	free	\$695 - \$2950	included in Windows	\$60	\$42	\$1350 - \$1500	\$220-\$800	\$300-\$800

Legend:

- supported
- ◉ supported partially (requires an additional code)

1.4 How it works: cameras

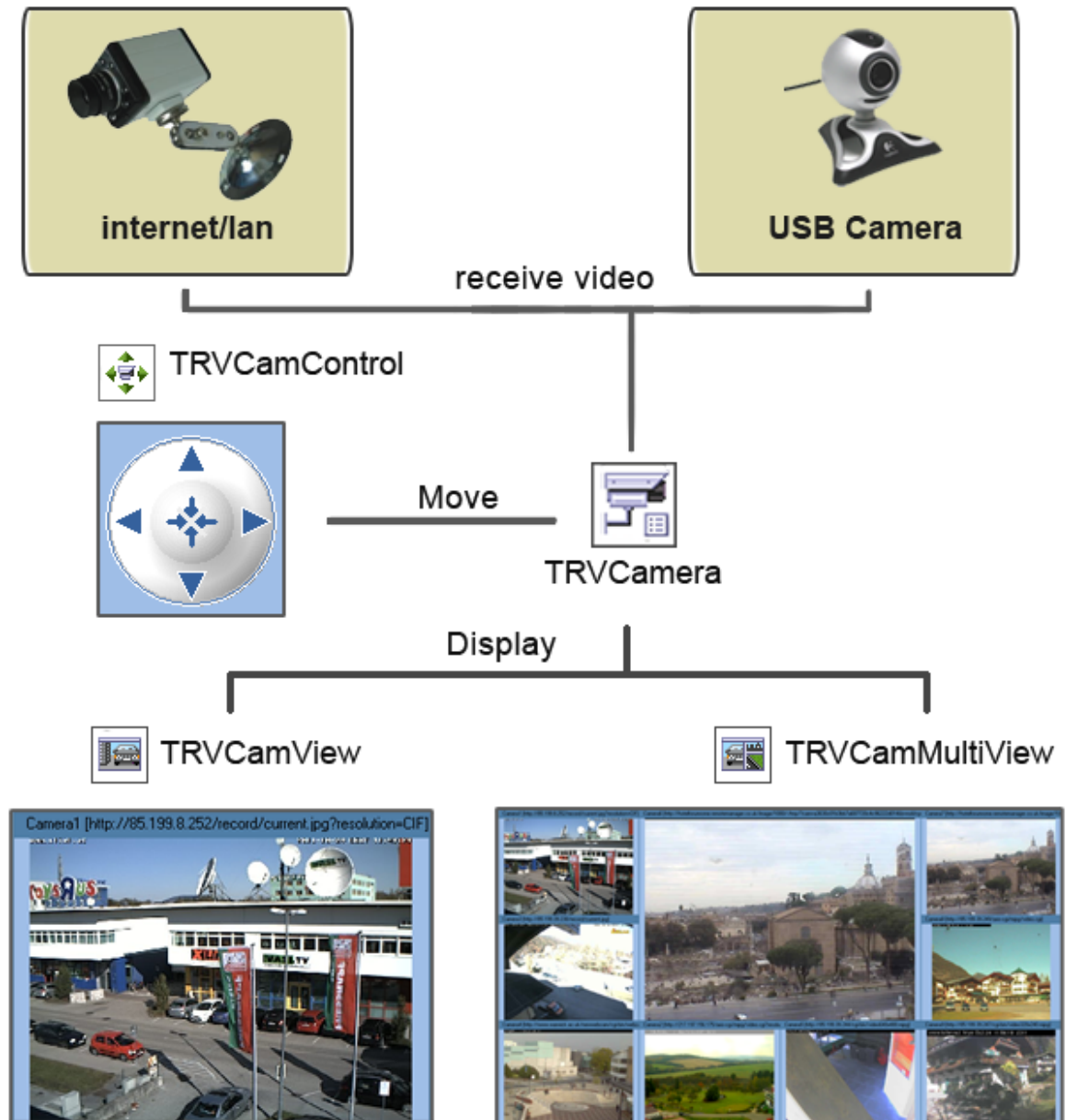
Overview



TRVCamera⁴² component works with cameras: searches, configures, receives a video stream, saves or plays a video file.

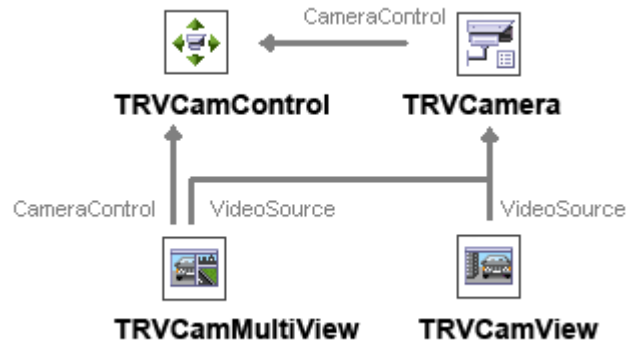
A video received by TRVCamera can be displayed in TRVCamView⁹³ or

TRVCamMultiView⁷⁴ components. The camera movement is controlled by TRVCamControl³⁹ component or by the viewer itself.



How the components are linked


The components are linked using CameraControl and VideoSource properties:






1.5 How it works: video chat without a server



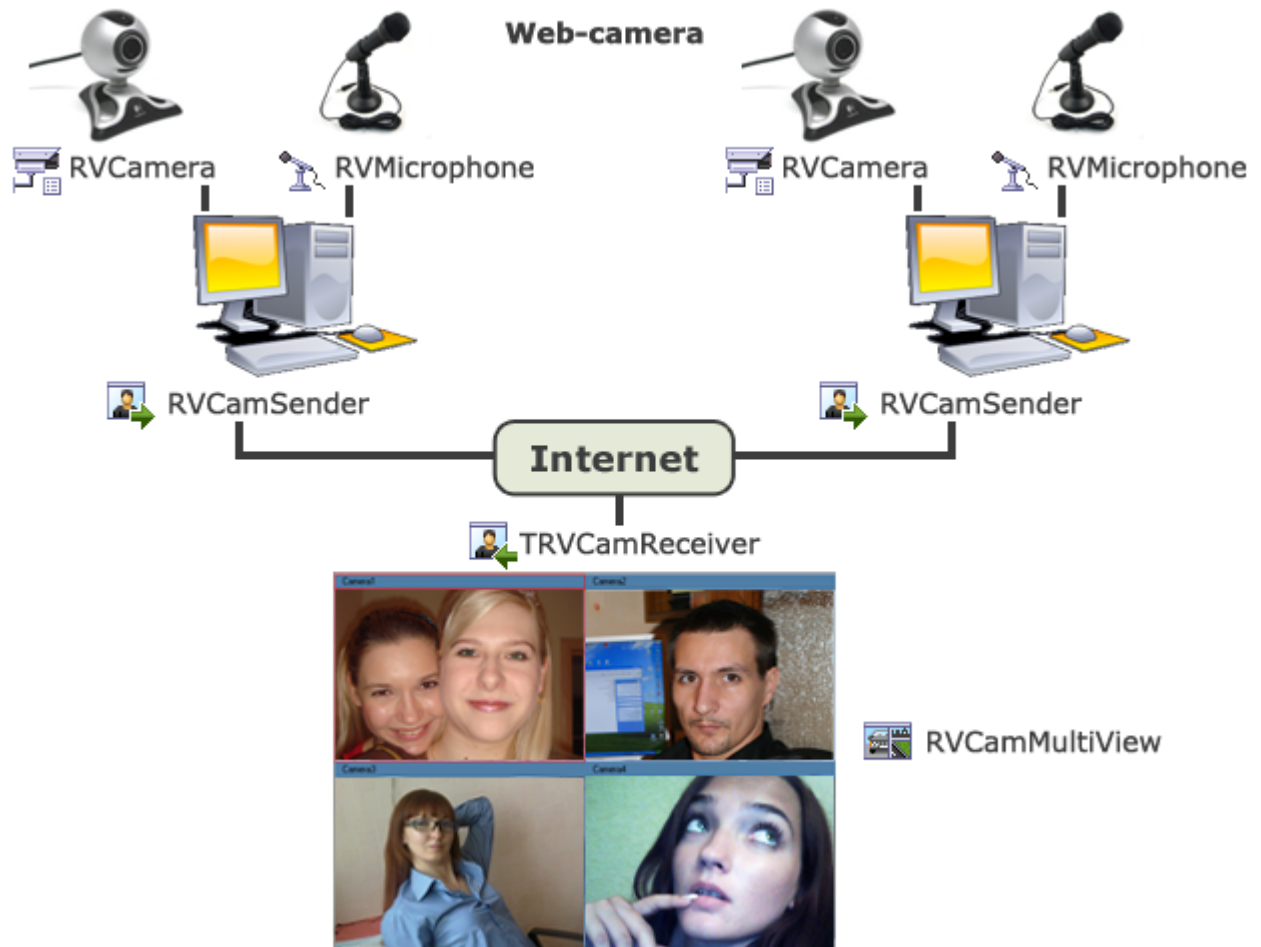
Overview

 TRVCamSender⁽¹³⁹⁾ and  TRVCamReceiver⁽¹²⁰⁾ allow sending and receiving video and audio streams, creating a video conference or a video chat.

A sender side: TRVCamSender receives a video from  TRVCamera⁽⁴²⁾ component, a sound from  TRVMicrophone⁽¹¹⁷⁾ or  TRVCamSound⁽¹¹⁵⁾ component and sends them to the Internet.

A receiver side: TRVCamReceiver receives video and audio from the Internet, from a single or multiple senders. These videos are displayed in TRVCamView⁽⁹³⁾ or TRVCamMultiview⁽⁷⁴⁾ components.

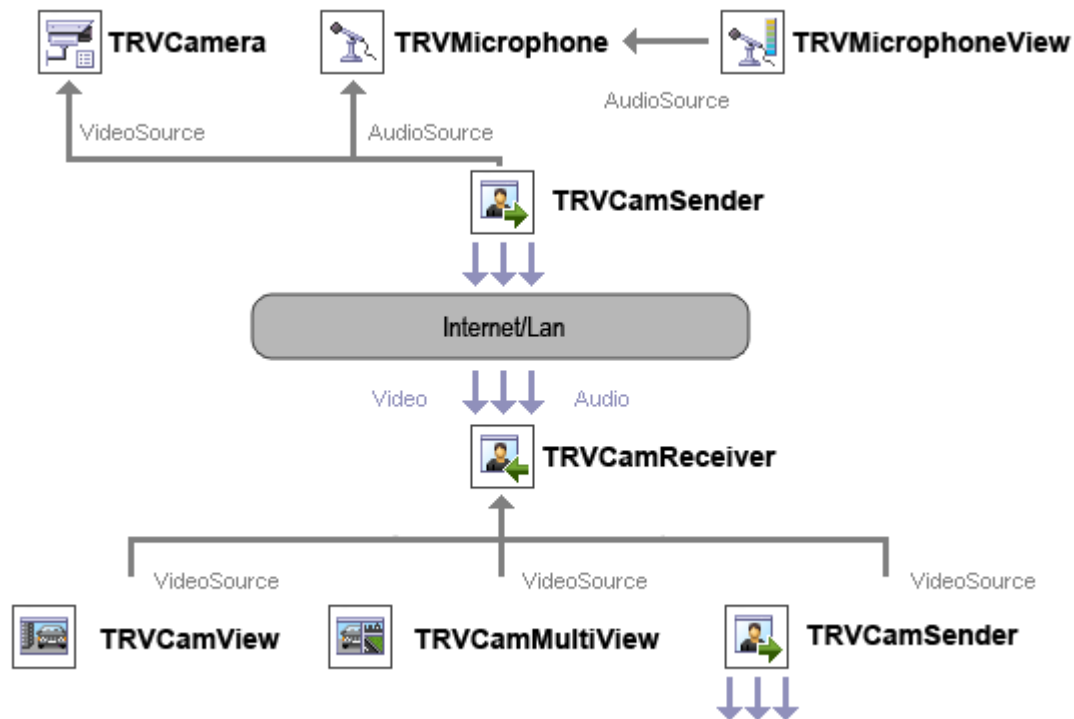
In addition to video and audio data, the components can send files, commands and arbitrary data.



To create a video chat, you need a sender and a receiver at the both sides.

How the components are linked



The components are linked using `VideoSource` and `AudioSource` properties




1.6 How it works: video chat with a server



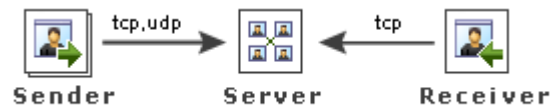
Overview

Although  **TRVCamSender**⁽¹³⁹⁾ and  **TRVCamReceiver**⁽¹²⁰⁾ can be directly connected⁽¹⁸⁾ to each other, this type of connection is not very convenient to connect multiple clients, because you need to implement many features yourself: maintaining a list of clients, resending received data to another clients, etc.

To solve these problems, we implemented a server component:  **TRVMediaServer**⁽¹⁶⁵⁾.

TRVMediaServer can receive data (video, audio, files, commands, etc.) from multiple senders and send them to multiple receivers. It implements several mechanisms allowing to determine which receivers receive data from each sender: private communications, groups, personal lists of allowed senders, personal lists of default receivers.

Each client of a server may consist of one more more TRVCamSenders and a single TRVCamReceiver. Multiple senders are useful to implement sending different types of data using different protocols (video and audio using UDP, other data using TCP).



1.7 Modes of connections

This topic explains how data (video, audio, files, commands, etc.) may be transferred between two applications via the network.

One application (which sends data) uses TRVCamSender⁽¹³⁹⁾ component. Another application (which receives data) uses TRVCamReceiver⁽¹²⁰⁾ component.

There are several modes of connections possible. The modes has the following differences: which side initiates a connection; TRVMediaServer⁽¹⁶⁵⁾ is used or not.

Working in different modes of connections

Information how to establishing different modes of connection can be found in the topic about TRVCamSender⁽¹³⁹⁾.

Connection from Sender⁽¹³⁹⁾ to Receiver⁽¹²⁰⁾ or MediaServer⁽¹⁶⁵⁾ (UDP, TCP or HTTP)

In this mode, the Sender does not make permanent connections to the Receiver (so *channels* and *sessions* are not established). The Sender sends new data when they become available. The Receiver (or the MediaServer) listens to the port and processes received data.

The Receiver uses the following events (with the parameters SessionKey = 0 и MediaTypes= []): OnConnecting, OnConnected, OnDisconnect, OnConnectError⁽¹³⁰⁾.

The Sender uses the following events (SessionKey = Sender.SessionKey, and MediaTypes contains the type of data to send): OnConnected, OnConnecting, OnDisconnect, OnConnectError⁽¹⁶³⁾.

In this mode, the receiver does not use he following events: OnOpenChannel, OnCloseChannel⁽¹³⁵⁾, OnSessionConnected, OnSessionDisconnected⁽¹³⁷⁾.

Connection from Receiver⁽¹²⁰⁾ to Sender⁽¹³⁹⁾ or MediaServer⁽¹⁶⁵⁾ (TCP or HTTP)

The receiver establishes a permanent connection to the Sender (or the MediaServer). For each data type⁽²⁴⁴⁾, a *channel* is created. All channels make up a *session*. When all channels are opened, a session is created. If at least one channel is closed, a session is terminated.

In this mode, all the Receiver's events are used: OnConnecting, OnConnected, OnDisconnect, OnConnectError⁽¹³⁰⁾, OnOpenChannel, OnCloseChannel⁽¹³⁵⁾, OnSessionConnected, OnSessionDisconnected⁽¹³⁷⁾. In all events, except for OnSessionConnected, OnSessionDisconnected⁽¹³⁷⁾, the parameter SessionKey=0.

The sequence of the Receiver's events when opening/closing a channel:

1. OnOpenChannel⁽¹³⁵⁾
2. OnConnecting⁽¹³⁰⁾
3. OnConnected/OnConnectError⁽¹³⁰⁾

4. OnDisconnect⁽¹³⁰⁾ (if no OnConnectError⁽¹³⁰⁾)
5. OnCloseChannel⁽¹³⁵⁾

When all channels are opened, OnSessionConnected⁽¹³⁷⁾ occurs.

If at least one of channels is closed, OnSessionDisconnected⁽¹³⁷⁾ occurs.

▼ Example

For example, only two channels are used: video and audio.

The events are called in the following order:

```

OnOpenChannel (for video)
OnConnecting
OnConnected
OnOpenChannel (for audio)
OnConnecting
OnConnected
OnSessionConnected (a session is established)
...
OnSessionDisconnected
OnDisconnect
OnCloseChannel
OnDisconnect
OnCloseChannel

```

Connections between clients

Option 1: A direct connection without a server

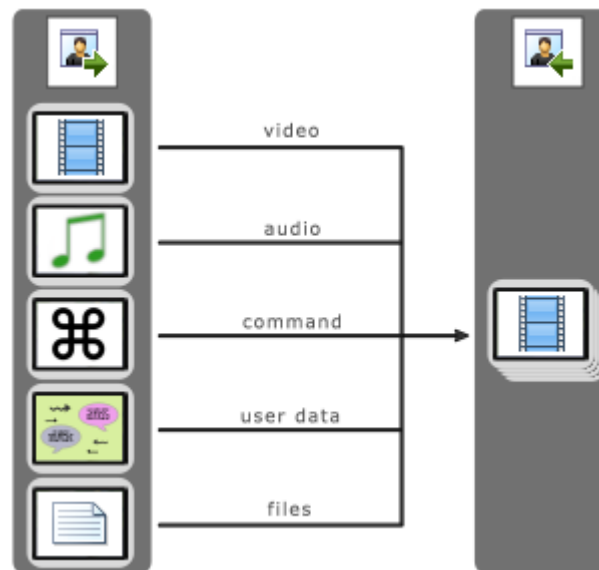
In the Option 1, there are two possible modes for transferring data from Client1 to Client2. The both modes requires that at least one side has a "white IP address" (available for another side)

1.a) Client1 sends data to Client2, when new data are available

In this mode Client2 must be visible for Client1, i.e. Client2 must have an IP address which Client1 can use to make a connection. Otherwise, this mode is not possible to use.

This mode is the fastest, because data are sent from Client1 to Client2 immediately when they become available, without delays.

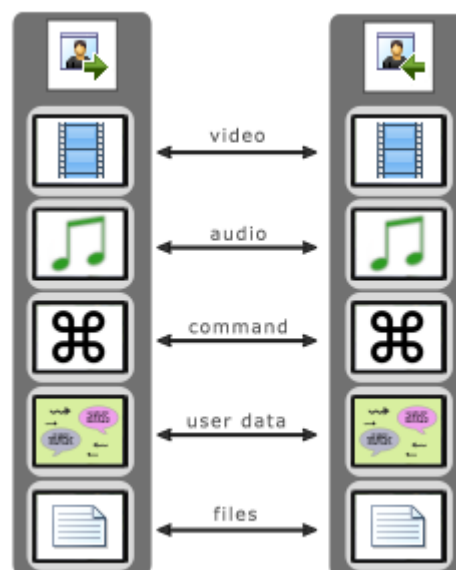
In this mode, a connection session is not created, because there are no permanent connections between the clients, a connection is created only when data are being sent, and it is terminated when the transfer is finished.



2.b) Client2 connects to Client1 and requests new data

In this mode Client1 must be visible for Client2, i.e. Client1 must have an IP address which Client2 can use to make a connection. Otherwise, this mode is not possible to use.

In this mode, permanent channels for each data type are opened. Client2 requests new data from each channel periodically. This type of connection was organized to provide a maximal possible bandwidth (if it would have been organized as a single channel, video data may overfill the channel, slowing down transfer of data of other types).

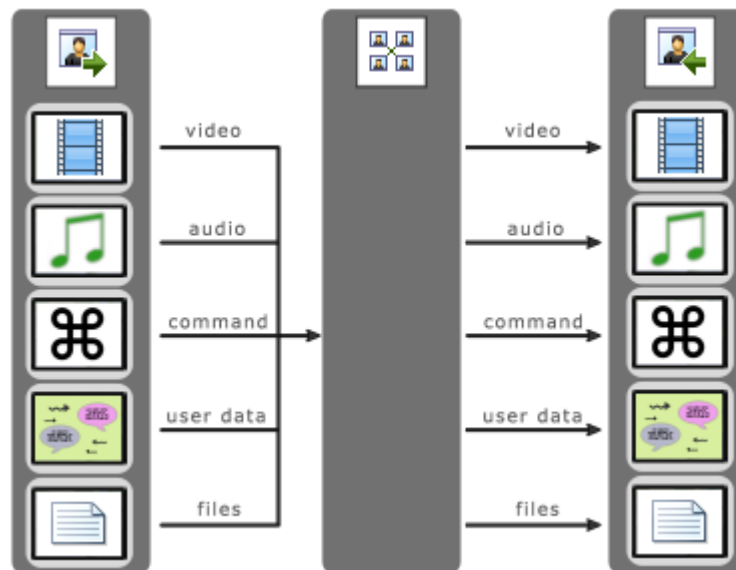


Option 2: Client-Server-Client

In this mode, Client1 sends data to Client2 via the Server. In this mode, only the Server must have a "white IP address".

Client1 connects to the Server directly. It does not create permanent channels, so the Server does not need to request data from the Client. When new data become available, Client1 connects to the Server and transfers them.

A situation for Client2 is different. It creates permanent channels for each data type and requests data from the Server periodically. It makes it possible for Client2 to work without a white IP address. Multiple channels allow to transfer data efficiently, but increase the server load.



This option of connection is used to organize transfers between multiple clients, creating groups of clients, contact lists, etc. A Server can help to transfer data between clients that cannot connect to each other directly.

1.8 Third-party libraries

RVMedia can use the third-party libraries listed below.

FFmpeg

FFmpeg is a free open-source multimedia framework.

Supported OS: Windows, Linux, macOS

Web site: www.ffmpeg.org

License: LGPL

Linked: dynamically

Supported versions: 1-7 (remuxing⁽²⁰³⁾ requires 4+)

Used by RVMedia for: receiving video stream in TRVCamera⁽⁴²⁾ (see FFMpegProperty⁽⁵⁰⁾ property), remuxing (saving to a file without re-encoding) of video streams in TRVCamera⁽⁴²⁾, recording video in TRVCamRecorder⁽⁸⁵⁾, recording audio in TRVAudioPlayer⁽¹¹⁰⁾.

See also: functions from MRVFFmpeg⁽²¹⁹⁾ and MRVFFMpegLists⁽²²⁰⁾ units (LoadFFMpegLibraries⁽²¹⁹⁾ contains important instructions for FFmpeg installation).

GStreamer

GStreamer is a free open-source multimedia framework.

Supported OS: Windows, Linux, macOS

Web site: gstreamer.freedesktop.org

License: LGPL

Linked: dynamically

Supported versions: 0.1 and 1 (1 is highly recommended)

Used by RVMedia for: receiving video stream in TRVCamera⁽⁴²⁾ (additional operation are possible by customizing a launch string; see GStreamerProperty⁽⁵²⁾ property)

See also: functions from MRVGStreamer⁽²²⁵⁾ unit (LoadGStreamerLibraries⁽²²⁵⁾ contain important instructions for GStreamer installation)

RNNoise

RNNoise is a noise suppression library based on a recurrent neural network.

Supported OS: Windows, Linux, macOS

Web site: github.com/xiph/rnnoise (you can download binaries from github.com/mjwells2002/rnnoise-bin/releases)

License: requires distribution including the file "COPYING"

Linked: dynamically

Used by RVMedia for: noise reduction in TRVMicrophone⁽¹¹⁷⁾ (see NoiseReduction⁽¹⁸⁰⁾ property)

See also: functions from MRVRNNoise⁽²²⁷⁾ unit (LoadRNNoise⁽²²⁷⁾ contains important instructions for RNNoise installation).

Fast Jpeg Decoder

JpegDec - Fast JPEG Decoder for Delphi

Supported OS: Windows 32-bit, Delphi VCL only (not supported in Lazarus and C++Builder)

Web site: www.marktg.com/jpegdec/

License: MPL

Linked: statically (included in the full version of RVMedia). Not used by default, requires a manual activation. See additional information in the comments at the beginning of **MRVJpegDec.pas**

Used by RVMedia for: Jpeg decoding (when reading Mjpeg video streams without Ffmpeg and GStreamer, in Mjpeg modes of local cameras), in TRVCamReceiver⁽¹²⁰⁾.

See also: RVMedia license, comments in **MRVJpegDec.pas**.

1.9 Additional information

RVMedia license, install and uninstall instructions, and a privacy statement can be found in ReadMe.chm located in the root folder of RVMedia installation.

1.10 Version history

 marks changes that may affect existing projects.

▼ After version 11

RAD Studio 13 Florence

Delphi and C++Builder 13 are supported.

RAD Studio 12.3 Athens

64-bit RAD Studio IDE is supported (for installation of Delphi+CBUILDER packages).

FFmpeg

Streaming: To start MPEG-TS streaming instead of file recording, assign a *udp://* or *srt://* URL to either TRVCamRecorder⁽⁸⁵⁾.OutputFileName⁽⁸⁹⁾ or TRVCamera⁽⁴²⁾.FFmpegProperty⁽⁵⁰⁾.RemuxProperty⁽²⁰¹⁾.FileName⁽²⁰³⁾.

Remuxing: additional property for remuxing (saving to a file or streaming without changing video and audio formats): TRVCamera.FFmpegProperty⁽⁵⁰⁾.Remuxing2⁽²⁰¹⁾; TRVFFmpegRemuxProperty⁽²⁰³⁾. You can use Remuxing⁽²⁰¹⁾ and Remuxing2⁽²⁰¹⁾ independently from each other (for example, one for recording, another one for streaming).

Error handling

The error handling mechanism for video receiving and recording has been revised. The result is accessible to the developer via TRVCamera.OnError⁽⁷⁰⁾, TRVCamRecorder.OnError⁽⁹²⁾, TRVAudioPlayer.OnError⁽¹¹⁴⁾ events.

Lazarus

The separation of runtime and designtime packages for Lazarus has been removed: instead of the two packages, *rvmedialaz.dpk* (runtime) and *rvmedialaz_dsgn.lpk* (designtime), there is now a single package, *rvmedialaz.dpk* (runtime + designtime).

▼ Version 11 (Delphi 12, FMX macOS, FFmpeg 6-7, remux, Mobotix, RNNoise)

RAD Studio 12 Athens

Delphi and C++Builder 12 are supported (including support of "Windows 64-bit modern" platform).

macOS

FireMonkey for macOS is supported (64-bit Intel and ARM)

Linux

- More settings are available in TRVWebCamDialog⁽¹⁰⁷⁾ for Linux (such as exposure, pan, tilt, zoom, focus, iris), if they are supported by the camera.
- PTZ control works for local web cameras in Linux.

FFmpeg

- New versions of FFmpeg⁽²⁴⁾ (versions 6 and 7) are supported, so RVMedia can use FFmpeg versions from 1 to 7.

- New property for TRVCamera.FFmpegProperty⁽⁵⁰⁾: NoDelay⁽²⁰¹⁾; it is useful to display online video streams.
- New property for TRVCamera.FFmpegProperty⁽⁵⁰⁾: VideoInputFormat⁽²⁰¹⁾, allows supporting special video sources
- New feature: video remuxing (saving without changing format) for videos played using FFmpeg. New property: TRVCamera.FFmpegProperty⁽⁵⁰⁾.Remuxing⁽²⁰¹⁾: TRVFFmpegRemuxProperty⁽²⁰³⁾.

Drawing

- TRVMRenderMode⁽²⁴⁶⁾: the OpenGL drawing feature is restored (and now available not only in Delphi VCL, but in Lazarus for Windows). DirectX drawing is removed. Skia4Delphi drawing is added.
- New property: TRVCamMultiView⁽⁷⁴⁾.Viewers⁽⁸²⁾[].AudioViewerColor defines a background color of an audio viewer.

IP cameras

Support of Mobotix cameras (having MJpeg URL like '*/cgi-bin/faststream.jpg'). You can rotate them and change video brightness.

Sound

Noise reduction is improved:

- new properties for RVMedia's own noise reduction procedure: TRVMicrophone.NoiseReductionLevel⁽¹⁸⁰⁾ and TRVAudioPlayer.NoiseReductionLevel⁽¹⁹⁴⁾
- ability to use RNNoise⁽²⁵⁾ for noise reduction in TRVMicrophone, new property UseRNNoise⁽¹⁸⁰⁾

Video decoding

- Support for video frames represented by Jpeg images not having a Huffman table (may be produced by some MJpeg or local cameras).
- Ability to use **Fast JPEG Decoder for Delphi** to decode MJpeg streams and Jpeg frames from local cameras faster. Only for Win32 Delphi VCL projects (not for C++Builder, FireMonkey, Lazarus, Win64 (the standard Delphi TJpegImage is already fast for Win64)). Used under the Mozilla Public License (MPL). Not used by default, requires a manual activation. Details can be found in the comment at the beginning of **MRVJpegDec.pas** and in the RVMedia license.

Desktop streaming


New function GetVisibleWindowsHandles⁽²¹⁸⁾ returns potential values for TRVCamera.DesktopWindowHandle⁽⁴⁷⁾ property.

Other changes

- The default value of TRVCamera.Latency⁽⁵³⁾ is changed to 0.
- Optimization: internal buffer is not used for frames if TRVCamera.Latency⁽⁵³⁾ = 0, so TRVCamera works more efficiently.
- New DescribeVideoMode⁽²²⁸⁾ and DescribeVideoModePixelFormat⁽²²⁸⁾ functions
- New TRVCamRecorder⁽⁸⁵⁾.OnGetImage⁽⁹³⁾ event.

- New TRVCamReceiver⁽¹²⁰⁾.UseTempFiles⁽¹²⁸⁾ property; improved efficiency and reliability of receiving files.

Refactoring

GUID-related functions are moved to the new MRVGUIDFuncs unit (fmxMRVGUIDFuncs for FireMonkey) . This unit is added to "uses" of demo projects, when necessary.

▼ Version 10 (FMX Linux)

FireMonkey for Linux

RVMedia supports Linux not only in Lazarus version, but also in FireMonkey version (for Delphi 10.3 and newer).

Changes in visual components


New properties:

- TRVCamMultiView⁽⁷⁴⁾.RefreshRate⁽⁸⁰⁾ can improve redrawing efficiency when displaying a large count of videos.
- TRVCamView⁽⁹³⁾.FullScreenView⁽⁹⁸⁾ and TRVCamMultiView⁽⁷⁴⁾.FullScreenMultiView⁽⁷⁹⁾ provide read-only access to full-screen versions of video viewers.
- TRVCamControl⁽³⁹⁾.DiskColor, DiskBrightness⁽⁴¹⁾ define the color of the control.

Changes in FFmpeg support

- TRVCamera⁽⁴²⁾.FFmpegProperty⁽⁵⁰⁾.UseVideoScale⁽²⁰¹⁾ supports proportional resizing (if only one video size is defined).
- New value in TRVVideoCodec⁽²⁵¹⁾: *rvcHEVC*.

Changes in GStreamer support




- Since this version, TRVCamera⁽⁴²⁾.VideoResolution⁽⁶¹⁾ does not affect size of video played using GStreamer  (only sizes defined in GStreamerProperty⁽⁵²⁾ can be applied).
- RVMedia supports both MSVC and MinGW builds of GStreamer for Windows

Other changes

New properties:

- TRVCamera⁽⁴²⁾.Parameters.AutoChangeAlias⁽⁵⁴⁾ allows to turn off updating values of Alias property
- TRVCamera⁽⁴²⁾.DesktopForm⁽⁴⁷⁾ (FireMonkey only)
- TRVMediaServer⁽¹⁶⁵⁾.BufferOptions⁽¹⁶⁷⁾.LimitType⁽¹⁹⁶⁾ property; not-limiting file buffers by default; increasing the default buffer limit for transferring files
- New value for TRVBoundsTestMode⁽²³⁷⁾: *rvstmRectangles*.

Refactoring


- Command-related classes (TRVCmd⁽¹⁹⁷⁾, TRVCmdParamCollection⁽¹⁹⁸⁾, TRVCmdParamItem⁽¹⁹⁸⁾) and events are moved from MRVType unit to the new MRVCmd unit .
- All units for Lazarus for Linux are renamed, from mrvlcl_lin_*.pas to MRVLin*.pas (for example, mrvlcl_lin_Player.pas is renamed to MRVLinPlayer.pas) .
- Type of all relative time values (tick counts) are changed from Cardinal to Int64 . As a result:
 - type of AStartTime parameter is changed in the events: TRVCamReceiver.OnDecodeAudio⁽¹³⁰⁾, all events of TRVAudioEvent⁽²³¹⁾ type

- networking protocol between TRVCamSender⁽¹³⁹⁾ and TRVCamReceiver⁽¹²⁰⁾ is changed, you need to rebuild both sides with the new RVMedia version

▼ Version 9 (Delphi 11, TRVCamSound, wait animation, FFmpeg 5, Linux sound)

Delphi and C++Builder **11 Alexandria** are supported.

Sound from videos:

New  TRVCamSound⁽¹¹⁵⁾ component allows reading sound from videos received by TRVCamera⁽⁴²⁾.

In the following events, the type of ASamplesPerSec parameter is changed to Integer[!] to allow arbitrary values: TCustomRVAudioOutput⁽¹⁹¹⁾.OnGetAudio⁽¹⁹²⁾, TCustomRVMicrophone⁽¹⁷⁷⁾.OnGetAudio⁽¹⁸⁷⁾, TRVCamSender⁽¹³⁹⁾.OnEncodeAudio⁽¹⁶⁴⁾, TRVCamReceiver⁽¹²⁰⁾.OnDecodeAudio⁽¹³⁰⁾.

New animation in video viewers:

TRVCamView⁽⁹³⁾ and TRVCamMultiView⁽⁷⁴⁾ components can display an animation indicating that the component waits for a video frame.

New properties: TRVCamView.WaitAnimationDelay⁽¹⁰⁵⁾ and TRVCamMultiView.WaitAnimationDelay⁽⁸⁴⁾.

Displaying videos:

TRVCamView⁽⁹³⁾.FrameScaleQuality⁽⁹⁷⁾, TRVCamMultiView⁽⁸⁴⁾.Viewers[].FrameScaleQuality⁽⁸²⁾ allows to lower video scale quality in VCL and LCL for Windows (to use less CPU time).

Encoding and decoding using FFmpeg:

New version of FFmpeg (version 5) is supported, so RVMedia can use FFmpeg versions from 1 to 5.

New property: TRVCamRecorder⁽⁸⁵⁾.VideoEncodingParameters⁽⁹¹⁾.

New event: TRVCamera⁽⁴²⁾.OnGetVideoStreamIndex⁽⁷¹⁾.

Ability to choose the specific codec for encoding/decoding. For decoding: TRVCamera⁽⁴²⁾.FFmpegProperty⁽⁵⁰⁾.DecodeAudioCodecName, DecodeVideoCodecName⁽²⁰¹⁾. For encoding: TRVCamRecorder⁽⁸⁵⁾.AudioCodecName, VideoCodecName⁽⁸⁸⁾, TRVAudioPlayer⁽¹¹⁰⁾.EncodeAudioCodecName⁽¹¹²⁾. You can use GetListOfAudioEncoders, GetListOfVideoEncoders, GetListOfAudioDecoders, GetListOfVideoDecoders⁽²²⁰⁾ functions to get possible values of these properties.

Linux version

Linux sound (using ALSA) in Lazarus is reworked.


Design-time support

A version checker is added. It allows to check for a new version of RVMedia and download the updated installer. The first check is performed when you click on a button in the version checker window. Subsequent checks are performed when Delphi/Lazarus IDE starts, only if you allowed them by leaving the "check on start" checkbox checked. On these checks, a version checker window is shown only if a new version is found (and no more than once a day).

▼ Version 8 (Delphi 10.4, TRVWebCamDialog, full-screen, HTTP login)

Delphi and C++Builder **10.4 Sydney** are supported.

Better support of local USB cameras (both in Windows and Linux):

- all camera rotation methods are supported for local USB webcams (for Windows);
- Brightness, Contrast, Saturation, Sharpness, Hue⁽⁴⁵⁾ properties are applied to local USB webcams;
- new  TRVWebCamDialog⁽¹⁰⁷⁾ component; it displays a dialog window for changing local USB webcam properties (for Windows and Linux);
- more camera video modes⁽⁶³⁾ are supported in Windows; added support for camera video modes in Linux;
- unsupported video modes are excluded from a list of video modes.⁽⁶³⁾

Color control:

New method GetColorControlPropertyRange⁽⁶⁵⁾ returns an allowed range for Brightness, Contrast, Saturation, Sharpness, Hue⁽⁴⁵⁾ properties of the selected video source.

Full-screen mode for video viewers:

- new TRVCamView⁽⁹³⁾.AllowFullScreen⁽⁹⁵⁾ and TRVCamMultiView⁽⁷⁴⁾.AllowFullScreen⁽⁷⁶⁾ properties to display full-screen buttons
- new TRVCamView⁽⁹³⁾.FullScreen⁽⁹⁷⁾ and TRVCamMultiView⁽⁷⁴⁾.FullScreen⁽⁷⁹⁾ properties to switch between a full-screen and a normal mode.

Resizing video viewers in TRVCamMultiView⁽⁷⁴⁾:


- new ScaleViewers⁽⁸¹⁾ property
- new Anchors and AlignVideoViewer properties for TRVCamMultiView⁽⁷⁴⁾.Viewers⁽⁸²⁾

New server methods:

- SendCmdToUser⁽¹⁷¹⁾
- SendCmdToGroup⁽¹⁷¹⁾

Proxy configuration, new properties:


- TRVCamera⁽⁴²⁾.ProxyProperty⁽⁵⁵⁾
- TRVCamSender⁽¹³⁹⁾.ProxyProperty⁽¹⁴⁹⁾
- TRVCamReceiver⁽¹²⁰⁾.ProxyProperty⁽¹²⁵⁾

TRVCamSender.ProxyHost and ProxyPort are removed .

Authentication:

- HTTP authentication code is improved and corrected
- ability to request a user name and a password from the user: new TRVCamera⁽⁴²⁾.LoginPrompt⁽⁵³⁾, Language⁽⁵²⁾ properties, OnLogin⁽⁷¹⁾ event
- new TRVCamera⁽⁴²⁾.OnLoginFailed⁽⁷²⁾ event.



FFmpeg configuration (new properties in TRVCamera⁽⁴²⁾.FFMpegProperty⁽⁵⁰⁾):

- UseFFMpegProperty allows/disallows assigning FFMpeg parameters
- RTFPPFlags, ProbeSize properties
- FrameDrop: Boolean allows/disallows dropping frames if they are received too late.
- CustomProperties: TStringList: additional parameters for FFMpeg.
- RTSPTTransport has the new default value: *[rvpeTCP, rvpeUDP]* .

GStreamer 1.0 is supported. Since this version, RVMedia can use both GStreamer 0.1 and GStreamer 1.0. The following special features are supported for GStreamer 1.0:

- listening the specified port to receive UDP video stream (see the comments about TRVCamera.DeviceType⁽⁴⁹⁾ property)
 - detecting HTTP/HTTPS/RTSP/UDP protocol by URL
 - using a custom pipeline string (TRVCamera.GStreamerProperty⁽⁵²⁾.LaunchString and LaunchStringMiddle⁽²⁰⁴⁾); using these low-level properties you can implement other video sources, or add additional video processing options (such as video recording or streaming).
 - TRVCamera.GStreamerProperty⁽⁵²⁾.UseQueue⁽²⁰⁴⁾
- New function: LoadGStreamerLibraries⁽²²⁵⁾.

Other changes:

- *rvccpDate* and *rvccpTime* are excluded from TRVCamView⁽⁹³⁾.CaptionParts⁽¹⁰¹⁾ by default 
- TRVCamera⁽⁴²⁾.UpdateUsers  is removed. AddUser and ModifyUser⁽⁶²⁾ are added instead.
- TRVCamSender⁽¹³⁹⁾.OnEncodeVideo⁽¹⁶⁴⁾ and TRVCamReceiver⁽¹²⁰⁾.OnDecodeVideo⁽¹³¹⁾ events allow using custom compression or encryption for video.
- TRVCamSender⁽¹³⁹⁾ can auto-detect sending buffer size, the default value of BufferSize⁽¹⁴⁴⁾ property is changed to 0 (meaning auto-detect)
- optimization: much faster sending and receiving, especially for large files and video frames, much more efficient reading of MJPEG files and streams.

▼ Version 7 (FMX Windows, Delphi 10.3)


FireMonkey for Windows is supported (for Delphi and C++Builder XE6 and newer).


Delphi and C++Builder **10.3 Rio** are supported. **Lazarus 2** is supported.

TRVCamReceiver⁽¹²⁰⁾.VideoLatency⁽¹²²⁾ is implemented.


New properties for TRVCamView⁽⁹³⁾: IconStyle⁽⁹⁹⁾, SearchPanelColor, SearchPanelTextColor⁽⁹⁹⁾, they define the appearance of a camera search panel. The same properties for TRVCamMultiView⁽⁷⁴⁾.


New properties for TRVCamera⁽⁴²⁾.FFMpegProperty⁽⁵⁰⁾ (Protocol is removed, replaced by UDPTransport )

Declaration of TRVSocket is moved to MRVSocket unit 

New properties: TRVCamSender⁽¹³⁹⁾.PixelColorThreshold⁽¹⁴⁸⁾ and TRVMotionDetector⁽²⁰⁹⁾.PixelColorThreshold⁽²¹⁰⁾ for red, green, blue colors. Default value is changed from 15 to 8 .

When sending a command⁽¹⁶⁰⁾, it's no longer necessary to wait until the previous command is sent. New TRVCamSender.OnSentCmd⁽¹⁶⁵⁾ event.

 Since this version, it is highly not recommended to display modal forms in certain events, mainly of TRVCamReceiver⁽¹²⁰⁾. We modified the "ChatRoom" demo to use a non-modal chat form.

 DirectX rendering mode⁽⁹⁶⁾ is abandoned (probably, it will be revived in future versions).


▼ Version 6 (Media channels, TRVAudioPlayer, TRVCamRecorder, FFmpeg 4, custom video source)

Media channels

In previous versions, TRVCamSender⁽¹³⁹⁾ component could send video from a single source; the same for audio. It was OK for direct sender-receiver connections, because a receiver can receive data from multiple senders. However, it was a problem for client-server applications, if a client wanted to translate multiple video and sound sources. In this version, media channels allow to solve this problem.

The main new property related to media channels is TRVCamSender⁽¹³⁹⁾.ExtraMediaSources⁽¹⁴⁶⁾. A new optional parameter (AMediaIndex, index of the media channel) is added to the methods for sending commands, files and user data.

New parameter (AMediaIndex) is added to the events of TRVCamReceiver⁽¹²⁰⁾:

OnReceiveFileData⁽¹³⁶⁾, OnReceiveCmdData⁽¹³⁵⁾, OnReceiveUserData⁽¹³⁷⁾, OnDataRead⁽¹⁸⁵⁾ . If you already used these events, you need to add this parameter in the event handlers.

New property IndexFrom⁽⁹⁸⁾ (index of media channel) is added to TRVCamView⁽⁹³⁾ and TRVCamMultiView⁽⁷⁴⁾.Viewers⁽⁸²⁾ [].

Sound



TRVAudioPlayer⁽¹¹⁰⁾ is a new component allowing to play sound and to record it to a file. It can be linked with TRVMicrophone⁽¹¹⁷⁾ or TRVCamReceiver⁽¹²⁰⁾ components.

The sound sub-system of TRVCamReceiver is rewritten, so sound is much clearer now.

The components now support stereo sound.

Video recording



TRVCamRecorder⁽⁸⁵⁾ is a new component for recording video and sound files. It gets audio and video data from TRVCamera⁽⁴²⁾, TRVMicrophone⁽¹¹⁷⁾ or TRVCamReceiver⁽¹²⁰⁾ components.

Local web cameras

RVMedia supports more video formats (I420, NV12, IYUV, UYVY), even if corresponding decoders are not installed in the system.


IP cameras

RVMedia supports cameras having different ports for commands and RTSP video. A new property is added: TRVCamera⁽⁴²⁾.RTSPPort⁽⁴⁶⁾.


Custom video

A new type of video source is added: DeviceType⁽⁴⁹⁾ = *rvidtUserData*. In this mode, video frames are requested from an application in OnNewImage⁽⁷²⁾ event.

Other changes in cameras

TRVCamera.FramePerSec⁽¹⁹⁰⁾ is a fractional value now. 


FFmpeg support

New TRVCamera⁽⁴²⁾.FFmpegProperty⁽⁵⁰⁾ property contains sub-properties to configure FFmpeg. TRVCamera⁽⁴²⁾.UseFFmpeg property is moved to TRVCamera.FFmpegProperty⁽⁵⁰⁾.UseFFmpeg 

FFmpeg is used in TRVCamRecorder⁽⁸⁵⁾ and TRVAudioPlayer⁽¹¹⁰⁾ components for video and audio recording.

FFmpeg version 4 is supported (as well as versions 3 and 2).

Microphone


The type of TRVMicrophone⁽¹¹⁷⁾.VolumeMultiplier⁽¹⁸²⁾ is changed from Byte to Double, to allow decreasing sound volume .

Camera viewers

TRVCamMultiView⁽⁷⁴⁾ is now implemented not as a parent window for internal TRVCamView⁽⁹³⁾ components. Since this version, it draws everything in a single window. It allows implementing DirectX and OpenGL drawing modes more efficiently.

New property: TRVCamView.CaptionHeight⁽¹⁰¹⁾ and TRVCamMultiView.CaptionHeight⁽⁷⁷⁾.

Demo projects


All Delphi demo projects are moved from "Demos" to "Demos\Delphi" folder . New folder for Lazarus demo project: "Demos\Lazarus".

If you installed this new version without uninstalling the previous version, delete all folders from "Demo", except for "Delphi" and "Lazarus" folders.

New demo projects:

- SendAndReceive\TwoSides: how to connect two applications, if IP address is available is only for one side
- ClientServer\VideoChats\Lecture: one client (Lecturer) shows video from two sources to other clients (Students)
- Recording\AudioRecorder: how to use new TRVAudioPlayer component
- Recording\VideoRecorder: how to use new TRVCamRecorder component

Other changes

- The following events have new parameter (RemoteSessionKey): TRVCamSender.OnConnected, OnConnecting, OnDisconnect, OnConnectError⁽¹⁶³⁾, TRVCamReceiver.OnConnected, OnConnecting, OnDisconnect, OnConnectError⁽¹³⁰⁾ . If you already used these events, you need to add this parameter in the event handlers.
- All custom cursors now have 32x32, 48x48, and 64x64 versions.
- New VideoDefaultAcceptAll, AudioDefaultAcceptAll, UserDefaultAcceptAll, FileDefaultAcceptAll, CmdDefaultAcceptAll properties of items in TRVCamReceiver⁽¹²⁰⁾.Senders⁽¹²⁵⁾ collection (default values correspond to the old behavior).
- Chinese user interface⁽²⁴⁵⁾.
- new property: TRVCamera.SmoothImage⁽⁵⁶⁾ and TRVCamReceiver.SmoothImage⁽¹²⁶⁾.

▼ Version 5 (Groups on server, multiple microphones, FFmpeg 3, Foscam)

Groups on a media server (TRVMediaServer⁽¹⁶⁵⁾ component): named and password-protected groups

Starting from this version, a group may have a name and a password. They may be specified in the parameters of TRVCamSender⁽¹³⁹⁾.JoinGroup⁽¹⁵⁸⁾. The group name and the identifier of the group

creator may be requested using `TRVCamSender.GetGroupInfo`⁽¹⁵⁸⁾ method, and returned in `TRVCamReceiver`⁽¹²⁰⁾.`OnGetGroupInfo`⁽¹³¹⁾. To join a password-protected group, other users must specify the same password in `TRVCamSender`⁽¹³⁹⁾.`JoinGroup`⁽¹⁵⁸⁾.

A client may receive a list of all groups from the server, using `TRVCamSender`⁽¹³⁹⁾.`GetAllGroups`⁽¹⁵⁸⁾ and `TRVCamReceiver`⁽¹²⁰⁾.`OnGetAllGroups`⁽¹³³⁾.

The count of groups on the server may be limited in `TRVMediaServer`⁽¹⁶⁵⁾.`MaxGroupCount`⁽¹⁷⁰⁾ property.

Some server features (like getting the list of all groups, getting the list of all online users, restarting the server) may be undesirable, so they must be turned on in `TRVMediaServer`⁽¹⁶⁵⁾.`CmdOptions`⁽¹⁶⁸⁾

Other new features of a media server


A client may restart the server by calling `TRVCamSender`⁽¹³⁹⁾.`RestartServer`⁽¹⁶⁰⁾ (works only if this feature is turned on in `TRVMediaServer`⁽¹⁶⁵⁾.`CmdOptions`⁽¹⁶⁸⁾)

A client may request a list of all [online] users on the server: `TRVCamSender`⁽¹³⁹⁾.`GetAllUsers`, `GetAllOnlineUsers`⁽¹⁵⁸⁾ (works only if this feature is turned on in `TRVMediaServer`⁽¹⁶⁵⁾.`CmdOptions`⁽¹⁶⁸⁾)



Changes in sound processing

- `TRVMicrophone`⁽¹¹⁷⁾ allows choosing the microphone (or another audio input device), new properties: `AudioInputDeviceIndex`, `AudioInputDeviceCount`, `AudioInputDeviceList`⁽¹⁷⁹⁾.
- You can encode audio data before sending them to the network (`TRVCamSender`⁽¹³⁹⁾.`OnEncodeAudio`⁽¹⁶⁴⁾) and decode them when they are received (`TRVCamReceiver`⁽¹²⁰⁾.`OnDecodeAudio`⁽¹³⁰⁾).




Changes in local USB cameras

- RVMedia can decode different formats of video from local web cameras (YV12, YUYV, YUY2, YUYU, UYVY, NV12, etc.) itself. Previously, it relied on a converter that converts these formats to RGB (it may be installed or not).
- `TRVCamera.VideoResolution`⁽⁶¹⁾ now affects web cameras 

Other changes

- Our motion detection class (used in `TRVCamSender` to detect changed areas to send) is now available as `TRVMotionDetector`⁽²⁰⁹⁾.
- RVMedia supports FFmpeg 3.0 (as well as previous versions of FFmpeg libraries)
- RVMedia supports newer (H.264) Foscam IP cameras (rotation and administration functions). New properties to configure these cameras: `Hue`, `Saturation`, `Sharpness`⁽⁴⁵⁾, `Bitrate`⁽⁴⁵⁾
- `TRVCamSender.UseVideoResolution` property is removed . Instead, a new option is added to `TRVVideoResolution`⁽²⁵²⁾: *rvDefault*. This value is now used as default for `TRVCamera.VideoResolution`⁽⁶¹⁾ и `TRVCamSender.VideoResolution`⁽¹⁵⁴⁾ . New resolutions are added to `TRVVideoResolution`⁽²⁵²⁾.
- new package scheme: 32+64 bit runtime packages + 32 bit design-time packages.

Renamed objects

- TRVCamReceiver.OnVideoAccessRequest and OnVideoAccessCancelRequest are renamed to OnMediaAccessRequest and OnMediaAccessCancelRequest⁽¹³⁹⁾ (as they already were called in this help file) . A new parameter ADataType is added to these events . The same parameter is added as an optional parameter to TRVCamSender.SendMediaAccessRequest⁽¹⁶²⁾ and SendMediaAccessCancelRequest⁽¹⁶²⁾.
- TRVCamera.GStreamerProperty⁽⁵²⁾.Bitrate is renamed to KBitrate .
- TRVCamera.Decoder.Baudrate was replaced by Bitrate.

Changes in demo projects


- new demo: ClientServer\VideoChats\ChatRooms\ shows how to use named password-protected rooms, and how a room creator can choose a user who will transmit video to all other group members
- Cameras\MotionDetect is moved to Cameras\MotionDetect_Old, a new demo (using TRVMotionDetector⁽²⁰⁹⁾) is placed in Cameras\MotionDetect\
- previously, TRVCamReceiver⁽¹²⁰⁾.Senders⁽¹²⁵⁾ [].GUIDFrom did not filter data of all types, and this property was used in video chats to filter video; in the new version, TRVCamReceiver⁽¹²⁰⁾.Senders⁽¹²⁵⁾ [].VideoSenders property is used, as it should;
- changes related to changes in VideoResolution and UseVideoResolution properties (see above)
- ClientServer\VideoChat demos allow choosing a microphone
- changes related to renaming "VideoAccess" events to "MediaAccess" (see above)

▼ Version 4 (Lazarus, H.264 camera search)

TRVCamera.SearchCamera⁽⁶⁸⁾ can search not only for MJPEG, but also for H.264 cameras. H.264 cameras are supported in DeviceType⁽⁴⁹⁾=rvdtIPCamera mode, if FFmpeg is available.

Lazarus for Windows and Linux is supported.

You can specify path of FFmpeg libraries⁽²¹⁹⁾.

Changes affecting compatibility:

- TRVCamera.OnGetImage⁽⁷¹⁾: the type of Img parameter is changed to TRVMBitmap⁽²⁰⁶⁾; this event is now called in a thread context.
- TRVMediaServer.OnDataRead⁽¹⁷²⁾ has a new parameter ADataType.
- TRVCamera.OnGetSnapshot event is removed, because GetSnapshot⁽⁶⁵⁾ returns the last frame in any camera mode.
- TRVImageWrapper⁽²⁰⁶⁾.GetBitmap now returns TRVMBitmap⁽²⁰⁶⁾ instead of TBitmap.
- TRVCamera.SearchCamera⁽⁶⁸⁾ work depends on VideoFormat⁽⁵⁹⁾ property
- TRVCmdParamItem⁽¹⁹⁸⁾.Value property is removed, a value must be accessed using TRVCmdParamItem's methods.


Since this version, TRVCamera.OnGetImage⁽⁷¹⁾ is called before saving data to MJpeg file, so if you painted something on a video frame in this event, this painting will be saved to a file.

Default value for TRVCamera.MaxCameraSearchThreadCount⁽⁵³⁾ is increased.

▼ Version 3 (FFmpeg)

FFmpeg support. New properties and methods:

- UseFFMPEG (note: in v 6.0, this property is moved to FFMpegProperty⁽⁵⁰⁾)
- IsSupportedFFMPEG⁽⁶⁵⁾

Renamed items :


- in the units names, "Win" is moved from the end to the beginning (for example, MRVMicrophoneWin.pas is renamed to MRVWinMicrophone.pas);
- TRVCamViewItem.GUID⁽⁸²⁾, TRVCamView.GUID⁽⁹⁸⁾, TRVAudioViewer.GUID⁽¹⁸⁹⁾ are renamed to GUIDFrom.


▼ Version 2.2

TRVCamSender.Encoding⁽¹⁴⁵⁾ = *rvetPNG* and *rvetPNGChange* are implemented for Delphi 2009 and newer.


Path to **GStreamer** libraries can be taken from environment variables written by the GStreamer installer, it's not necessary to add this path to PATH environment variable any more.

▼ Version 2.1

More formats are added in TRVCamera.VideoFormat⁽⁵⁹⁾; *rvvfMPEG4* is renamed to *rvvfAVI_MPEG* .

Multi-monitor support for TRVCamera.DeviceType⁽⁴⁹⁾ = *rvdtDesktop*: VideoDeviceIndex⁽⁵⁸⁾ allows switching the source monitor. New property DesktopZoomPercent⁽⁴⁷⁾ allows scaling frames of video received in this mode (previously, VideoResolution⁽⁶¹⁾ property was applied )

The list of camera models supported by TRVCamera.SearchCamera⁽⁶⁸⁾ is greatly increased. This method has a new optional parameter allowing to narrow down a search.

TRVCamera.IPCameraType property is changed to IPCameraTypes⁽⁵²⁾ .

New TRVCamera.VideoDeviceIdList⁽⁵⁸⁾ property returns unique identifiers of local web cameras.

▼ Version 2.0

GStreamer support

In TRVCamera⁽⁴²⁾ + **GStreamer**, protocols and video formats are separated now.

TRVCamera.DeviceType⁽⁴⁹⁾ property is changed , TRVCamera.VideoFormat⁽⁵⁹⁾ property is added.

TRVCamera.GStreamerProperty⁽⁵²⁾ allows configuring GStreamer.

TRVCamera.UseGStreamer property is moved to TRVCamera.GStreamerProperty⁽⁵²⁾.UseGStreamer .

Sending⁽¹³⁹⁾

New property for TRVCamSender: SendOptions⁽¹⁵¹⁾.

New event OnSendCmd⁽¹⁶⁵⁾ helps to debug command sending.

Changed areas are detected in reduced frame images, to make processing faster: new property ChangedAreaProcessingMode⁽¹⁴⁴⁾.

Receiving

TRVCamReceiver⁽¹²⁰⁾ and TRVCamera⁽⁴²⁾ now use three threads for video data: for receiving, for decoding, and for drawing. In this way, a camera can receive video very fast without waiting for processing, to prevent lags.

OnReceiveFileData⁽¹³⁶⁾ and OnDataRead⁽¹⁸⁵⁾ events are now called in a thread context .


New events: OnReceivingFile and OnReceivedFile⁽¹³⁶⁾.


The event OnReceiveUserData⁽¹³⁷⁾ is called in a thread or the main process context depending on SynchronizedReceiveUserData⁽¹²⁷⁾ property.


New property FilterSystemCmd⁽¹²³⁾.

Media server⁽¹⁶⁵⁾


You can send a command to users from a server: new SendCommandToGUID⁽¹⁷²⁾ method.

In OnServerCmd⁽¹⁷³⁾ you can process not only server commands, but commands that clients send to each other, if you turn off FilterUserCmd⁽¹⁶⁸⁾. Parameters of OnServerCmd⁽¹⁷³⁾ are changed .

ConnectionProperties property is split into two properties: SenderConnectionProperties and ReceiverConnectionProperties⁽¹⁷⁰⁾ .

OnServerCmd⁽¹⁷³⁾, OnUserConnect, OnUserDisconnect⁽¹⁷⁴⁾, OnDataRead⁽¹⁷²⁾ events are called in a thread context .

Microphone and sound

TRVMicrophone⁽¹¹⁷⁾ has new properties defining sound quality: SamplesPerSec and BitsPerSample⁽¹⁷⁹⁾. TRVMicrophone can read sound from a WAV-file instead of a microphone (see SourceType⁽¹⁸²⁾ property and new properties and events related to WAV-files). Default value for SoundMinLevel⁽¹⁸¹⁾ is changed to 10 . You can also change sound buffer size: BufferDuration⁽¹⁷⁹⁾.

TRVCamReceiver now can mix sound from different sources. New properties: Volume⁽¹²⁸⁾ and Mute⁽¹²⁴⁾.

TRVMicrophoneView⁽¹¹⁷⁾ now can display sound activity not only from TRVMicrophone, but also from TRVCamReceiver: new properties RecieverSource and GUID⁽¹⁸⁹⁾.

TRVCamMultiView⁽⁷⁴⁾ now can display audio viewers next to video viewers. New property: AudioSource⁽⁷⁶⁾, Viewers[].AudioViewer and Viewers[].AlignAudioViewer⁽⁸²⁾.

Localization

New properties: TRVCamView.Language⁽⁹⁹⁾, TRVCamMultiView.Language⁽⁸⁰⁾, TRVTrafficMeter.Language⁽¹⁷⁶⁾.

64 bit

Both Win32 and Win64 projects are supported in this version.

Other

TRVCamera.PlayVideoFile⁽⁶⁷⁾ uses FramePerSec⁽¹⁹⁰⁾ property.

▼ Version 1.1 (GStreamer)


GStreamer


TRVCamera⁽⁴²⁾ supports of **GStreamer** SDK to play H.264 via RTSP and MPEG-4 via HTTP. New values for TRVCamera.DeviceType⁽⁴⁹⁾. New method IsSupportedGStreamer⁽⁶⁶⁾, new property UseGStreamer.


2 Components

The package includes the following VCL/LCL/FireMonkey components.


Video


 TRVCamera⁽⁴²⁾ – a component allowing to receive a video stream from the camera (and other sources) and configure it.

 TRVCamControl⁽³⁹⁾ – a visual component allowing to control the camera movement.


 TRVCamView⁽⁹³⁾ – a visual component displaying video from TRVCamera or TRVCamReceiver.


 TRVCamMultiView⁽⁷⁴⁾ – a visual component displaying videos from multiple sources.


 TRVCamRecorder⁽⁸⁵⁾ – a component for recording audio and video files from TRVCamera (maybe with TRVCamSound⁽¹¹⁵⁾), TRVMicrophone⁽¹¹⁷⁾ or TRVCamReceiver⁽¹²⁰⁾.


 TRVWebCamDialog⁽¹⁰⁷⁾ – a component for displaying a dialog window allowing to change USB webcam properties

Sound


 TRVMicrophone⁽¹¹⁷⁾ – a component for reading sound from a microphone (and from uncompressed WAV files)


 TRVCamSound⁽¹¹⁵⁾ – a component for reading sound from videos that are received by TRVCamera⁽⁴²⁾


 TRVMicrophoneView⁽¹¹⁷⁾ – a visual component showing a microphone (our other audio source) activity.

 TRVAudioPlayer⁽¹¹⁰⁾ – a component for playing and recording sound from TRVMicrophone⁽¹¹⁷⁾, TRVCamSound⁽¹¹⁵⁾ or TRVCamReceiver⁽¹²⁰⁾.

Network

 TRVCamSender⁽¹³⁹⁾ – a component for sending video (from TRVCamera or TRVCamReceiver) and/or audio (from TRVMicrophone or TRVCamReceiver) to TRVCamReceiver or TRVMediaServer via the network.


 TRVCamReceiver⁽¹²⁰⁾ – a component for receiving video and audio (from TRVCamSender or TRVMediaServer) via the network.


 TRVMediaServer⁽¹⁶⁵⁾ – a component for sending data (video, audio, commands, files) from multiple TRVCamSenders to multiple TRVCamReceivers via the network.


 TRVTrafficMeter⁽¹⁷⁵⁾ – a component for displaying traffic charts.

2.1 Video


Video


 TRVCamera⁽⁴²⁾ – a component allowing to receive a video stream from the camera (and other sources) and configure it.

 TRVCamControl⁽³⁹⁾ – a visual component allowing to control the camera movement.

 TRVCamView⁽⁹³⁾ – a visual component displaying video from TRVCamera or TRVCamReceiver.

 TRVCamMultiView⁽⁷⁴⁾ – a visual component displaying videos from multiple sources.

 TRVCamRecorder⁽⁸⁵⁾ – a component for recording audio and video files from TRVCamera (maybe with TRVCamSound⁽¹¹⁵⁾), TRVMicrophone⁽¹¹⁷⁾ or TRVCamReceiver⁽¹²⁰⁾.

 TRVWebCamDialog⁽¹⁰⁷⁾ – a component for displaying a dialog window allowing to change USB webcam properties

2.1.1 TRVCamControl

TRVCamControl controls the camera movement (if the camera supports a rotation and the user has enough rights to operate the camera).

Unit [VCL and LCL] MRVCamControl;

Unit [FMX] fmxMRVCamControl;

Syntax

```
TRVCamControl = class (TCustomControl)
```

▼ Hierarchy

VCL and LCL:

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl

FMX:

TObject
TPersistent
TComponent

TFmxObject
TControl

Description

The component has 5 main active areas working like push buttons: move left, move up, move right, move down, move to the home position.

(there are 4 additional active areas between arrows: for example, you can move to the top and up direction at the same time).



This component is optional: you can control the camera movement directly in TRVCamView⁹³ /TRVCamMultiView⁷⁴ components, or implement your own user interface by calling TRVCamera⁴².Move***⁶⁶ methods.

To move the camera to the home position, click at the center button.

To move the camera in the specified direction, push the corresponding arrow and hold the mouse button (the longer you hold it, the further the camera moves).

How to use

Option 1: Create TRVCamControl component and assign it to CameraControl property of TRVCamera⁴² component.

Option 2: Create TRVCamControl component and assign it to CameraControl property of TRVCamMultiView⁷⁴ component. In this mode, the component controls the movement of the camera from the selected view.

2.1.1.1 Properties

In TRVCamControl

- ArrowColor⁴¹
- ArrowColorClicked⁴¹
- ArrowColorHot⁴¹
- ArrowLineColor⁴¹
- BorderColor⁴¹
- DiskBrightness⁴¹
- DiskColor⁴¹
- ShowFocus⁴²

Inherited from TCustomControl

- Color⁴¹
- ParentColor³⁹

2.1.1.1.1 TRVCamControl's colors

Colors

```

property Color: TRVMColor244;
property BorderColor: TRVMColor244;
property ArrowLineColor: TRVMColor244;
property ArrowColor: TRVMColor244;
property ArrowColorClicked: TRVMColor244;
property ArrowColorHot: TRVMColor244;
property DiskColor: TRVMColor244;
property DiskBrightness: Integer;

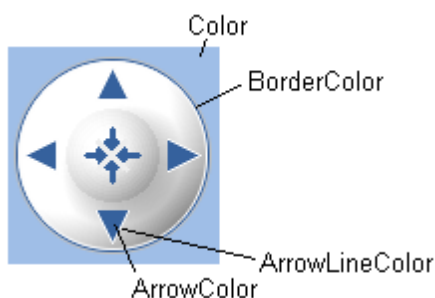
```

VCL and LCL:

```

property ParentColor: Boolean;

```



Color defines the background color (around the control). If **ParentColor**=*True*, the background is not painted, so the control is transparent.

BorderColor is a color of the bordering circle.

ArrowLineColor is a color of arrows' borders.

ArrowColor, **ArrowColorClicked**, **ArrowColorHot** are colors of arrows: normal, clicked, below the mouse pointer respectively.

DiskColor and **DiskBrightness** define the color of the control itself (disk). **DiskColor** defines hue, **DiskBrightness** defines brightness (the larger the brighter).

When activating Delphi XE2+ styles, system colors are changed to the corresponding style colors.

Default values [VCL and LCL]

- ParentColor: *True*
- BorderColor: \$009D6135
- ArrowLineColor: *c/White*
- ArrowColor: \$009D6135
- ArrowColorClicked: \$000E94DC
- ArrowColorHot: \$00DDA67D
- DiskColor: \$007F7F7F
- DiskBrightness: 100

Default values [FMX]

- BorderColor: \$FF35619D
- ArrowLineColor: *TAlphaColorRec.White*
- ArrowColor: \$FF35619D

- ArrowColorClicked: \$FFDC940E
- ArrowColorHot: \$FF7DA6DD
- DiskColor: \$FF7F7F7F
- DiskBrightness: 100

2.1.1.1.2 TRVCamControl.ShowFocus

Specifies whether the component should draw a dotted circle when focused.

property ShowFocus: Boolean;

Default value

True

2.1.2 TRVCamera

TRVCamera works with cameras: searches, configures, receives a video stream, saves or plays a video file.

Unit [VCL and LCL] MRVCamera;

Unit [FMX] fmxMRVCamera;

Syntax

TRVCamera = **class** (TRVVideoSource¹⁹⁰)

▼ **Hierarchy**

TObject

TPersistent

TComponent

*TRVMediaSource*¹⁸⁹

*TRVVideoSource*¹⁹⁰

Description

A video received by TRVCamera can be displayed in TRVCamView⁹³ or TRVCamMultiView⁷⁴ components. The camera movement is controlled by TRVCamControl³⁹ component or by the viewer itself.

TRVCamera can be assigned as a VideoSource to TRVCamSender¹³⁹.

You can record video from TRVCamera using TRVCamRecorder⁸⁵.

TRVCamera can receive video from the following sources:

- IP camera from the network (**MJPEG**, updated JPEG, set of JPEG files updated in a cycle, **H.264 streams****)
- **RTSP*** (MJPEG streams, H.264 streams, AVI and MP4 files containing H.264 and **MPEG-4 Part 2** video data)
- HTTP (MJPEG streams, H.264 streams*, AVI and MP4 files containing H.264 and MPEG-4 Part 2 video data*)
- USB web camera;
- screen (desktop);
- file (if the necessary codec is installed);

- video frames provided in an event.

You can choose the video source using `DeviceType`⁴⁹ property. After assigning necessary properties (depending on the source type), call `PlayVideoStream`⁶⁷.

Video can be recorded to a MJPEG file⁵⁰ and played from a MJPEG file⁶⁷.

* requires either **GStreamer** or **FFmpeg**

** requires **FFmpeg**

Sound

If video is received using **FFmpeg** (from IP camera, RTSP or HTTP video stream, or a local file), sound can be read by `TRVCamSound`¹¹⁵ component linked to this `TRVCamera` component.

If video is received from a local file using **DirectX**, sound is played directly on the default audio output device.

Lazarus note

In Lazarus, this component must have a windowed control as an owner: you can place it on a form, but you cannot place it on a datamodule.

If the component is created with `Owner = nil`, it assigns the main form as an owner.

2.1.2.1 Properties

In TRVCamera

- `Agent`⁴⁴
- `Bitrate`⁴⁵
- `Brightness`⁴⁵
- `CameraControl`⁴⁶
- `CameraHost`⁴⁶
- `CameraPort`⁴⁶
- `CameraSearchTimeOut`⁴⁶
- `CommandMode`⁴⁷
- `Contrast`⁴⁵
- `CycleVideoImages`⁴⁷
- `DeviceType`⁴⁹
- `DesktopForm`⁴⁷ [FMX]
- `DesktopMode`⁴⁷
- `DesktopRect`⁴⁷
- `DesktopWindowHandle`⁴⁷ [Windows]
- `DesktopZoomPercent`⁴⁷
- `FFMpegProperty`⁵⁰
- `FileName`⁵⁰
- `FlipHorizontally`⁵¹
- `FlipVertically`⁵¹
- `FocusDistance`⁵¹

- FocusType ⁽⁵¹⁾
- GStreamerProperty ⁽⁵²⁾
- Hue ⁽⁴⁵⁾
- ▶ IPCameraTypes ⁽⁵²⁾
- JpegIntegrity ⁽⁵²⁾
- Language ⁽⁵²⁾
- Latency ⁽⁵³⁾
- LoginPrompt ⁽⁵³⁾
- MaxCameraSearchThreadCount ⁽⁵³⁾
- ▶ MaxUsers ⁽⁵⁸⁾
- Parameters ⁽⁵⁴⁾
- ProxyProperty ⁽⁵⁵⁾
- RTSPPort ⁽⁴⁶⁾
- Quality ⁽⁵⁵⁾
- Rotation ⁽⁵⁵⁾
- Saturation ⁽⁴⁵⁾
- ▶ Searching ⁽⁵⁵⁾
- Sharpness ⁽⁴⁵⁾
- SmoothImage ⁽⁵⁶⁾
- ToFile ⁽⁵⁰⁾
- URL ⁽⁵⁷⁾
- UserAccess ⁽⁵⁷⁾
- UserName ⁽⁵⁷⁾
- UserPassword ⁽⁵⁷⁾
- Users ⁽⁵⁸⁾
- ▶ VideoDeviceCount ⁽⁵⁸⁾
- VideoDeviceIndex ⁽⁵⁸⁾
- ▶ VideoDeviceIdList ⁽⁵⁸⁾
- ▶ VideoDeviceList ⁽⁵⁸⁾
- VideoFormat ⁽⁵⁹⁾
- VideoImagesURLs ⁽⁴⁷⁾
- VideoMode ⁽⁶⁰⁾
- VideoResolution ⁽⁶¹⁾

Inherited from TRVVideoSource ⁽¹⁹⁰⁾

- ▶ Aborting ⁽¹⁹⁰⁾
- FramePerSec ⁽¹⁹⁰⁾
- ▶ FramePerSecInt ⁽¹⁹⁰⁾

2.1.2.1.1 TRVCamera.Agent

This property is send as a user-agent field when connecting using HTTP protocol.

```
property Agent: String;
```

This property is optional.

Default value

" (empty string)

2.1.2.1.2 TRVCamera.Bitrate

Returns and allows changing video bitrate

property Bitrate: Integer;

Property	Supported by
• Bitrate	DeviceType ⁴⁹ = <i>rvdtRTSP</i> and <i>rvdtHTTP</i> , assigns GStreamer's bitrate, range 0..2097152 (0 means autocalculated bitrate) DeviceType ⁴⁹ = <i>rvdtIPCamera</i> , MJPEG Foscam cameras, ranges: for MJPEG 1200..115200, for H.264 20480..2097152

To apply these values, change them when TRVCamera is connected to a capable device.

If the camera supports these properties, their values are received from the camera when the component is connected to it, see SearchCamera⁶⁸.

Default value

- 2097152

(however, the value is not applied and is overridden when connected to the camera)

2.1.2.1.3 TRVCamera.Brightness, Contrast, Hue, Saturation, Sharpness

Returns and allows changing color properties for the current video source.

property Brightness: Integer;

property Contrast: Integer;

property Hue: Integer;

property Saturation: Integer;

property Sharpness: Integer;

A range of allowed values is returned by GetColorControlPropertyRange⁶⁵ method.

Property	DeviceType ⁴⁹			
	<i>rvdtIPCamera</i>	<i>rvdtWebCamera</i>	<i>rvdtDesktop</i>	Others
• Brightness	Supported for some Foscam, Axis and Panasonic cameras (or compatible)	supported, if the camera supports this property	Supported by RVMedia itself	—
• Contrast	Supported for some			
• Hue • Saturation • Sharpness	Foscam cameras (or compatible)		—	

See also

- ResetImageSetting⁶⁸

2.1.2.1.4 TRVCamera.CameraControl

Specifies a TRVCameraControl component used to control the camera movement (if the current camera supports it).

property CameraControl: TRVCamControl⁽³⁹⁾;

If this component is assigned to VideoSource of one of viewers in TRVCamMultiView⁽⁷⁴⁾, this property is maintained by TRVCamMultiView⁽⁷⁴⁾, see TRVCamMultiView.CameraControl⁽⁷⁶⁾ property.

2.1.2.1.5 TRVCamera.CameraHost, CameraPort

The properties specify the IP camera's host and port.

property CameraHost: String;

property CameraPort: Word;

property RTSPPort: Word;

These properties are used only if DeviceType⁽⁴⁹⁾=rvdtIPCamera.

If camera's video protocol is **RTSP**, it may use different ports for commands and for video stream. In this case, specify the port for commands in **CameraPort**, and the port for video in **RTSPPort**.

CameraPort/RTSPPort is also used together with URL⁽⁵⁷⁾ property.

Default values:

- CameraHost: "" (empty string)
- CameraPort: 0 (= INTERNET_INVALID_PORT_NUMBER, using the protocol-specific default)
- RTSPPort: 0 (= INTERNET_INVALID_PORT_NUMBER, using the protocol-specific default)

If CycleVideoImages⁽⁴⁷⁾=True, a video from the IP camera is read from the files specified in VideoImagesURL⁽⁴⁷⁾.

To play video from the camera, first call SearchCamera⁽⁶⁸⁾, then PlayVideoStream⁽⁶⁷⁾.

Note: if SearchCamera⁽⁶⁸⁾ finds the camera, and this camera is not a controllable camera manufactured by Foscam, Axis, D-Link or Panasonic, CameraHost and CameraPort properties are cleared, and the address of video stream is assigned to URL⁽⁵⁷⁾.

See also

- URL⁽⁵⁷⁾
- UserName, UserPassword⁽⁵⁷⁾

2.1.2.1.6 TRVCamera.CameraSearchTimeout

Specifies the time-out interval for camera searching, in milliseconds.

property CameraSearchTimeout: Word;

A lesser value makes searching faster, but increases a chance of overlooking cameras.

Default value

3000 (i.e. 3 seconds)

See also

- SearchCamera⁽⁶⁸⁾

2.1.2.1.7 TRVCamera.CommandMode

Defines the mode how the component sends commands to the camera

type

```
TRVSendMode = (rvsmWait, rvsmNoWait); // defined in MRVType unit
```

property CommandMode: TRVSendMode

Value	Meaning
<i>rvsmWait</i>	The component waits for the commands to complete
<i>rvsmNoWait</i>	The component does not wait for the commands. When commands are finished, events are called.

Default value

rvsmWait

See also

- Example: Working in a wait mode ⁽²²⁸⁾
- Example: Working in a no-wait mode ⁽²²⁹⁾

2.1.2.1.8 TRVCamera.CycleVideoImages, VideoImagesURLs

The properties allow using a list of file names for downloading video frames.

property CycleVideoImages: Boolean;

property VideoImagesURLs: TStringList;

These properties are used only if DeviceType ⁽⁴⁹⁾ = *rvdtIPCamera* or *rvdtHTTP*.

If CycleVideoImages=*True*, video frames are read cyclically from the locations specified in VideoImagesURL. These locations are added to the address specified in URL ⁽⁵⁷⁾.

Default values

- CycleVideoImages: *False*
- VideoImagesURLs: empty string-list

2.1.2.1.9 TRVCamera.DesktopMode, DesktopRect, ...

These properties specify a part of desktop for encoding into a video.

type

```
// defined in MRVType unit
```

```
TRVDesktopMode = (rvdmFull, rvdmRect, rvdmWindow);
```

property DesktopMode: TRVDesktopMode;

property DesktopRect: TRVMRect ⁽²⁴⁶⁾;

property DesktopZoomPercent: Integer;

```
// only for Windows and macOS
```

property DesktopWindowHandle: TRVMWindowHandle ⁽²⁴⁷⁾;

```
// only for FireMonkey
```

property DesktopForm: TCustomForm;

These properties are used if DeviceType⁽⁴⁹⁾=*rvdtDesktop*.

VCL, Lazarus for Windows, FireMonkey for Windows and macOS

A source region of screen is chosen depending in DesktopMode property.

DesktopMode	Meaning
<i>rvdmFull</i>	A whole desktop or a whole monitor is used as a source, see VideoDevice*** ⁽⁵⁸⁾ properties.
<i>rvdmRect</i>	A rectangle defined in DesktopRect is used
<i>rvdmWindow</i>	A rectangle of the window specified in DesktopWindowHandle is used. If DesktopWindowHandle =0, the main form is used.

macOS note: the current implementation supports only the primary monitor.

If you want to use other application's window as a source, you can use `GetVisibleWindowsHandles`⁽²¹⁸⁾ to get possible values of **DesktopWindowHandle**.

FireMonkey

FireMonkey version of RVMedia supports all DesktopMode options only for Windows and macOS platforms.

On other platforms, it can stream only forms of this application. DesktopMode must be *rvdmWindow*. The source form is specified in **DesktopForm**. On Windows and macOS platforms, if both **DesktopWindowHandle** and **DesktopForm** are defined, **DesktopForm** is chosen for streaming.

When streaming, RVMedia draws **DesktopForm** onto the bitmap (instead of streaming a screen area covered by this form).

If **DesktopWindowHandle** and **DesktopForm** are not defined, the main form is used.

Common

DesktopZoomPercent scales the resulting video frames. For example, DesktopZoomPercent=100 leaves frames unchanged (100% size), DesktopZoomPercent=50 shrinks its width and height to 50%. Assign values in the range 1..99 to reduce frame size and thus to reduce traffic. It is also possible to assign values larger than 100 to make frames larger, but it makes no sense.

Default value

DesktopMode: *rvdmFull*

DesktopWindowHandle: 0

DesktopForm: *nil*

DesktopZoomPercent: 100

See also

- DesktopVideoMode methods⁽⁶³⁾

2.1.2.1.10 TRVCamera.DeviceType

Specifies the source video device type.

type

```
// defined in MRVType unit
```

```
TRVDeviceType = (rvdtIPCamera, rvdtWebCamera, rvdtDesktop,  
rvdtFile, rvdtRTSP, rvdtHTTP, rvdtUserData);
```

property DeviceType: TRVDeviceType;

Value	Meaning	GStreamer	FFmpeg
<i>rvdtIPCamera</i>	IP Camera producing MJPEG or H.264 video stream via HTTP or TCP. CameraHost:CameraPort ⁽⁴⁶⁾ or URL ⁽⁵⁷⁾ properties are used. Video format is specified in VideoFormat ⁽⁵⁹⁾ .	not used	not required for MJPEG; required otherwise
<i>rvdtWebCamera</i>	USB webcam. VideoDeviceIndex ⁽⁵⁸⁾ property is used.	not used	
<i>rvdtDesktop</i>	Screen or window. DesktopMode ⁽⁴⁷⁾ property is used.	not used	
<i>rvdtFile</i>	Video from a file. SourceFileName ⁽⁵⁶⁾ property is used.	not used	used optionally**
<i>rvdtRTSP</i>	Video via RTSP (Real Time Streaming Protocol)*. URL ⁽⁵⁷⁾ property is used. Video format is specified in VideoFormat ⁽⁵⁹⁾ *.	either GStreamer or FFmpeg is required	
<i>rvdtHTTP</i>	Video via HTTP*. URL ⁽⁵⁷⁾ property is used. Video format is specified in VideoFormat ⁽⁵⁹⁾ *.	not required for MJPEG; either GStreamer or FFmpeg is required otherwise	
<i>rvdtUserData</i>	Video provided by the application, using OnNewImage ⁽⁷²⁾ event	not used	

* the specified video format and network protocol are used in GStreamer; for FFmpeg, they are detected automatically.

** local files can be played either using FFmpeg or DirectX. DirectX can be used only in VCL and LCL for Windows. For FireMonkey and other OS, FFmpeg is necessary.

If FFmpeg or GStreamer 1.0 are used, *rvdtRTSP* and *rvdtHTTP* are processed identically: the real protocol (rtsp, http, https or udp) is autodetected. There is only one difference: for *rvdtRTSP*, the default port (which is used if not specified in URL) is RTSPPort⁽⁴⁶⁾.

If GStreamer 0.1 is used, *rvdtRTSP* and *rvdtHTTP* must strictly correspond to the stream protocol.

For GStreamer 1.0, RVMedia can display UDP stream received by the given computer. You just need to specify the port to listen. The URL format must be 'udp://:5600' (for listening the port 5600), or simply 'udp://', if the port is specified in CameraPort or RTSPPort⁽⁴⁶⁾. The supported stream formats are H.264 or MJpeg (defined in VideoFormat⁽⁵⁹⁾ property).

For GStreamer 1.0, you can specify your own GStreamerProperty⁽⁵²⁾.LaunchString⁽²⁰⁴⁾, if you need a protocol or format that cannot be defined in DeviceType and VideoFormat properties.

Default value

rvidtIPCamera

2.1.2.1.11 TRVCamera.FFMpegProperty

Properties to configure FFMpeg⁽²⁴⁾.

property FFMpegProperty: TRVFFMpegProperty⁽²⁰¹⁾;

You can check the presence of FFMpeg using IsSupportedFFMPEG⁽⁶⁵⁾ function.

The main property is FFMpegProperty.UseFFMpeg, it turns FFMpeg support on/off.

You may wish to disable FFMpeg in the following cases:

- if you want to use GStreamer instead of FFMpeg to play HTTP or RTSP video streams;
- if you want to use native RVMedia JPEG/MJPEG decoding. Some cameras provide videos in format of updated JPEG pictures. FFMpeg may thought that this is a static picture, so it stops after the first frame.
- if you want to use DirectX instead of FFMpeg to play local files (only VCL and LCL for Windows); DirectX plays videos with sound.

Default value

True

See also:

- DeviceType⁽⁴⁹⁾

2.1.2.1.12 TRVCamera.FileName, ToFile

The properties allow to record a video to the file (in MJPEG format)

property ToFile: Boolean;

property FileName: String;

If ToFile=*True*, a video from the camera is recorded to the file FileName.

Default values

ToFile: False

FileName: "" (empty string)

2.1.2.1.13 TRVCamera.FlipHorizontally, FlipVertically

The properties allow flipping video horizontally and/or vertically, if the camera supports these settings.

property FlipHorizontally: Boolean;

property FlipVertically: Boolean;

If the camera supports these properties, their values are received from the camera when the component is connected to it, see SearchCamera⁶⁸.

Default values:

False

2.1.2.1.14 TRVCamera.FocusType, FocusDistance

The properties control the camera focus, if the camera supports it.

type

// defined in MRVType unit

TRVCamFocus = (rvcfFocusAuto, rvcfFocusNear, rvcfFocusFar);

property FocusType: TRVCamFocus;

property FocusDistance: Integer;

For IP cameras

These properties are supported for Panasonic cameras (see SearchCamera⁶⁸). These are write-only properties: RVMedia cannot read these properties from the camera, but can assign them.

FocusType:

Value	Meaning
<i>rvcfFocusAuto</i>	automatic focus
<i>rvcfFocusNear</i>	focus near
<i>rvcfFocusFar</i>	focus far

FocusDistance is used only if FocusType<>*rvcfFocusAuto*. Possible values:

Value	Meaning
1	small displacement
2	large displacement

For local web cameras

FocusType is not supported

FocusDistance can be get and set (for cameras that support it). Minimum and maximum values depend on the driver.

Default values

- FocusType: *rvcfFocusAuto*
- FocusDistance: 1

2.1.2.1.15 TRVCamera.GStreamerProperty

Properties to configure **GStreamer**

property GStreamerProperty: TRVGStreamerProperty⁽²⁰⁴⁾;

You can check the presence of GStreamer using IsSupportedGStreamer⁽⁶⁶⁾ function.

The main property is GStreamerProperty.UseGStreamer, it turn the GStreamer support on/off

Default value

True

See also:

- DeviceType⁽⁴⁹⁾

2.1.2.1.16 TRVCamera.IPCameraTypes

When connected to an IP camera (after a camera searching⁽⁶⁸⁾ is complete), returns the camera model.

property IPCameraTypes: TRVCameraTypes⁽²³⁸⁾;

If the value is one of [rvctFoscam], [rvctPanasonic], [rvctAxis], [rvctDLink], TRVCamera supports not only displaying, but configuring and controlling the camera movement (if the camera allows it). Even if the found camera is created by another manufacturer but supports the protocol of these camera models, it will be detected as [rvctFoscam], [rvctPanasonic], [rvctAxis] or [rvctDLink], and can be controlled in the same way. For example, Samsung cameras supporting the Axis protocol will be detected as [Axis]. More specific camera model can be read from Parameters⁽⁵⁴⁾.Alias property.

Many manufactures creates clones of cameras of manufactures listed in TRVCameraTypes, or cameras having compatible interface. These cameras will be detected too.

If this property contains multiple values, this means that these camera models have identical address of MJPEG stream, so TRVCamera is not sure which manufacturer created this camera.

In any case, TRVCamera does not guarantee that the camera is really created by a manufacturers listed in this property.

2.1.2.1.17 TRVCamera.JpegIntegrity

Specifies how received jpeg images (video frames) are checked

property JpegIntegrity: TRVJpegIntegrity⁽²⁴³⁾;

Default value

rvjiNone

2.1.2.1.18 TRVCamera.Language

A user interface language for the login prompt dialog.

property Language: TRVMLanguage⁽²⁴⁵⁾;

This dialog is displayed if LoginPrompt⁽⁵³⁾ = *True* and OnLogin event is not assigned.

Default value*rvmlEnglish***2.1.2.1.19 TRVCamera.Latency**

Maximal allowed time interval (in milliseconds) between the moments when a video frame is received and when it is shown.

property Latency: Integer;

If this property has a positive value, TRVCamera accumulates received frames and shows them with a delay. This feature allows displaying frames at a regular interval.

If this property's value is zero, frames are displayed as soon as possible, but more frames can be dropped, if they come at irregular intervals and cannot be displayed fast enough. On the other hand, this option reduces the overall overhead (because of skipping buffering steps), so it may increase a display frame rate.

Default value*0***2.1.2.1.20 TRVCamera.LoginPrompt**

Specifies whether the component should request a user name and a password from the user (if authentication failed).

property LoginPrompt: Boolean;

If the specified video source (IP camera, HTTP or RTSP stream⁽⁴⁹⁾) needs authentication, the component uses a user name and a password specified in URL⁽⁵⁷⁾, or in UserName and UserPassword⁽⁵⁷⁾ and UserPassword⁽⁵⁷⁾ properties.

If authentication fails, and **LoginPrompt** = *True*, the component requests new user name and password from the user. If OnLogin⁽⁷¹⁾ event is assigned, it is called; otherwise the login prompt dialog is displayed. If authentication still fails, this process is repeated several times.

A language of the login prompt dialog is specified in the Language⁽⁵²⁾ property.

If authentication finally fails, the following events occur: OnLoginFailed⁽⁷²⁾; OnEndVideoStream⁽⁷⁰⁾ event with an error code.

Default value*False***2.1.2.1.21 TRVCamera.MaxCameraSearchThreadCount**

Specifies the maximum count of simultaneously working IP-camera searching threads.

property MaxCameraSearchThreadCount: Word;

Higher values make searches faster, but they consume more system resources. Also, there is a possibility that a remote server may decide that it is under attack and reject connections.

This count does not include threads for searching Axis, D-Link, Foscam and Panasonic (or compatible) cameras. RVMedia has special support for these cameras, and they are searched in a special way.

The actual count of created thread is $\min(\text{MaxCameraSearchThreadCount}, \text{glRVInetMaxConnect}^{(216)})$.

Default value

40

See also

- SearchCamera⁽⁶⁸⁾

2.1.2.1.22 TRVCamera.Parameters

This property contains sub-properties returning information about the IP-camera (if the camera supports it) and allowing to change them (if the camera supports it).

property Parameters: TRVCameraParameters;

TRVCameraParameters includes the following properties:

- ID: String;
- Version: String;
- Alias: String;*
- Network: TRVCamNetworkProperties;
- DateTimeParameter: TRVDateTimeParameter;
- WirelessLan: TRVWirelessLan;
- ADSL: TRVADSL;
- UPnPtoMapPort: Boolean;
- MailService: TRVMailService;
- FtpService: TRVFtpService;
- AlarmService: TRVAlarmService;
- PTZSettings: TRVPTZSettings;
- Decoder: TRVDecoder;

* Note about Alias property

Alias is assigned:

- for IP cameras: by SearchCamera⁽⁶⁸⁾ method. If the found camera's API allows it, Alias receives the camera name. Otherwise, it is assigned to the name of the found group of camera models (which is not necessary correct, because different camera models may have the same video URL)
- for local web cameras: when changing a value of VideoDeviceIndex⁽⁵⁸⁾, Alias receives the name of the chosen local camera
- for desktop: when changing a value of VideoDeviceIndex⁽⁵⁸⁾, Alias receives the description of the chosen display.

If you assign property AutoUpdateAlias = False, Alias will not be updated automatically, so it keeps the value that was assigned to it.

Alias can be displayed in captions of video viewers, see TRVCamView⁽⁹³⁾.CaptionParts⁽¹⁰¹⁾ property.

TRVCameraParameters includes the following read-only properties:

- CameraHost: String;
- IPCameraTypes: TRVCameraTypes;
- DeviceType: TRVDeviceType;

- `UserAccess`: `TRVUserAccess`;
- `IPCameraTypesStr`: `String`;
- `UserAccessStr`: `String`;

For detailed information about these properties, see the demo

- **Demos\Cameras\MediaTest_Delphi** (for Delphi)
- **Demos\Cameras\MediaTest_Lazarus** (for Lazarus)

2.1.2.1.23 TRVCamera.ProxyProperty

This property contains sub-properties for proxy settings.

property `ProxyProperty`: `TRVProxyProperty`²¹¹;

2.1.2.1.24 TRVCamera.Quality

Defines the video quality preference, if the camera supports it.

type

// defined in MRVType unit

`TRVQualityType = (qtStandard, qtFavorMotion, qtFavorClarity);`

property `Quality`: `TRVQualityType`;

Value	Meaning
<i>qtStandard</i>	standard quality
<i>qtFavorMotion</i>	optimizing for a fast motion
<i>qtFavorClarity</i>	optimizing for a quality static video frames

If the camera supports this property, its value is received from the camera when the component is connected to it, see `SearchCamera`⁶⁸.

Default value

qtStandard

2.1.2.1.25 TRVCamera.Rotation

Allows to rotate video frames before by 90, 180 or 270 degrees.

type

// defined in MRVType unit

`TRVRotation = (rvrNone, rvr90, rvr180, rvr270);`

property `Rotation`: `TRVRotation`;

Default value

rvrNone

2.1.2.1.26 TRVCamera.Searching

Returns *True* is the component is searching for the camera

property `Searching`: `Boolean`; *// read-only*

See also

SearchCamera⁽⁶⁸⁾

2.1.2.1.27 TRVCamera.SmoothImage

Smooths video frames by creating images from several last received video frames.

property SourceFileName: **String**;

Experimental property.

If *True*, video frames are smoothed by creating an image from several (3) last received frames.

Pros: removes noise in images, especially if lighting is insufficient.

Contras: blurs moving objects.

We do not recommend using this mode for videos containing fast-changing timers, or if the camera has a low frame rate (see FramePerSec⁽¹⁹⁰⁾).

Default value

False

See also:

- TRVCamReceiver⁽¹²⁰⁾.SmoothImage⁽¹²⁶⁾

2.1.2.1.28 TRVCamera.SourceFileName

Specifies the source video file name.

property SourceFileName: **String**;

The component gets video from the specified file if DeviceType⁽⁴⁹⁾=*rvdtFile*.

All platforms

If FFMpegProperty.UseFFMPEG⁽⁵⁰⁾=*True*, and **FFmpeg** is available, the component uses FFmpeg to play this file. In this case, video sound can be read by TRVCamSound⁽¹¹⁵⁾ component linked to this TRVCamera (and then played or recorded using TRVAudioPlayer⁽¹¹⁰⁾ component or sent by TRVCamSender⁽¹³⁹⁾ component).

Windows

If FFmpeg is disabled or not available, the component uses DirectX to play file. The system must have a codec for this file installed. If the video contains sound, it is played on the default audio device.

Linux

FFmpeg is required to play video file.

macOS

If FFmpeg is disabled or not available, the component uses AVFoundation to play file.

If the video contains sound:

- if TRVCamSound⁽¹¹⁵⁾ component linked to this TRVCamera, and it has TRVAudioPlayer⁽¹¹⁰⁾ component assigned AudioOutput⁽¹⁸⁸⁾, sound is played on the device selected in this TRVAudioPlayer⁽¹¹⁰⁾ (however, TRVAudioPlayer⁽¹¹⁰⁾ itself is not used to play sound)
- otherwise, sound is played on the default audio device.

Note: with AVFoundation, you can play video not only from a local file but also from an URL. So, this mode may be useful to play Internet streams when FFmpeg and GStreamer are not available.

2.1.2.1.29 TRVCamera.URL

Specifies the address of a video stream from an IP camera (or another video source available in the Internet)

property URL: String;

This property is used if DeviceType⁽⁴⁹⁾=*rvdtIPCamera* and CameraHost⁽⁴⁶⁾ is unassigned, and if DeviceType⁽⁴⁹⁾=*rvdtRTSP*, *rvdtHTTP*.

Avoid including user name, password and port in this property. Use CameraPort/RTSPPort⁽⁴⁶⁾, UserName, UserPassword⁽⁵⁷⁾ instead. However, if user name and password are specified in **URL**, they will be used. If authentication fails, the component can request a new username and a password, if LoginPrompt⁽⁵³⁾ = *True*.

This property may be assigned automatically as a result of camera searching⁽⁶⁸⁾.

To play video from the camera, call PlayVideoStream⁽⁶⁷⁾.

Default value

" (empty string)

2.1.2.1.30 TRVCamera.UserAccess

When connected to a camera (after a camera searching⁽⁶⁸⁾ is complete), returns the access mode for the user UserName:UserPassword⁽⁵⁷⁾.

type // defined in MRVType/fmxMRVType unit

TRVUserAccess = (uaNone, uaPresent, uaVisitor, uaOperator, uaAdmin);

property UserAccess: TRVUserAccess; // read-only

2.1.2.1.31 TRVCamera.UserName, UserPassword

The properties define the user name and the password to access the camera.

property UserName: String;

property UserPassword: String;

The component can request a user name and a password from the user, if LoginPrompt⁽⁵³⁾ = *True*.

Default value

" (empty string)

See also

- UserAccess⁽⁵⁷⁾
- CameraHost⁽⁴⁶⁾
- URL⁽⁵⁷⁾

2.1.2.1.32 TRVCamera.Users

Allows managing users for an IP camera.

property Users: TRVCamUserCollection;

This property allows reading and modifying a list of IP camera users (the camera must support it, and admin rights are required).

The current DeviceType⁽⁴⁹⁾ must be *rvdtIPCamera*, and the camera must be found by SearchCamera⁽⁶⁸⁾.

If the camera supports this property (Users), its value is received from the camera when the component is connected to it, see SearchCamera⁽⁶⁸⁾. Some cameras provide a user list without passwords.

To check if the connected camera supports managing user list, check the presence of *rvcp_Users* in GetAccessibleCamProperties⁽⁶⁴⁾.

TRVCamUserCollection is a collection of TRVCamUser items. Each item has the properties:

- UserName: String
- Password: String;
- Access: TRVUserAccess

type // defined in MRVType unit

```
TRVUserAccess = (uaNone, uaPresent, uaVisitor, uaOperator, uaAdmin);
```

If connected to the camera, any change to this collection or to its items is sent to the camera as it is performed.

It may be inefficient for multiple changes, so TRVCamera implements AddUser and ModifyUser methods.

2.1.2.1.33 TRVCamera.VideoDeviceIndex, VideoDeviceCount, VideoDeviceList, VideoDeviceIdList

Properties controlling USB webcams and monitors (displays).

property VideoDeviceIndex: Integer;

property VideoDeviceCount: Integer; // read-only

property VideoDeviceList[Index: Integer]: String; // read-only

property VideoDeviceIdList[Index: Integer]: String; // read-only

These properties are used if DeviceType⁽⁴⁹⁾=*rvdtWebCamera* or *rvdtDesktop*. VideoDeviceIdList is used only for web cameras.

DeviceType⁽⁴⁹⁾=*rvdtWebCamera*

VideoDeviceCount returns the count of webcams (video capture devices).

VideoDeviceList[Index] (where Index is in the range 0..VideoDeviceCount-1) returns names of webcams. These names can be shown to users in the application UI.

VideoDeviceIdList[Index] (where Index is in the range 0..VideoDeviceCount-1) returns identifiers of webcams. These identifiers are unique on the computer.

Assign a value in the range 0..VideoDeviceCount-1 to VideoDeviceIndex to choose the webcam.

To play video from the camera, call `PlayVideoStream`⁽⁶⁷⁾.

DeviceType⁽⁴⁹⁾ = *rvdtDesktop*

`VideoDeviceCount` returns the count of monitors + 1.

The 0-th device corresponds to the whole desktop, the indexes from 1 to `VideoDeviceCount-1` correspond to monitors (the *i*-th device corresponds to `Screen.Monitors[i-1]`).

`VideoDeviceList[Index]` (where `Index` is in the range `0..VideoDeviceCount-1`) returns a description of a device (a desktop or a monitor).

Assign a value in the range `0..VideoDeviceCount-1` to `VideoDeviceIndex` to choose the desktop or a monitor as a video source (if `DesktopMode`⁽⁴⁷⁾ = *rvdmFull*)

To play video from the desktop, call `PlayVideoStream`⁽⁶⁷⁾.

[FMX note]: in FireMonkey, multiple monitors are supported since Delphi XE7. For Delphi XE6, `VideoDeviceCount` returns 1 and `VideoDeviceList[0]` returns the description of the primary monitor.

See also

- `FillVideoDeviceList`⁽⁶³⁾
- `CamVideoMode` methods⁽⁶³⁾
- `DesktopVideoMode` methods⁽⁶³⁾
- `TRVWebCamDialog`⁽¹⁰⁷⁾ component

2.1.2.1.34 TRVCamera.VideoFormat

Specifies a video format.

type

```
TRVVideoFormat = (rvvfMJPEG, rvvfH264, rvvfAVI_H264, rvvfMP4_H264,
  rvvfAVI_MPEG, rvvfMP4_MPEG); // defined in MRVType unit
```

property `VideoFormat`: `TRVVideoFormat`;

This property is used if `DeviceType`⁽⁴⁹⁾ = *rvdtHTTP*, *rvdtRTSP*, or *rvdtIPCamera*.

All video formats (except for MJPEG in *rvdtHTTP* and *rvdtIPCamera* modes) require either **GStreamer** or **FFmpeg**. They do nothing if they are not available or turned off.

- GStreamer: checking for availability⁽⁶⁶⁾, turning on/off⁽⁵²⁾;
- FFmpeg: checking for availability⁽⁶⁵⁾, turning on/off⁽⁵⁰⁾.

This property is taken into account in two cases:

- when playing video (`PlayVideoStream`⁽⁶⁷⁾)
- when searching for cameras (`SearchCamera`⁽⁶⁸⁾)

Value	Meaning	GStreamer usage	FFmpeg usage
<i>rvvfMJPEG</i>	MJPEG video stream	optional, if <code>DeviceType</code> ⁽⁴⁹⁾ = <i>rvdtHTTP</i> required, if <code>DeviceType</code> ⁽⁴⁹⁾ = <i>rvdtRTSP</i>	optional, if <code>DeviceType</code> ⁽⁴⁹⁾ = <i>rvdtHTTP</i> or <i>rvdtIPCamera</i> , required otherwise;

<i>rvvfH264</i>	H.264 video stream	required	the specified video format is ignored: it is autodetected
<i>rvvfAVI_H264</i>	Video from AVI file. Video must be encoded as H.264		
<i>rvvfMP4_H264</i>	Video from MP4 (or QuickTime) file Video must be encoded as H.264		
<i>rvvfAVI_MPEG</i>	Video from AVI file. Video must be encoded as MPEG-4 Part 2		
<i>rvvfMP4_MPEG</i>	Video from MP4 (or QuickTime) file Video must be encoded as MPEG-4 Part 2		

Note 1: MPEG-4 Part 2 and H.264 contain patented technologies, the use of which requires licensing in countries that acknowledge software algorithm patents.

Note 2: The main advantage of using GStreamer and FFmpeg is supporting H.264 video streams from cameras and other sources (*rvvfH264* video format). For displaying video files, we recommend downloading them to the local computer and use DeviceType⁽⁴⁹⁾=*rvdtFile*. It does not require GStreamer and can play any video file that can be played in Windows Media Player on this computer.

Default value

rvvfMJPEG

2.1.2.1.35 TRVCamera.VideoMode

Sets the camera mode to prevent a jitter impact on the image because of the electricity frequency, if supported by the camera.

type

```
TRVVideoMode = (vm50HZ, vm60HZ, vmOutdoor); // defined in MRVType unit
```

property VideoMode: TRVVideoMode;

Value	Meaning
<i>vm50HZ</i>	50 Hz, typically when the camera is used indoor
<i>vm60HZ</i>	60 Hz, typically when the camera is used indoor
<i>vmOutdoor</i>	Recommended if the camera is used outdoor

50/60 Hz modes should be set according to the electricity standard in your country.

If the camera supports this property, its value is received from the camera when the component is connected to it, see SearchCamera⁽⁶⁸⁾.

Default value*vm50HZ***2.1.2.1.36 TRVCamera.VideoResolution**

Changes the video resolution, if the camera supports it.

property VideoResolution: TRVVideoResolution⁽²⁵²⁾;

If the camera supports this property, its value is received from the camera when the component is connected to it, see SearchCamera⁽⁶⁸⁾.

If the current DeviceType⁽⁴⁹⁾ = *rvdtWebCamera*, the component sets video mode having width the most close to the specified in the resolution.

If the current DeviceType⁽⁴⁹⁾ = *rvdtIPCamera*, the component sets the camera video resolution, if the camera supports it (currently, it is implemented for Foscam cameras).

If the current DeviceType⁽⁴⁹⁾ = *rvdtRTSP* or *rvdtHTTP*, this property is not used. If GStreamer is used to play video, video can be scaled using GStreamerProperty⁽⁵²⁾.UseVideoScale⁽²⁰⁴⁾ and related properties. If FFmpeg is used to play video, video can be scaled using FFMpegProperty⁽⁵⁰⁾.UseVideoScale⁽²⁰¹⁾ and related properties.

Default value*rvDefault***See also**

- CamVideoMode methods⁽⁶³⁾ for USB web cameras
- TRVCamSender.VideoResolution⁽¹⁵⁴⁾

2.1.2.2 Methods**In TRVCamera**

AddUser⁽⁶²⁾
 FillVideoDeviceList⁽⁶³⁾
 GetAccessibleCamMethods⁽⁶⁴⁾
 GetAccessibleCamProperties⁽⁶⁴⁾
 GetAvailableCamMethods⁽⁶⁴⁾
 GetAvailableCamProperties⁽⁶⁴⁾
 GetCamCurrentVideoMode⁽⁶³⁾
 GetCamVideoMode⁽⁶³⁾
 GetCamVideoModeCount⁽⁶³⁾
 GetCamVideoModeIndex⁽⁶³⁾
 GetDesktopCurrentVideoMode⁽⁶³⁾
 GetDesktopVideoMode⁽⁶³⁾
 GetDesktopVideoModeCount⁽⁶³⁾
 GetDesktopVideoModeIndex⁽⁶³⁾
 GetSnapshot⁽⁶⁵⁾
 IsSupportedFFMPEG⁽⁶⁵⁾
 IsSupportedGStreamer⁽⁶⁶⁾
 LockCommands⁽⁶⁶⁾

ModifyUser⁽⁶²⁾
 Move***⁽⁶⁶⁾
 PlayVideoFile⁽⁶⁷⁾
 PlayVideoStream⁽⁶⁷⁾
 ResetImageSetting⁽⁶⁸⁾
 SearchCamera⁽⁶⁸⁾
 SetCamVideoMode⁽⁶³⁾
 SetDesktopVideoMode⁽⁶³⁾
 SwitchLEDOff⁽⁶⁹⁾
 SwitchLEDOn⁽⁶⁹⁾
 UnlockCommands⁽⁶⁶⁾
 WaitForVideo⁽⁶⁹⁾
 WaitForVideoFile⁽⁶⁹⁾
 WaitForVideoStream⁽⁶⁹⁾

Inherited from TRVVideoSource⁽¹⁹⁰⁾

Abort⁽¹⁹¹⁾
 GetOptimalVideoResolution⁽¹⁹¹⁾

2.1.2.2.1 TRVCamera.AddUser, ModifyUser

The methods make changes in Users⁽⁵⁸⁾ collection and apply changes to the IP camera.

```

procedure AddUser(const AUserName, APassword: String;
  AAccess: TRVUserAccess(58));
procedure ModifyUser(Index: Integer; const AUserName, APassword: String;
  AAccess: TRVUserAccess(58));
  
```

The current DeviceType⁽⁴⁹⁾ must be *rvdtIPCamera*, and the camera must be found by SearchCamera⁽⁶⁸⁾, the camera must support a list of users, and TRVCamera must be connected to this camera as an admin.

To check if the connected camera supports managing user list, check the presence of *rvcp_Users* in GetAccessibleCamProperties⁽⁶⁴⁾.

AddUser adds a new item Users⁽⁵⁸⁾ and assigns its properties, then applies changes to the camera.

ModifyUser changes properties of Users⁽⁵⁸⁾ [**Index**], then applies changes to the camera.

Each change in Users⁽⁵⁸⁾ collection is applied to the camera. When you add item to this collection, delete and item, assign a new value to item property, the corresponding command it sent to the connected IP camera (if the camera supports it).

However, it can be inefficient. For example, you can add a new item to the collection (one operation), sets its properties (3 operations). Instead, you can call AddUser that does it as a single operation.

2.1.2.2.2 TRVCamera.FillVideoDeviceList

Fills a list USB webcams or monitors (video capture and video display devices).

procedure FillVideoDeviceList(List: TStrings);

The method can be used if DeviceType⁽⁴⁹⁾=rvdtWebCamera or rvdtDesktop.

This method fills List with names of available webcams or monitors, depending on DeviceType properties (see VideoDeviceList⁽⁵⁸⁾ property)

2.1.2.2.3 TRVCamera.Get- SetCamVideoMode, etc.

A set of methods for managing video modes of local web cameras (i.e. local video capture devices).

```
function GetCamVideoModeCount: Integer;
function GetCamVideoModeIndex: Integer;
function GetCamCurrentVideoMode(
    var CamVideoMode: TRVCamVideoMode(239)): Boolean;
function SetCamVideoMode(const Index: Integer): Boolean;
function GetCamVideoMode(const Index: Integer;
    var VideoMode: TRVCamVideoMode(239)) : Boolean;
```

The methods can be used if DeviceType⁽⁴⁹⁾=rvdtWebCamera.

GetCamVideoModeCount returns the count of available modes for the current web camera.

GetCamVideoMode returns information about the specified mode (Index parameter must be in range 0..**GetCamVideoModeCount**-1). **SetCamVideoMode** changes the web camera mode to the Index-th mode (but it may fail for some modes).

GetCamCurrentVideoMode returns information about the current video mode.

GetCamVideoModeIndex returns the index of the current video mode (this method is inaccurate: it returns the index of the first video mode having TRVCamVideoMode matching the current video mode; however, video modes may include additional parameters, not included in TRVCamVideoMode record; they are ignored when finding the index).

Assigning a new value to VideoResolution⁽⁶¹⁾ may change the current video mode (the most similar video mode is selected). After calling **SetCamVideoMode**, VideoResolution⁽⁶¹⁾ is ignored until you assign a different value to this property. When initializing a web camera, a video mode is chosen according to VideoResolution⁽⁶¹⁾.

See also functions:

- DescribeVideoMode⁽²²⁸⁾
- DescribeVideoModePixelFormat⁽²²⁸⁾

2.1.2.2.4 TRVCamera.Get- SetDesktopVideoMode, etc.

A set of methods for managing primary monitor video modes.

```
function GetDesktopVideoModeCount: Integer;
function GetDesktopVideoModeIndex: Integer;
function GetDesktopCurrentVideoMode(
```

```

var DesktopVideoMode: TRVDesktopVideoMode(241): Boolean;
function SetDesktopVideoMode(const Index: Integer): Boolean;
function GetDesktopVideoMode(const Index: Integer;
    var VideoMode: TRVDesktopVideoMode(241)): Boolean;

```

The methods can be used if DeviceType⁽⁴⁹⁾=rvdtDesktop.

GetDesktopVideoModeCount returns the count of available video modes for the primary monitor.

GetDesktopVideoMode returns information about the specified mode (Index parameter must be in range 0..**GetDesktopVideoModeCount**-1). **SetDesktopVideoMode** changes the desktop mode to the Index-th mode.

GetDesktopCurrentVideoMode returns information about the current video mode.

GetDesktopVideoModelIndex returns the index of the current video mode.

This feature is supported only on Windows platform.

2.1.2.2.5 TRVCamera.GetAvailableCamProperties, GetAvailableCamMethods, GetAccessibleCamProperties, GetAccessibleCamMethods

The method return available IP camera properties and methods.

```

type // defined in MRVType unit
    TRVCamProperty = (rvcp_DateTimeParameter, rvcp_dtpSyncWithPC,
        rvcp_dtpNow, rvcp_dtpTimeZone, rvcp_dtpNTPEnabled,
        rvcp_dtpNTPServer, rvcp_Network, rvcp_netDynamicIP,
        rvcp_netIPAddr, rvcp_netSubnetMask, rvcp_netGateway,
        rvcp_netDNSServer, rvcp_netHttpPort, rvcp_WirelessLan,
        rvcp_wlSSID, rvcp_wlEncryption, rvcp_wlNetworkType,
        rvcp_wlWEP, rvcp_wlWPA, rvcp_wlShareKey, rvcp_wlAuthetication,
        rvcp_wlKeyFormat, rvcp_wlDefaultTXKey, rvcp_wlKey, rvcp_wlKeyBits,
        rvcp_ADSL, rvcp_adslUser, rvcp_adslPassword, rvcp_MailService,
        rvcp_msSender, rvcp_msReceivers, rvcp_msSMTPServer, rvcp_msSMTPPort,
        rvcp_msReportInternetIPbyMail, rvcp_msNeedAuthentication,
        rvcp_msSMTPUser, rvcp_msSMTPPassword, rvcp_FtpService,
        rvcp_fsServer, rvcp_fsPort, rvcp_fsUser, rvcp_fsPassword,
        rvcp_fsUploadFolder, rvcp_fsMode, rvcp_fsUploadImageEnabled,
        rvcp_fsUploadInterval, rvcp_AlarmService,
        rvcp_asMotionDetectEnabled, rvcp_asMotionDetectSensitivity,
        rvcp_asAlarmInputEnabled, rvcp_asAlarmTriggerLevel,
        rvcp_asIOLinkage, rvcp_asOutputLevel, rvcp_asSendMail,
        rvcp_asUploadImageEnabled, rvcp_asUploadImageInterval,
        rvcp_asSchedulerEnabled, rvcp_Decoder, rvcp_dBaudrate,
        rvcp_PTZSettings, rvcp_ptzsLedMode, rvcp_ptzsCenterOnStart,
        rvcp_ptzsAutoPatrolInterval, rvcp_ptzsAutoPatrolType,
        rvcp_ptzsPatrolHoriz, rvcp_ptzsPatrolVert,
        rvcp_ptzsPatrolRate, rvcp_ptzsPatrolUpRate, rvcp_ptzsPatrolDownRate,
        rvcp_ptzsPatrolLeftRate, rvcp_ptzsPatrolRightRate, rvcp_dmID,
        rvcp_dmVersion, rvcp_dmAlias, rvcp_dmUPnPtoMapPort, rvcp_Users,
        rvcp_Switch, rvcp_FlipHorizontally, rvcp_FlipVertically,
        rvcp_VideoResolution, rvcp_VideoMode, rvcp_Brightness,
        rvcp_Contrast, rvcp_Parameters);

    TRVCamMethod = (rvul_dtpGetTimeZoneStr, rvcm_wlScan,

```

```
rvcm_dmUpgradeDeviceFirmware, rvcm_dmBackup, rvcm_dmRestore,
rvcm_dmRestoreFactorySettings, rvcm_dmRebootDevice, rvcm_dmLog,
rvcm_VideoStream, rvcm_SnapShot, rvcm_Move);
```

function GetAvailableCamProperties: TRVCamProperties;

function GetAvailableCamMethods: TRVCamMethods;

function GetAccessibleCamProperties: TRVCamProperties;

function GetAccessibleCamMethods: TRVCamMethods;

GetAvailableCamProperties and GetAvailableCamMethods return all the properties and the methods supported by the camera, even if they are not available for the user `UserName:UserPassword`⁽⁵⁷⁾.

GetAccessibleCamProperties and GetAccessibleCamMethods take the user rights into account.

These methods can be used only after the camera is found, see SearchCamera⁽⁶⁸⁾.

2.1.2.2.6 TRVCamera.GetColorControlPropertyRange

Returns a range, a default value and a step size for the specified property.

```
function GetColorControlPropertyRange(Prop: TRVColorControlProperty(240);
    out MinValue, MaxValue, DefValue, Step: Integer): Boolean;
```

The function returns *True* if the selected camera supports the specified property.

You can use the results when assigning values to Brightness, Contrast, Saturation, Sharpness, Hue⁽⁴⁵⁾ properties.

Input parameter:

Prop – property (brightness / contrast / saturation / sharpness / hue).

Output parameters:

MinValue, MaxValue – minimum and maximum allowed property values.

DefValue – default (neutral) property value.

Step – step size for the property (the smallest increment by which the property can change).

2.1.2.2.7 TRVCamera.GetSnapShot

Returns the current frame from the camera as an image.

```
function GetSnapShot: TRVImageWrapper(206);
```

You should free this image yourself.

2.1.2.2.8 TRVCamera.IsSupportedFFMPEG

Returns *True* if **FFmpeg** is installed in the system and can be used.

```
function IsSupportedFFMPEG: Boolean;
```

RVMedia requires the following minimal set of FFmpeg libraries:

- avformat-*.dll
- avcodec-*.dll
- avutil-*.dll
- swscale-*.dll

See also:

- IsSupportedGStreamer⁽⁶⁶⁾
- FFMpegProperty.UseFFMPEG⁽⁵⁰⁾
- LoadFFMpegLibraries⁽²¹⁹⁾ global procedure.

2.1.2.2.9 TRVCamera.IsSupportedGStreamer

Returns True if **GStreamer** is installed in the system and can be used.

function IsSupportedGStreamer: Boolean

Note: GStreamer may be installed without necessary decoders (for playing MPEG4 and H.264)

See also:

- GStreamerProperty⁽⁵²⁾
- IsSupportedFFMPEG⁽⁶⁵⁾
- DeviceType⁽⁴⁹⁾

2.1.2.2.10 TRVCamera.LockCommands, UnlockCommands

LockCommands prevents sending commands to the camera after property changes, UnlockCommands restores the sending mode.

procedure LockCommands;

procedure UnlockCommands;

2.1.2.2.11 TRVCamera.Move***

The methods for camera movement (rotation), if the camera supports movement.

Start movement:

```
function MoveLeft : Boolean;
function MoveRight : Boolean;
function MoveUp : Boolean;
function MoveDown : Boolean;
function MoveLeftUp : Boolean;
function MoveLeftDown : Boolean;
function MoveRightUp : Boolean;
function MoveRightDown : Boolean;
function MoveHPatrol : Boolean;
function MoveVPatrol : Boolean;
```

Stop movement:

```
function MoveLeftStop : Boolean;
function MoveRightStop : Boolean;
function MoveUpStop : Boolean;
function MoveDownStop : Boolean;
function MoveHPatrolStop : Boolean;
function MoveVPatrolStop : Boolean;
function MoveStop : Boolean; // stops all
```

Move to the home position:

```
function MoveCenter : Boolean;
```

Description

The methods start or stop movement to the specified direction, vertical or horizontal patrolling.

Wait mode ⁽⁴⁷⁾: the methods return only when the command is executed. Return value: the movement command was successfully sent to the camera.

No-wait mode ⁽⁴⁷⁾: the method initializes a thread (for sending the command to the camera) and returns immediately. Return value: *False*. When the command is complete, OnMoved ⁽⁷²⁾ event occurs.

In the both cases, all **Move***** methods only initiate movement; this movement continues until the corresponding **Move***Stop** method is called. The only exception is **MoveCenter** that moves the camera to the home position and does not need a stop method.

The methods take FlipHorizontally and FlipVertically ⁽⁵¹⁾ into account.

Other ways to move a camera

In addition to calling these methods, you can:

- use TRVCamControl ⁽³⁹⁾ component
- use the mouse in TRVCamView ⁽⁹³⁾ component (see CamMoveMode ⁽⁹⁵⁾)
- use the mouse in TRVCamMultiView ⁽⁷⁴⁾ component (see CamMoveMode ⁽⁷⁷⁾)

Supported cameras

- If DeviceType ⁽⁴⁹⁾ = *rvdtIPCamera*: movement is supported for some Foscam, Axis, Panasonic IP cameras. The camera model must be detected by calling SearchCamera ⁽⁶⁸⁾.
- If DeviceType ⁽⁴⁹⁾ = *rvdtWebCamera*: movement is supported for some local cameras (Windows only)

2.1.2.2.12 TRVCamera.PlayVideoStream, PlayVideoFile

Starts playing video.

```
procedure PlayVideoStream;
procedure PlayVideoFile(FileName : string);
```

PlayVideoStream starts playing a video from a camera or a network. A video source depends on DeviceType ⁽⁴⁹⁾ property.

- a wait mode example ⁽²²⁸⁾
- a no-wait mode example ⁽²²⁹⁾

If the video from the camera starts successfully (when the first frame is received), OnStartVideoStream ⁽⁷³⁾ event occurs. When the video stops, OnEndVideoStream ⁽⁷⁰⁾ event occurs.

PlayVideoFile starts playing an MJPEG file specified in the **FileName** parameter. If the video from the file starts successfully (when the first frame is read), OnStartVideoFile ⁽⁷³⁾ event occurs. When the video stops, OnEndVideoFile ⁽⁷⁰⁾ event occurs. FramePerSec ⁽¹⁹⁰⁾ property is used.

To stop a video, call Abort ⁽¹⁹¹⁾.

You can wait until a video finished using WaitForVideo, WaitForVideoStream, WaitForVideoFile ⁽⁶⁹⁾ (although, using the events is strongly recommended).

2.1.2.2.13 TRVCamera.ResetImageSetting

Resets video color parameters to default values

function TRVCamera.ResetImageSetting: Boolean;

See also

- Brightness, Contrast, Hue, Saturation, Sharpness ⁽⁴⁵⁾

2.1.2.2.14 TRVCamera.SearchCamera

Searches for the camera at CameraHost:CameraPort ⁽⁴⁶⁾ having the specified video format.

function SearchCamera(CameraTypes: TRVCameraTypes ⁽²³⁸⁾ = []) : Boolean;

The method connects to the specified address, recognizes the camera model (if there is a camera at this address), reads its properties.

MJpeg vs H.264 cameras

If VideoFormat ⁽⁵⁹⁾ = *rvvfMJPEG*, the method searches for cameras providing MJPEG video streams; otherwise it searches for cameras providing H.264 streams (they require FFmpeg to play).

Camera types

If you know a model of the camera, you can specify its type in **CameraTypes** parameter. The method only searches for the camera types specified in this parameter (the empty set works like the full set, i.e. the component searches for all supported camera models).

The following camera types are the most important, they allow searching for cameras that support a special API:

- *rvctFoscam*: searching for cameras supporting Foscam API
- *rvctPanasonic*: searching for cameras supporting Panasonic API
- *rvctAxis*: searching for cameras supporting Axis API
- *rvctDLink*: searching for cameras supporting DLink API
- *rvctMobotix*: searching for cameras supporting Mobotix API

For these cameras, TRVCamera can not only display video, but also rotate and configure the camera (to the extent allowed by API and access rights).

Other camera types allow searching for video URLs in location common for the specified camera types. The most comprehensive search is performed if *rvctUnknown* is included.

Wait/no-wait modes

Wait mode ⁽⁴⁷⁾: the method returns only when the search is complete. Return value: a camera is found. Not recommended, because searching is a relatively long process.

No-wait mode ⁽⁴⁷⁾: the method initializes a searching thread and returns immediately. Return value: False. When the search is complete, OnSearchComplete ⁽⁷³⁾ event occurs.

Search steps

When the search is successfully completed, the following changes are made:

- IPCameraTypes ⁽⁵²⁾ property is assigned
- VideoResolution ⁽⁶¹⁾ property is assigned if the camera supports it

- for all camera models, except for controllable Foscam, Axis, D-Link and Panasonic (or compatible) cameras, CameraHost:CameraPort⁽⁴⁶⁾ are cleared, and URL of MJPEG stream is assigned to URL⁽⁵⁷⁾ property
- Parameters⁽⁵⁴⁾ property is filled

After searching, call PlayVideoStream⁽⁶⁷⁾ to receive video from this camera.

See also

- WaitForSearch⁽⁶⁹⁾
- MaxCameraSearchThreadCount⁽⁵³⁾
- CameraSearchTimeout⁽⁴⁶⁾
- Searching⁽⁵⁵⁾

2.1.2.2.15 TRVCamera.SwitchLEDOn, SwitchLEDOff

The methods turn the camera LED on/off, if the camera supports it.

```
function SwitchLEDOn : Boolean;
function SwitchLEDOff : Boolean;
```

2.1.2.2.16 TRVCamera.WaitForVideoStream, WaitForVideoFile, WaitForVideo, WaitForSearch

The methods wait for the specified operation to finish.

```
procedure WaitForVideoStream;
procedure WaitForVideoFile;
procedure WaitForVideo;
procedure WaitForSearch;
```

WaitForVideoStream waits until a video from a camera (started in PlayVideoStream⁽⁶⁷⁾) stops.

WaitForVideoFile waits until a video from a file (started in PlayVideoFile⁽⁶⁷⁾) stops.

WaitForVideo waits until a video (either from a file or from a camera) stops.

WaitForSearch waits until a camera searching (started in SearchCamera⁽⁶⁸⁾) stops.

These methods may be useful in a no-wait⁽⁴⁷⁾ mode, but, when possible, avoid using these methods and use the events instead: OnEndVideoStream, OnEndVideoFile⁽⁷⁰⁾, OnSearchComplete⁽⁷³⁾. These methods may be useful after calling Abort⁽¹⁹¹⁾.

The methods call Application.ProcessMessages inside, so it is possible that the user tried to close the application while these methods were working. After calling these methods, you must check Application.Terminated property and exit if necessary.

2.1.2.3 Events

In TRVCamera

```
OnDownload(70)
OnDownloaded(70)
OnEndVideoFile(70)
OnEndVideoStream(70)
```

OnError⁽⁷⁰⁾
 OnGetImage⁽⁷¹⁾
 OnGetVideoStreamIndex⁽⁷¹⁾
 OnLogin⁽⁷¹⁾
 OnLoginFailed⁽⁷²⁾
 OnMoved⁽⁷²⁾
 OnNewImage⁽⁷²⁾
 OnPrepared⁽⁷²⁾
 OnProgress⁽⁷³⁾
 OnSearchComplete⁽⁷³⁾
 OnSetProperty⁽⁷³⁾
 OnStartVideoFile⁽⁷³⁾
 OnStartVideoStream⁽⁷³⁾
 OnVideoStart⁽⁷⁴⁾

2.1.2.3.1 TRVCamera.OnDownload, OnDownloaded

reserved for future use

2.1.2.3.2 TRVCamera.OnEndVideoFile, OnEndVideoStream

The events occur when a video is stopped.

```

property OnEndVideoStream: TRVCamDoneEvent(232);
property OnEndVideoFile: TRVCamDoneEvent(232);
  
```

OnEndVideoStream occurs when the component stops playing a video from the current video source.

OnEndVideoFile occurs when the component stops playing a video started by PlayVideoFile⁽⁶⁷⁾.

Note: playing video from a file using DeviceType⁽⁴⁹⁾ = *rvdtFile* calls **OnEndVideoStream**, not **OnEndVideoFile**.

See also

- PlayVideoStream, PlayVideoFile⁽⁶⁷⁾
- Abort⁽¹⁹¹⁾
- OnStartVideoStream, OnStartVideoFile⁽⁷³⁾

2.1.2.3.3 TRVCamera.OnError

The event occurs in response to an error.

```

property OnError: TRVCamErrorEvent(232);
  
```

This event allows handling errors that may occur while receiving video from different sources or during video remuxing.

Some errors are critical and interrupt video playback or other operations. Others are more like warnings, but this event enables you to treat them as errors and stop the current operation if necessary.

This event may be triggered from a background thread, so avoid direct interaction with the user interface. If you need to update the UI, use `TThread.Synchronize` or `TThread.Queue` to safely execute code in the main thread.

See also

- `TRVCamRecorder.OnError`⁽⁹²⁾
- `TRVAudioPlayer.OnError`⁽¹¹⁴⁾

2.1.2.3.4 TRVCamera.OnGetImage

This event occurs when the component reads a frame from a camera stream or a file.

property `OnGetImage: TRVImageEvent`⁽²³⁵⁾;

The image is returned in **Img** parameter. You must not free **Img** yourself. You can modify this image.

This event is called in a thread context.

2.1.2.3.5 TRVCamera.OnGetVideoStreamIndex

Allows to choose a video stream to play.

property `OnGetVideoStreamIndex: TRVGetMediaStreamIndexEvent`⁽²³⁴⁾;

This event occurs before playing a video file, if it is played using **FFmpeg**.

This event may be useful if a video file contains more than one video stream. It is a very rare case.

If the event is not processed, the first video stream is played.

Warning: this event is called in a thread context.

You cannot change a video stream while the video is played: you need to restart video.

See also

`TRVCamSound.OnGetAudioStreamIndex`⁽¹¹⁶⁾

2.1.2.3.6 TRVCamera.OnLogin

The event for requesting a user name and a password from the user.

type

```
// Defined in MRVType/fmxMRVType unit
TRVOnLoginEvent = procedure(Sender: TObject; const ARealm : string;
    var AUserName, APassword : String; var Proceed: Boolean) of object;
```

property `OnLogin: TRVOnLoginEvent`;

This event occurs if authentication fails and `LoginPrompt`⁽⁵³⁾ = `True`.

If this event is not assigned, the component displays its own login prompt dialog. The event allows to implement your own user interface for requesting a user name and a password.

Input parameters:

ARealm describes the resource that needs authentication.

AUserName, APassword – user name and password from the previous (failed) login attempt.

Proceed = `True`.

Output parameters:

UserName, APassword – new user name and password.

Assign **Proceed** = *True* to proceed with the new user name and password. Assign **Proceed** = *False* to stop login attempts.

2.1.2.3.7 TRVCamera.OnLoginFailed

Occurs when authentication fails.

type

```
// Defined in MRVType/fmxMRVType unit
TRVOnLoginFailedEvent = procedure (Sender: TObject;
  const AURL : string) of object;
```

property OnLoginFailed: TRVOnLoginFailedEvent;

If LoginPromt⁽⁵³⁾ = *True*, this event occurs when all attempts to login failed (it does not occur on each attempt).

You can use this event to display a message to the user.

It is not necessary to process this event. If authentication fails, OnEndVideoStream⁽⁷⁰⁾ occurs with a non-zero Status parameter.

2.1.2.3.8 TRVCamera.OnMoved

Occurs when a command for a camera movement is finished.

property OnMoved: TRVCamDoneEvent⁽²³²⁾;

The parameters Status=0 means that the command was successful, other values mean errors.

2.1.2.3.9 TRVCamera.OnNewImage

The event allows providing custom video frames.

property OnNewImage: TRVImageEvent⁽²³⁵⁾;

This event occurs if DeviceType⁽⁴⁹⁾ = *rvdtUserData*.

In this event, programmers can provide their own video frames. They must be assigned to **img** parameter.

Example:

```
img.Assign(MyFrame);
```

In this example, MyFrame is a variable of a graphic class (for example, TBitmap, or TJPEGImage) containing the current video frame.

2.1.2.3.10 TRVCamera.OnPrepared

The event occurs when the component reads properties from the IP camera.

property OnPrepared: TRVCamDoneEvent⁽²³²⁾;

This event occurs while searching for the camera. It occurs before OnSearchComplete⁽⁷³⁾.

A search is started in SearchCamera⁽⁶⁸⁾ method.

2.1.2.3.11 TRVCamera.OnProgress

The event occurs when the next portion of video data (from a file or a camera) is received.

```
type // defined in MRVType unit
  TRVCamProgressEvent = procedure (Sender: TObject;
    BytesRead: Integer) of object;
property OnProgress: TRVCamProgressEvent;
```

BytesRead is a size of this new data portion. You can use this event to calculate how many bytes are received from a camera while playing a video.

2.1.2.3.12 TRVCamera.OnSearchComplete

The event occurs when a camera search is complete.

```
property OnSearchComplete: TRVCamDoneEvent232;
```

The parameters Status=0 means that a camera is found, other values mean errors.

A search is started in SearchCamera⁶⁸ method.

2.1.2.3.13 TRVCamera.OnSetProperty

Occurs when a command for assigning a camera property is finished.

```
property OnSetProperty: TRVCamDoneEvent232;
```

The parameters Status=0 means that the command was successful, other values mean errors.

2.1.2.3.14 TRVCamera.OnStartVideoFile, OnStartVideoStream

The events occur when a video starts playing (usually, when the first video frame is received).

```
property OnStartVideoStream: TNotifyEvent;
```

```
property OnStartVideoFile: TNotifyEvent;
```

OnStartVideoStream occurs when the component starts playing a video from a camera.

OnStartVideoFile occurs when the component starts playing a video from a file in PlayVideoFile⁶⁷ method.

Note: playing video from a file using DeviceType⁴⁹ = *rvdtFile* calls **OnStartVideoStream**, not **OnStartVideoFile**.

See also

- PlayVideoStream, PlayVideoFile⁶⁷
- OnEndVideoStream, OnEndVideoFile⁷⁰
- OnVideoStart⁷⁴

2.1.2.3.15 TRVCamera.OnVideoStart

The event occurs when a video starts

property OnVideoStart: TNotifyEvent;

This event occurs when the component initializes playing a video (from a camera or a file). It occurs before receiving the first video frame, so this event does not guarantee that this video will actually start playing.

See also

OnStartVideoFile and OnStartVideoStream ⁽⁷³⁾

2.1.3 TRVCamMultiView

TRVCamMultiView shows videos from several video sources. Optionally, it can show activity of audio sources.

Unit [VCL and LCL] MRVCamMultiView;

Unit [FMX] fmxMRVCamMultiView;

Syntax [VCL and LCL]

```
TRVCamMultiView = class(TScrollingWinControl)
```

Syntax [FMX]

```
TRVCamMultiView = class(TCustomScrollBox)
```

▼ Hierarchy

VCL and LCL:

TObject
TPersistent
TComponent
TControl
TWinControl
TScrollingWinControl

FMX:

TObject
TPersistent
TComponent
TFmxObject
TControl
TStyledControl
TCustomScrollBox

How to use

Fill the collection of Viewers ⁽⁸²⁾. Each item in this collection defines a position and properties of one video window inside the control. Properties of items of this collections are similar to properties of TRVCamView ⁽⁹³⁾ component.

Full-screen mode

You can display this viewer expanded to the full screen size by assigning *True* to `FullScreen`⁽⁷⁹⁾.

You can allow users to switch to/from a full-screen mode by assigning *True* to `AllowFullScreen`⁽⁷⁶⁾.

You can detect switching modes in `OnFullScreen`⁽⁸⁵⁾ event.

Drawing efficiency

See the notes about drawing efficiency in the topics about:

- `RefreshRate`⁽⁸⁰⁾ property
- `Viewers`⁽⁸²⁾`[]`.`ViewModel`⁽¹⁰³⁾ property.

2.1.3.1 Properties

In TRVCamMultiView

- `AllowFullScreen`⁽⁷⁶⁾
- `AudioSource`⁽⁷⁶⁾
- `CameraControl`⁽⁷⁶⁾
- `CamMoveMode`⁽⁷⁷⁾
- `CaptionColor`⁽⁷⁷⁾
- `CaptionFont`⁽⁷⁷⁾ [VCL]
- `CaptionTextSettings`⁽⁷⁷⁾ [FMX]
- `CaptionHeight`⁽⁷⁷⁾
- ▶ `CurRenderMode`⁽⁷⁸⁾ [VCL and LCL]
- `HoverLineColor`⁽⁸⁰⁾
- `FocusLineColor`⁽⁷⁸⁾
- `FullScreen`⁽⁷⁹⁾
- ▶ `FullScreenMultiView`⁽⁷⁹⁾
- `IconStyle`⁽⁸⁰⁾
- `Language`⁽⁸⁰⁾
- `RefreshRate`⁽⁸⁰⁾
- `RememberLastFrame`⁽⁸¹⁾
- `RenderMode`⁽⁷⁸⁾ [VCL and LCL]
- `SearchPanelColor`⁽⁸¹⁾
- `SearchPanelTextColor`⁽⁸¹⁾
- `TextSettings`⁽⁷⁹⁾ [FMX]
- `ViewerColor`⁽⁷⁸⁾
- `ViewerIndex`⁽⁸²⁾
- `Viewers`⁽⁸²⁾
- `WaitAnimationDelay`⁽⁸⁴⁾

Inherited from TScrollingWinControl

- `Align`
- `Anchors`
- `Color`⁽⁷⁸⁾
- `Cursor`

- Enabled
- Font⁽⁷⁹⁾ [VCL]
- Height
- Hint
- ParentFont⁽⁷⁹⁾ [VCL]
- PopupMenu
- ShowHint
- TabOrder
- TabStop
- Visible

2.1.3.1.1 TRVCamMultiView.AllowFullScreen

Allows switching to a full-screen mode.

property AllowFullScreen: Boolean;

If *True*, a special icon is displayed when the user moves the mouse pointer above the control.

To close the full-screen viewer, press a similar icon or **Esc**.

Default value

False

See also

- FullScreen⁽⁷⁹⁾
- OnFullScreen⁽⁸⁵⁾
- TRVCamView⁽⁹³⁾.AllowFullScreen⁽⁹⁵⁾

2.1.3.1.2 TRVCamMultiView.AudioSource

Specifies a TRVMicrophone⁽¹¹⁷⁾ or TRVCamSound⁽¹¹⁵⁾ component for which an activity can be displayed.

property AudioSource: TRVAudioSource⁽¹⁸⁵⁾;

Viewer panels inside TRVCamMultiView component can display an audio viewer, if the corresponding item in Viewers⁽⁸²⁾ collection has AudioViewer property = *True*. If this viewer panel is linked to TRVCamera⁽⁴²⁾ (not to TRVCamReceiver⁽¹²⁰⁾), its audio viewer shows activity of this AudioSource.

2.1.3.1.3 TRVCamMultiView.CameraControl

Specifies a TRVCameraControl component used to control the camera movement (if the current camera supports it).

property CameraControl: TRVCamControl⁽³⁹⁾;

This component controls the camera in the selected viewer, see ViewerIndex⁽⁸²⁾.

If this property is not empty, it is assigned to the CameraControl⁽⁴⁶⁾ property of TRVCamera component linked to the selected viewer.

2.1.3.1.4 TRVCamMultiView.CamMoveMode

Specifies how the selected viewer controls the camera motion.

property `CamMoveMode: TRVCamMoveMode(238);`

This property works if the viewer⁽⁸²⁾'s VideoSource is TRVCamera⁽⁴²⁾. The user can control the camera motion (rotation) using the mouse, if supported by the camera.

Default value

`vcmmDrag`

See also:

- TRVCamView⁽⁹³⁾.CamMoveMode⁽⁹⁵⁾
- TRVCamera⁽⁴²⁾.Move***⁽⁶⁶⁾ methods
- TRVCamControl⁽³⁹⁾ component

2.1.3.1.5 TRVCamMultiView.CaptionColor, CaptionFont, CaptionHeight

The properties define how the caption of viewer windows is displayed.

property `CaptionColor: TRVMColor(244);`

property `CaptionHeight: Integer;`

VCL and LCL:

property `CaptionFont: TFont;`

FMX:

property `CaptionTextSettings: TTextSettings;`

CaptionColor is a default background color for captions of viewer windows (it can be overridden by Viewers⁽⁸²⁾ [].CaptionColor).

[FMX note]: semi-transparency is not supported in the caption yet, please specify full opacity (\$FF) in the alpha channel of this color.

CaptionFont (or **CaptionTextSettings**) is a font for a text in captions.

CaptionHeight defines the caption height in 96 DPI screen mode. When the screen DPI is different, the caption height is changed accordingly.

Other caption properties are customized for each viewer⁽⁸²⁾: Title, CaptionParts, ShowCaption, CaptionColor.

Default values

- CaptionHeight: 20

Default values [VCL and LCL]:

- CaptionFont: Tahoma, 8
- CaptionColor: \$00A77E4F

• Default values [FMX]:

- CaptionColor: \$FF4F7EA7

See also

- caption properties of TRVCamView⁽¹⁰¹⁾

2.1.3.1.6 TRVCamMultiView.Color, ViewerColor

Background colors.

```
property Color: TRVMColor(244);
property ViewerColor: TRVMColor(244);
```

Color is a background color of this component. ViewerColor is a default background color for viewer windows (it can be overridden by Viewers⁽⁸²⁾ [.Color]).

Default value [VCL and LCL]

- Color: *\$00E7BE9F*
- ViewerColor: *\$00E7BE9F*

Default value [FMX]

- Color: *TAlphaColorRec.White*
- ViewerColor: *\$FF9FBEE7*

See also

- FocusLineColor⁽⁷⁸⁾
- CaptionColor⁽⁷⁷⁾
- HoverLineColor⁽⁸⁰⁾

2.1.3.1.7 TRVCamMultiView.CurRenderMode, RenderMode

RenderMode specifies a video rendering method for all viewers.

CurRenderMode returns a rendering method currently used.

```
property RenderMode: TRVMRenderMode(246);
property CurRenderMode: TRVMRenderMode(246);
```

If the mode specified in RenderMode cannot be initialized, the component falls back to *rvmrmSoftware*.

See the requirements in the topic about TRVMRenderMode⁽²⁴⁶⁾.

Default value

rvmrmSoftware

2.1.3.1.8 TRVCamMultiView.FocusLineColor

```
property FocusLineColor: TRVMColor(244);
```

FocusLineColor defines the color of a frame around the viewer when it is selected.

Default values [VCL and LCL]

- FocusLineColor: *c!Red*

Default values [FMX]

- FocusLineColor: *TAlphaColorRec.Red*

Assign *c!None* (VCL and LCL) or *TAlphaColorRec.Null* (FMX) to to use a normal frame color for focused windows.

See also

- Color, ViewerColor⁽⁷⁸⁾
- CaptionColor⁽⁷⁷⁾

- HoverLineColor⁽⁸⁰⁾

2.1.3.1.9 TRVCamMultiView.Font, ParentFont

The properties specify the font used in:

- the search panel;
- the viewer itself to display “No Video” text.

VCL and LCL:

```
property Font: TFont;
property ParentFont: Boolean;
```

FMX:

```
property Font: TTextSettings;
```

To have a control use the same font as its parent control, set **ParentFont** to *True*. If **ParentFont** is *False*, the control uses its own **Font** property.

Default value

ParentFont: *True*

See also

CaptionFont⁽⁷⁷⁾
search panel in TRVCamView⁽¹⁰⁰⁾

2.1.3.1.10 TRVCamMultiView.FullScreen

Switches to/from a full-screen mode.

```
property FullScreen: Boolean;
```

Returns *True* when the viewer is in a full-screen mode.

Assign *True* to this property to show this viewer in a full-screen mode, assign *False* to return back to a normal mode. This assignment works even if AllowFullScreen⁽⁷⁶⁾ = *False*.

Initial value

False

See also

- FullScreenMultiView⁽⁷⁹⁾
- OnFullScreen⁽⁸⁵⁾
- TRVCamView⁽⁹³⁾.FullScreen⁽⁹⁷⁾

2.1.3.1.11 TRVCamMultiView.FullScreenMultiView

Returns a full-screen multi-viewer.

```
property FullScreenMultiView: TRVCamMultiView(74);
```

When this multi-viewer is in a full-screen mode⁽⁷⁹⁾, this property returns a multi-view control representing this control on a full screen.

When this multi-viewer is not in a full-screen mode, this property returns *nil*.

See also

- FullScreen⁽⁷⁹⁾

- OnFullScreen⁽⁸⁵⁾
- TRVCamView⁽⁹³⁾.FullScreenView⁽⁹⁸⁾

2.1.3.1.12 TRVCamMultiView HoverLineColor

Specifies the color of a rectangle shown when a viewer window is below the mouse pointer.

property HoverLineColor: TRVMColor⁽²⁴⁴⁾;

Default value [VCL and LCL]

\$00B78E5F

Default value [FMX]

\$FF5F8EB7

Assign *cNone* (VCL and LCL) or *TAlphaColorRec.Null* (FMX) to use a normal frame color for focused windows.

See also

- FocusLineColor⁽⁷⁸⁾
- CaptionColor⁽⁷⁷⁾
- Color⁽⁷⁸⁾

2.1.3.1.13 TRVCamMultiView.IconStyle

Specifies an animation that is displayed while the component searches for an IP camera.

property IconStyle: TRVMIconStyle⁽²⁴⁵⁾;

This panel is shown when TRVCamera.SearchCamera⁽⁶⁸⁾ is called.

This property is applied to all viewer windows.

This property defines not only an animation, but also background and text colors of a search panel (if they are not defined explicitly in SearchPanelColor and SearchPanelTextColor⁽⁸¹⁾ properties).

Default value

rvisClassic

2.1.3.1.14 TRVCamMultiView.Language

Specifies the language for user interface of all video viewers.

property Language: TRVMLanguage⁽²⁴⁵⁾;

Default value

rvmlEnglish

2.1.3.1.15 TRVCamMultiView.RefreshRate

Specifies how often the control can be redrawn when playing videos.

property RefreshRate: Integer;

If **RefreshRate** = 0, the component is redrawn when receiving each new video frame. While in VCL version the component draws new video frames directly on its canvas (without redrawing other its parts), in LCL and FireMonkey the whole control is redrawn (with CllipRect optimization, if possible), and displaying a large count of videos may be inefficient.

If **RefreshRate** > 0, it defines the maximum count of repaints in a second. In this mode, the component checks periodically for updated video viewers⁽⁸²⁾, and redraws itself if necessary (with CllipRect optimization, if possible). This mode is efficient for displaying a large count of videos, especially in LCL and FireMonkey.

Default value

0

2.1.3.1.16 TRVCamMultiView.RememberLastFrame

Indicates whether to remember the last video frame.

property RememberLastFrame: Boolean;

If *True*, viewers display the last frame when a video is interrupted or stopped. A blank screen is shown otherwise.

Default value

True

2.1.3.1.17 TRVCamMultiView.ScaleViewers

Specifies whether video viewers are resized together with the component.

property ScaleViewers: Boolean;

Video viewers are defined in Viewers⁽⁸²⁾ collection property.

If *True*, when the component is resized, positions and sizes of all viewers are changed proportionally.

If *False*, video viewers are resized according to their AlignVideoViewer and Anchors properties.

Default value

True

2.1.3.1.18 TRVCamMultiView.SearchPanelColor, SearchPanelTextColor

These properties specify colors of a camera search panel.

property SearchPanelColor: TRVMColor⁽²⁴⁴⁾;

property SearchPanelTextColor: TRVMColor⁽²⁴⁴⁾;

This panel is shown when TRVCamera.SearchCamera⁽⁶⁸⁾ is called.

These properties are applied to all viewer windows.

Additional information can be found in the topic about properties of TRVCamView of the same name⁽⁹⁹⁾.

Default value [VCL and LCL]

c/None

Default value [FMX]

TAlphaColorRec.Null

2.1.3.1.19 TRVCamMultiView.ViewerIndex

Specifies the selected viewer.

property `ViewerIndex: Integer;`

This is the index in the collection `Viewers`⁽⁸²⁾.

The selected viewer has a frame of `FocusLineColor`⁽⁷⁸⁾ color.

A video from the selected viewer can be displayed not only in its window, but in all viewers having the property `MainViewer=True`.

2.1.3.1.20 TRVCamMultiView.Viewers

A collection of viewer windows.

type

```
TRVCamViewCollection = class (TCollection);
```

```
TRVCamViewItem = class (TCollectionItem)
```

property `Viewers: TRVCamViewCollection;`

This is a collection of `TRVCamViewItem` items. Each item defines properties of one viewer window.

Properties of each item in this collection are described below.

Properties for video viewer (TRVCamViewItem)

Video properties

An item in this collection has the same properties as `TRVCamView`⁽⁹³⁾ component (the links below are to the corresponding properties of `TRVCamView`⁽⁹³⁾ component):

- `VideoSource`⁽¹⁰³⁾
- `GUIDFrom`, `IndexFrom`⁽⁹⁸⁾
- `Title`, `CaptionParts`, `ShowCaption`, `CaptionColor`⁽¹⁰¹⁾
- `Color`⁽⁹⁶⁾
- `ShowCameraSearch`⁽¹⁰⁰⁾
- `UseOptimalVideoResolution`⁽¹⁰²⁾
- `ViewMode`⁽¹⁰³⁾
- `FrameScaleQuality`⁽⁹⁷⁾ [VCL and LCL]

Additionally, if `UseFramePerSec=True` (default is *False*), `FramePerSec` is assigned to the corresponding `TRVCamView`⁽⁹³⁾.`VideoSource`⁽¹⁰³⁾.`FramePerSec`⁽¹⁹⁰⁾ (see also about main viewers).

By default, `Color` and `CaptionColor` properties are equal to *c/None*. It means that the properties of the parent `TRVCamMultiView`⁽⁷⁴⁾ is used instead (`ViewerColor`⁽⁷⁸⁾ and `CaptionColor`⁽⁷⁷⁾)

Some viewers can be chosen as main viewers: assign `MainViewer=True`. Main viewers display videos from the selected viewer, see `ViewerIndex`⁽⁸²⁾. Normally, it makes sense to assign only one main viewer, and make its window larger than other viewers. When the viewer becomes selected, `FramePerSec` and `UseFramePerSec` of the main viewer is used instead of the properties of the selected viewer (if there are several main viewers, the maximum of their `FramePerSec` is used). So you can specify the larger FPS value for the selected window than for normal windows.

Positioning

Left, Top, Width, Height properties define the position and the size of the video viewer. These values are measured in logical pixels at 96 DPI. If the screen DPI is different, the size and position are recalculated accordingly.

When the multiviewer is resized, all video viewers may be resized too. A resizing mode depends on the multiviewer's `ScaleViewers`⁽⁸¹⁾ property.

If it is *True*, **Left, Top, Width, Height** of viewers are changed proportionally to the new size.

If it is *False*, viewers are resized according to their **AlignVideoViewer** and **Anchor** properties.

type

```
TRVMAnchor = (rvmaLeft, rvmaTop, rvmaRight, rvmaBottom);
TRVMAnchors = set of TRVMAnchor;
```

property Anchors: TRVMAnchors;

Use **Anchors** to ensure that the video viewer maintains its current position relative to an edge of multiviewer, even if the multiviewer is resized. When the multiviewer is resized, the video viewer holds its position relative to the edges to which it is anchored. If the video viewer is anchored to opposite edges of its multiviewer, the video viewer stretches when its multiviewer is resized. For example, if the video viewer has its Anchors property set to [rvmaLeft, rvmaRight], the video viewer stretches when the width of its multiviewer changes. The default value for Anchors is [rvmaLeft, rvmaTop]. **Anchors** are used only if the multiviewer's `ScaleViewers`⁽⁸¹⁾ = *False*.

type

```
TRVAlignVideoViewer = (rvavvTop, rvavvBottom, rvavvLeft, rvavvRight,
    rvavvClient, rvavvNone);
```

property AlignVideoViewer: TRVAlignVideoViewer;

Use **AlignVideoViewer** to align the video viewer to the top, bottom, left, or right of its multiviewer and have it remain there even if the size of the multiviewer changes. When the multiviewer is resized, an aligned video viewer also resizes so that it continues to span the top, bottom, left, or right edge of the multiviewer. The default value of **AlignVideoViewer** is *rvavvNone*, which means the video viewer remains where it is positioned on the multiviewer. If **AlignVideoViewer** is set to *rvavvClient*, the video viewer fills the entire multiviewer area. **AlignVideoViewer** is used only if the multiviewer's `ScaleViewers`⁽⁸¹⁾ = *False*.

To get the viewer coordinates, you can use the method of `TRVCamViewItem`:

procedure `.GetViewerRects(out VideoRect, AudioRect: TRVMRect(246));`

This method returns areas where video and audio viewers are displayed in the parent `TRVCamMultiView`⁽⁷⁴⁾ control. You can use this method in a custom drawing event⁽⁸⁵⁾.

Audio viewer properties

In addition to video viewer, a viewer window may have an optional audio viewer (an integrated `TRVMicrophoneView`⁽¹¹⁷⁾ component). It is controlled by the following properties:

- **AudioViewer**: Boolean shows/hides an audio viewer (default value = *False*)
- **AlignAudioViewer**: `TRVAlignAudioViewer` defines the position of the audio viewer (default value = *rvaavRight*)
- **AudioViewerColor**: `TRVMColor`⁽²⁴⁴⁾ defines the background color of the audio viewer.

type

```
TRVAlignAudioViewer = (rvaavTop, rvaavBottom, rvaavLeft, rvaavRight,
    rvaavClient);
```

Value	Meaning
<i>rvaavTop</i>	Audio viewer above video viewer
<i>rvaavBottom</i>	Audio viewer below video viewer
<i>rvaavLeft</i>	Audio viewer to the left side of video viewer
<i>rvaavRight</i>	Audio viewer to the right side of video viewer
<i>rvaavClient</i>	Audio viewer occupies the whole area, hiding video viewer

If **VideoSource** property points to TRVCamReceiver⁽¹²⁰⁾, an audio viewer displays activity of this receiver: **VideoSource** and **GUIDFrom** are assigned to ReceiverSource and GUIDFrom⁽¹⁸⁹⁾ properties of the integrated TRVMicrophoneView⁽¹¹⁷⁾ component, respectively.

If **VideoSource** property points to TRVCamera⁽⁴²⁾, an audio viewer displays activity of the component linked to AudioSource⁽⁷⁶⁾ property of the parent TRVCamMultiView⁽⁷⁴⁾.

2.1.3.1.21 TRVCamMultiView.WaitAnimationDelay

Specifies the delay (in ms) before displaying an animation that shows that the component waits for a next video frame.

property WaitAnimationDelay: Integer;

This property is applied to all viewer windows.

See TRVCamView⁽⁹³⁾.WaitAnimationDelay⁽¹⁰⁵⁾ for details.

Default value

3000 (3 seconds)

2.1.3.2 Events

In TRVCamMultiView

- OnFullScreen⁽⁸⁵⁾
- OnSelectViewer⁽⁸⁵⁾
- OnViewerPaint⁽⁸⁵⁾

Inherited from TScrollingWinControl

- OnClick
- OnContextPopup
- OnEnter
- OnExit
- OnKeyDown
- OnKeyPress
- OnKeyUp
- OnMouseDown
- OnMouseMove
- OnMouseUp

2.1.3.2.1 TRVCamMultiView.OnFullScreen

Occurs when switching to/from a full-screen view.

property OnFullScreen: TRVFullScreenEvent⁽²³⁴⁾;

2.1.3.2.2 TRVCamMultiView.OnSelectViewer

Occurs when one of viewer windows becomes selected (focused).

type

```
TRVSelectCamViewEvent = procedure (Sender: TObject;  
    Viewer: TRVCamView(93); ViewerIndex: Integer) of object;
```

property OnSelectViewer: TRVSelectCamViewEvent;

A viewer window may become selected when the user clicked it with the mouse, or moved to it using Tab or Shift+Tab, or when assigning a value to ViewerIndex⁽⁸²⁾ property.

Parameters:

Viewer – a viewer component corresponding to Viewers⁽⁸²⁾[ViewerIndex].

ViewerIndex = ViewerIndex⁽⁸²⁾

2.1.3.2.3 TRVCamMultiView.OnViewerPaint

Occurs when the component needs to paint a video frame.

type

```
TRVCamViewerPaintEvent = procedure (Sender: TObject;  
    Viewer: TRVCamView(93); ViewerIndex: Integer;  
    VideoFrame : TBitmap; ACanvas : TCanvas;  
    var CanDrawFrame : Boolean) of object;
```

property OnViewerPaint: TRVCamViewerPaintEvent;

Parameters

Viewer – a viewer component corresponding to Viewers⁽⁸²⁾[ViewerIndex].

ViewerIndex – an index of the viewer that needs to draw a video frame.

VideoFrame – a bitmap image containing a video frame to draw.

ACanvas – a canvas where you can draw. However, the recommended way to use this event is modifying **VideoFrame**.

CanDraw specifies whether the component should draw **VideoFrame** onto **ACanvas**.

See also

- Viewers⁽⁸²⁾[].GetViewerRects()

2.1.4 TRVCamRecorder

A video (and audio) recording and streaming component.

Unit [VCL and LCL] MRVCamRecorder;

Unit [FMX] fmxMRVCamRecorder;

Syntax

```
TRVCamRecorder = class (TComponent)
```

▼ Hierarchy

TObject

TPersistent

TComponent

Description

This component can be used to record and stream video and audio files.

The component works only if *FFmpeg*⁽²⁴⁾ library is available to the application.

To record or stream video, assign TRVCamera⁽⁴²⁾ or TRVCamReceiver⁽¹²⁰⁾ component to VideoSource⁽⁹²⁾ property.

To record sound, assign o TRVMicrophone⁽¹¹⁷⁾ or TRVCamSound⁽¹¹⁵⁾ or TRVCamReceiver⁽¹²⁰⁾ component to AudioSource⁽⁸⁹⁾ property.

Recording or streaming to OutputFileName⁽⁸⁹⁾ is started when you assign *True* to Active⁽⁸⁷⁾.

▼ Notes

Note 1: the properties work slightly different comparing to properties of the same name in TRVCamSender⁽¹³⁹⁾. In TRVCamSender, sound can be taken from TRVCamReceiver assigned to VideoSource, and TRVCamReceiver cannot be assigned to AudioSource.

Note 2: both TRVCamRecorder and TRVAudioPlayer⁽¹¹⁰⁾ components can record sound files. When connected to TRVCamReceiver, TRVAudioPlayer records all sounds, TRVCamRecorder records sound from the chosen source.

Video format is specified in VideoCodec⁽⁸⁸⁾, audio format in AudioCodec⁽⁸⁸⁾ properties (or, if you know the exact codec names, in VideoCodecName and AudioCodecName⁽⁸⁸⁾ properties).

Warning: Some video and audio formats may be patent-protected in some countries, and supporting these formats will require from you obtaining licenses from the patent owners.

The following properties define audio parameters: AudioBitrate, AudioSampleRate, AudioChannels, AudioSampleFormat⁽⁸⁷⁾.

The following properties define video parameters: VideoBitrate, VideoFramePerSec, VideoWidth, VideoHeight, VideoAutoSize⁽⁹¹⁾, VideoEncodeParameters⁽⁹¹⁾.

Note: If the source video is received using FFmpeg, there is an alternative to TRVCamRecorder: remuxing using TRVCamera.FFMpegProperty⁽⁵⁰⁾.Remuxing⁽²⁰¹⁾ (saving video as it is, without changing formats of video and audio streams)

2.1.4.1 Properties

In TRVCamRecorder

- Active⁽⁸⁷⁾
- AudioBitrate⁽⁸⁷⁾
- AudioChannels⁽⁸⁷⁾
- AudioCodec⁽⁸⁸⁾

- AudioCodecName ⁸⁸
- AudioSampleRate ⁸⁷
- AudioSource ⁸⁹
- OutputFileName ⁸⁹
- Paused ⁸⁹
- SourceAudioGUID ⁹⁰
- SourceAudioIndex ⁹⁰
- SourceVideoGUID ⁹⁰
- SourceVideoIndex ⁹⁰
- UseAudio ⁸⁹
- UseVideo ⁹²
- VideoAutoSize ⁹¹
- VideoBitrate ⁹¹
- VideoCodec ⁸⁸
- VideoCodecName ⁸⁸
- VideoEncodingParameters ⁹¹
- VideoFramePerSec ⁹¹
- VideoHeight ⁹¹
- VideoSource ⁹²
- VideoWidth ⁹¹

2.1.4.1.1 TRVCamRecorder.Active

Turn on/off recording

property Active: Boolean;

Assign *True* to start video recording, assign *False* to stop it.

2.1.4.1.2 TRVCamRecorder.Audio*

Audio encoding parameters.

property AudioBitrate: Integer;

property AudioSampleRate: Integer;

property AudioChannels: Integer;

property AudioSampleFormat: TRVSampleFormat ²⁴⁹;

AudioSampleRate is a number of audio samples played in 1 second. **AudioSampleFormat** defines format of each audio sample.

AudioChannels is a number of audio channels (1 – mono, 2 – stereo, etc.).

AudioBitrate is a number of bits processed in 1 second when playing sound from a recorded file, it affects compression quality.

Possible values of these properties depend on the chosen AudioCodec ⁸⁸. RVMedia tries to correct values of these properties when passing them to **FFmpeg** (without changing values of the properties themselves), but it is not always possible.

You can use GetListOfAvailableSampleFormats ²²³ to get possible values for **AudioSampleFormat**.

You can use GetListOfAvailableSampleRates ²²³ to get possible values for **AudioSampleRate**.

Unfortunately, not all codecs provide these lists.

Default values:

- AudioBitrate: 64000
- AudioSampleRate: 44100
- AudioChannels: 1
- AudioSampleFormat: *rvsfFloat*

See also

- AudioSource⁽⁸⁹⁾
- AudioCodec⁽⁸⁸⁾

2.1.4.1.3 TRVCamRecorder.AudioCodec, VideoCodec

The properties specify codec types for encoding audio and video.

```
property AudioCodec: TRVAudioCodec(236);
property VideoCodec: TRVVideoCodec(251);
```

These properties are used if AudioCodecName/VideoCodecName⁽⁸⁸⁾ are empty.

You can use GetListOfAvailableFFmpegAudioCodecs⁽²²¹⁾ and GetListOfAvailableFFmpegVideoCodecs⁽²²²⁾ to find possible values of these properties.

For successful file encoding, codecs must be available and compatible with the file format (determined by OutputFileName⁽⁸⁹⁾ file extension).

When codecs are default, the recorder tries to choose default codecs for the file format.

GetVideoFileExts⁽²²⁴⁾ may help to choose a proper video codec for the given file extension.

Warning: Some video and audio formats may be patent-protected in some countries, and supporting these formats will require from you obtaining licenses from the patent owners.

Default values:

rvacDefault, rvvcDefault

2.1.4.1.4 TRVCamRecorder.AudioCodecName, VideoCodecName

The properties specify FFmpeg codecs for encoding audio and video.

```
property AudioCodecName: String;
property VideoCodecName: String;
```

Normally, you should use AudioCodec and VideoCodec⁽⁸⁸⁾ properties instead of these low-level properties.

You can use these properties to define the exact codecs that are not default for specific audio/video stream formats (such as codecs that use hardware acceleration and depend on specific hardware).

For successful file encoding, codecs must be available and compatible with the file format (determined by OutputFileName⁽⁸⁹⁾ file extension).

If codecs with these names are not available, the component falls back to AudioCodec and VideoCodec⁽⁸⁸⁾.

You can use GetListOfAudioEncoders and GetListOfVideoEncoders⁽²²⁰⁾ functions to get possible values of these properties.

Warning: Some video and audio formats may be patent-protected in some countries, and supporting these formats will require from you obtaining licenses from the patent owners.

Default values:

" (empty strings)

2.1.4.1.5 TRVCamRecorder.AudioSource, UseAudio

AudioSource specifies an audio source for recording

property AudioSource: TRVMediaSource⁽¹⁸⁹⁾;

property UseAudio: Boolean;

Assign TRVMicrophone⁽¹¹⁷⁾, TRVCamSound⁽¹¹⁵⁾ or TRVCamReceiver⁽¹²⁰⁾ component to **AudioSource** to use it as a sound source for recording.

AudioSource is used only if **UseAudio** = *True*.

If TRVCamReceiver⁽¹²⁰⁾ receives sound from multiple sources, assign SourceAudioGUID and SourceAudioIndex⁽⁹⁰⁾ properties

Default values:

nil, True

See also:

VideoSource⁽⁹²⁾

2.1.4.1.6 TRVCamRecorder.OutputFileName

Specifies the file name (for recording) or the URL address (for streaming).

property OutputFileName: String;

- If you assign a file name to this property, video/audio will be written to this file when you assign Active⁽⁸⁷⁾ = *True*.
You can use GetListOfAvailableFFmpegFileFormats⁽²²²⁾ function to choose a video file extension.
- If you assign UDP URL to this property, UDP streaming will be started when you assign Active⁽⁸⁷⁾ = *True* (as MPEG-TS).

Default value:

" (*empty string*)

2.1.4.1.7 TRVCamRecorder.Paused

Pauses recording

property Paused: Boolean;

This property may be assigned only when recording is started⁽⁸⁷⁾ (if you assign *True* while not recording, an exception occurs).

2.1.4.1.8 TRVCamRecorder.SourceAudioGUID, SourceAudioIndex

These properties specify the unique identifier of the audio source, if AudioSource⁽⁸⁹⁾ is TRVCamReceiver⁽¹²⁰⁾, and index of its media channel.

property SourceAudioGUID: **String**;

property SourceAudioIndex: **Integer**;

These properties are ignored when this recorder writes sound from TRVMicrophone⁽¹¹⁷⁾.

If AudioSource⁽⁸⁹⁾ is TRVCamReceiver⁽¹²⁰⁾, the recorder writes sound from the specified sender and its media channel.

Initially, a sender identifier is generated in TRVCamSender⁽¹³⁹⁾ component and identifies this sender (TRVCamSender⁽¹³⁹⁾.GUIDFrom⁽¹⁴⁷⁾ property). Then you need to add this identifier to Senders⁽¹²⁵⁾ property of TRVCamReceiver⁽¹²⁰⁾ component. Then you need to connect a recorder to this receiver, and assign its **SourceAudioGUID** equal to one of receiver.Senders[]⁽¹²⁵⁾.GUIDFrom.

If TRVCamSender⁽¹³⁹⁾ has multiple media channels, assign the index of the desired channel to **SourceAudioIndex**; otherwise, leave **SourceAudioIndex** = 0.

Default values

- SourceAudioGUID: "" (empty string)
- SourceAudioIndex: 0

2.1.4.1.9 TRVCamRecorder.SourceVideoGUID, SourceVideoIndex

These properties specify the unique identifier of the Video source, if VideoSource⁽⁹²⁾ is TRVCamReceiver⁽¹²⁰⁾, and index of its media channel.

property SourceVideoGUID: **String**;

property SourceVideoIndex: **Integer**;

These properties are ignored when this recorder writes video from TRVCamera⁽⁴²⁾.

If VideoSource is TRVCamReceiver⁽¹²⁰⁾, the recorder writes video from the specified sender and its media channel.

Initially, a sender identifier is generated in TRVCamSender⁽¹³⁹⁾ component and identifies this sender (TRVCamSender⁽¹³⁹⁾.GUIDFrom⁽¹⁴⁷⁾ property). Then you need to add this identifier to Senders⁽¹²⁵⁾ property of TRVCamReceiver⁽¹²⁰⁾ component. Then you need to connect a recorder to this receiver, and assign its **SourceVideoGUID** equal to one of receiver.Senders[]⁽¹²⁵⁾.GUIDFrom.

If TRVCamSender⁽¹³⁹⁾ has multiple media channels, assign the index of the desired channel to **SourceVideoIndex**; otherwise, leave **SourceVideoIndex** = 0.

Default values

- SourceVideoGUID: "" (empty string)
- SourceVideoIndex: 0

2.1.4.1.10 TRVCamRecorder.Video*

Video encoding parameters

```
property VideoBitrate: Integer;
property VideoFramePerSec: Integer;
property VideoWidth: Integer;
property VideoHeight: Integer;
property VideoAutoSize: Boolean;
```

VideoFramePerSec is a number of video frames playing in 1 second (also known as frame rate).

VideoBitrate is a number of bits processed in 1 second when playing video from a recorded file, it affects compression quality.

If **VideoAutoSize** = *True*, values of **VideoWidth** and **VideoHeight** are ignored, and size of the source video⁽⁹²⁾ is used.

VideoWidth and **VideoHeight** must be even numbers (if not, RVMedia corrects them automatically when passing to *FFmpeg*).

Default values:

- VideoBitRate: 800000
- VideoFramePerSec: 25;
- VideoWidth: 720;
- VideoHeight: 576
- VideoAutoSize: *True*

See also

- VideoSource⁽⁹²⁾
- VideoCodec⁽⁸⁸⁾
- VideoEncodingParameters⁽⁹¹⁾

2.1.4.1.11 TRVCamRecorder.VideoEncodingParameters

Video encoding parameters.

```
property VideoEncodingParameters: TRVVideoEncodingParameters;
```

This property has the following sub-properties:

GroupOfPicturesSize: Integer: the number of pictures in a *group of pictures*. Default value: 12.

MaxBFrameCount: Integer: if it has a non-negative value, defines the maximum number of *B-frames* between non-B-frames. Default value: -1.

ReferenceFrameCount: Integer: if it has a non-negative value, defines the number of *reference frames*. Default value: -1.

See also

- VideoBitrate, VideoFramePerSec, VideoWidth, VideoHeight, VideoAutoSize⁽⁹¹⁾
- VideoSource⁽⁹²⁾
- VideoCodec⁽⁸⁸⁾

2.1.4.1.12 TRVCamRecorder.VideoSource, UseVideo

VideoSource specifies a video source for recording

property VideoSource: TRVMediaSource⁽¹⁸⁹⁾;

property UseVideo: Boolean;

Assign TRVCamera⁽⁴²⁾ or TRVCamReceiver⁽¹²⁰⁾ component to **VideoSource** to use it as a video source for recording.

VideoSource is used only if **UseVideo** = *True*.

If TRVCamReceiver⁽¹²⁰⁾ receives video from multiple sources, assign SourceVideoGUID and SourceVideoIndex⁽⁹⁰⁾ properties.

Default values:

nil, True

See also:

AudioSource⁽⁸⁹⁾

2.1.4.2 Events

In TRVCamRecorder

- OnError⁽⁹²⁾
- OnFirstFrame⁽⁹²⁾
- OnGetImage⁽⁹³⁾

2.1.4.2.1 TRVCamRecorder.OnError

The event occurs in response to an error.

property OnError: TRVCamErrorEvent⁽²³²⁾;

This event allows handling errors that may occur while recording video (with Source parameter = *rvcesFFmpeg*).

Some errors are critical and interrupt video recording. Others are more like warnings, but this event enables you to treat them as errors and stop the current operation if necessary.

This event may be triggered from a background thread, so avoid direct interaction with the user interface. If you need to update the UI, use TThread.Synchronize or TThread.Queue to safely execute code in the main thread.

See also

- TRVCamera.OnError⁽⁷⁰⁾
- TRVAudioPlayer.OnError⁽¹¹⁴⁾

2.1.4.2.2 TRVCamRecorder.OnFirstFrame

This event occurs when a first video frame is received and is ready to be written to a file.

property OnFirstFrame: TNotifyEvent;

2.1.4.2.3 TRVCamRecorder.OnGetImage

This event occurs before the component writes a video frame to a file.

property OnGetImage: TRVImageEvent⁽²³⁵⁾;

If video is scaled by TRVCamRecorder (

The image is returned in **Img** parameter. You must not free **Img** yourself. You can modify this image.

This event is called in a thread context.

2.1.5 TRVCamView

TRVCamView shows video from the specified video source: either from TRVCamera⁽⁴²⁾ or from TRVCamReceiver⁽¹²⁰⁾. Optionally, it can control the camera movement.

Unit [VCL and LCL] MRVCamView;

Unit [FMX] fmxMRVCamView;

Syntax

```
TRVCamView = class (TCustomControl)
```

▼ Hierarchy

VCL and LCL:

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl

FMX:

TObject
TPersistent
TComponent
TFmxObject
TControl
TStyledControl
TTextControl

How to use

Assign TRVCamera⁽⁴²⁾ or TRVCamReceiver⁽¹²⁰⁾ component to VideoSource⁽¹⁰³⁾ property of TRVCamView component, start playing a video.

When using TRVCamReceiver⁽¹²⁰⁾ that receives videos from multiple sources, you can specify the video to display in GUIDFrom and IndexFrom⁽⁹⁸⁾ properties.

Full-screen mode

You can display this viewer expanded to the full screen size by assigning *True* to FullScreen⁽⁹⁷⁾.

You can allow users to switch to/from a full-screen mode by assigning *True* to AllowFullScreen⁽⁹⁵⁾.

You can detect switching modes in OnFullScreen⁽¹⁰⁷⁾ event.

Drawing efficiency

See the note about drawing efficiency in the topic about ViewMode⁽¹⁰³⁾ property.

2.1.5.1 Properties

In TRVCamView

- AllowFullScreen⁽⁹⁵⁾
- AutoSize⁽⁹⁵⁾
- ▶ Camera⁽¹⁰³⁾
- CamMoveMode⁽⁹⁵⁾
- CaptionColor⁽¹⁰¹⁾
- CaptionFont⁽¹⁰¹⁾ [VCL and LCL]
- CaptionTextSettings⁽¹⁰¹⁾ [FMX]
- CaptionHeight⁽¹⁰¹⁾
- CaptionParts⁽¹⁰¹⁾
- ▶ CurRenderMode⁽⁹⁶⁾ [VCL and LCL]
- FocusLineColor⁽⁹⁶⁾
- FrameScaleQuality⁽⁹⁷⁾ [VCL and LCL]
- FullScreen⁽⁹⁷⁾
- ▶ FullScreenView⁽⁹⁸⁾
- GUIDFrom⁽⁹⁸⁾
- HoverLineColor⁽⁹⁸⁾
- IconStyle⁽⁹⁹⁾ (LCL; D2009+)
- IndexFrom⁽⁹⁸⁾
- Language⁽⁹⁹⁾
- ▶ Receiver⁽¹⁰³⁾
- RememberLastFrame⁽⁹⁹⁾
- RenderMode⁽⁹⁶⁾ [VCL and LCL]
- SearchPanelColor⁽⁹⁹⁾
- SearchPanelTextColor⁽⁹⁹⁾
- ShowCameraSearch⁽¹⁰⁰⁾
- ShowCaption⁽¹⁰¹⁾
- Title⁽¹⁰¹⁾
- UseOptimalVideoResolution⁽¹⁰²⁾
- VideoSource⁽¹⁰³⁾
- ViewMode⁽¹⁰³⁾
- WaitAnimationDelay⁽¹⁰⁵⁾

Inherited from TCustomControl

- Align
- Anchors
- Color⁽⁹⁶⁾
- Cursor
- DoubleBuffered

- Enabled
- Font⁽⁹⁷⁾ [VCL and LCL]
- Height
- Hint
- ParentFont⁽⁹⁷⁾ [VCL and VCL]
- PopupMenu
- ShowHint
- TabOrder
- TabStop
- TextSettings⁽⁹⁷⁾ [FMX]
- Visible

2.1.5.1.1 TRVCamView.AllowFullScreen

Allows switching to a full-screen mode.

property AllowFullScreen: Boolean;

If *True*, a special icon is displayed when the user moves the mouse pointer above the control. The user can click this icon to display this viewer in a full screen mode.

To close the full-screen viewer, press a similar icon or **Esc**.

Default value

False

See also

- FullScreen⁽⁹⁷⁾
- OnFullScreen⁽¹⁰⁷⁾
- TRVCamMultiView⁽⁷⁴⁾.AllowFullScreen⁽⁷⁶⁾

2.1.5.1.2 TRVCamView.AutoSize

Specifies whether the control sizes itself automatically to accommodate its contents.

property AutoSize: Boolean;

Default value

False

See also

ViewMode⁽¹⁰³⁾

2.1.5.1.3 TRVCamView.CamMoveMode

Specifies how the viewer controls the camera motion.

property CamMoveMode: TRVCamMoveMode⁽²³⁸⁾;

This property works if VideoSource⁽¹⁰³⁾ is TRVCamera⁽⁴²⁾. The user can control the camera motion (rotation) using the mouse, if supported by the camera.

Default value

vcmmDrag

See also

- OnBeginMove, OnEndMove⁽¹⁰⁷⁾
- TRVCamMultiView⁽⁷⁴⁾.CamMoveMode⁽⁷⁷⁾
- TRVCamera⁽⁴²⁾.Move***⁽⁶⁶⁾ methods
- TRVCamControl⁽³⁹⁾ component

2.1.5.1.4 TRVCamView.Color

Background color.

property Color: TRVMColor⁽²⁴⁴⁾;

Default value [VCL and LCL]

\$00E7BE9F

Default value [FMX]

\$FF9FBEE7

See also

- FocusLineColor⁽⁹⁶⁾
- CaptionColor⁽¹⁰¹⁾
- HoverLineColor⁽⁹⁸⁾

2.1.5.1.5 TRVCamView.CurRenderMode, RenderMode

RenderMode specifies a video rendering method.

CurRenderMode returns a rendering method currently used.

property RenderMode: TRVMRenderMode⁽²⁴⁶⁾;

property CurRenderMode: TRVMRenderMode⁽²⁴⁶⁾;

These properties work only in VCL and LCL for Windows.

If the mode specified in RenderMode cannot be initialized, the component falls back to *rvmrmSoftware*.

Default value

rvmrmSoftware

2.1.5.1.6 TRVCamView.FocusLineColor

Specifies the color of a rectangle shown when the control has the input focus.

property FocusLineColor: TRVMColor⁽²⁴⁴⁾;

Default value [VCL and LCL]

clRed

Default value [FMX]

TAlphaColorRec.Red

See also

- Color⁽⁹⁶⁾
- CaptionColor⁽¹⁰¹⁾
- HoverLineColor⁽⁹⁸⁾

2.1.5.1.7 TRVCamView.Font, ParentFont

The properties specify the font used in:

- the search panel ⁽¹⁰⁰⁾;
- the viewer itself to display “No Video” text.

VCL and LCL:

```
property Font: TFont;
property ParentFont: Boolean;
```

FMX:

```
property Font: TTextSettings;
```

[VCL and LCL note]: To have a control use the same font as its parent control, set **ParentFont** to *True*. If **ParentFont** is *False*, the control uses its own **Font** property.

[FMX note]: TextSettings include Font and FontColor properties.

Default value [VCL and LCL]:

ParentFont: *True*

See also

CaptionFont ⁽¹⁰¹⁾

2.1.5.1.8 TRVCamView.FrameScaleQuality

Specifies the image scaling algorithm.

```
type // defined in MRVType unit
  TRVMFrameScaleQuality = (rvmfscLow, rvmfscNormal);
property FrameScaleQuality: TRVMFrameScaleQuality
```

This property is used only in VCL and LCL for Windows, for all values of RenderMode ⁽⁹⁶⁾.

Value	Animation
<i>rvmfscLow</i>	Low quality, quick, low CPU (or GPU) usage
<i>rvmfscNormal</i>	Normal quality

Frames are scaled according to ViewMode ⁽¹⁰³⁾.

Initial value

rvmfscNormal

2.1.5.1.9 TRVCamView.FullScreen

Switches to/from a full-screen mode.

```
property FullScreen: Boolean;
```

Returns *True* when the viewer is in a full-screen mode.

Assign *True* to this property to show this viewer in a full-screen mode, assign *False* to return back to a normal mode. This assignment works even if AllowFullScreen ⁽⁹⁵⁾ = *False*.

Initial value

False

See also

- FullScreenView⁽⁹⁸⁾
- OnFullScreen⁽¹⁰⁷⁾
- TRVCamMultiView⁽⁷⁴⁾.AllowFullScreen⁽⁷⁹⁾

2.1.5.1.10 TRVCamView.FullScreenView

Returns a full-screen viewer.

property FullScreenView: TRVCamView⁽⁹³⁾;

When this viewer is in a full-screen mode⁽⁹⁷⁾, this property returns a viewer control representing this control on a full screen.

When this viewer is not in a full-screen mode, this property returns *nil*.

See also

- FullScreen⁽⁹⁷⁾
- OnFullScreen⁽¹⁰⁷⁾
- TRVCamMultiView⁽⁷⁴⁾.FullScreenMultiView⁽⁷⁹⁾

2.1.5.1.11 TRVCamView.GUIDFrom, IndexFrom

These properties specify the unique identifier of the video source, if VideoSource⁽¹⁰³⁾ is TRVCamReceiver⁽¹²⁰⁾, and index of its media channel.

property GUIDFrom: String;

property IndexFrom: Integer;

These properties are ignored when this viewer displays video from TRVCamera⁽⁴²⁾.

If VideoSource⁽¹⁰³⁾ is TRVCamReceiver⁽¹²⁰⁾, the viewer displays a video stream having the same GUID value.

Initially, a video identifier is generated in TRVCamSender⁽¹³⁹⁾ component and identifies this sender (TRVCamSender⁽¹³⁹⁾.GUIDFrom⁽¹⁴⁷⁾ property). Then you need to add this identifier to Senders⁽¹²⁵⁾ property of TRVCamReceiver⁽¹²⁰⁾ component. Then you need to connect a viewer to this receiver, and assign its **GUIDFrom** equal to one of receiver.Senders⁽¹²⁵⁾ GUIDFrom.

If TRVCamSender⁽¹³⁹⁾ has multiple media channels, assign the index of the desired channel to **IndexFrom**; otherwise, leave **IndexFrom** = 0.

Default values

- GUIDFrom: '' (empty string)
- IndexFrom: 0

2.1.5.1.12 TRVCamView HoverLineColor

Specifies the color of a rectangle shown when the control is below the mouse pointer;

property HoverLineColor: TRVMColor⁽²⁴⁴⁾;

Default value [VCL and LCL]

\$00B78E5F

Default value [FMX]

\$FF5F8EB7

See also

- FocusLineColor⁽⁹⁶⁾
- CaptionColor⁽¹⁰¹⁾
- Color⁽⁹⁶⁾

2.1.5.1.13 TRVCamView.IconStyle

Specifies an animation that is displayed while the component searches for an IP camera.

property IconStyle: TRVMIconStyle⁽²⁴⁵⁾;

This panel is shown when TRVCamera.SearchCamera⁽⁶⁸⁾ is called, if ShowCameraSearch⁽¹⁰⁰⁾ = *True*.

This property defines not only an animation, but also background and text colors of a search panel (if they are not defined explicitly in SearchPanelColor and SearchPanelTextColor⁽⁹⁹⁾ properties).

Default value

rvisClassic

2.1.5.1.14 TRVCamView.Language

Specifies the language for user interface.

property Language: TRVMLanguage⁽²⁴⁵⁾;

Default value

rvmlEnglish

2.1.5.1.15 TRVCamView.RememberLastFrame

Indicates whether to remember the last video frame.

property RememberLastFrame: Boolean;

If *True*, the control displays the last frame when a video is interrupted or stopped. A blank screen is shown otherwise.

Default value

True

2.1.5.1.16 TRVCamView.SearchPanelColor, SearchPanelTextColor

These properties specify colors of a camera search panel.

property SearchPanelColor: TRVMColor⁽²⁴⁴⁾;

property SearchPanelTextColor: TRVMColor⁽²⁴⁴⁾;

This panel is shown when TRVCamera.SearchCamera⁽⁶⁸⁾ is called, if ShowCameraSearch⁽¹⁰⁰⁾ = *True*.

By default (value of this property is *c/None/Null*), the viewer uses default colors that depend on IconStyle⁽⁹⁹⁾ property.

SearchPanelColor overrides a background color of a search panel.

SearchPanelTextColor overrides a text color of a search panel.

[FMX note]: these colors can be semitransparent (opacity is defined in the color's alpha channel)

Default value [VCL and LCL]*c/None***Default value [FMX]***TAlphaColorRec.Null***2.1.5.1.17 TRVCamView.ShowCameraSearch**

Indicates whether the control displays a special panel while searching for the camera.

property ShowCameraSearch: Boolean;

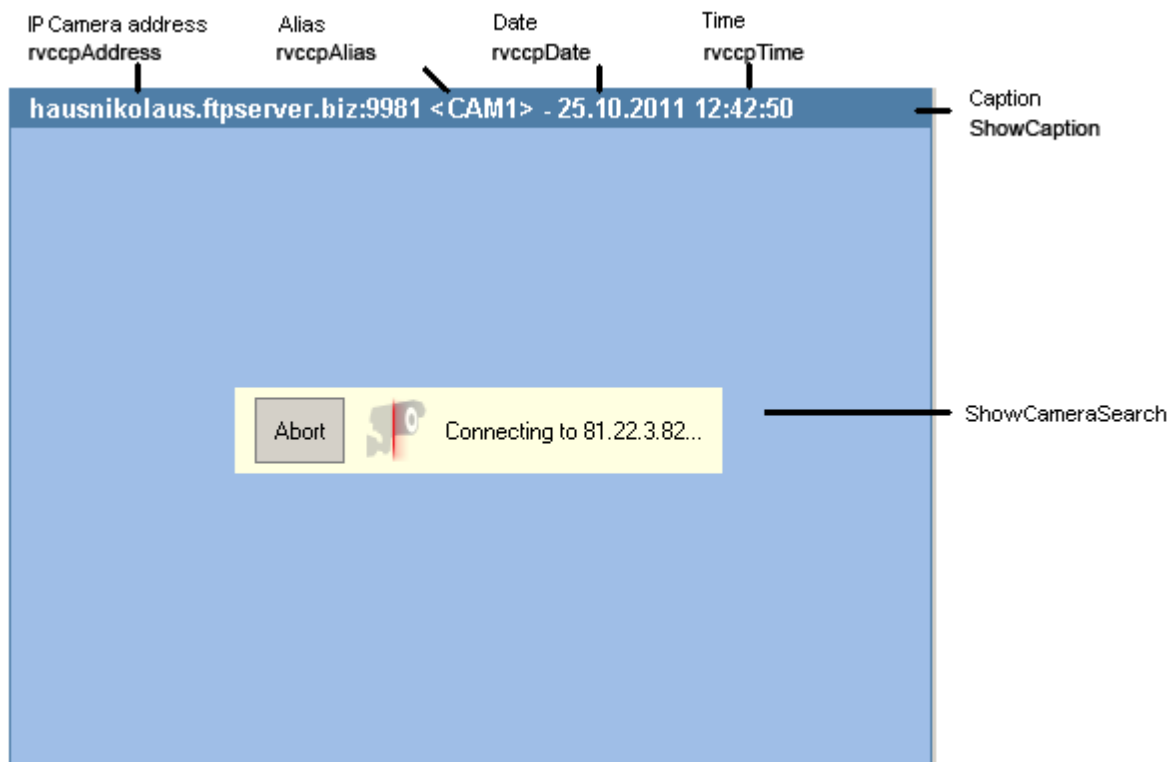
The panel contains some text, animation and “Abort” button.

The “Abort” button uses Font⁽⁹⁷⁾ property for text, its colors are not customizable.

The animation is defined in IconStyle⁽⁹⁹⁾ property.

A color of the panel itself is chosen automatically depending on IconStyle⁽⁹⁹⁾ property, or can be specified explicitly in SearchPanelColor⁽⁹⁹⁾ property.

A text uses Font⁽⁹⁷⁾ property, its color is chosen automatically depending on IconStyle⁽⁹⁹⁾ property, or can be specified explicitly in SearchPanelTextColor⁽⁹⁹⁾.

**Default value***True*

2.1.5.1.18 TRVCamView.ShowCaption, CaptionParts, Title, CaptionColor, CaptionFont, CaptionHeight

The properties define how the window caption is displayed.

```
type // defined in MRVType unit
  TRVCameraCaptionPart = (rvccpAddress, rvccpAlias,
    rvccpDate, rvccpTime);
  TRVCameraCaptionParts = set of TRVCameraCaptionPart;
property ShowCaption: Boolean;
property Title: String;
property CaptionParts: TRVCameraCaptionParts;
property CaptionColor: TRVMColor(244);
property CaptionHeight: Integer;
```

VCL and LCL:

```
property CaptionFont: TFont;
```

FMX:

```
property CaptionTextSettings: TTextSettings;
```

A caption is displayed if **ShowCaption**=*True*. Its background is painted with **CaptionColor**. When activating Delphi XE2+ styles, system colors are changed to the corresponding style colors.

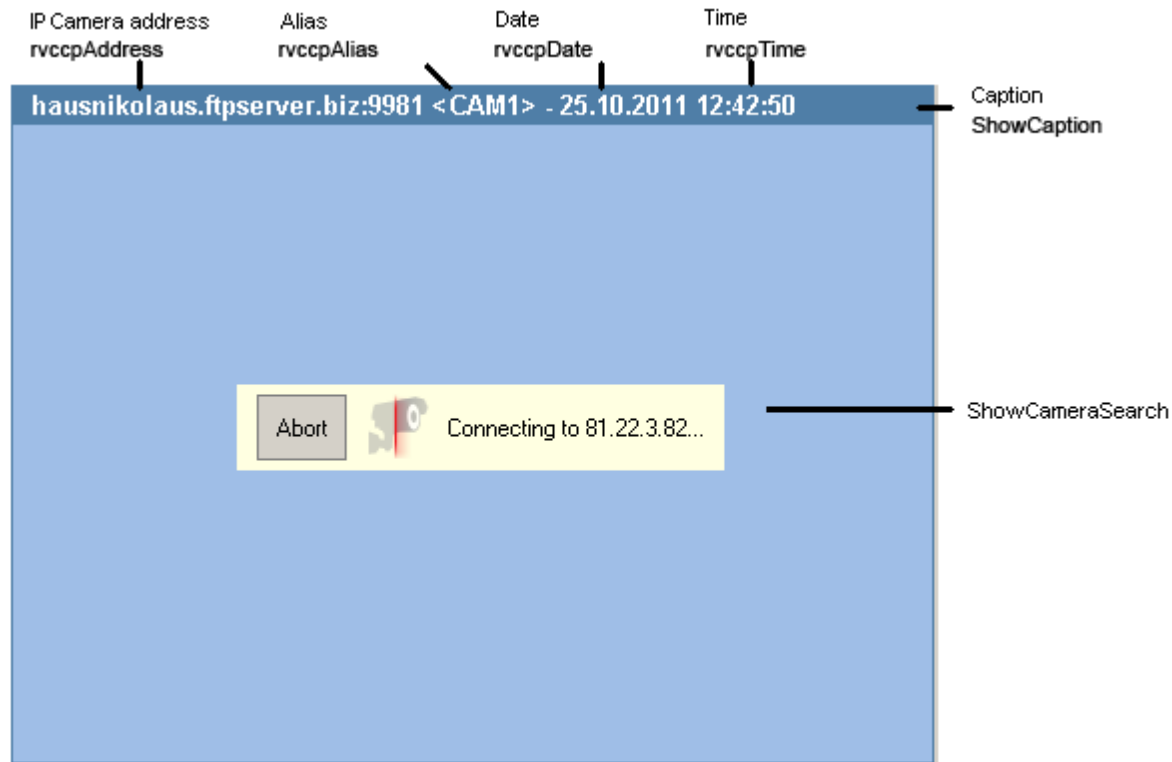
CaptionColor is also used for drawing a frame around the whole window.

[FMX note]: **CaptionColor** can be semi-transparent. In this case, video area is extended to the caption area, and caption is drawn above video.

The caption's text is drawn using **CaptionFont** and consists of **Title** and additional information specified in **CaptionParts**:

Value	Meaning
<i>rvccpAddress</i>	IP camera address
<i>rvccpAlias</i>	RVCamera.Parameters ⁽⁵⁴⁾ .Alias (if playing a video stream, not a file played by PlayVideoFile ⁽⁶⁷⁾)
<i>rvccpDate</i>	the current date
<i>rvccpTime</i>	the current time

The information specified in **CaptionParts** is shown only if VideoSource⁽¹⁰³⁾ is TRVCamera⁽⁴²⁾.



CaptionHeight defines the caption height in 96 DPI screen mode. When the screen DPI is different, the caption height is changed accordingly.

Default values

- ShowCaption: *True*
- Title: "" (empty string)
- CaptionParts: [*rvccpAddress*, *rvccpAlias*]
- CaptionHeight: 20

Default values [VCL and LCL]:

- CaptionFont: Tahoma, 8
- CaptionColor: [\\$00A77E4F](#)

Default values [FMX]:

- CaptionColor: [\\$FF4F7EA7](#)

2.1.5.1.19 TRVCamView.UseOptimalVideoResolution

Indicates whether the control tries to set the video resolution which is optimal for its size.

property UseOptimalVideoResolution;

This property works if VideoSource⁽¹⁰³⁾ is TRVCamera⁽⁴²⁾. If *True*, the control assigns VideoSource.VideoResolution⁽⁶¹⁾ calculated using VideoSource.GetOptimalVideoResolution⁽¹⁹¹⁾, specifying its size in the parameters.

Default value

False

2.1.5.1.20 TRVCamView.VideoSource, Camera, Receiver

VideoSource specifies a video source.

property VideoSource: TRVVideoSource;

You can assign either TRVCamera⁽⁴²⁾ or TRVCamReceiver⁽¹²⁰⁾ component to this property.

property Camera: TRVCamera⁽⁴²⁾; // read-only

property Receiver: TRVCamReceiver⁽¹²⁰⁾; // read-only

If **VideoSource** is TRVCamera⁽⁴²⁾, **Camera** returns the value of **VideoSource**, typecasted to TRVCamera⁽⁴²⁾. Otherwise, **Camera** returns *nil*.

If **VideoSource** is TRVCamReceiver⁽¹²⁰⁾, **Receiver** returns the value of **VideoSource**, typecasted to TRVCamReceiver⁽¹²⁰⁾. Otherwise, **Receiver** returns *nil*.

See also

GUIDFrom⁽⁹⁸⁾

2.1.5.1.21 TRVCamView.ViewMode

Specifies how video frames are positioned and scaled in the viewer.

type

// defined in MRVType unit

TRVCamViewMode = (vvmNormal, vvmCenter, vvmCut, vvmAspect,
vvmStretch);

property ViewMode: TRVCamViewMode;

View Mode	Sample
<p><i>vvmNormal</i></p> <p>A video is displayed at the top left corner of the window, without stretching</p>	

vvmCenter

A video is displayed in the center of the window, without stretching

*vvmCut*

A video is stretched proportionally so that the smallest side fits the window, the largest side is truncated

*vvmAspect* (default)

A video is stretched proportionally to fit the window, keeping the side proportions.



vvmStretch

A video is stretched to fit the window.



[FMX note]: If CaptionColor⁽¹⁰¹⁾ is semi-transparent, the area for placing video includes the caption area. Otherwise (also in VCL and LCL) the area for placing video is below the caption area.

Note about efficiency

Stretched drawing of large frames may take noticeable time and CPU resources, especially in VCL and LCL, so "normal" and "center" modes are faster.

How to make drawing more efficient:

- [VCL and LCL for Windows] lower quality of scaling using FrameScaleQuality⁽⁹⁷⁾ property;
- using "normal" or "center" modes and pre-scale video.

The following TRVCamera⁽⁴²⁾ settings affect video size:

- VideoResolution⁽⁶¹⁾ property;
- GStreamerProperty⁽⁵²⁾.UseVideoScale⁽²⁰⁴⁾ and related properties;
- FFMpegProperty⁽⁵⁰⁾.UseVideoScale⁽²⁰¹⁾ and related properties;
- SetCamVideoMode⁽⁶³⁾ method.

Resizing video using FFMpeg or GStreamer instead of stretch-drawing has the following advantages:

- they resize frames in a thread context, while drawing is performed in the context of the main process;
- if frames are downscaled by FFMpeg or GStreamer, RVMedia can work with smaller images that can be processed faster and take less memory.

Default value

vvmAspect

See also

Autosize⁽⁹⁵⁾

2.1.5.1.22 TRVCamView.WaitAnimationDelay

Specifies the delay (in ms) before displaying an animation that shows that the component waits for a next video frame.

property WaitAnimationDelay: Integer;

This animation is displayed only if VideoSource⁽¹⁰³⁾ is TRVCamera⁽⁴²⁾. A countdown to the animation starts when video starts playing (when TRVCamera.OnStartVideoStream or OnStartVideoFile⁽⁷³⁾ is called) and resets when a next video frame is drawn.

Note 1: Video must be started after linking this viewer to the camera (i.e. OnStartVideoStream or OnStartVideoFile⁽⁷³⁾ must happen when the components are linked), otherwise an animation will not be displayed.

Note 2: The following events stops a countdown and an animation: VideoSource⁽¹⁰³⁾ is changed; video is stopped (when TRVCamera.OnEndVideoStream or OnEndVideoFile⁽⁷⁰⁾ is called).

Assign a zero value to this property to disable this feature.

Default value

3000 (3 seconds)

See also

- TRVCamMultiView⁽⁷⁴⁾.WaitAnimationDelay⁽⁸⁴⁾

2.1.5.2 Events

In TRVCamView

- OnBeginMove⁽¹⁰⁷⁾
- OnEndMove⁽¹⁰⁷⁾
- OnFullScreen⁽¹⁰⁷⁾
- OnMouseEnter⁽¹⁰⁷⁾
- OnMouseLeave⁽¹⁰⁷⁾
- OnPaint⁽¹⁰⁷⁾

Inherited from TCustomControl

- OnClick
- OnContextPopup
- OnEnter
- OnExit
- OnKeyDown
- OnKeyPress
- OnKeyUp
- OnMouseDown
- OnMouseMove
- OnMouseUp
- OnMouseWheel
- OnMouseWheelDown
- OnMouseWheelUp

2.1.5.2.1 TRVCamView.OnBeginMove, OnEndMove

The event occur when the user begins/ends the camera movement in this viewer.

property OnBeginMove: TNotifyEvent

property OnEndMove: TRVCamDoneEvent²³²;

See also

- CamMoveMode⁹⁵

2.1.5.2.2 TRVCamView.OnFullScreen

Occurs when when switching to/from a full-screen view.

property OnFullScreen : TRVFullScreenEvent²³⁴;

See also properties:

- AllowFullScreen⁹⁵
- FullScreen⁹⁷

2.1.5.2.3 TRVCamView.OnMouseEnter, OnMouseLeave

The events occur when the user moves the mouse pointer inside/outside the component.

property OnMouseEnter: TNotifyEvent;

property OnMouseLeave: TNotifyEvent;

2.1.5.2.4 TRVCamView.OnPaint

Occurs when the component needs to paint a video frame.

type // defined in MRVType unit

TRVCamPaintEvent = **procedure** (Sender : TObject; VideoFrame : TBitmap;
ACanvas : TCanvas; **var** CanDrawFrame : Boolean) **of object**;

property OnPaint: TRVCamPaintEvent;

Parameters

VideoFrame – a bitmap image containing a video frame to draw.

Canvas – a canvas where you can draw. However, the recommended way to use this event is modifying **VideoFrame**.

CanDrawFrame specifies whether the component should draw **VideoFrame** onto **Canvas**.

2.1.6 TRVWebCamDialog

TRVWebCamDialog shows a dialog window for changing parameters of local cameras.

Unit [VCL and LCL] MRVWebCamDlg;

Unit [FMX] fmxMRVWebCamDlg;

Syntax

TRVWebCamDialog = **class** (TComponent)

▼ **Hierarchy**

TObject

TPersistent
TComponent

Platform

Dialogs for Windows and Linux have different appearance.

How to use

Assign a TRVCamera⁽⁴²⁾ component to Camera⁽¹⁰⁹⁾ property. In this component, choose the camera for changing properties (assign Camera⁽¹⁰⁹⁾.DeviceType⁽⁴⁹⁾ = *rvdtWebCamera*, assign Camera⁽¹⁰⁹⁾.VideoDeviceIndex⁽⁵⁸⁾).

You can change the dialog language by assigning Language⁽¹¹⁰⁾ property.

Supported properties (Windows)

The dialog allows changing the properties listed below. A property can be changed only if the selected camera supports it. For some properties, users can switch between auto/manual mode.

Video processing amplifier properties:

- brightness
- contrast
- hue
- saturation
- sharpness
- gamma
- color enable (yes/no)
- white balance
- backlight compensation
- gain

Camera control properties:

- pan
- tilt
- roll
- zoom
- exposure
- aperture (iris)
- focus

Supported properties (Linux)

The dialog shows properties supported by the web camera.

Supported properties (macOS)

The dialog shows the following properties:

- zoom
- exposure

- ISO
- focus distance
- white balance

However, macOS does not support changing properties for most cameras.

See also

- TRVCamera⁽⁴²⁾.Brightness, Contrast, Hue, Saturation, Sharpness⁽⁴⁵⁾
- TRVCamera⁽⁴²⁾.Move*⁽⁶⁶⁾ methods

2.1.6.1 Properties

In TRVWebCamDialog

- Camera⁽¹⁰⁹⁾
- Language⁽¹¹⁰⁾

Inherited from TCustomControl

- Align
- Anchors
- Color⁽⁹⁶⁾
- Cursor
- DoubleBuffered
- Enabled
- Font⁽⁹⁷⁾ [VCL]
- Height
- Hint
- ParentFont⁽⁹⁷⁾ [VCL]
- PopupMenu
- ShowHint
- TabOrder
- TabStop
- TextSettings⁽⁹⁷⁾ [FMX]
- Visible

2.1.6.1.1 TRVWebCamDialog.Camera

Specifies the camera for changing its properties.

property Camera: TRVCamera⁽⁴²⁾;

Camera.DeviceType⁽⁴⁹⁾ must be *rvtWebCamera*, otherwise the dialog window is not shown.

See also

- Execute⁽¹¹⁰⁾ method

2.1.6.1.2 TRVWebCamDialog.Language

Specifies a language for the dialog's user interface.

property Language: TRVMLanguage⁽²⁴⁵⁾;

Default value

rvmlEnglish

2.1.6.2 Methods

In TRVWebCamDialog

Execute⁽¹¹⁰⁾

2.1.6.2.1 TRVWebCamDialog.Execute

Shows the dialog window.

function Execute: Boolean;

The dialog window is shown if Camera⁽¹⁰⁹⁾ is assigned, its DeviceType⁽⁴⁹⁾ = *rvdtWebCamera*, and VideoDeviceIndex⁽⁵⁸⁾ is assigned.

The method returns *True* if the dialog was shown (even if changes were canceled).

Changes made in this dialog are applied immediately, so the user can preview results while the dialog is displayed. If the user pressed "Cancel", all properties are restored to the original state.

2.2 Audio

Sound



TRVMicrophone⁽¹¹⁷⁾ – a component for reading sound from a microphone (and from uncompressed WAV files)



TRVCamSound⁽¹¹⁵⁾ – a component for reading sound from videos that are received by TRVCamera⁽⁴²⁾



TRVMicrophoneView⁽¹¹⁷⁾ – a visual component showing a microphone (our other audio source) activity.



TRVAudioPlayer⁽¹¹⁰⁾ – a component for playing and recording sound from TRVMicrophone⁽¹¹⁷⁾, TRVCamSound⁽¹¹⁵⁾ or TRVCamReceiver⁽¹²⁰⁾.

2.2.1 TRVAudioPlayer

A sound playing and recording component.

Unit [VCL and LCL] MRVAudioPlayer;

Unit [FMX] fmxMRVAudioPlayer;

Syntax

```
TRVAudioPlayer = class (TCustomRVAudioPlayer192)
```

▼ Hierarchy

TObject

TPersistent

TComponent

*TCustomRVAudioOutput*¹⁹¹

*TCustomRVAudioPlayer*¹⁹²

Description

This component must be linked to TRVMicrophone¹¹⁷, TRVCamSound¹¹⁵ or TRVCamReceiver¹²⁰ components to play sound.

Without **TRVAudioPlayer**, TRVMicrophone¹¹⁷ cannot play or record sound.

Without **TRVAudioPlayer**, TRVCamSound¹¹⁵ cannot play or record sound, and cannot synchronize video playback speed to audio playback speed.

Without **TRVAudioPlayer**, TRVCamReceiver¹²⁰ can play sound even without **TRVAudioPlayer**, but only on the default audio output device, with default sound parameters, and without recording to a file.

To play sound with this component, assign it to TRVMicrophone¹¹⁷.AudioOutput¹⁸⁸, TRVCamSound¹¹⁵.AudioOutput¹⁸⁸, or TRVCamReceiver¹²⁰.AudioOutput¹⁸⁵ properties.

Playing sound

See the topic on TCustomRVAudioPlayer¹⁹² for properties controlling sound playing.

Recording

The component can record sound to a file. **FFmpeg** library must be available to the application for this feature.

The component supports recording to mp3, ogg, wav, flac and other formats, see EncodeAudioCodec¹¹² property.

Warning: Some audio formats may be patent-protected in some countries, and supporting these formats will require from you obtaining licenses from the patent owners.

Recording is started when you assign *True* to Recording¹¹³ property. Sound is recorded to OutputFileName¹¹³.

Recording is stopped when you assign *False* to Recording¹¹³ property, or change any of Encode*¹¹² properties. When it is stopped, OnStopRecording¹¹⁴ occurs.

Alternatively, you can use TRVCamRecorder⁸⁵ component for sound recording.

Platform notes

Linux (Lazarus): RVMedia uses ALSA (Advanced Linux Sound Architecture) to play sound. If ALSA is not available, it falls back to OSS (Open Sound System). However, an OSS support has less functionality, so ALSA is highly recommended.

2.2.1.1 Properties

In TRVAudioPlayer

- EncodeAudioCodec ⁽¹¹²⁾
- EncodeAudioCodecName ⁽¹¹²⁾
- EncodeBitRate ⁽¹¹²⁾
- EncodeChannels ⁽¹¹²⁾
- EncodeSampleFormat ⁽¹¹²⁾
- EncodeSampleRate ⁽¹¹²⁾
- OutputFileName ⁽¹¹³⁾
- Recording ⁽¹¹³⁾
- UseFFMpeg ⁽¹¹⁴⁾

Inherited from TCustomRVAudioPlayer ⁽¹⁹²⁾

- ▶ AudioOutputDeviceCount ⁽¹⁹⁴⁾
- ▶ AudioOutputDeviceIndex ⁽¹⁹⁴⁾
- ▶ AudioOutputDeviceList ⁽¹⁹⁴⁾
- BufferDuration ⁽¹⁹⁴⁾
- Mute ⁽¹⁹⁴⁾
- NoiseReduction ⁽¹⁹⁴⁾
- NoiseReductionLevel ⁽¹⁹⁴⁾
- Pitch ⁽¹⁹⁵⁾
- VolumeMultiplier ⁽¹⁹⁵⁾

Inherited from TCustomRVAudioOutput ⁽¹⁹¹⁾

- Active ⁽¹⁹²⁾
- Volume ⁽¹⁹²⁾

2.2.1.1.1 TRVAudioPlayer.Encode*

The properties defining parameters for sound recording.

```
property EncodeAudioCodec: TRVAudioCodec (236);
property EncodeAudioCodecName: String;
property EncodeBitrate: Integer;
property EncodeSampleRate: Integer;
property EncodeChannels: Integer;
property EncodeSampleFormat: TRVSampleFormat (249);
```

EncodeSampleRate is a number of audio samples played in 1 second. **EncodeSampleFormat** defines format of each audio sample.

EncodeChannels is a number of audio channels (1 – mono, 2 – stereo, etc.).

EncodeBitrate is a number of bits processed in 1 second when playing sound from a recorded file, it affects compression quality.

EncodeAudioCodec is a codec type. It must correspond to the extension of OutputFileName ⁽¹¹³⁾.

This is a recommended way to specify the file format. You can use

GetListOfAvailableFFmpegAudioCodecs ⁽²²¹⁾ function to get possible values of this property.

EncodeAudioCodecName is a codec name. This is an alternative way to specify the file format. It is useful if you want to use a codec that is not default for the file format (such as codecs that uses a hardware acceleration and depends on a specific hardware). If the specified codec is not available, the component falls back to **EncodeAudioCodec**. You can use `GetListOfAudioEncoders`⁽²²⁰⁾ function to get possible values of this property.

Assign `Recording`⁽¹¹³⁾ = *True* to start recording.

If you assign values to these properties while sound is being recorded, recording is stopped.

Warning: Some audio formats may be patent-protected in some countries, and supporting these formats will require from you obtaining licenses from the patent owners.

Default values

- `EncodeBitrate`: 64000
- `EncodeSampleRate`: 8000
- `EncodeChannels`: 1
- `EncodeAudioCodec`: *rvmeacWAV*
- `tEncodeSampleFormat`: *rvsf8*

2.2.1.1.2 TRVAudioPlayer.OutputFileName

Specifies the file name for sound recording.

property `OutputFileName`: `String`;

If you assign `Recording`⁽¹¹³⁾ = *True*, sound will be written to this file.

The file must have the proper extension corresponding to `EncodeAudioCodec`⁽¹¹²⁾. You can use `GetAudioFileExt`⁽²²⁴⁾ function to choose the extension.

Default value:

" (empty string)

See also

`Recording`⁽¹¹³⁾

2.2.1.1.3 TRVAudioPlayer.Recording

Turn on/off sound recording to a file.

property `Recording`: `Boolean`;

Assign *True* to this property to start recording, assign *False* to stop recording.

This property is independent of `Active`⁽¹⁹²⁾. The component can be inactive (does not play sound on any output device) but still records sound.

When recording is finished, `OnStopRecording`⁽¹¹⁴⁾ occurs.

In the current version, **FFmpeg** is **required** for sound recording.

Default value:

False

See also

- `UseFFMpeg`⁽¹¹⁴⁾

2.2.1.1.4 TRVAudioPlayer.UseFFmpeg

Specifies whether **FFmpeg** must be used for sound recording.

property UseFFmpeg: Boolean;

In the current version, FFmpeg is **required** for sound recording. No sound recording happens if **UseFFmpeg** = *False*.

FFmpeg libraries must be available for the application, otherwise recording will not be performed.

Default value:

True

See also

Recording ⁽¹¹³⁾

2.2.1.2 Events

In TRVAudioPlayer

- OnError ⁽¹¹⁴⁾
- OnGetAudio ⁽¹¹⁴⁾

Inherited from TCustomRVAudioOutput ⁽¹⁹¹⁾

- OnGetAudio ⁽¹⁹²⁾

2.2.1.2.1 TRVAudioPlayer.OnStopRecording

Occurs when recording is stopped

property OnStopRecording;

See also:

Volume ⁽¹¹³⁾

2.2.1.2.2 TRVAudioPlayer.OnError

The event occurs in response to an error.

property OnError: TRVCamErrorEvent ⁽²³²⁾;

This event allows handling errors that may occur while recording audio (with Source parameter = *rvcesFFmpeg*).

Some errors are critical and interrupt audio recording. Others are more like warnings, but this event enables you to treat them as errors and stop the current operation if necessary.

This event may be triggered from a background thread, so avoid direct interaction with the user interface. If you need to update the UI, use TThread.Synchronize or TThread.Queue to safely execute code in the main thread.

See also

- TRVCamera.OnError ⁽⁷⁰⁾
- TRVCamRecorder.OnError ⁽⁹²⁾

2.2.2 TRVCamSound

TRVCamSound reads sound from videos received by TRVCamera⁽⁴²⁾ component.

Unit [VCL and LCL] MRVCamSound;

Unit [FMX] fmxMRVCamSound

Syntax

```
TRVCamSound = class(TCustomRVMicrophone(177))
```

▼ Hierarchy

TObject

TPersistent

TComponent

TRVMediaSource⁽¹⁸⁹⁾

TRVAudioSource⁽¹⁸⁵⁾

TRVAudioSourceWithOutput⁽¹⁸⁷⁾

Description

This component can be considered is an extension of TRVCamera⁽⁴²⁾ component (linked using Camera⁽¹¹⁶⁾ property).

Without **TRVCamSound**, TRVCamera can receive only video without sound.

A pair of TRVCamera+**TRVCamSound** components can receive video with sound.

Limitation

1. This component works only if video is played using **FFmpeg** (from IP camera, network video stream or a local file).
2. The component supports only the following sound formats: mono or stereo sound, with 8 or 16 bits per sample. If a video has another sound format, sound is converted the the supported format.

How to use

Assign TRVCamera⁽⁴²⁾ component to Camera⁽¹¹⁶⁾ property. Make sure that RVCamera.FFmpegProperty⁽⁵⁰⁾.Audio⁽²⁰¹⁾ = *True* (otherwise sound will not be read).

TRVCamSound can be assigned as an AudioSource⁽¹⁴³⁾ to TRVCamSender⁽¹³⁹⁾ to send sound to the network.

TRVCamSound can be assigned as an AudioSource⁽¹⁸⁹⁾ to TRVMicrophoneView⁽¹¹⁷⁾ to visualize sound.

This component does not play sound itself. To play sound, assign TRVAudioPlayer⁽¹¹⁰⁾ component to AudioOutput⁽¹⁸⁸⁾ property. Assigned AudioOutput is important to synchronize video playback to audio playback (otherwise, they are read without attempts to synchronize video and audio).

2.2.2.1 Properties

In TRVCamSound

- Camera⁽¹¹⁶⁾
- GUID⁽¹¹⁶⁾

Inherited from TRVAudioSourceWithOutput⁽¹⁸⁷⁾

- AudioOutput⁽¹⁸⁸⁾

2.2.2.1.1 TRVCamSound.Camera

Defines the component that receives video.

property Camera: TRVCamera⁽⁴²⁾;

No more than one TRVCamSound component can be assigned to each TRVCamera component.

Without TRVCamSound, TRVCamera can receive only video without sound.

2.2.2.1.2 TRVCamSound.GUID

An unique identifier of this component.

property GUID: TRVMAnsiString⁽²⁴⁴⁾;

An unique identifier is used to distinguish different audio sources in TRVAudioPlayer⁽¹¹⁰⁾ component (if TRVAudioPlayer⁽¹¹⁰⁾ plays sound from several **TRVCamSound** components).

Initial value:

auto-generated GUID

2.2.2.2 Events

In TRVCamSound

- OnGetAudioStreamIndex⁽¹¹⁶⁾

2.2.2.2.1 TRVCamSound.OnGetAudioStreamIndex

Allows to choose an audio stream to play.

property OnGetAudioStreamIndex: TRVGetMediaStreamIndexEvent⁽²³⁴⁾;

This event occurs before playing a video file.

This event may be useful if a video source contains more than one audio stream.

If the event is not processed, the first audio stream is played.

Warning: this event is called in a thread context.

You cannot change an audio stream while the video is played: you need to restart video.

See also

TRVCamera.OnGetVideoStreamIndex⁽⁷¹⁾

2.2.3 TRVMicrophone

TRVMicrophone reads sound from a microphone.

Unit [VCL and LCL] MRVMicrophone;

Unit [FMX] fmxMRVMicrophone

Syntax

```
TRVMicrophone = class(TCustomRVMicrophone177)
```

▼ Hierarchy

TObject
TPersistent
TComponent
*TRVMediaSource*¹⁸⁹
*TRVAudioSource*¹⁸⁵
*TRVAudioSourceWithOutput*¹⁸⁷
*TCustomRVMicrophone*¹⁷⁷

Description

This component publishes properties inherited from TCustomRVMicrophone¹⁷⁷.

How to use

TRVMicrophone can be assigned as an AudioSource¹⁴³ to TRVCamSender¹³⁹ to send sound to the network.

TRVMicrophone can be assigned as an AudioSource¹⁸⁹ to TRVMicrophoneView¹¹⁷ to visualize its activity.

This component does not play sound itself. To play sound, assign TRVAudioPlayer¹¹⁰ component to AudioOutput¹⁸⁸ property.

Other audio source components

- TRVCamSound¹¹⁵ (sound from videos received by TRVCamera⁴² component).

Platform notes

Linux (Lazarus): RVMedia uses ALSA (Advanced Linux Sound Architecture) to record sound. If ALSA is not available, it falls back to OSS (Open Sound System). However, an OSS support has less functionality, so ALSA is highly recommended.

2.2.4 TRVMicrophoneView

TRVMicrophoneView shows a microphone (or another audio source) activity. Alternatively, it can indicate sound data received via the network.

Unit [VCL and LCL] MRVMicrophoneViewer;

Unit [FMX] fmxMRVMicrophoneViewer;

Syntax

```
TRVMicrophone = class(TRVAudioViewer188)
```

▼ Hierarchy

VCL and LCL:

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
*TRVAudioViewer*¹⁸⁸

FMX:

TObject
TPersistent
TComponent
TFmxObject
TControl
*TRVAudioViewer*¹⁸⁸

Description

The component displays a volume of a sound signal read from an audio source (*AudioSource*¹⁸⁹) or received via the network (*ReceiverSource*¹⁸⁹).

This component may be used separately, or it can be integrated in *TRVCamMultiView*⁷⁴.

This component does not play sound, it only displays an activity of an audio source visually.

2.2.4.1 Properties

In TRVMicrophoneView

- *Orientation*¹¹⁸
- *Style*¹¹⁹

Inherited from TRVAudioViewer¹⁸⁸

- *AudioSource*¹⁸⁹
- *GUIDFrom*¹⁸⁹
- *ReceiverSource*¹⁸⁹

2.2.4.1.1 TRVMicrophoneView.Orientation

Specifies how the content is oriented.

type // defined in *MRVType* unit

```
TRVMicrophoneOrientation = (rvmpoHorizontal, rvmpoVertical);
```

property Orientation: TRVMicrophoneOrientation;

Default value

rvmpoVertical

See also





- [Style](#)⁽¹¹⁹⁾

2.2.4.1.2 TRVMicrophoneView.Style

Specifies how the sound volume is displayed.

```
type // defined in MRVType unit
  TRVMicrophoneStyle = (rvmpsSimple, rvmpsHistogram, rvmpsGradient,
    rvmpsOwnerDraw);
property Style: TRVMicrophoneStyle;
```

If `Style=rvmpsOwnerDraw`, you can draw the component yourself in `OnPaint`⁽¹¹⁹⁾ event.

Value	Sample
<i>rvmpsSimple</i>	
<i>rvmpsHistogram</i>	
<i>rvmpsGradient</i>	
<i>rvmpsOwnerDraw</i>	 (any image drawn in OnPaint)

Default value

rvmpsHistogram

2.2.4.2 Events**In TRVMicrophoneView**

- [OnPaint](#)⁽¹¹⁹⁾

2.2.4.2.1 TRVMicrophoneView.OnPaint

Allows to draw a custom image showing a sound volume.

```
type // defined in MRVType unit
  TRVMicrophonePaintEvent = procedure (Sender : TObject;
    Level : Integer) of object;
property OnPaint: TRVMicrophonePaintEvent;
```

This event is used in `Style`⁽¹¹⁹⁾=*rvmpsOwnerDraw*.

The sound volume can be calculated basing on the **Level** parameter as `|Level-127|`.

2.3 Network

Network



TRVCamSender⁽¹³⁹⁾ – a component for sending video (from TRVCamera or TRVCamReceiver) and/or audio (from TRVMicrophone or TRVCamReceiver) to TRVCamReceiver or TRVMediaServer via the network.



TRVCamReceiver⁽¹²⁰⁾ – a component for receiving video and audio (from TRVCamSender or TRVMediaServer) via the network.



TRVMediaServer⁽¹⁶⁵⁾ – a component for sending data (video, audio, commands, files) from multiple TRVCamSenders to multiple TRVCamReceivers via the network.



TRVTrafficMeter⁽¹⁷⁵⁾ – a component for displaying traffic charts.

2.3.1 TRVCamReceiver

TRVCamReceiver receives video and audio from TRVCamSender⁽¹³⁹⁾(s) or TRVMediaServer⁽¹⁶⁵⁾ via the network.

Unit [VCL and LCL] MRVReceiver;

Unit [FMX] fmxMRVReceiver;

Syntax

```
TRVCamReceiver = class (TCustomRVReceiver(184))
```

▼ Hierarchy

```

TObject
  TPersistent
    TComponent
      TRVMediaSource(189)
      TRVVideoSource(190)
      TCustomRVReceiver(184)
        TRVVideoSource(190)

```

Description

Assign Active⁽¹²²⁾ = *True* to activate the receiver. This component receives video and audio data via the network from one or more TRVCamSender⁽¹³⁹⁾ components. If the receiver can receive from multiple senders, you should add identifiers of senders to Senders⁽¹²⁵⁾.

In addition to audio and video, a receiver also can receive commands⁽¹³⁵⁾, files⁽¹³⁶⁾, and binary data⁽¹³⁷⁾.

Videos received by the receiver can be displayed in TRVCamView⁽⁹³⁾ or TRVCamMultiView⁽⁷⁴⁾ components.

Video and audio received by the receiver can be sent further using TRVCamSender⁽¹³⁹⁾ components.

Audio data are played on the default sound device, see Volume⁽¹²⁸⁾ and Mute⁽¹²⁴⁾ properties. If you want to choose a sound device, configure advanced sound properties, or record sound to a file, assign TRVAudioPlayer⁽¹¹⁰⁾ component to AudioOutput⁽¹⁸⁵⁾ property.

Lazarus note

In Lazarus, this component must have a windowed control as an owner: you can place it on a form, but you cannot place it on a datamodule.

If the component is created with Owner = nil, it assigns the main form as an owner.

2.3.1.1 Properties

In TRVCamReceiver

- Active⁽¹²²⁾
- AudioLatency⁽¹²²⁾
- BufferDuration⁽¹²²⁾
- BufferSize⁽¹²²⁾
- Color⁽¹²²⁾
- ConnectionProperties⁽¹²³⁾
- FilterSystemCmd⁽¹²³⁾
- GUIDMy⁽¹²³⁾
- JpegIntegrity⁽¹²⁴⁾
- Mute⁽¹²⁴⁾
- Port⁽¹²⁴⁾
- Protocol⁽¹²⁴⁾
- ProxyProperty⁽¹²⁵⁾
- ReceiveMediaTypes⁽¹²⁵⁾
- RetryCount⁽¹²⁵⁾
- Senders⁽¹²⁵⁾
- ▶ SessionKey⁽¹²⁶⁾
- ▶ SessionKey2⁽¹²⁶⁾
- SmoothImage⁽¹²⁶⁾
- ▶ State⁽¹²⁷⁾
- SynchronizedReceiveUserData⁽¹²⁷⁾
- TCPConnectionType⁽¹²⁸⁾
- UseTempFiles⁽¹²⁸⁾
- VideoLatency⁽¹²²⁾
- Volume⁽¹²⁸⁾

Inherited from TCustomRVReceiver⁽¹⁸⁴⁾

- AudioOutput⁽¹⁸⁵⁾

Inherited from TRVVideoSource⁽¹⁹⁰⁾

- ▶ Aborting⁽¹⁹⁰⁾
- FramePerSec⁽¹⁹⁰⁾
- ▶ FramePerSecInt⁽¹⁹⁰⁾

2.3.1.1.1 TRVCamReceiver.Active

Enables receiving.

property Active: Boolean;

Every time when **Active** is changed from *False* to *True*, a value of SessionKey⁽¹²⁶⁾ is changed.

Default value

False

See also

- State⁽¹²⁷⁾

2.3.1.1.2 TRVCamReceiver.AudioLatency, VideoLatency

Latency, in milliseconds.

property VideoLatency: Word;

property AudioLatency: Word;

Video frames/audio fragments are not played, if the specified time has not elapsed.

AudioLatency is important, it makes sure that all audio fragments have been received before playing.

VideoLatency allows to synchronize video with audio. If you do not plan to receive audio, it's recommended to assign **VideoLatency** = 0 to save resources.

Default values

1000

2.3.1.1.3 TRVCamReceiver.BufferDuration

Specifies the buffer size for audio data, in milliseconds.

property BufferDuration: Word;

Default value

1000

2.3.1.1.4 TRVCamReceiver.BufferSize

Used internally

property BufferSize: Cardinal;

See also

- TRVCamSender.BufferSize⁽¹⁴⁴⁾

2.3.1.1.5 TRVCamReceiver.Color

Specifies the color for lost fragments of video frames.

property Color: TRVMColor⁽²⁴⁴⁾;

Default value [VCL and LCL]:

c/None (transparent)

Default value [FMX]:

TRVAlphaColorRec.Null (transparent)

2.3.1.1.6 TRVCamReceiver.ConnectionProperties

Defines connection properties.

property ConnectionProperties: TRVConnectionProperties⁽²⁰⁰⁾;

2.3.1.1.7 TRVCamReceiver.FilterSystemCmd

Specifies whether system commands invoke OnReceiveCmdData⁽¹³⁵⁾ event.

property FilterSystemCmd: Boolean;

Commands having names starting from 'RV_' and 'RVS_' are system commands.

If *True*, this event is not called for system commands.

If *False*, this event is called for system commands. It may be useful for debugging.

Default value

True

2.3.1.1.8 TRVCamReceiver.GUIDMy

An identifier of this receiver.

property GUIDMy: String;

If defined, this receiver accepts only data containing the same identifier of a receiver (TRVCamSender⁽¹³⁹⁾.GUIDTo⁽¹⁴⁷⁾). It helps to ignore unauthorized senders during network attacks.

If empty, this receiver accepts all data.

Additionally, this property is used when this receiver is connected to TRVMediaServer⁽¹⁶⁵⁾ as a part of a client.

Do not assign all-zero GUID (i.e. '{00000000-0000-0000-0000-000000000000}') to this property.

Default value

"" (empty string)

2.3.1.1.9 TRVCamReceiver.IgnoreCorruptedFrames

Specifies how missing or corrupted frames are processed in a chain of frames.

property IgnoreCorruptedFrames: Boolean;

A chain of frames is sent, if a positive value is assigned to TRVCamSender⁽¹³⁹⁾.FrameDifferenceInterval⁽¹⁴⁶⁾. The first frame in a chain is sent as it is, subsequent frames are sent as differences between the new and the old frame.

This property specifies how the receiver works if one frame in a chain is missing or corrupt (it is possible in UDP connection).

If *True*, frames are still shown, but visual artifacts are possible.

If *False*, the rest of chain after a corrupted/missing frame is dropped.

Default value

False

2.3.1.1.10 TRVCamReceiver.JpegIntegrity

Specifies how received jpeg images (video frames) are checked

property JpegIntegrity: TRVJpegIntegrity⁽²⁴³⁾;

Default value

rvjiNone

2.3.1.1.11 TRVCamReceiver.Mute

Allows/disallows playing sound received from the network.

property Mute: Boolean;

This property does not change the system "mute" property of speakers, it mutes only sound received by this component.

If AudioOutput⁽¹⁸⁵⁾ is assigned, this property is ignored, and AudioOutput.Mute⁽¹⁹⁴⁾ is used instead.

See also:

- Volume⁽¹²⁸⁾

2.3.1.1.12 TRVCamReceiver.Port

Specifies the port for a connection.

property Port: Word;

The same value must be specified in ReceiverPort⁽¹⁵⁰⁾ property of all sender⁽¹³⁹⁾ components that will connect to this receiver.

Default value

0

2.3.1.1.13 TRVCamReceiver.Protocol

Defines a protocol used to receive data.

property Protocol: TRVProtocol⁽²⁴⁸⁾;

Recommendation for connection between TRVCamSender⁽¹³⁹⁾ and TRVCamReceiver⁽¹²⁰⁾:

If you need to send not only video and audio, but also other kinds of data (commands, files, etc.), create 2 pairs of Sender+Receiver. For the first Sender, assign Protocol⁽¹⁴⁹⁾=*rvpUDP*, connect it to the first Receiver (having the same value of Protocol property) and use to transfer video and audio data. For the second Sender, assign Protocol⁽¹⁴⁹⁾=*rvpTCP* (or *rvpHTTP*), connect it to the second Receiver (having the same value of Protocol property) and use to transfer commands, files, etc.

Recommendation for connection between TRVCamReceiver⁽¹²⁰⁾ and TRVMediaServer⁽¹⁶⁵⁾:

A connection Server-Receiver must always be HTTP or TCP.

2.3.1.1.14 TRVCamReceiver.ProxyProperty

This property contains sub-properties for proxy settings.

property ProxyProperty: TRVProxyProperty⁽²¹¹⁾;

2.3.1.1.15 TRVCamReceiver.ReceiveMediaTypes

Defines which types of media (video, audio, etc.) the component receives.

property ReceiveMediaTypes: TRVMediaTypes⁽²⁴⁴⁾;

Default value

[rvmtVideo, rvmtAudio, rvmtUserData, rvmtFileData, rvmtCmdData]

See also properties of TRVCamSender⁽¹³⁹⁾:

- SendMediaTypes⁽¹⁵⁰⁾

2.3.1.1.16 TRVCamReceiver.RetryCount

Specifies the maximal count of attempts to connect.

property RetryCount: Integer;

Default value

const MAX_RETRY_COUNT = 5

2.3.1.1.17 TRVCamReceiver.Senders

The collection of potential senders.

property Senders: TRVSenderCollectionEx⁽²¹³⁾;

This collection works differently depending on the side which initiates a connection.

Option 1: Connection from TRVCamSender⁽¹³⁹⁾ to TRVCamReceiver

This type of connection happens if Protocol⁽¹²⁴⁾=rvpUDP, or TCPConnectionType⁽¹²⁸⁾=rvtcpSenderToReceiver.

Senders are used as filters defining which connections can be accepted.

Each item (TRVSenderItemEx⁽²¹⁴⁾) in **Senders** specifies which connections can be accepted from the address specified in item.SenderHost. If the item has non-empty GUIDFrom, the receiver accepts only senders having the same value of GUIDFrom⁽¹⁴⁷⁾. If item.GUIDFrom is empty, the item properties AudioSenders, VideoSenders, UserDataSenders, CmdSenders, FileSenders are used as filters (however, these properties make more sense when communicating with TRVMediaServers⁽²⁴⁴⁾, see below).

If **Senders** is empty, connections from any senders are accepted.

Option 2: Connection from TRVCamReceiver to TRVCamSender⁽¹³⁹⁾

This type of connection happens when Protocol⁽¹²⁴⁾=rvpTCP (or rvpHTTP), and TCPConnectionType⁽¹²⁸⁾=rvtcpReceiverToSender.

The receiver connects to all senders listed in **Senders**. For each item, the receiver connects to the address `item.SenderHost:item.SenderPort`. To make a successful connection, `item.GUIDFrom` must be equal to `sender.GUIDFrom`⁽¹⁴⁷⁾.

Option 3: Connection from TRVMediaServer⁽¹⁶⁵⁾ to TRVCamReceiver

The following settings are required for this mode: Protocol⁽¹²⁴⁾=*rvpTCP* (or *rvpHTTP*), and TCPConnectionType⁽¹²⁸⁾=*rvtcpReceiverToSender*.

In general, properties should have the same settings as with the option 2. However:

- `item.SenderHost` must specify the server address
- `item.GUIDFrom` should be empty,
- the item properties `AudioSenders`, `VideoSenders`, `UserDataSenders`, `CmdSenders`, `FileSenders` may contain lists of clients allowing to send data to this receiver; if they are empty, data from all clients of this server are accepted/rejected, depending on `VideoDefaultAcceptAll`, `AudioDefaultAcceptAll`, `UserDefaultAcceptAll`, `FileDefaultAcceptAll`, `CmdDefaultAcceptAll` properties.

2.3.1.1.18 TRVCamReceiver.SessionKey, SessionKey2

Returns the identifier of the current session.

```
property SessionKey: TRVSessionKey(250);
property SessionKey2: TRVSessionKey(250);
```

Value of **SessionKey** is changed (incremented by 1) every time when `Active`⁽¹²²⁾ becomes *True*. When `Active`⁽¹²²⁾=*False*, this property returns 0.

Value of **SessionKey** is passed as a parameter to events of TRVCamReceiver. If you perform time consuming operations inside an event, it makes sense to compare values of this property and `SessionKey` parameter, to make sure that a connection was not closed or reopened.

Value of **SessionKey** is not necessary equal to the `SessionKey`⁽¹⁵¹⁾ of the connected TRVCamSender⁽¹³⁹⁾ (they are completely independent from each other).

Note: **SessionKey** returns a non-zero value for the whole period of connection, not only when all channels are open (and even in the mode when channels are not created).

SessionKey2 returns the same value as `SessionKey` when `Active`⁽¹²²⁾=*True*. But when `Active`⁽¹²²⁾=*False*, it returns the key of the previous session.

See also

- `State`⁽¹²⁷⁾

2.3.1.1.19 TRVCamReceiver.SmoothImage

Smooths video frames by creating images from several last received video frames.

```
property SourceFileName: String;
```

Experimental property.

If *True*, video frames are smoothed by creating an image from several (3) last received frames.

Pros: removes noise in images, especially if lighting is insufficient.

Contras: blurs moving objects.

We do not recommend using this mode for videos containing fast-changing timers, or if the video has a low frame rate (see `FramePerSec`⁽¹⁹⁰⁾).

Default value

False

See also:

- `TRVCamera`⁽⁴²⁾.`SmoothImage`⁽⁵⁶⁾

2.3.1.1.20 TRVCamReceiver.State

Returns the receiver state.

```
type // Defined in MRVType unit
  TRVMState = (rvmsDisconnect, rvmsConnecting, rvmsConnect,
               rvmsDisconnecting);
property State: TRVMState;
```

Value	Meaning	Corresponding value of Active ⁽¹²²⁾
<i>rvmsDisconnect</i>	The receiver is disconnected	<i>False</i>
<i>rvmsConnecting</i>	The receiver is connecting	<i>True</i>
<i>rvmsConnect</i>	The receiver is connected	<i>True</i>
<i>rvmsDisconnecting</i>	The receiver is disconnecting	<i>True</i>

The component can receive data when `State=rvmsConnect`. It cannot receive data if `State=rvmsDisconnect`. When `State` is equal to `rvmsConnecting` and `rvmsDisconnecting`, the component is waiting, and operations are not available.

See also:

- `SessionKey`⁽¹²⁶⁾

2.3.1.1.21 TRVCamReceiver.SynchronizedReceiveUserData

Defines the thread context for `OnReceiveUserData`⁽¹³⁷⁾ event

```
property SynchronizedReceiveUserData: Boolean;
```

If *False*, this event is called in a thread context.

If *True*, this event is called in the context on the main process.

Default value

True

2.3.1.1.22 TRVCamReceiver.TCPConnectionType

Specifies which side starts a TCP/HTTP connection.

property TCPConnectionType: TRVTCPCConnectionType⁽²⁵¹⁾;

This property is used if Protocol⁽¹²⁴⁾=*rvpTCP* or *rvpHTTP*.

When connecting with TRVCamSender⁽¹³⁹⁾, this value must be equal to the sender's TCPConnectionType⁽¹⁵³⁾.

Default value

rvtcpSenderToReceiver

2.3.1.1.23 TRVCamReceiver.UseTempFiles

Allows using temporary files as buffers.

property UseTempFiles: Boolean;

The receiver can use temporary files as buffers as buffers when receiving files via TRVMediaServer⁽¹⁶⁵⁾.

This settings is required to accept large files.

Default value

True

2.3.1.1.24 TRVCamReceiver.Volume

Returns or changes the volume of speakers for this application.

property Volume: Word;

The range of values is 0..65535.

This property changes the system volume of speakers for the application.

If AudioOutput⁽¹⁸⁵⁾ is assigned, use AudioOutput.Volume⁽¹⁹²⁾ instead.

See also:

- Mute⁽¹²⁴⁾
- TRVAudioSource.Volume⁽¹⁸⁶⁾ (microphone volume)

2.3.1.2 Methods

In TRVCamReceiver

GetMaxChannelCount⁽¹²⁹⁾

GetOpenChannelCount⁽¹²⁹⁾

Inherited from TRVVideoSource⁽¹⁹⁰⁾

Abort⁽¹⁹¹⁾

GetOptimalVideoResolution⁽¹⁹¹⁾

2.3.1.2.1 TRVCamReceiver.GetOpenChannelCount, GetMaxChannelCount

The methods return the count of opened channels and the maximal possible count of channels.

function GetOpenChannelCount: Integer;

function GetMaxChannelCount: Integer;

If TRVCamSender⁽¹³⁹⁾ initiates the connection, the maximal count of channels is 1. If TRVCamReceiver initiates the connection, a channel is created for each data type specified in ReceiveMediaTypes⁽¹²⁵⁾. See the topic about the modes of connections⁽²¹⁾ for the explanations.

See also:

- OnOpenChannel, OnCloseChannel⁽¹³⁵⁾

2.3.1.3 Events

Inherited from TCustomRVReceiver⁽¹⁸⁴⁾

- OnDataRead⁽¹⁸⁵⁾

In TRVCamReceiver

- OnCloseChannel⁽¹³⁵⁾
- OnConnected⁽¹³⁰⁾
- OnConnectError⁽¹³⁰⁾
- OnConnecting⁽¹³⁰⁾
- OnDecodeAudio⁽¹³⁰⁾
- OnDecodeVideo⁽¹³¹⁾
- OnDisconnect⁽¹³⁰⁾
- OnGetAllGroups⁽¹³³⁾
- OnGetAllOnlineUsers⁽¹³³⁾
- OnGetAllUsers⁽¹³³⁾
- OnGetGroupInfo⁽¹³¹⁾
- OnGetGroupUsers⁽¹³⁴⁾
- OnGetImage⁽¹³⁴⁾
- OnMediaAccessCancelRequest⁽¹³⁹⁾
- OnMediaAccessRequest⁽¹³⁹⁾
- OnOpenChannel⁽¹³⁵⁾
- OnReceiveCmdData⁽¹³⁵⁾
- OnReceivedFile⁽¹³⁶⁾
- OnReceiveFileData⁽¹³⁶⁾
- OnReceiveUserData⁽¹³⁷⁾
- OnReceivingFile⁽¹³⁶⁾
- OnSessionConnected⁽¹³⁷⁾
- OnSessionDisconnected⁽¹³⁷⁾
- OnUserEnter⁽¹³⁸⁾
- OnUserExit⁽¹³⁸⁾
- OnUserJoinsGroup⁽¹³⁸⁾
- OnUserLeavesGroup⁽¹³⁸⁾

2.3.1.3.1 TRVCamReceiver.OnConnected, OnConnecting, OnDisconnect, OnConnectError

The events occurring on connection/disconnection to TRVMediaServer⁽¹⁶⁵⁾ or TRVCamSender⁽¹³⁹⁾.

```
property OnConnecting: TRVSocketEvent(235);
property OnConnected: TRVSocketEvent(235);
property OnConnectError: TRVSocketEvent(235);
property OnDisconnect: TRVSocketEvent(235);
```

OnConnecting occurs when a sender starts a connection to the receiver, or when the receiver starts a connection to a sender/server. See the topic about the modes of connections⁽²¹⁾ for the explanations.

After **OnConnecting**, either **OnConnected** or **OnConnectError** occurs. **OnConnected** occurs on a successful connection. **OnConnectError** occurs on a failed connection.

OnDisconnect occurs on a disconnection.

If the sender starts a connection, **MediaTypes** parameter is empty.

If the receiver starts a connection, these events are called in the process of opening channels⁽¹³⁵⁾ for a specific data type, so **MediaTypes** parameter contains a single data type, identifying the channel.

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey⁽¹²⁶⁾ property, to make sure that a connection was not closed or reopened.

See also

- OnOpenChannel, OnCloseChannel⁽¹³⁵⁾
- TRVCamSenders.OnConnected, OnConnecting, OnDisconnect, OnConnectError⁽¹⁶³⁾

2.3.1.3.2 TRVCamReceiver.OnDecodeAudio

Occurs when audio data are received.

```
type // defined in MRVType unit
  TRVDecodeAudioEvent = procedure (Sender: TObject;
    AStream: TMemoryStream; var ADataSize: Integer;
    const AAudioIndex: Word;
    var AStartTime: Int64, var ADuration: Cardinal; var ASamplesPerSec: Integer;
    var ABitsPerSample: TRVBitsPerSample(237); var AChannels: Integer;
    var DoDefault: Boolean) of object;

property OnDecodeAudio: TRVDecodeAudioEvent;
```

Parameters:

AStream contains audio data. Only initial **ADataSize** bytes in this stream are used. Normally, it is raw data with the parameters **ASamplesPerSec**, **ABitsPerSample**, **AChannels**, but they could be encoded in TRVCamSender⁽¹³⁹⁾.OnEncodeAudio⁽¹⁶⁴⁾ event.

ADuration is a sound duration, in milliseconds.

You can play sound yourself. If you do it, assign *False* to **DoDefault** parameter.

Otherwise, you can use this event to decode sound encoded in TRVCamSender⁽¹³⁹⁾.
 .OnEncodeAudio⁽¹⁶⁴⁾.

See also

- OnDecodeVideo⁽¹³¹⁾

2.3.1.3.3 TRVCamReceiver.OnDecodeVideo

Occurs when a video frame (or a changed area of a video frame) is received.

```
type // defined in MRVType unit
  TRVDecodeVideoEvent = procedure (Sender: TObject;
    AStream: TMemoryStream;
    var ImageType: Byte) of object;
```

```
property OnDecodeVideo: TRVDecodeVideoEvent;
```

Use this event to decode images that were encoded in TRVCamSender⁽¹³⁹⁾.OnEncodeVideo⁽¹⁶⁴⁾.

Parameters:

AStream contains image data. **ImageType** identifies the image type. See TRVCamSender⁽¹³⁹⁾.OnEncodeVideo⁽¹⁶⁴⁾ for possible values.

After encoding, if you changed the image format, modify **ImageType** accordingly.

See also

- OnDecodeAudio⁽¹³⁰⁾

2.3.1.3.4 TRVCamReceiver.OnGetGroupInfo

Occurs in response to TRVCamSender⁽¹³⁹⁾.GetGroupInfo⁽¹⁵⁸⁾

```
type
  TRVGroupInfoEvent = procedure (Sender: TRVCamReceiver(120);
    SessionKey: TRVSessionKey(250);
    const GUIDGroup: TRVMAnsiString(244);
    const AGUIDOwner, AGroupName : TRVMAnsiString(244);
    ACmd : TRVCmd) of object;
```

```
property OnGetGroupInfo: TRVGroupInfoEvent;
```

This event occurs when a pair of TRVCamSender⁽¹³⁹⁾ and TRVCamReceiver⁽¹²⁰⁾ (inside a single application) is connected to TRVMediaServer⁽¹⁶⁵⁾ via the network as a client.

TRVCamSender⁽¹³⁹⁾.GetGroupInfo⁽¹⁵⁸⁾ requests information about the group. The server sends this information to a receiver, and **OnGetGroupInfo** occurs.

Parameters:

GUIDGroup – identifier of the group (the same as the parameter of TRVCamSender⁽¹³⁹⁾.GetGroupInfo⁽¹⁵⁸⁾)

AGUIDOwner – identifier of the client who created this group.

AGroupName – name of the group (the same as the parameter of TRVCamSender⁽¹³⁹⁾.JoinGroup⁽¹⁵⁸⁾, when this group was created)

ACmd – a command containing this information.

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey⁽¹²⁶⁾ property, to make sure that a connection was not closed or reopened.

Do not display modal forms in this event

See also:

- OnGetAllGroups⁽¹³³⁾

2.3.1.3.5 TRVCamReceiver.OnRequestJoinGroup

Occurs in response to TRVCamSender⁽¹³⁹⁾.JoinGroup⁽¹⁵⁸⁾

type

```
TRVJoinGroupEvent = procedure (Sender: TRVCamReceiver(120);  
    SessionKey: TRVSessionKey(250); const GUIDGroup: TRVMansiString(244);  
    const AAccess: Boolean; const AError: Integer) of object;
```

property OnRequestJoinGroup: TRVJoinGroupEvent;

This event occurs when a pair of TRVCamSender⁽¹³⁹⁾ and TRVCamReceiver⁽¹²⁰⁾ (inside a single application) is connected to TRVMediaServer⁽¹⁶⁵⁾ via the network as a client.

TRVCamSender⁽¹³⁹⁾.JoinGroup⁽¹⁵⁸⁾ requests to join the specified group on the server. The server sends this information to a receiver, and **OnRequestJoinGroup** occurs.

Parameters:

GUIDGroup – identifier of the group (the same as the parameter of TRVCamSender⁽¹³⁹⁾.JoinGroup⁽¹⁵⁸⁾)

AAccess – *True* if the user successfully (extended information can be received from **AError** parameter)

AError – error (or success) code.

AError Value	Meaning
RV_ERROR_CMD_SUCCESS	The user successfully joined the group (no error).
RV_ERROR_CMD_BAD_PASSWORD	The user did not join the group because the supplied password was incorrect.
RV_ERROR_CMD_GROUP_EXISTS	The user did not join the group because this group does not exist on the server.

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey⁽¹²⁶⁾ property, to make sure that a connection was not closed or reopened.

Warning: Do not display modal forms in this event.

See also:

- OnGetAllGroups⁽¹³³⁾

2.3.1.3.6 TRVCamReceiver.OnGetAllGroups

Occurs in response to TRVCamSender⁽¹³⁹⁾.GetAllGroups⁽¹⁵⁸⁾

property OnGetAllGroups: TRVCmdEvent⁽²³³⁾;

This event occurs when a pair of TRVCamSender⁽¹³⁹⁾ and TRVCamReceiver⁽¹²⁰⁾ (inside a single application) is connected to TRVMediaServer⁽¹⁶⁵⁾ via the network as a client.

This command is supported only if *rvcpUseSystemCmd* and *rvcpCmdAllGroups* are included in TRVMediaServer⁽¹⁶⁵⁾.CmdOptions⁽¹⁶⁸⁾.

TRVCamSender⁽¹³⁹⁾.GetAllGroups⁽¹⁵⁸⁾ requests a list of all groups on the server. The server sends this list to a receiver, and **OnGetAllGroups** occurs.

The list of groups is contained in **ACmd** parameter.

This command has the following parameters:

'GUIDCount' (integer) – count of returned groups

'GUIDGroup1', 'GUIDGroup2', ... (string) – group identifiers (from 1 to the value of 'GUIDCount')

Example

```

procedure TfrmMain.RVCamReceiver1GetAllGroups(Sender: TRVCamReceiver(120);
  SessionKey: TRVSessionKey(250); const GUIDGroup,
  GUIDUser: TRVMAnsiString(244); ACmd: TRVCmd(197));
var
  i, Count : Integer;
  GUIDGroup : TRVMAnsiString(244);
begin
  Count := ACmd.ParamByName('GUIDCount').AsInteger;
  for i := 1 to Count do
  begin
    GUIDGroup :=
      ACmd.ParamByName(TRVMAnsiString('GUIDGroup'+IntToStr(i))).AsString;
    RVCamSender1.GetGroupInfo(GUIDGroup);
    ...
  end;
end;

```

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey⁽¹²⁶⁾ property, to make sure that a connection was not closed or reopened.

Warning: Do not display modal forms in this event.

2.3.1.3.7 TRVCamReceiver.OnGetAllUsers, OnGetAllOnlineUsers

The events occur in response to TRVCamSender⁽¹³⁹⁾.GetAllUsers / GetAllOnlineUsers⁽¹⁵⁸⁾

property OnGetAllUsers: TRVCmdEvent⁽²³³⁾;

property OnGetAllOnlineUsers: TRVCmdEvent⁽²³³⁾;

These events occur when a pair of TRVCamSender⁽¹³⁹⁾ and TRVCamReceiver⁽¹²⁰⁾ (inside a single application) is connected to TRVMediaServer⁽¹⁶⁵⁾ via the network as a client.

These commands are supported only if *rvcpUseSystemCmd* and *rvcpCmdAllUsers* are included in `TRVMediaServer(165).CmdOptions(168)`.

`TRVCamSender(139).GetAllUsers` (or `GetAllOnlineUsers(158)`) request a list of all users on the server. The server sends this list to a receiver, and **OnGetAllUsers** (or **GetAllOnline**) occurs.

The list of users is contained in **ACmd** parameter.

This command has the following parameters:

'GUIDCount' (integer) – count of returned users.

'GUIDUser1', 'GUIDUser2', ... (string) – user identifiers (from 1 to the value of 'GUIDCount')

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and `SessionKey(126)` property, to make sure that a connection was not closed or reopened.

Warning: Do not display modal forms in this event.

2.3.1.3.8 TRVCamReceiver.OnGetGroupUsers

Occurs in response to `TRVCamSender(139).GetUsersFromGroup(158)`

type

```
TRVGetGroupUsersEvent = procedure (Sender: TRVCamReceiver(120);  
    SessionKey: TRVSessionKey(250); const GUIDGroup: TRVMansiString(244);  
    GUIDUsers: TStrings) of object;
```

property OnGetGroupUsers : TRVGetGroupUsersEvent

This event occurs when a pair of `TRVCamSender(139)` and `TRVCamReceiver(120)` (inside a single application) is connected to `TRVMediaServer(165)` via the network as a client.

`TRVCamSender(139).GetUsersFromGroup(158)` requests a list of clients belonging to some group from the server. The server sends this list to a receiver, and **OnGetGroupUsers** occurs.

Parameters:

GUIDGroup – identifier of the group (the same as the parameter of `TRVCamSender(139).GetUsersFromGroup(158)`)

GUIDUsers – a list of identifiers of users belonging to this group.

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and `SessionKey(126)` property, to make sure that a connection was not closed or reopened.

Warning: Do not display modal forms in this event.

See also

- `OnUseJoinsGroup`, `OnUserLeavesGroup(138)`

2.3.1.3.9 TRVCamReceiver.OnGetImage

This event occurs when the component receives a video frame.

property OnGetImage: TRVImageEvent⁽²³⁵⁾;

The image is returned in **Img** parameter. You must not free **Img** yourself. You can modify this image.

AGUIDFrom is an identifier of the sender which sent this image (see TRVCamSender⁽¹³⁹⁾.GUIDFrom⁽¹⁴⁷⁾).

AGUIDFrom is an identifier of the receiver (see TRVCamSender⁽¹³⁹⁾.GUIDTo⁽¹⁴⁷⁾).

This event is called in a thread context.

2.3.1.3.10 TRVCamReceiver.OnOpenChannel, OnCloseChannel

The events occur before and after opening a connection for the specific media type⁽²⁴⁴⁾.

```
property OnOpenChannel: TRVSocketEvent(235);
property OnCloseChannel: TRVSocketEvent(235);
```

A *channel* is a connection for transferring data of one of media types. A receiver can accept up to 5 media types, specified in ReceiveMediaTypes⁽¹²⁵⁾ property. When all channels are opened, a *session* is established.

Channels and sessions are used only when a receiver initiates a connection to a sender/server. See the topic about the modes of connections⁽²¹⁾ for the explanations.

The sequence of events on (successful) connection:

1. For each channel: **OnOpenChannel**, then OnConnecting⁽¹³⁰⁾, then OnConnected⁽¹³⁰⁾;
2. OnSessionConnected⁽¹³⁷⁾.

The sequence of events on disconnection

1. For each channel: OnDisconnect⁽¹³⁰⁾, then **OnCloseChannel**;
2. OnSessionDisconnected⁽¹³⁷⁾.

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey⁽¹²⁶⁾ property, to make sure that a connection was not closed or reopened.

See also

- GetOpenChannelCount, GetMaxChannelCount⁽¹²⁹⁾

2.3.1.3.11 TRVCamReceiver.OnReceiveCmdData

Occurs in response to TRVCamSender⁽¹³⁹⁾.SendCmd⁽¹⁶⁰⁾

```
type // defined in MRVCmd unit
  TRVCmdReadEvent = procedure (Sender: TObject;
    SessionKey: TRVSessionKey(250); Cmd: TRVCmd(197);
    ASocket: TRVSocket(251);
    nGUIDFrom, nGUIDTo, nGUIDGroup : TGUID;
    AMediaIndex : Word) of object;
property OnReceiveCmdData: TRVCmdReadEvent;
```

This event occurs after the receiver receives a command sent by TRVCamSender⁽¹³⁹⁾ (either directly or via TRVMediaServer⁽¹⁶⁵⁾).

Parameters:

Cmd – the command and its parameters.

nGUIDFrom – identifier of the sender which sent the command (TRVCamSender.GUIDFrom⁽¹⁴⁷⁾)

nGUIDGroup – identifier of the group⁽¹⁵⁸⁾ on the server⁽¹⁶⁵⁾, if this command was sent to a group.

AMediaIndex – index of the sender's media channel.

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey⁽¹²⁶⁾ property, to make sure that a connection was not closed or reopened.

Warning: Do not display modal forms in this event.

By default, this event is not called when receiving system commands (having names starting from 'RV_' or 'RVS_'). If you want to call this event for these commands (for example, for debugging), assign FilterSystemCmd⁽¹²³⁾=False.

2.3.1.3.12 TRVCamReceiver.OnReceiveFileData

The event occur in response to TRVCamSender⁽¹³⁹⁾.SendFile⁽¹⁶¹⁾

```
type // defined in MRVType unit
  TRVFileReadEvent = procedure (Sender: TObject;
    SessionKey: TRVSessionKey(250);
    FileName: String; FileOffs, TotalFileSize: Int64;
    AData: TStream; ASocket: TRVSocket(251);
    GUIDFrom, GUIDTo, GUIDGroup: TGUID; AMediaIndex : Word) of object;
  TRVFileEvent = procedure (Sender: TObject; SessionKey : TRVSessionKey;
    FileName: String; TotalFileSize: Int64; ASocket: TRVSocket(251);
    GUIDFrom, GUIDTo, GUIDGroup: TGUID) of object;
property OnReceiveFileData: TRVFileReadEvent;
property OnReceivingFile: TRVFileEvent;
property OnReceivedFile: TRVFileEvent;
```

OnReceivingFile occurs when the receiver starts to receive a file.

OnReceiveFileData occurs (multiple times) while receiving file data.

OnReceivedFile occurs when a file has been received.

Parameters:

AData – received content

nGUIDFrom – identifier of the sender which sent the command (TRVCamSender.GUIDFrom⁽¹⁴⁷⁾)

nGUIDGroup – identifier of the group⁽¹⁵⁸⁾ on the server⁽¹⁶⁵⁾, if this command was sent to a group.

FileName, **FileOffs** correspond to the parameters of TRVCamSender.SendFile⁽¹⁶¹⁾. **TotalFileSize** – the size of the original file.

AMediaIndex – index of the sender's media channel.

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey⁽¹²⁶⁾ property, to make sure that a connection was not closed or reopened.

Warning: Do not display modal forms in this event.

OnReceivingFile and **OnReceivedFile** are called in the context of the main process.

OnReceiveFileData is called in a thread context. Do not update user interface (or make any other operation that requires a context of the main process) in this event.

2.3.1.3.13 TRVCamReceiver.OnReceiveUserData

The event occur in response to TRVCamSender⁽¹³⁹⁾.SendUserData⁽¹⁶¹⁾

```
type // defined in MRVType unit
  TRVDataReadEvent = procedure (Sender: TObject;
    SessionKey: TRVSessionKey(250);
    AData: TStream; ASocket: TRVSocket(251);
    GUIDFrom, GUIDTo, GUIDGroup: TGUID;
    AMediaIndex : Word) of object;
property OnReceiveUserData: TRVDataReadEvent;
```

Parameters:

AData – received content

nGUIDFrom – identifier of the sender which sent the command (TRVCamSender.GUIDFrom⁽¹⁴⁷⁾)

nGUIDGroup – identifier of the group⁽¹⁵⁸⁾ on the server⁽¹⁶⁵⁾, if this command was sent to a group.

AMediaIndex – index of the sender's media channel.

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey⁽¹²⁶⁾ property, to make sure that a connection was not closed or reopened.

Warning: Do not display modal forms in this event.

The event is called in a thread context or a main process context, depending on SynchronizedReceiveUserData⁽¹²⁷⁾ property.

Do not update user interface (or make any other operation that requires a context of the main process) in events called in a thread context.

2.3.1.3.14 TRVCamReceiver.OnSessionConnected, OnSessionDisconnected

The events occur when all channels are connected/disconnected.

```
type // Defined in MRVType unit
  TRVSessionEvent = procedure (Sender: TObject;
    SessionKey: TRVSessionKey(250)) of object;
property OnSessionConnected: TRVSessionEvent;
property OnSessionDisconnected: TRVSessionEvent;
```

Channels and sessions are used only when a receiver initiates a connection to a sender/server. See the topic about the modes of connections⁽²¹⁾ for the explanations.

A *channel* is a connection for transferring data of one of media types. A receiver can accept up to 5 media types, specified in ReceiveMediaTypes⁽¹²⁵⁾ property (even if TRVCamSender does not send⁽¹⁵⁰⁾ some of these data, channels are still opened). These channels make up a *session*. When all channels are opened, a session is created (and **OnSessionConnected** occurs). If at least one channel is closed, a session is terminated (and **OnSessionDisconnected** occurs).

OnSessionDisconnected occurs after an unsuccessful attempt to connect as well (without calling **OnSessionConnected**).

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey⁽¹²⁶⁾ property, to make sure that a connection was not closed or reopened.

2.3.1.3.15 TRVCamReceiver.OnUserEnter, OnUserExit

Occurs in response to TRVCamSender⁽¹³⁹⁾.HelloToDefaultReceivers/GoodbyeToDefaultReceivers⁽¹⁵⁷⁾ and HelloToAllowedSenders/GoodbyeToAllowedSenders⁽¹⁵⁶⁾.

type

```
TRVUserEvent = procedure (Sender: TRVCamReceiver(120);  
    SessionKey: TRVSessionKey(250);  
    const GUIDUser: TRVMAnsiString(244)) of object;
```

property OnUserEnter: TRVUserEvent;

property OnUserExit: TRVUserEvent;

This event occurs if a client is connected to TRVMediaServer⁽¹⁶⁵⁾ via the network.

Parameters:

GUIDUser – identifier of the client which enters or exits.

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey⁽¹²⁶⁾ property, to make sure that a connection was not closed or reopened.

Warning: Do not display modal forms in this event.

2.3.1.3.16 TRVCamReceiver.OnUserJoinsGroup, OnUserLeavesGroup

Occurs in response to TRVCamSender⁽¹³⁹⁾.JoinGroup and LeaveGroup⁽¹⁵⁸⁾.

type

```
TRVGroupEvent = procedure (Sender: TRVCamReceiver(120);  
    SessionKey: TRVSessionKey(250);  
    const GUIDGroup, GUIDUser: TRVMAnsiString(244)) of object;
```

property OnUserJoinsGroup: TRVGroupEvent;

property OnUserLeavesGroup: TRVGroupEvent;

This event occurs if a client is connected to TRVMediaServer⁽¹⁶⁵⁾ via the network.

Parameters:

GUIDGroup – identifier of the group (the same as the parameter of TRVCamSender⁽¹³⁹⁾ JoinGroup and LeaveGroup⁽¹⁵⁸⁾)

GUIDUser – identifier of the client which joins or leaves the group.

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey⁽¹²⁶⁾ property, to make sure that a connection was not closed or reopened.

Warning: Do not display modal forms in this event.

See also

- OnGetGroupUsers⁽¹³⁴⁾

2.3.1.3.17 TRVCamReceiver.OnMediaAccessRequest, OnMediaAccessCancelRequest

Occurs in response to TRVCamSender⁽¹³⁹⁾.SendMediaAccessRequest and SendMediaCancelAccessRequest⁽¹⁶²⁾

type

```
TRVMediaAccessRequestEvent = procedure (Sender: TRVCamReceiver(120);
  SessionKey: TRVSessionKey(250);
  const GUIDGroup, GUIDUser: TRVMAnsiString(244); var Request: Boolean;
  ADataType: Word) of object;
TRVMediaAccessCancelRequestEvent = procedure (
  Sender: TRVCamReceiver(120); SessionKey: TRVSessionKey(250);
  const GUIDGroup, GUIDUser: TRVMAnsiString(244);
  ADataType: Word) of object;
```

property OnMediaAccessRequest: TRVVideoAccessRequestEvent;

property OnMediaAccessCancelRequest: TRVVideoAccessCancelRequestEvent;

When another client calls SendMediaAccessRequest⁽¹⁶²⁾ to this client, **OnMediaAccessRequest** occurs. In this event you can allow sending video and audio to that client by calling TRVCamSender⁽¹³⁹⁾.AllowMediaAccess⁽¹⁵⁸⁾.

When another client calls SendMediaAccessCancelRequest⁽¹⁶²⁾ to this client, **OnMediaAccessCancelRequest** occurs. In this event you should disallow sending data (usually video and audio) to that client by calling TRVCamSender⁽¹³⁹⁾.CancelMediaAccess⁽¹⁵⁸⁾.

Note: these events help to manage a list of default receivers⁽¹⁵⁷⁾ on a media server⁽¹⁶⁵⁾. This is a list of default addressees for data (not only video and audio, but all types of data) that are sent to unspecified addressees.

Parameters:

GUIDGroup – identifier of the group (if the request was made to a group)

GUIDUser – identifier of the client-requester.

ADataType – reserved for future use (planned: it will identify data types for which the request is made, see ***_DATA⁽²¹⁵⁾ constants (this parameter may contain more than one constant, combined using "or" operator)

If you perform time consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey⁽¹²⁶⁾ property, to make sure that a connection was not closed or reopened.

Warning: Do not display modal forms in this event.

2.3.2 TRVCamSender

TRVCamSender sends video and audio to the network.

Unit [VCL and LCL] MRVSender;

Unit [FMX] fmxMRVSender;

Syntax

```
TRVCamSender = class (TCustomRVSender185)
```

▼ Hierarchy

TObject
TPersistent
TComponent
*TCustomRVSender*¹⁸⁵

Description

If Active¹⁴³ = *True*, the component gets the video from VideoSource¹⁵⁵ and audio from AudioSource¹⁴³, and sends them to the network. These video and audio streams can be received by TRVCamReceiver¹²⁰ and TRVMediaServer¹⁶⁵. In addition to audio and video, a sender can send commands¹⁶⁰ and files¹⁶¹.

A video format is specified in Encoding¹⁴⁵.

Media channels

Data may be organized in multiple media channels.

Media channels are useful when you send information via TRVMediaServer¹⁶⁵; for example, they allow for clients to send video from multiple cameras. For direct connections (to TRVCamReceiver¹²⁰), you can use multiple TRVCamSender components instead.

Audio and video for the default (0th) channel are defined in properties VideoSource¹⁵⁵ and AudioSource¹⁴³.

Audio and video for the 1st, 2nd channels and so on are defined in VideoSource and AudioSource properties of items in ExtraMediaSource¹⁴⁶ collection property.

Commands, files, and user data can also be linked to a channel. The corresponding methods have MediaIndex parameter where you can specify the index of media channel.

Connection to TRVCamReceiver¹²⁰ - 1. Making connection

UDP connection is recommended for video and audio, especially for video. It's highly not recommended for other types of data.

HTTP connection is required if proxy servers are used.

Settings common for all options:

Settings for TRVCamSender:

- Active¹⁴³ = *True*

Settings for TRVCamReceiver¹²⁰:

- Active¹²² = *True*

▼ Option 1: UDP connection from Sender to Receiver



Settings for TRVCamSender:

- Protocol⁽¹⁴⁹⁾ = *rvpUDP*
- ReceiverHost:ReceiverPort⁽¹⁵⁰⁾ must be specified.

Settings for TRVCamReceiver⁽¹²⁰⁾:

- Protocol⁽¹²⁴⁾ = *rvpUDP*
- Port⁽¹²⁴⁾ (must be the same as Sender.ReceiverPort⁽¹⁵⁰⁾)

▼ Option 2: TCP (or HTTP) connection from Sender to Receiver



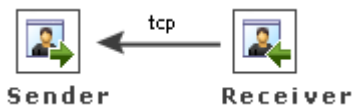
Settings for TRVCamSender:

- Protocol⁽¹⁴⁹⁾ = *rvpTCP* (or *rvpHTTP*)
- ReceiverHost:ReceiverPort⁽¹⁵⁰⁾ must be specified.
- ProxyProperty⁽¹⁴⁹⁾ (optionally, only for *rvpHTTP*)
- TCPConnectionType⁽¹⁵³⁾ = *rvtcpSenderToReceiver*

Settings for TRVCamReceiver⁽¹²⁰⁾:

- Protocol⁽¹²⁴⁾ = *rvpTCP* (or *rvpHTTP*)
- Port⁽¹²⁴⁾ (must be the same as Sender.ReceiverPort⁽¹⁵⁰⁾)
- TCPConnectionType⁽¹²⁸⁾ = *rvtcpSenderToReceiver*

▼ Option 3: TCP (or HTTP) connection from Receiver to Sender



Settings for TRVCamSender:

- Protocol⁽¹⁴⁹⁾ = *rvpTCP* (or *rvpHTTP*)
- SenderPort⁽¹⁵⁰⁾
- TCPConnectionType⁽¹⁵³⁾ = *rvtcpReceiverToSender*

Settings for TRVCamReceiver⁽¹²⁰⁾:

- Protocol⁽¹²⁴⁾ = *rvpTCP* (or *rvpHTTP*)
- Senders⁽¹²⁵⁾ must contain an item with SenderHost:SenderPort⁽²¹⁴⁾ pointing to this sender (the item's SenderPort must be equal to this sender's SenderPort)
- TCPConnectionType⁽¹²⁸⁾ = *rvtcpReceiverToSender*

Connection to TRVCamReceiver⁽¹²⁰⁾ - 2. Using unique identifiers

Unique identifiers may be used to identify senders and receivers.

Identifying receivers

A receiver may (optionally) check if data are really addressed to it. It helps to ignore unauthorized senders during network attacks.

To enable this feature, assign a valid value to TRVCamReceiver.GUIDMy⁽¹²³⁾. If it is assigned, the receiver accepts only data from senders containing the same value in GUIDTo⁽¹⁴⁷⁾ properties.

Identifying senders

In the simplest case, when a single sender is connected to a single receiver, a sender identification is not necessary.

However, a receiver can receive videos from multiple senders; GUIDFrom⁽¹⁴⁷⁾ property allows to distinguish senders. A receiver can either accept data from the specified senders (listed in TRVCamReceiver.Senders⁽¹²⁵⁾), or it can accept data from all senders (in this case, TRVCamReceiver.Senders⁽¹²⁵⁾ must be empty).

Connection to TRVMediaServer⁽¹⁶⁵⁾

See the topic about TRVMediaServer⁽¹⁶⁵⁾.

More information about connection modes

See the overview topic⁽²¹⁾.

Lazarus note

In Lazarus, this component must have a windowed control as an owner: you can place it on a form, but you cannot place it on a datamodule.

If the component is created with Owner = nil, it assigns the main form as an owner.

2.3.2.1 Properties

In TRVCamSender

- Active⁽¹⁴³⁾
- AudioSource⁽¹⁴³⁾
- BufferSize⁽¹⁴⁴⁾
- ChangedAreaProcessingMode⁽¹⁴⁴⁾
- CompressionOptions⁽¹⁴⁴⁾
- CompressionQuality⁽¹⁴⁵⁾
- ConnectionProperties⁽¹⁴⁵⁾
- Encoding⁽¹⁴⁵⁾
- ExtraMediaSources⁽¹⁴⁶⁾
- FilterBlur⁽¹⁴⁶⁾
- FrameDifferenceInterval⁽¹⁴⁶⁾
- FullFrameInterval⁽¹⁴⁷⁾
- GUIDFrom⁽¹⁴⁷⁾
- GUIDTo⁽¹⁴⁷⁾
- GUIDGroup⁽¹⁴⁷⁾
- MinChangeAreaSize⁽¹⁴⁸⁾
- PixelColorThreshold⁽¹⁴⁸⁾
- PixelColorThresholdB⁽¹⁴⁸⁾
- PixelColorThresholdG⁽¹⁴⁸⁾
- PixelColorThresholdR⁽¹⁴⁸⁾
- Protocol⁽¹⁴⁹⁾
- ProxyProperty⁽¹⁴⁹⁾

- ReceiverHost⁽¹⁵⁰⁾
- ReceiverPort⁽¹⁵⁰⁾
- SenderPort⁽¹⁵⁰⁾
- SendMediaTypes⁽¹⁵⁰⁾
- SendOptions⁽¹⁵¹⁾
- ▶ SessionKey⁽¹⁵¹⁾
- ShowCmd⁽¹⁵¹⁾
- SourceAudioIndex⁽¹⁵¹⁾
- SourceGUID⁽¹⁵⁵⁾
- SourceVideoIndex⁽¹⁵²⁾
- TestMode⁽¹⁵⁴⁾
- UseGUID⁽¹⁴⁷⁾
- VideoResolution⁽¹⁵⁴⁾
- VideoSendType⁽¹⁵⁴⁾
- VideoSource⁽¹⁵⁵⁾

2.3.2.1.1 TRVCamSender.Active

Enables/disables video and audio sending.

property Active: Boolean;

Default value

False

See also

- VideoSource⁽¹⁵⁵⁾
- AudioSource⁽¹⁴³⁾
- SendMediaTypes⁽¹⁵⁰⁾

2.3.2.1.2 TRVCamSender.AudioSource

Defines the audio source for the default (the 0th) media channel.

property AudioSource: TRVAudioSource⁽¹⁸⁵⁾;

You can assign TRVMicrophone⁽¹¹⁷⁾ or TRVCamSound⁽¹¹⁵⁾ component to this property. Alternatively, if TRVCamReceiver⁽¹²⁰⁾ is used as a video source⁽¹⁵⁵⁾, it also can be used as an audio source.

Media channels

TRVCamSender can send video and audio in multiple channels. The default (0th) channel is defined by **AudioSource** and VideoSource⁽¹⁵⁵⁾ properties. Other channels are defined in ExtraMediaSources⁽¹⁴⁶⁾ property.

Properties necessary to specify audio source completely

In the simplest case, sound is read from TRVMicrophone or TRVCamSound component assigned to this property. No additional property settings are required.

But when reading sound from TRVCamReceiver, you need to specify additional properties, because this receiver can receive data from multiple senders, and each sender may send multiple media channels. In this case, in addition to assigning TRVCamReceiver to VideoSource⁽¹⁵⁵⁾, you need to specify:

- SourceGUID⁽¹⁵⁵⁾ to specify the sender

- SourceAudioIndex⁽¹⁵¹⁾ to specify media channel of the sender specified in SourceGUID
- See the scheme in the topic about SourceAudioIndex⁽¹⁵¹⁾.

See also

- OnEncodeAudio⁽¹⁶⁴⁾

2.3.2.1.3 TRVCamSender.BufferSize

Defines the maximal size of data fragment for sending.

property BufferSize: Cardinal;

The component sends data fragment by fragment, each fragment has size no more than **BufferSize**.

The larger **BufferSize**, the faster data are sent. However, some OS has limits (that can be different for TCP and UDP connections).

0 is a special value:

- for Windows, a maximum possible fragment size is used (recommended)
- for Linux, 8192 bytes is used.

Warning: some OS do not allow using **BufferSize** larger than 16384 for UDP connections.

Default value

0

See also

- TRVCamReceiver.BufferSize⁽¹²²⁾

2.3.2.1.4 TRVCamSender.ChangedAreaProcessingMode

Specifies how video frames are preprocessed before detecting changed areas

property ChangedAreaProcessingMode: TRVChangedAreaProcessingMode⁽²⁴⁰⁾;

if Encoding⁽¹⁴⁵⁾ = *rvet*Change*, the component compares the previous and the current video frames to find the changed areas. Additional information about this process can be found in the topic about MinChangeAreaSize and PixelColorThreshold⁽¹⁴⁸⁾ properties.

Default value:

rvcapmAuto

See also

- TRVMotionDetector⁽²⁰⁹⁾.ChangedAreaProcessingMode⁽²⁰⁹⁾

2.3.2.1.5 TRVCamSender.CompressionOptions

Defines compression options for video, audio, commands, files, and user data.

property CompressionOptions: TRVCompressionOptions⁽¹⁹⁹⁾;

For different types of data, you can choose one of the following options:

- no compression,
- compression by parts (packets),
- compression as a whole.

2.3.2.1.6 TRVCamSender.CompressionQuality

Specifies the compression quality for images.

property `CompressionQuality: Integer;`

Possible values are in range 1..100.

Jpeg and HWL

Use `CompressionQuality` to set the compression quality of Jpeg and HWL images. The higher a property value (up to a maximum of 100), the better the image quality, but the larger the size. The lower the property value (to a minimum of 1), the smaller the resulting size, but at the expense of picture quality.

Png

For Png images, this option does not affect image quality, but affects compression quality. The higher a property value, the smaller the picture, but more CPU resources are required for encoding it.

Value	Small values of <code>CompressionQuality</code>	Large values of <code>CompressionQuality</code>
<i>Jpeg, HWL</i>	Small data size, low image quality, low CPU usage	Large data size, high image quality, high CPU usage
<i>Png</i>	Large data size, low CPU usage	Small data size, high CPU usage

Default value

90

2.3.2.1.7 TRVCamSender.ConnectionProperties

Defines connection properties.

property `ConnectionProperties: TRVConnectionProperties(200);`

2.3.2.1.8 TRVCamSender.Encoding

Specifies the video encoding.

property `Encoding: TRVEncodingType(242);`

In *rvet*Change* modes, the component sends only changed areas⁽¹⁴⁸⁾ of video frames.

Default value

rvetJPEG

See also

- `FullFrameInterval`⁽¹⁴⁷⁾
- `FrameDifferenceInterval`⁽¹⁴⁶⁾

2.3.2.1.9 TRVCamSender.ExtraMediaSources

Defines additional sources of audio and video which will be sent by this TRVCamSender component.

property ExtraMediaSources: TRVMediaSourceCollection⁽²⁰⁷⁾;

TRVCamSender component may send data organized in multiple media channels.

Audio and video for the default (0th) channel are defined in its properties VideoSource⁽¹⁵⁵⁾, AudioSource⁽¹⁴³⁾ (and SourceGUID⁽¹⁵⁵⁾, SourceVideoIndex⁽¹⁵²⁾, SourceAudioIndex⁽¹⁵¹⁾).

Audio and video for the 1st, 2nd channels and so on are defined in properties of items in its ExtraMediaSource⁽¹⁴⁶⁾ collection property:

- ExtraMediaSource[0] defines the 1st channel,
- ExtraMediaSource[1] defines the 2nd channel,
- and so on.

The type of items of this collection is TRVMediaSourceItem⁽²⁰⁸⁾.

2.3.2.1.10 TRVCamSender.FilterBlur

Allows applying the blur filter to video frames before sending.

property FilterBlur: Boolean;

Default value

False

2.3.2.1.11 TRVCamSender.FrameDifferenceInterval

Enables a mode of sending differences between frames.

property FrameDifferenceInterval: Word

This mode works only if Encoding⁽¹⁴⁵⁾ is *rvet*Change*.

Assigning a positive value to this property turns on a mode where only differences between frames is sent.

The value specified a number of video frames in a chain. For example, if

FrameDifferenceInterval=5, a chain consists of 5 frames: the first frame is sent as it is, the subsequent 4 frames are sent as differences between the new and the old frame.

This settings reduces traffic, but:

- it requires more calculations on the sender side;
- if at least one frame is lost or corrupted (it is possible in UDP mode), drawing subsequent frames leads to artifacts appearing (see also TRVCamReceiver⁽¹²⁰⁾.IgnoreCorruptedFrames⁽¹²³⁾)

The frame counter is reset not only when reaching FrameDifferenceInterval, but also by reaching FullFrameInterval⁽¹⁴⁷⁾, so it does not make sense to have FrameDifferenceInterval>FullFrameInterval⁽¹⁴⁷⁾.

Default value

0

2.3.2.1.12 TRVCamSender.FullFrameInterval

Defines an interval for sending full video frames.

property FullFrameInterval: Integer;

if Encoding⁽¹⁴⁵⁾ = *rvet*Change*, only parts of video frames are sent. If, for some reasons, an initial frame containing the full image was lost, the destination side would display a partial image. To solve this problem, the sender sends a full frame every **FullFrameInterval** time.

Default value

25

See also

- FrameDifferenceInterval⁽¹⁴⁶⁾

2.3.2.1.13 TRVCamSender.GUIDFrom, UseGUID

The properties allow to specify an unique identifier for this sender.

property UseGUID: Boolean

property GUIDFrom: String;

If **UseGUID**=*True*, a **GUIDFrom** is sent with data, allowing for the receiver to distinguish videos from different senders.

If you want to communicate with TRVMediaServer⁽¹⁶⁵⁾ component, **UseGUID** must be *True* (otherwise the server rejects data from this sender).

If **UseGUID**=*False*, data can be sent only to TRVCamReceiver⁽¹²⁰⁾ component accepting data from any GUIDs.

If **UseGUID**=*True*, **GUIDFrom** must not be all-zero (i.e. '{00000000-0000-0000-0000-000000000000}').

Default values

- UseGUID: *True*
- GUID: auto-generated value

2.3.2.1.14 TRVCamSender.GUIDTo, GUIDGroup

The properties specify identifiers of a receiver and a group of receivers.

property GUIDTo: String;

property GUIDGroup: String;

If UseGUID⁽¹⁴⁷⁾=*False*, GUIDTo and GUIDGroup are ignored.

When connected to TRVCamReceiver⁽¹²⁰⁾

If the receiver's GUIDMy⁽¹²³⁾ is empty, it accepts data from senders having any value of **GUIDTo** (empty or not).

If the receiver's GUIDMy⁽¹²³⁾ is assigned, it accepts data only from senders having the same value of **GUIDTo**.

GUIDGroup is ignored.

When connected to TRVMediaServer¹⁶⁵

If **GUIDTo** is not empty, it identifies the client to send data to: the server sends data from this sender to the specified client only.

Otherwise, if **GUIDGroup** is not empty, the server sends data from this sender to clients belonging to the group¹⁵⁸ identified by **GUIDGroup**.

Otherwise, the sender sends data to default receivers¹⁵⁷. If this list is empty, data go to nowhere.

Default values

" (empty string)

See also:

- GUIDFrom¹⁴⁷

2.3.2.1.15 TRVCamSender.MinChangeAreaSize, PixelColorThreshold

The properties specify how the previous and the current video frames are compared, if Encoding¹⁴⁵ = *rvet*Change*

property MinChangeAreaSize: Integer;

property PixelColorThreshold: Integer;

property PixelColorThresholdR: Integer;

property PixelColorThresholdG: Integer;

property PixelColorThresholdB: Integer;

if Encoding¹⁴⁵ = *rvet*Change*, the component compares the previous and the current video frames to find the changed areas.

Let the first pixel is (R1 G1 B1), the second pixel is (R2 G2 B2). If **PixelColorThreshold** >= 0, they are treated as different if $(|R1-R2| + |G1-G2| + |B1-B2|)/3 > \text{PixelColorThreshold}$. Otherwise, they are treated as different if $(|R1-R2| > \text{PixelColorThresholdR})$ or $(|G1-G2| > \text{PixelColorThresholdG})$ or $(|B1-B2| > \text{PixelColorThresholdB})$.

The component calculates rectangles covering changed pixels optimally. Rectangles containing less than **MinChangeAreaSize** changed pixels are ignored.

Then the sender sends only rectangles covering changed areas. Optimal settings for these properties reduce a network traffic and allow to filter out a noise.

Default values

- MinChangeAreaSize: 10
- PixelColorThreshold -R -G -B: 8

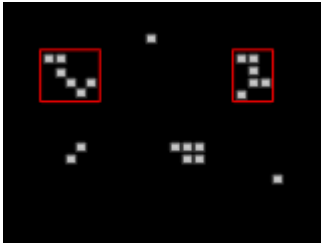
See also

- TestMode¹⁵⁴
- ChangedAreaProcessingMode¹⁴⁴

Example

For example, the sender receives an image from a camera, and then it calculates changed (above the threshold) pixels. On the picture below, unchanged pixels are painted with a black color, changed pixels are shown in a gray color. If MinChangeAreaSize=6, all areas containing less than 6 pixels are ignored.

Two areas containing 6 or more pixels are framed with a red rectangle.



The sender sends only these two areas, and the receiver places them



See also

- `TRVMotionDetector`⁽²⁰⁹⁾.`MinChangeAreaSize`, `PixelColorThreshold`⁽²¹⁰⁾

2.3.2.1.16 TRVCamSender.Protocol

Defines a protocol to use for sending data.

```
property Protocol: TRVProtocol(248);
```

Recommendation for connection between TRVCamSender⁽¹³⁹⁾ and TRVCamReceiver⁽¹²⁰⁾:

If you need to send not only video and audio, but also other kinds of data (commands, files, etc.), create 2 pairs of Sender+Receiver. For the first Sender, assign `Protocol=rvpUDP`, connect it to the first Receiver (having the same value of `Protocol`⁽¹²⁴⁾ property) and use to transfer video and audio data. For the second Sender, assign `Protocol=rvpTCP` (or `rvpHTTP`), connect it to the second Receiver (having the same value of `Protocol`⁽¹²⁴⁾ property) and use to transfer commands, files, etc.

Recommendation for connection between TRVCamSender⁽¹³⁹⁾ and TRVMediaServer⁽¹⁶⁵⁾:

If you need to send not only video and audio, but also other kinds of data (commands, files, etc.), a client may consists of 3 components having the same GUID: two Senders and one Receiver. The first Sender may send video and audio using UDP. The second Sender may send commands and files using HTTP or TCP. A connection Server-Receiver must always be HTTP or TCP.

Default value

rvpUDP

2.3.2.1.17 TRVCamSender.ProxyProperty

This property contains sub-properties for proxy settings.

```
property ProxyProperty: TRVProxyProperty(211);
```

2.3.2.1.18 TRVCamSender.ReceiverHost, ReceiverPort

The properties define the address and port of the receiver (TRVCamReceiver⁽¹²⁰⁾ or TRVMediaServer⁽¹⁶⁵⁾).

property ReceiverHost: String;

property ReceiverPort: Word;

When connecting to TRVCamReceiver⁽¹²⁰⁾, a value of **ReceiverPort** must be equal to its Port⁽¹²⁴⁾ property.

When connecting to TRVMediaServer⁽¹⁶⁵⁾, a value of **ReceiverPort** must be equal either to its HTTPPort⁽¹⁶⁹⁾ or UDPPort⁽¹⁷⁰⁾, depending on Protocol⁽¹⁴⁹⁾.

These properties are used:

- if Protocol⁽¹⁴⁹⁾ = *rvpUDP*, or
- if TCPConnectionType⁽¹⁵³⁾ = *rvtcpSenderToReceiver*

Default values:

- ReceiverHost: ""
- ReceiverPort: 7777

See also:

- ProxyProperty⁽¹⁴⁹⁾

2.3.2.1.19 TRVCamSender.SenderPort

Defines the sender's port in the mode when a receiver initiates a connection to the sender.

property SenderPort: Word;

This property is used if Protocol⁽¹⁴⁹⁾ = *rvpTCP* (or *rvpHTTP*) and TCPConnectionType⁽¹⁵³⁾ = *rvtcpReceiverToSender*.

Default values:

0

2.3.2.1.20 TRVCamSender.SendMediaTypes

Defines which types of media (video, audio, etc.) the component sends.

property SendMediaTypes: TRVMediaTypes⁽²⁴⁴⁾;

Default value

[*rvmtVideo*, *rvmtAudio*, *rvmtUserData*, *rvmtFileData*, *rvmtCmdData*]

See also

- VideoSource⁽¹⁵⁵⁾
- AudioSource⁽¹⁴³⁾
- methods for sending commands⁽¹⁶⁰⁾
- methods for sending files and binary data⁽¹⁶¹⁾

See also properties of TRVCamReceiver⁽¹²⁰⁾:

- ReceiveMediaTypes⁽¹²⁵⁾

2.3.2.1.21 TRVCamSender.SendOptions

Specifies options for sending different types of data.

property SendOptions: TRVSendOptions⁽²¹⁴⁾;

2.3.2.1.22 TRVCamSender.SessionKey

Returns the identifier of the current session.

property SessionKey: TRVSessionKey⁽²⁵⁰⁾;

Value of this property is changed (incremented by 1) every time when Active⁽¹⁴³⁾ becomes *True*. When Active⁽¹⁴³⁾=*False*, this property returns 0.

Value of this property is passed as a parameter to events⁽¹⁶³⁾ of TRVCamSender. If you perform time consuming operations inside an event, it makes sense to compare values of this property and SessionKey parameter, to make sure that a connection was not closed or reopened.

Value of this property is not transferred to connected TRVCamReceiver⁽¹²⁰⁾ or TRVMediaServer⁽¹⁶⁵⁾, so it is a local property. It is not necessary equal to the SessionKey⁽¹²⁶⁾ of the connected TRVCamReceiver (these properties are independent from each other).

2.3.2.1.23 TRVCamSender.ShowCmd

Specifies whether to call OnSendCmd and OnSentCmd⁽¹⁶⁵⁾ events.

property ShowCmd: Boolean;

Default value

False

2.3.2.1.24 TRVCamSender.SourceAudioIndex

Defines the index of media channel of TRVCamReceiver⁽¹²⁰⁾ assigned to VideoSource⁽¹⁵⁵⁾, which will be used for **audio** in the default (0th) media channel.

property SourceAudioIndex: Integer;

In the simplest case, sound is read from TRVMicrophone⁽¹¹⁷⁾ component assigned to AudioSource⁽¹⁴³⁾ property. No additional property settings are required (SourceGUID⁽¹⁵⁵⁾ must be empty, **SourceAudioIndex** must be 0).

But more complex case is possible: this sender is used to re-translate sound received from the network. In this case, sound is taken from TRVCamReceiver⁽¹²⁰⁾ component assigned to VideoSource⁽¹⁵⁵⁾ property.

This receiver can receive data from multiple senders, and each sender may send multiple media channels. In this case, you need to specify:

- SourceGUID⁽¹⁵⁵⁾ to specify the sender which sends audio to this receiver
- **SourceAudioIndex** to specify media channel of the sender specified in SourceGUID

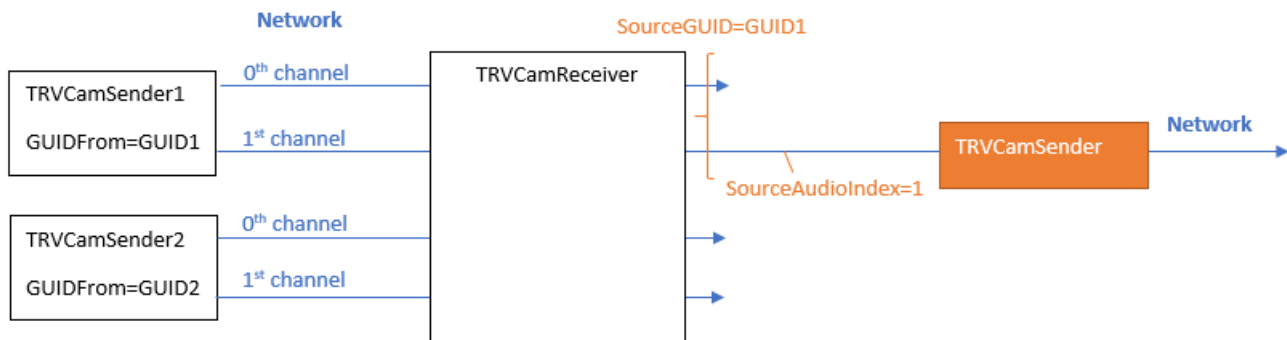
Example

Let we have TRVCamReceiver which receives data from two senders via the network (either directly, or via TRVMediaServer⁽¹⁶⁵⁾): TRVCamSender1 and TRVCamSender2.

Each of these source senders has two media channel (0th and 1st)

We want to re-translate sound from the 1st channel of TRVCamSender1.

Our TRVCamSender component and its properties are colored in orange.



As you can see, we assign SourceGUID⁽¹⁵⁵⁾ property equal to the value of GUIDFrom⁽¹⁴⁷⁾ of TRVCamSender1, and **SourceAudioIndex** = 1.

TRVCamReceiver component is assigned to VideoSource⁽¹⁵⁵⁾ property.

Multiple media channels

AudioSource⁽¹⁴³⁾/VideoSource⁽¹⁵⁵⁾, SourceGUID⁽¹⁵⁵⁾, **SourceAudioIndex** properties define a sound source for the default (0th) media channel of this TRVCamSender component.

Similarly, VideoSource, SourceGUID, SourceVideoIndex⁽¹⁵²⁾ properties define a video source for the default media channel.

More media channels (indexed from 1) can be defined in properties of items of ExtraMediaSources⁽¹⁴⁶⁾ collection property.

Default value

0

2.3.2.1.25 TRVCamSender.SourceVideoIndex

Defines the index of media channel of TRVCamReceiver⁽¹²⁰⁾ assigned to VideoSource⁽¹⁵⁵⁾, which will be used for **video** in the default (0th) media channel.

property SourceVideoIndex: Integer;

In the simplest case, video is read from TRVCamera⁽⁴²⁾ component assigned to VideoSource⁽¹⁵⁵⁾ property. No additional property settings are required (SourceGUID⁽¹⁵⁵⁾ must be empty, **SourceVideoIndex** must be 0).

But more complex case is possible: this sender is used to re-translate video received from the network. In this case, sound is taken from TRVCamReceiver⁽¹²⁰⁾ component assigned to VideoSource⁽¹⁵⁵⁾ property.

This receiver can receive data from multiple senders, and each sender may send multiple media channels. In this case, you need to specify:

- SourceGUID⁽¹⁵⁵⁾ to specify the sender which sends video to this receiver
- **SourceVideoIndex** to specify media channel of the sender specified in SourceGUID

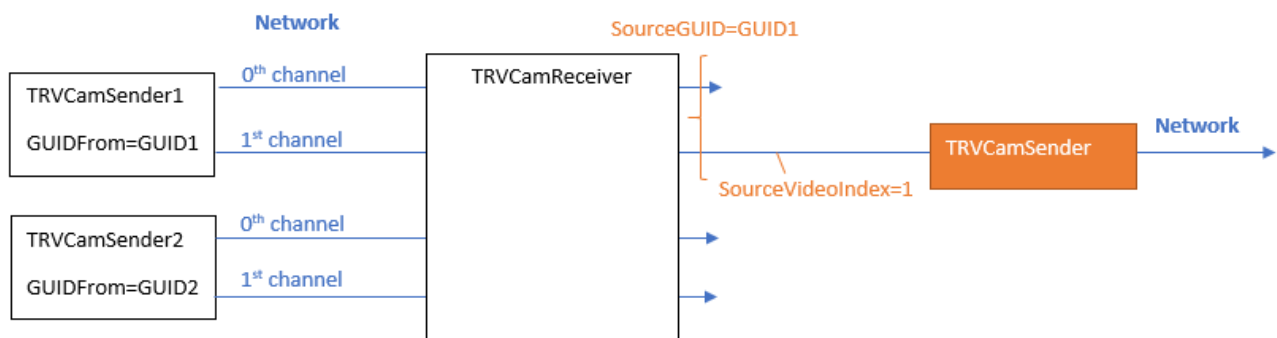
Example

Let we have TRVCamReceiver which receives data from two senders via the network (either directly, or via TRVMediaServer⁽¹⁶⁵⁾): TRVCamSender1 and TRVCamSender2.

Each of these source senders has two media channel (0th and 1st)

We want to re-translate video from the 1st channel of TRVCamSender.

Our TRVCamSender component and its properties are colored in orange.



As you can see, we assign `SourceGUID`⁽¹⁵⁵⁾ property equal to the value of `GUIDFrom`⁽¹⁴⁷⁾ of TRVCamSender1, and **SourceVideoIndex** = 1.

TRVCamReceiver component is assigned to `VideoSource`⁽¹⁵⁵⁾ property.

Multiple media channels

`VideoSource`⁽¹⁵⁵⁾, `SourceGUID`⁽¹⁵⁵⁾, **SourceVideoIndex** properties define a video source for the default (0th) media channel of this TRVCamSender component.

Similarly, `AudioSource`⁽¹⁴³⁾/`VideoSource`, `SourceGUID`, `SourceAudioIndex`⁽¹⁵¹⁾ properties define a audio source for the default media channel.

More media channels (indexed from 1) can be defined in properties of items of `ExtraMediaSources`⁽¹⁴⁶⁾ collection property.

Default value

0

2.3.2.1.26 TRVCamSender.TCPConnectionType

Specifies which side starts a TCP/HTTP connection.

property `TCPConnectionType`: `TRVTCPCConnectionType`⁽²⁵¹⁾;

This property is used if `Protocol`⁽¹⁴⁹⁾ = `rvpTCP` or `rvpHTTP`.

When connecting with TRVCamReceiver⁽¹²⁰⁾, this value must be equal to the receiver's `TCPConnectionType`⁽¹²⁸⁾.

Default value

rvtcpSenderToReceiver

2.3.2.1.27 TRVCamSender.TestMode

Allows sending test data.

property TestMode: TRVBoundsTestMode⁽²³⁷⁾;

rvstmChangedFragments may be used if Encoding⁽¹⁴⁵⁾=rvet*Change. In this mode, instead of sending changed fragments, the component sends special test images, where changed pixels are shown, and changed areas (matching MinChangeAreaSize⁽¹⁴⁸⁾ property) are framed with rectangles. This mode can be useful to find the optimal values of MinChangeAreaSize and PixelColorThreshold⁽¹⁴⁸⁾ properties for the given video source.

Default value

rvstmNone

2.3.2.1.28 TRVCamSender.VideoResolution

The properties allow to reduce the video resolution.

property VideoResolution: TRVVideoResolution⁽²⁵²⁾;

if **VideoResolution** <> *rvDefault*, and **VideoResolution** is less than the video resolution of VideoSource⁽¹⁵⁵⁾, the sender sends video in **VideoResolution**.

Default value

rvDefault

See also

TRVCamera.VideoResolution⁽⁶¹⁾

2.3.2.1.29 TRVCamSender.VideoSendType

Specifies how video is sent.

type

```
TRVMVideoSendType = (rvmvstImageStream, rvmvstVideoStream);
```

property VideoSendType : TRVMVideoSendType;

Value	Meaning
<i>rvmvstImageStream</i>	Frames are sent separately. A connection is established for sending each frame. Several frames can be sent in parallel.
<i>rvmvstVideoStream</i>	Frames are sent in a single connection. This connection is closed only when video is finished. Recommended for slow processors.

This property is used when a sender is connected to a receiver. Otherwise, a permanent connection is always established.

Default value:

rvmvstImageStream

2.3.2.1.30 TRVCamSender.VideoSource, SourceGUID

Defines the video (and may be audio) source.

property VideoSource: TRVVideoSource⁽¹⁹⁰⁾;

property SourceGUID: String

A video source can be either a TRVCamera⁽⁴²⁾ or TRVCamReceiver⁽¹²⁰⁾ component.

If TRVCamReceiver⁽¹²⁰⁾ is assigned to this property, it also can be used as an audio source.

If TRVCamReceiver⁽¹²⁰⁾ is assigned to this property, and this receiver receives videos from multiple sources, you can specify the video (and audio) to display in **SourceGUID** property.

Media channels

TRVCamSender can send video and audio in multiple channels. The default (0th) channel is defined by AudioSource⁽¹⁴³⁾ and **VideoSource** properties. Other channels are defined in ExtraMediaSources⁽¹⁴⁶⁾ property.

Properties necessary to specify video source completely

In the simplest case, video is read from TRVCamera component assigned to this property. No additional property settings are required.

But when reading video from TRVCamReceiver, you need to specify additional properties, because this receiver can receive data from multiple senders, and each sender may send multiple media channels. In this case, in addition to assigning TRVCamReceiver to VideoSource⁽¹⁵⁵⁾, you need to specify:

- **SourceGUID** to specify the sender
- SourceVideoIndex⁽¹⁵²⁾ to specify media channel of the sender specified in **SourceGUID**

See the scheme in the topic about SourceVideoIndex⁽¹⁵²⁾.

2.3.2.2 Methods

In TRVCamSender

AddAllowedSender⁽¹⁵⁶⁾
 AddAllowedSenders⁽¹⁵⁶⁾
 AddDefaultReceiver⁽¹⁵⁷⁾
 AllowMediaAccess⁽¹⁵⁸⁾
 BeginCmd⁽¹⁶⁰⁾
 CancelMediaAccess⁽¹⁵⁸⁾
 ClearAllowedSenders⁽¹⁵⁶⁾
 EndCmd⁽¹⁶⁰⁾
 JoinGroup⁽¹⁵⁸⁾
 HelloToAllowedSenders⁽¹⁵⁶⁾
 HelloToDefaultReceivers⁽¹⁵⁷⁾
 GetAllGroups⁽¹⁵⁸⁾
 GetGroupInfo⁽¹⁵⁸⁾
 GetUsersFromGroup⁽¹⁵⁸⁾
 GoodbyeToAllowedSenders⁽¹⁵⁶⁾
 GoodbyeToDefaultReceivers⁽¹⁵⁷⁾
 LeaveGroup⁽¹⁵⁸⁾

NeedSendFullFrame⁽¹⁶⁰⁾
 Reconnect⁽¹⁶⁰⁾
 RemoveAllowedSender⁽¹⁵⁶⁾
 RemoveDefaultReceiver⁽¹⁵⁷⁾
 RestartServer⁽¹⁶⁰⁾
 SendCmd⁽¹⁶⁰⁾
 SendFile⁽¹⁶⁰⁾
 SendMediaAccessRequest⁽¹⁶²⁾
 SendMediaAccessCancelRequest⁽¹⁶²⁾
 SendUserData⁽¹⁶⁰⁾
 WaitForSendCmd⁽¹⁶⁰⁾

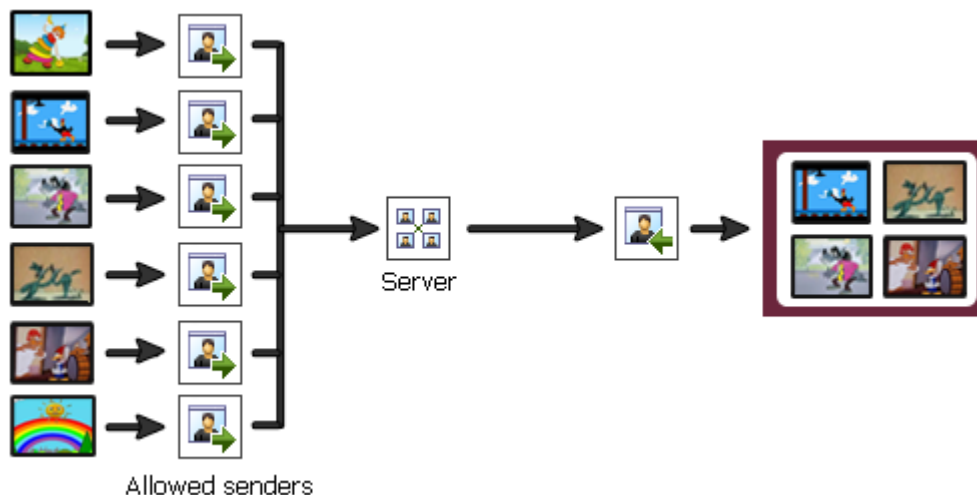
2.3.2.2.1 TRVCamSender.AddAllowedSender and others

Methods allowing to filter out senders on the server.

```

procedure AddAllowedSender (GUID : TGUID);
procedure RemoveAllowedSender (GUID : TGUID);
procedure AddAllowedSenders (GUID : array of TGUID; Count : Integer);
procedure ClearAllowedSenders (AllowAll: Boolean);
procedure HelloToAllowedSenders;
procedure GoodbyeToAllowedSenders;
  
```

The methods work only if this sender is connected to TRVMediaServer⁽¹⁶⁵⁾ via the network. They allow to define a list of clients that can send data to this client.



Initially, this list is empty, the server can send data to this client from all clients. If at least one client is added in this list, the server can send to this client data only from the clients included in this list.

AddAllowedSender/RemoveAllowedSender adds/deletes users to the list of allowed senders. **AddAllowedSenders** adds multiple users. **ClearAllowedSenders** removes all allowed senders.

If **ClearAllowedSenders** is called with **AllowAll=True**, this client can receive messages from all clients, like in the initial state. If **ClearAllowedSenders** is called with **AllowAll=False**, or the last allowed sender is removed by calling **RemoveAllowedSender**, this client does not accept data from any other client.

A client may inform its allowed senders about its presence by calling **HelloToAllowedSenders**, and inform them about exiting by calling **GoodbyeToAllowedSenders**. If **HelloToAllowedSenders** was called, and connection between this client and the sever is broken, the server itself informs allowed senders about this client exiting. Allowed senders are informed in `OnUserEnter` and `OnUserExit` ⁽¹³⁸⁾ events.

The list of allowed senders can be kept by the server when this client disconnects, if you change `TRVMediaServer.KeepClientInfoMode` ⁽¹⁶⁹⁾.

In addition to filtering on a server, a receiver may filter senders locally, see `TRVCamReceiver` ⁽¹²⁰⁾.
`.Senders` ⁽¹²⁵⁾.

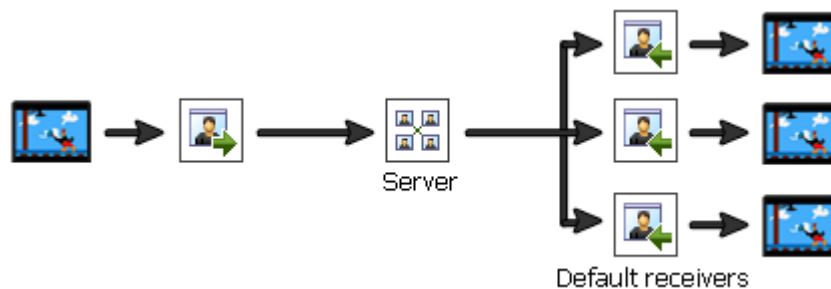
2.3.2.2.2 TRVCamSender.AddDefaultReceiver and others

Methods working with a list of default receivers on the server.

```
procedure AddDefaultReceiver(GUID : TGUID);
procedure RemoveDefaultReceiver(GUID : TGUID);
procedure HelloToDefaultReceivers;
procedure GoodbyeToDefaultReceivers;
```

The methods work only if this sender is connected to `TRVMediaServer` ⁽¹⁶⁵⁾ via the network.

Audio and video are sent to default receivers if `GUIDTo` and `GUIDGroups` ⁽¹⁴⁷⁾ are empty.
Commands ⁽¹⁶⁰⁾ and files ⁽¹⁶¹⁾ can be sent to default receivers as well.



A server may create a list of default receivers for each client. A client itself adds and removes default receivers by calling **AddDefaultReceiver/RemoveDefaultReceiver**.

A client may inform its default receivers about its presence by calling **HelloToDefaultReceivers**, and inform them about exiting by calling **GoodbyeToDefaultReceivers**. If **HelloToDefaultReceivers** was called, and connection between this client and the sever is broken, the server itself informs default receivers about this client exiting. Default receivers are informed in `OnUserEnter` and `OnUserExit` ⁽¹³⁸⁾ events.

The list of default receivers can be kept by the server when this client disconnects, if you change `TRVMediaServer.KeepClientInfoMode` ⁽¹⁶⁹⁾.

In addition, the component provides a set of methods for simplifying a management of default receivers:

- `SendMediaAccessRequest`, `SendMediaAccessCancelRequest` ⁽¹⁶²⁾ (requests for addition/removal in the list of default receivers)
- `AllowMediaAccess`, `CancelMediaAccess` ⁽¹⁵⁸⁾ (alternative methods for adding and removing to the list of default receivers)

2.3.2.2.3 TRVCamSender.AllowMediaAccess, CancelMediaAccess

The methods simplify management of default receivers ⁽¹⁵⁷⁾.

```
procedure AllowMediaAccess(const GUID: TRVMAnsiString(244));
procedure CancelMediaAccess(const GUID: TRVMAnsiString(244));
```

These methods are useful when TRVCamSender is connected to TRVMediaServer ⁽¹⁶⁵⁾ as a part of a client.

AllowMediaAccess allows sending video and audio to another client identified by **GUID** (it is implemented by adding that client to the list of default receivers ⁽¹⁵⁷⁾). Usually, this method is called from TRVCamReceiver ⁽¹²⁰⁾.OnMediaAccessRequest ⁽¹³⁹⁾ event.

CancelMediaAccess stops sending video and audio to another client identified by **GUID** (it is implemented by excluding that client from the list of default receivers ⁽¹⁵⁷⁾). Usually, this method is called from TRVCamReceiver ⁽¹²⁰⁾.OnMediaAccessCancelRequest ⁽¹³⁹⁾ event.

See the example in the topic about SendMediaAccessRequest and SendMediaAccessCancelRequest ⁽¹⁶²⁾.

2.3.2.2.4 TRVCamSender.GetAllUsers, GetAllOnlineUsers

The methods request a list of users on the server.

```
procedure GetAllUsers;
procedure GetAllOnlineUsers;
```

These methods work when a pair of TRVCamSender ⁽¹³⁹⁾ and TRVCamReceiver ⁽¹²⁰⁾ (inside a single application) is connected to TRVMediaServer ⁽¹⁶⁵⁾ via the network as a client.

The methods request a list of users. The server sends this information to a receiver, and OnGetAllUsers / OnGetAllOnlineUsers ⁽¹³³⁾ occurs.

If the server's KeepClientInfoMode ⁽¹⁶⁹⁾ = *rvkclmWhileOnline*, these methods return the same lists. Otherwise, the servers stores information about offline clients.

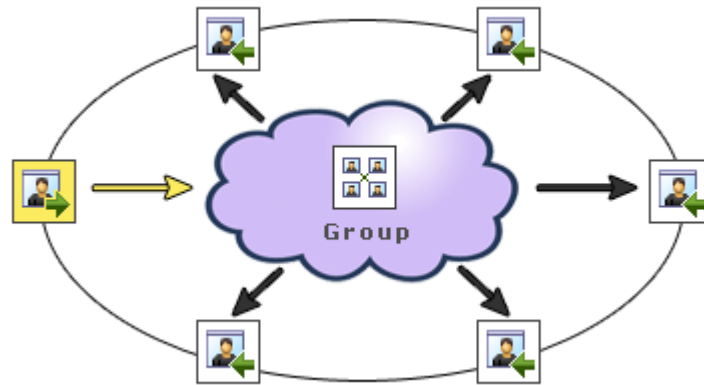
2.3.2.2.5 TRVCamSender.JoinGroup, LeaveGroup, etc.

Methods working with groups on a server.

```
procedure JoinGroup(AGUIDGroup: TGUID; Permanent: Boolean = False;
  AGroupName: TRVMAnsiString = ''; AGroupPassword: TRVMAnsiString = '';
  OnlyExistingGroup: Boolean = False);
procedure LeaveGroup(GUIDGroup: TGUID);
procedure GetUsersFromGroup(GUIDGroup: TGUID; Online: Boolean);
procedure GetAllGroups;
procedure GetGroupInfo(AGUIDGroup: TGUID);
```

The methods work only if this sender is connected to TRVMediaServer ⁽¹⁶⁵⁾ via the network.

A server may have several groups of clients (users). Users belonging to a group may exchange data with each other. A group is identified by an unique identifier (GUIDGroup). Groups can be used to implement chat rooms.



Joining and leaving

JoinGroup adds this sender to the group (identified by GUIDGroup) on the server. If the parameter **OnlyExistingGroup** = *True*, the user joins group only if it already exists on the server, otherwise a new group is created (if the count of groups on the server does not exceed TRVMediaServer.MaxGroupCount⁽¹⁷⁰⁾).

When a new user joins an existing group, all members of this group receive notifications about this new user (OnUserJoinsGroup⁽¹³⁸⁾ event of their receivers).

The user can specify the group name and password. If this method creates a new group, all other users must specify the same password to join this group.

If the parameter **Permanent** = *False*, the user becomes a member of the group until he/she calls **LeaveGroup**, or until the connection to this client is disconnected. All other users are informed when the user leaves this group (OnUserLeavesGroup⁽¹³⁸⁾ events of the receiver).

If the parameter **Permanent** = *True*, the user cannot be removed in the group. When the user calls **LeaveGroup**, or when he/she is disconnected to the server, he/she becomes "offline", but he/she still a member of the group. Such user can be removed from the group only after joining to it again with **Permanent** = *False*. This feature allows using groups as contact lists.

LeaveGroup deletes this sender from the group. All members of this group receive notifications about this user exiting the group (OnUserLeavesGroup⁽¹³⁸⁾ event of their receivers). If it was the last user of the group, the group is deleted on the server. If the connection between this client and the server is broken, the server itself informs the group members about this user exiting.

Receiving information

GetUsersFromGroup receives a list of users belonging to the group. If the parameter **Online**=*True*, it returns only group users which are currently connected. A list of users is returned in OnGetGroupUsers⁽¹³⁴⁾ of the receiver.

GetAllGroups returns a list of groups on the server. This command is supported only if *rvcpUseSystemCmd* and *rvcpCmdAllGroups* are included in TRVMediaServer⁽¹⁶⁵⁾.CmdOptions⁽¹⁶⁸⁾. A list of groups is returned in OnGetAllGroups⁽¹³³⁾ of the receiver.

GetGroupInfo returns information about the specific group (its name and creator). This information is returned in OnGetGroupInfo⁽¹³¹⁾ of the receiver.

See also:

- GUIDGroup⁽¹⁴⁷⁾ property

- information about groups in the topic about TRVMediaServer⁽¹⁶⁵⁾

2.3.2.2.6 TRVCamSender.NeedSendFullFrame

Request sending a full frame as soon as possible.

procedure NeedSendFullFrame;

This method may be useful if Encoding⁽¹⁴⁵⁾ = *rvet*Change*. In these modes, a full frame is sent every FullFrameInterval⁽¹⁴⁷⁾ frames. If full frames are not sent for a long time (when video frame rate is low, and/or video image is static so frames are not sent because they are identical), a receiver can wait a considerable time before it can start displaying video (because it can start displaying video only after receiving a full frame). This method allows sending a full frame at the specified time.

For example, this method is used in **ClientServer\VideoChat\Lecture** demo. Without this method, new students would receive slides only after the lecturer changed up to 10 slides (because the lecturer's FullFrameInterval = 10).

2.3.2.2.7 TRVCamSender.Reconnect

Reconnects to TRVCamReceiver⁽¹²⁰⁾ or TRVMediaServer⁽¹⁶⁵⁾ after the connection was broken.

procedure Reconnect;

2.3.2.2.8 TRVCamSender.RestartServer

Restarts TRVMediaServer⁽¹⁶⁵⁾.

procedure Reconnect;

If this sender is connected to a media server via the network, this method sends a command to the server to restart it.

This command is processed if the server has *rvcpUseSystemCmd* and *rvcpCmdResetServer* included in CmdOptions⁽¹⁶⁸⁾ property.

2.3.2.2.9 TRVCamSender.SendCmd and others

Methods for sending commands to TRVCamReceiver⁽¹²⁰⁾ (when the sender is connected via the network with TRVCamReceiver⁽¹²⁰⁾ either directly or through TRVMediaServer⁽¹⁶⁵⁾).

procedure BeginCmd;

procedure EndCmd;

procedure WaitForSendCmd(IsProcessMessages: Boolean;
AMediaIndex: Word = 0);

procedure SendCmd(Cmd: TRVCmd⁽¹⁹⁷⁾; vGUIDTo: TRVMAnsiString⁽²⁴⁴⁾ = '';
vGUIDGroup: TRVMAnsiString⁽²⁴⁴⁾ = ''; AMediaIndex: Word = 0);

When connected to TRVMediaServer⁽¹⁶⁵⁾:

Names of commands sent to a server must not start from 'RV_' or 'RVS_', These prefixes are reserved for system commands.

SendCmd sends Cmd to the client specified in vGUIDTo (if vGUIDTo is empty, GUIDTo⁽¹⁴⁷⁾ property is used instead). If they are empty, the command is sent to clients belonging to the group having identifier vGUIDGroup (if vGUIDGroup is empty, GUIDGroup⁽¹⁴⁷⁾ property is used instead). Otherwise,

the command is sent to the default receivers⁽¹⁵⁷⁾ (if they are defined). When the client's receiver receives a command, OnReceiveCmdData⁽¹³⁵⁾ event occurs.

Commands with names starting from 'RVSU_' are processed specially. They are sent to the server itself (GUID parameters are ignored). When the server receives it, OnServerCmd⁽¹⁷³⁾ event occurs. These commands are not resent by the server to clients. Commands to the server must not be large: the total command size, including all internal information, must not exceed 8912 bytes.

When connected to TRVCamReceiver⁽¹²⁰⁾:

SendCmd sends Cmd to the receiver, vGUIDTo may specify a receiver identifier⁽¹²³⁾ (if vGUIDTo is empty, GUIDTo⁽¹⁴⁷⁾ property is used instead). Group identifier is ignored. When the receiver receives a command, OnReceiveCmdData⁽¹³⁵⁾ event occurs.

Sending commands

SendCmd only initiates sending, and exits before this command is actually sent. Do not free Cmd object, it will be freed by TRVCamSender.

You can use **WaitForSendCmd** to wait until all commands are sent. If IsProcessMessages=True, **WaitForSendCmd** contains a cycle calling Application.ProcessMessages, so check Application.Terminated after.

If you need to send several commands, call **BeginCmd**, then call multiple **SendCmd**, then call **EndCmd**. In this mode, commands are accumulated when calling **SendCmd**, and sent only in **EndCmd**.

Media channels

The optional AMediaIndex parameter allows defining the index of media channel, thus linking this command to this channel. In the most applications, you can leave this parameter equal to 0.

See also

- OnSendCmd, OnSentCmd⁽¹⁶⁵⁾

2.3.2.2.10 TRVCamSender.SendFile, SendUserData

Methods for sending a file and arbitrary data.

```
procedure SendFile(FileName: String; FileSeek: Int64 = 0;
  vGUIDTo: TRVMAnsiString(244) = ''; vGUIDGroup: TRVMAnsiString(244) = '';
  AMediaIndex: Word = 0);
procedure SendUserData(AStream: TMemoryStream;
  vGUIDTo: TRVMAnsiString(244) = ''; TRVMAnsiString: TRVMAnsiString(244) = '';
  AMediaIndex: Word = 0);
```

When connected to TRVMediaServer⁽¹⁶⁵⁾: The methods send data to the client specified in vGUIDTo (if vGUIDTo is empty, GUIDTo⁽¹⁴⁷⁾ property is used instead). If they are empty, data are sent to clients belonging to the group having identifier vGUIDGroup (if vGUIDGroup is empty, GUIDGroup⁽¹⁴⁷⁾ property is used instead). Otherwise, the command is sent to the default receivers⁽¹⁵⁷⁾ (if they are defined).

When connected to TRVCamReceiver⁽¹²⁰⁾: The methods send data to the receiver, vGUIDTo may specify a receiver identifier⁽¹²³⁾ (if vGUIDTo is empty, GUIDTo⁽¹⁴⁷⁾ property is used instead). Group identifiers are ignored.

SendFile sends a file FileName (from the position (in bytes) defined in FileSeek parameter).

SendUserData sends data from AStream.

Media channels

The optional AMediaIndex parameter allows defining the index of media channel, thus linking this file/data to this channel. In the most applications, you can leave this parameter equal to 0.

Files and data in different channels can be sent at the same time.

On the receiver's side

When TRVCamReceiver⁽¹²⁰⁾ receives a file, OnReceivingFile, OnReceiveFileData, OnReceivedFile⁽¹³⁶⁾ events occur.

When TRVCamReceiver⁽¹²⁰⁾ receives user data, OnReceiveUserData⁽¹³⁷⁾ event occurs.

2.3.2.2.11 TRVCamSender.SendMediaAccessRequest, SendMediaAccessCancelRequest

The methods simplify management of default receivers⁽¹⁵⁷⁾.

```
procedure SendMediaAccessRequest(const GUID: TRVMAnsiString(244);
  ADataType: Word = RVMEDIA_DATA(215));
procedure SendMediaAccessCancelRequest(const GUID: TRVMAnsiString(244);
  ADataType: Word = RVMEDIA_DATA(215));
```

These methods are useful when TRVCamSender is connected to TRVMediaServer⁽¹⁶⁵⁾ as a part of a client.

SendMediaAccessRequest sends a request to the client identified by **GUID**; in response, that client can start sending data (usually video and audio) to the requester.

SendMediaAccessCancelRequest sends a request to the client identified by **GUID**; in response, that client should stop sending data to the requester.

Technically, these methods send special commands, like SendCmd⁽¹⁶⁰⁾.

Parameters:

GUID – identifier of other client, from where media is requested. If **GUID** is empty, GUIDTo⁽¹⁴⁷⁾ is used. If it is empty as well, this request is sent to the group GUIDGroup⁽¹⁴⁷⁾.

ADataType – reserved for future use (planned: it will list data types, for which the request is made, can contain ***_DATA⁽²¹⁵⁾ constants (combined using "or" operator). If this parameter is not specified, audio and video are assumed).

Example:

There are two clients, ClientA and ClientB, connected to a media server. ClientA consists of RVCamSenderA and RVCamReceiverA, ClientB consists of RVCamSenderB and RVCamReceiverB. ClientA's identifier is GUID_A, ClientB's identifier is GUID_B.

ClientA wants to receive video and audio from ClientB via the network. It calls `RVCamSenderA.SendMediaAccessRequest(Guid_B)`. In response, `TRVCamReceiverB.OnMediaAccessRequest`⁽¹³⁹⁾ event occurs. In this event, ClientB (if it agrees to send video and audio to ClientA) calls `TRVCamSenderB.AllowMediaAccess`⁽¹⁵⁸⁾(GUID_A).

Next, ClientA do not want to receive video and audio from ClientB any more. It calls `RVCamSenderA.SendMediaAccessCancelRequest(Guid_B)`. In response, `TRVCamReceiverB.OnMediaAccessCancelRequest`⁽¹³⁹⁾ event occurs. In this event, ClientB should call `TRVCamSenderB.CancelMediaAccess`⁽¹⁵⁸⁾(GUID_A).

2.3.2.3 Events

In TRVCamSender

- `OnConnected`⁽¹⁶³⁾
- `OnConnectError`⁽¹⁶³⁾
- `OnConnecting`⁽¹⁶³⁾
- `OnDisconnect`⁽¹⁶³⁾
- `OnEncodeAudio`⁽¹⁶⁴⁾
- `OnEncodeVideo`⁽¹⁶⁴⁾
- `OnSendCmd`⁽¹⁶⁵⁾
- `OnSentCmd`⁽¹⁶⁵⁾

2.3.2.3.1 TRVCamSender.OnConnected, OnConnecting, OnDisconnect, OnConnectError

The events occurring on connection/disconnection to `TRVMediaServer`⁽¹⁶⁵⁾ or `TRVCamReceiver`⁽¹²⁰⁾.

```
property OnConnecting: TRVSocketEvent(235);
property OnConnected: TRVSocketEvent(235);
property OnConnectError: TRVSocketEvent(235);
property OnDisconnect: TRVSocketEvent(235);
```

OnConnecting occurs when the sender starts a connection to a server/receiver, or when a receiver starts a connection to the sender. See the topic about the modes of connections⁽²¹⁾ for the explanations.

After **OnConnecting**, either **OnConnected** or **OnConnectError** occurs.

OnConnected occurs on a successful connection. **OnConnectError** occurs on a failed connection.

OnDisconnect occurs on a disconnection.

If the sender starts a connection, **MediaTypes** parameter contains types of data that will be sent. Senders create new connections to send specific data, so this parameter always contain a single data type.

If a receiver starts a connection, **MediaTypes** is empty.

If you perform time-consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and `SessionKey`⁽¹⁵¹⁾ property, to make sure that a connection was not closed or reopened.

2.3.2.3.2 TRVCamSender.OnEncodeAudio

Occurs when sender is about to send audio data from AudioSource⁽¹⁴³⁾ or ExtraMediaSources⁽¹⁴⁶⁾.

You can use this event to compress or encrypt audio data.

property OnEncodeAudio: TRVAudioEvent⁽²³¹⁾;

AStream contains raw audio data. Only initial **ADataSize** bytes in this stream are used. Parameters of these data are specified in **ASamplesPerSec**, **ABitsPerSample** and **AChannels**.

AAudioIndex identifies the audio source (0 for AudioSource⁽¹⁴³⁾, 1 or greater for ExtraMediaSources⁽¹⁴⁶⁾).

ADuration is an audio length, in milliseconds.

You can encode audio in another format and write it back to **AStream**. Update **ADataSize** as well.

You can also modify values of **ASamplesPerSec**, **ABitsPerSample**, **AChannels** that will be sent to the network. If your modifications changed the sound duration, modify **ADuration** as well.

See also:

- OnEncodeVideo⁽¹⁶⁴⁾
- TRVCamReceiver⁽¹²⁰⁾.OnDecodeAudio⁽¹³⁰⁾
- TRVMicrophone⁽¹¹⁷⁾.BitsPerSample, SamplesPerSec⁽¹⁷⁹⁾

2.3.2.3.3 TRVCamSender.OnEncodeVideo

Occurs when sender is about to send a video frame from VideoSource⁽¹⁵⁵⁾ (or ExtraMediaSources⁽¹⁴⁶⁾)

You can use this event to compress or encrypt video data.

type *// defined in MRVType unit*

```
TRVEncodeVideoEvent = procedure (Sender: TObject;
  AStream: TMemoryStream;
  var ADataSize: Integer; AVideoIndex: Word;
  ImageType: Byte) of object;
```

property OnEncodeVideo: TRVEncodeVideoEvent;

AStream contains a video frame. Only initial **ADataSize** bytes in this stream are used.

AVideoIndex identifies the audio source (0 for VideoSource⁽¹⁵⁵⁾, 1 or greater for ExtraMediaSources⁽¹⁴⁶⁾).

The format of data in **AStream** is identified by **ImageType** parameter. It may be one of:

```
rvtiJPEG = 1;
rvtiHWL = 2;
rvtiBMP = 3;
rvtiPNG = 4;
```

If the sender sends changed areas, this event is called for every changed area on a video frame. Otherwise, it is called for the whole video frame.

You can encode a video data, and write it back to **AStream**. Update **ADataSize** accordingly.

See also:

- Encoding⁽¹⁴⁵⁾
- OnEncodeAudio⁽¹⁶⁴⁾

- TRVCamReceiver⁽¹²⁰⁾.OnDecodeVideo⁽¹³¹⁾

2.3.2.3.4 TRVCamSender.OnSendCmd, OnSentCmd

Occurs when a command is sent.

```
type // defined in MRVCmd unit
  TRVCmdWriteEvent = procedure (Sender: TObject;
    SessionKey: TRVSessionKey(250);
    Cmd: TRVCmd(197); nGUIDFrom, nGUIDTo, nGUIDGroup: TGUID;
    AMediaIndex : Word) of object;
```

```
property OnSendCmd: TRVCmdWriteEvent;
```

```
property OnSentCmd: TRVCmdWriteEvent;
```

These events are called only if ShowCmd⁽¹⁵¹⁾=True.

Commands are sent by SendCmd⁽¹⁶⁰⁾ method (and other methods that call SendCmd internally to send special commands).

OnSendCmd occurs when command sending is initiated (i.e. when SendCmd⁽¹⁶⁰⁾ is called). This event can be used for debugging purposes.

OnSentCmd occurs after the command is sent.

These events may be called in a thread context, so do not update user interface (or make any other operation that requires a context of the main process) in this event.

2.3.3 TRVMediaServer

TRVMediaServer translates data (video, audio, commands, files) from multiple TRVCamSender⁽¹³⁹⁾s to multiple TRVCamReceiver⁽¹²⁰⁾s via the network.

Unit [VCL and LCL] MRVMediaServer;

Unit [FMX] fmxMRVMediaServer;

Syntax

```
TRVMediaServer = class (TComponent)
```

▼ Hierarchy

TObject

TPersistent

TComponent

Description

To start the server, define its HTTPPort⁽¹⁶⁹⁾ and UDPPort⁽¹⁷⁰⁾, and assign HTTPActive⁽¹⁶⁹⁾ and UDPActive⁽¹⁷⁰⁾ = True.

Multiple clients may be connected to a server via the network. Each client may consists of one or more TRVCamSender⁽¹³⁹⁾s and one TRVCamReceiver⁽¹²⁰⁾, having the same identifiers (TRVCamSender.GUIDFrom⁽¹⁴⁷⁾, TRVCamReceiver.GUIDMy⁽¹²³⁾; TRVCamSender.UseGUID⁽¹⁴⁷⁾ must be True). If the second client with the same identifier is connected, the server disconnects the first client.



Two senders in the same client may be necessary to use different protocols: the first sender may send video and audio using UDP, the second sender may send commands, binary data and files using TCP or HTTP. Protocol is specified in `TRVCamSender.Protocol`⁽¹⁴⁹⁾. TCP or HTTP senders must be connected to `HTTPPort`⁽¹⁶⁹⁾, UDP senders must be connected to `UDPPort`⁽¹⁷⁰⁾. TCP or HTTP senders must have `TCPConnectionType`⁽¹⁵³⁾ = `rvtcpSenderToReceiver`.

A receiver is always connected to the server using TCP or HTTP⁽¹²⁴⁾. It must have `TCPConnectionType`⁽¹²⁸⁾ = `rvtcpReceiverToSender`.

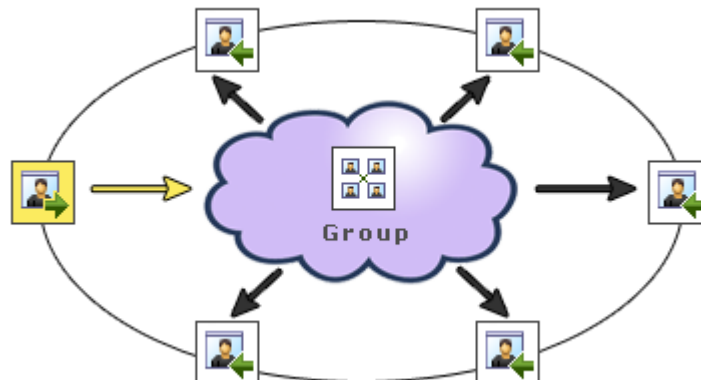
Client to client

If two clients know identifiers of each other, they can send data to each other via the server. The identifier of the other client can be specified in `TRVCamSender.GUIDTo`⁽¹⁴⁷⁾ (or similar parameters of methods sending commands⁽¹⁶⁰⁾, files⁽¹⁶¹⁾, etc.)

This type of communication can be used to implement private talks.

Groups

Another way to establish communication between clients is adding them to the same group.



A client may add itself to a group on the server by calling `TRVCamSender.JoinGroup`⁽¹⁵⁸⁾, and remove itself from the group by calling `TRVCamSender.LeaveGroup`⁽¹⁵⁸⁾.

Existing group members are notified in `TRVCamReceiver.OnUserJoinsGroup` and `OnUserLeavesGroup`⁽¹³⁸⁾.

A client may request a list of group members by calling `TRVCamSender.GetUsersFromGroup`⁽¹⁵⁸⁾; in response, `TRVCamReceiver.OnGetGroupUsers`⁽¹³⁴⁾ is called.

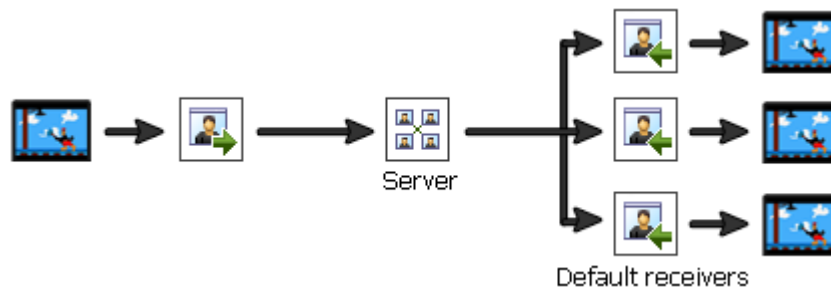
The target group for data can be specified in `TRVCamSender.GUIDTo`⁽¹⁴⁷⁾ (or similar parameters of methods sending commands⁽¹⁶⁰⁾, files⁽¹⁶¹⁾, etc.)

This type of communication can be used to implement chat rooms.

The count of groups may be limited using `MaxGroupCount`⁽¹⁷⁰⁾ property.

Default receivers

Another way to establish communication between clients is a list of default receivers.



If a sender does not have GUIDTo and GUIDGroup specified, data from it are sent to default receivers.

Default receivers may be used to implement contact lists (together with allowed senders).

See also:

- TRVCamSender's methods for management of default receivers ⁽¹⁵⁷⁾
- KeepClientInfoMode ⁽¹⁶⁹⁾ property
- TRVCamSender's methods for managing allowed senders ⁽¹⁵⁶⁾

2.3.3.1 Properties

In TRVMediaServer

- BufferOptions ⁽¹⁶⁷⁾
- CmdOptions ⁽¹⁶⁸⁾
- FilterUserCmd ⁽¹⁶⁸⁾
- GUIDMy ⁽¹⁶⁹⁾
- HTTPActive ⁽¹⁶⁹⁾
- HTTPPort ⁽¹⁶⁹⁾
- KeepClientInfoMode ⁽¹⁶⁹⁾
- MaxGroupCount ⁽¹⁷⁰⁾
- ReceiverConnectionProperties ⁽¹⁷⁰⁾
- SenderConnectionProperties ⁽¹⁷⁰⁾
- ▶ SessionKey ⁽¹⁷⁰⁾
- TempFolder ⁽¹⁶⁷⁾
- UDPActive ⁽¹⁷⁰⁾
- UDPPort ⁽¹⁷⁰⁾

2.3.3.1.1 TRVMediaServer.BufferOptions, TempFolder

The properties define buffering options.

```
property BufferOptions: TRVBufferOptions (196) ;
property TempFolder: String;
```

File buffers are created in **TempFolder**. If **TempFolder** is an empty string, a system directory for temporary files is used.

2.3.3.1.2 TRVMediaServer.CmdOptions

Options for processing commands from clients.

type

```
TRVCmdOption = (rvcpUseSystemCmd, rvcpCmdAllGroups, rvcpCmdAllUsers,
    rvcpCmdAllOnline, rvcpCmdResetServer);
TRVCmdOptions = set of TRVCmdOption;
```

property CmdOptions: TRVCmdOptions;

Value	Meaning
<i>rvcpUseSystemCmd</i>	Allows processing system commands on the server. All the options below works only if <i>rvcpUseSystemCmd</i> is included.
<i>rvcpCmdAllGroups</i>	Allows clients to receive a list of groups on the server. See: <ul style="list-style-type: none"> • TRVCamSender⁽¹³⁹⁾.GetAllGroups⁽¹⁵⁸⁾ • TRVCamReceiver⁽¹²⁰⁾.OnGetAllGroups⁽¹³³⁾
<i>rvcpCmdAllUsers</i>	Allows clients to receive a list of all clients on the server. See: <ul style="list-style-type: none"> • TRVCamSender⁽¹³⁹⁾.GetAllUsers⁽¹⁵⁸⁾ • TRVCamReceiver⁽¹²⁰⁾.OnGetAllUsers⁽¹³³⁾
<i>rvcpCmdAllOnline</i>	Allows clients to receive a list of all online clients on the server. See: <ul style="list-style-type: none"> • TRVCamSender⁽¹³⁹⁾.GetAllOnlineUsers⁽¹⁵⁸⁾ • TRVCamReceiver⁽¹²⁰⁾.OnGetAllOnlineUsers⁽¹³³⁾
<i>rvcpCmdResetServer</i>	Allows clients to restart the server. See: <ul style="list-style-type: none"> • TRVCamSender⁽¹³⁹⁾.RestartServer⁽¹⁶⁰⁾

Default value

[]

2.3.3.1.3 TRVMediaServer.FilterUserCmd

Specifies whether system commands invoke OnServerCmd⁽¹⁷³⁾ event.

property FilterUserCmd: Boolean;

If *True*, this event is not called only for commands addressed to the server (having names starting from 'RVS_' or 'RVSU_').

If *False*, this event is called for all commands, including commands that users send to each other, providing that these commands are sent uncompressed⁽¹⁴⁴⁾.

Default value

True

2.3.3.1.4 TRVMediaServer.GUIDMy

The server identifier.

property GUIDMy: TRVMAnsiString⁽²⁴⁴⁾;

This property is not used directly, but it is recommended to pass it as a parameter AGUIDFrom to SendCommandToGUID⁽¹⁷²⁾.

Initial value:

'{00000000-0000-0000-0000-000000000001}'

2.3.3.1.5 TRVMediaServer.HTTPActive, HTTPPort

HTTPActive turns on/off an HTTP server. HTTPPort defines the server port.

property HTTPActive: Boolean;

property HTTPPort: Word;

HTTPPort and UDPPort⁽¹⁷⁰⁾ must be different.

Default values

- HTTPActive: *False*
- HTTPPort: 8080

2.3.3.1.6 TRVMediaServer.KeepClientInfoMode

Defines how the server keeps information about clients.

type

TRVKeepClientInfoMode = (rvkclmWhileOnline, rvkclmAlways);

property KeepClientInfoMode: TRVKeepClientInfoMode;

This property specifies when information about a client is cleared on the server.

Value	Meaning
<i>rvkclmWhileOnline</i>	Information is stored while the client is online. When it disconnects, all information about this client is deleted. This mode can be used if GUIDs of clients are regenerated on each connection.
<i>rvkclmAlways</i>	Information is stored even when the client is offline (until the server is restarted). This mode can be used if GUIDs of clients are persistent (and identify users' profiles)

The main information stored for a client is the following lists:

- list of allowed senders⁽¹⁵⁶⁾
- list of default receivers⁽¹⁵⁷⁾

Default value:

rvkclmWhileOnline

TO-DO: a programming interface allowing to store lists and other data of clients in a database.

2.3.3.1.7 TRVMediaServer.MaxGroupCount

Defines the maximal allowed count of groups on the server.

property MaxGroupCount: Cardinal;

The value 0 means "unlimited".

It's recommended to limit the count of groups, because otherwise malicious clients may create new groups until the server runs out of memory.

Default value

0

2.3.3.1.8 TRVMediaServer.SenderConnectionProperties, ReceiverConnectionProperties

Defines connection properties (time out time and types of sockets)

property SenderConnectionProperties: TRVConnectionProperties⁽²⁰⁰⁾;
property ReceiverConnectionProperties: TRVConnectionProperties⁽²⁰⁰⁾;

SenderConnectionProperties contains properties for connected senders (data from clients to the server).

ReceiverConnectionProperties contains properties for connected receivers (data from the server to clients).

2.3.3.1.9 TRVMediaServer.SessionKey

Returns the identifier of the current session.

property SessionKey: TRVSessionKey⁽²⁵⁰⁾;

Value of this property is changed (incremented by 1) every time when HTTPActive⁽¹⁶⁹⁾ becomes *True*. When HTTPActive⁽¹⁶⁹⁾=*False*, this property returns 0.

2.3.3.1.10 TRVMediaServer.UDPAcive, UDPPort

UDPAcive turns on/offs a UDP server. UDPPort defines the server port.

property UDPAcive: Boolean;
property UDPPort: Word;

HTTPPort⁽¹⁶⁹⁾ and UDPPort must be different.

Default values

- UDPAcive: *False*
- UDPPort: 8081

2.3.3.2 Methods

In TRVMediaServer

SendCmdToGroup⁽¹⁷¹⁾
 SendCmdToUser⁽¹⁷¹⁾
 SendCommandToGUID⁽¹⁷²⁾

2.3.3.2.1 TRVMediaServer.SendCmdToGroup

Sends a command to the client specified in **AGUIDTo** parameter.

```
function SendCmdToGUIDGroup (Cmd: TRVCmd(197);
  AGUIDFrom, AGUIDGroup: TGUID): Integer;
```

Parameters

Cmd is a command to send. The method frees this object itself.

AGUIDFrom must be equal to GUIDMy⁽¹⁶⁹⁾ property of the server (unless you want to simulate a command from another client).

AGUIDGroup identifies the group of addressee. The command will be send to all members of this group.

Return value

0 if not sent; count of group members otherwise.

See also

- SendCmdToUser⁽¹⁷¹⁾

2.3.3.2.2 TRVMediaServer.SendCmdToUser

Sends a command to the client specified in **AGUIDTo** parameter.

```
function SendCmdToGUIDUser (Cmd: TRVCmd(197);
  AGUIDFrom, AGUIDTo: TGUID): Boolean;
```

This method is similar to SendCommandToGUID⁽¹⁷²⁾, but does not have a group parameter and it frees the command object itself.

Parameters

Cmd is a command to send. The method frees this object itself.

AGUIDFrom must be equal to GUIDMy⁽¹⁶⁹⁾ property of the server (unless you want to simulate a command from another client).

AGUIDTo identifies the addressee (GUIDMy⁽¹²³⁾ property of the client's receiver)

Return value

False if the command cannot be sent (for example, this user is not connected), *True* otherwise.

See also

- SendCmdToGroup⁽¹⁷¹⁾

2.3.3.2.3 TRVMediaServer.SendCommandToGUID

Sends a command to the client specified in **AGUIDTo** parameter.

```
function SendCommandToGUID(Cmd: TRVCmd197;
    AGUIDFrom, AGUIDTo, AGUIDGroup: TGUID): Boolean;
```

Parameters

Cmd is a command to send. After calling this method, free this object yourself.

AGUIDFrom must be equal to GUIDMy¹⁶⁹ property of the server (unless you want to simulate a command from another client).

AGUIDTo identifies the addressee (GUIDMy¹²³ property of the client's receiver)

AGUIDGroup can be used to simulate a command sent to a group of user.

Return value

False if the command cannot be sent (for example, this user is not connected), *True* otherwise.

See also

- SendCmdToUser¹⁷¹
- SendCmdToGroup¹⁷¹
- TRVCamSender¹³⁹.SendCmd¹⁶⁰

2.3.3.3 Events

In TRVMediaServer

- OnConnectionCountChanged¹⁷³
- OnDataRead¹⁷²
- OnError¹⁷⁴
- OnPacketProcessing¹⁷³
- OnServerCmd¹⁷³
- OnStart¹⁷⁴
- OnStop¹⁷⁴
- OnUserConnect¹⁷⁴
- OnUserDisconnect¹⁷⁴

2.3.3.3.1 TRVMediaServer.OnDataRead

Occurs when new data are received.

```
type // defined in MRVType unit
    TRVServerDataReadEvent = procedure (Sender: TObject;
        SessionKey: TRVSessionKey250;
        ADataType: Word; AData: TStream; ASocket: TRVSocket251;
        AGUIDFrom, AGUIDTo, AGUIDGroup: TGUID) of object;
property OnDataRead: TRVServerDataReadEvent;
```

This is a low level event. It is rarely needs to be processed.

AData – received data.

ADataType – data type; it may be one of *****_DATA**⁽²¹⁵⁾ constants, or other values identifying data used by the server to maintain connections.

ASocket – a socket from which data are read.

All GUID parameters are available only for TCP connections. For UDP connections, they are equal to 0.

GUIDFrom – identifier of the data sender (if available)

GUIDTo – identifier of the data receiver (if available)

GUIDGroup – identifier of the group (if available)

This event is called in a thread context. Do not update user interface (or make any other operation that requires a context of the main process) in this event.

2.3.3.3.2 TRVMediaServer.OnPacketProcessing, OnConnectionCountChanged

The events allowing to display information about the server status.

```
type
  TRVMSCountEvent = procedure (Sender: TRVMediaServer(165);
    const Count : Integer) of object;
property OnPacketProcessing: TRVMSCountEvent;
property OnConnectionCountChanged: TRVMSCountEvent;
```

OnPacketProcessing occurs when a new packet is received or processed. **Count** is the number of accumulated packets.

OnConnectionCountChanged occurs when a count of connections is changed. **Count** is the number of connections (a *connection* is a channel⁽²¹⁾ to TRVCamReceiver⁽¹²⁰⁾)

These events can be useful to show information about the server status; you can show how much the server is busy.

See also:

- OnUserConnect, OnUserDisconnect⁽¹⁷⁴⁾

2.3.3.3.3 TRVMediaServer.OnServerCmd

Occurs when a command is received from a client.

```
type
  TRVMSServerCmdEvent = procedure (Sender: TRVMediaServer(165);
    const GUIDUser, GUIDToUser, GUIDGroup: TRVMAnsiString(244);
    ServerCMD: TRVCmd(197));
property OnServerCmd: TRVMSServerCmdEvent;
```

This event occurs when a command sent by TRVCamSender⁽¹³⁹⁾.SendCmd⁽¹⁶⁰⁾ (or by some other sender methods) is received by the server.

By default, this event is called only by commands specially addressed to the server. Names of these commands have prefixes either 'RVS_' or 'RVSU_'. You can set FilterUserCmd⁽¹⁶⁸⁾=False to call this event for all commands.

Parameters:

GUIDUser is a client identifier of the sender (TRVCamSender.GUIDFrom⁽¹⁴⁷⁾)

GUIDToUser is a client identifier of the addressee (TRVCamReceiver.GUIDFrom⁽¹²³⁾), this parameter is valid only for commands addressed from a client to a client.

GUIDGroup is an identifier of a group⁽¹⁵⁸⁾, if a command is sent to a group.

ServerCmd contains the command data.

This event is called in a thread context. Do not update user interface (or make any other operation that requires a context of the main process) in this event.

If you perform time consuming operations inside an event, it makes to check that SessionKey⁽¹⁷⁰⁾ property is not changed (to make sure that a connection is not closed or reopened).

See also:

- TRVCamReceiver.OnReceiveCmdData⁽¹³⁵⁾

2.3.3.3.4 TRVMediaServer.OnStart, OnStop, OnError

The events occurring when the server starts or stops.

type

```
TRVServerEvent = procedure (Sender: TRVMediaServer(165);
  const Protocol: TRVProtocol(248)) of object;
property OnStart: TRVServerEvent;
property OnStop: TRVServerEvent;
property OnError: TRVServerEvent;
```

OnStart occurs when the server started its work successfully.

OnError occurs if the server fails to start.

OnStop occurs when the server ends its work.

2.3.3.3.5 TRVMediaServer.OnUserConnect, OnUserDisconnect

Occurs when a connection to client is created/closed.

type

```
TRVMSUserEvent = procedure (Sender: TRVMediaServer(165);
  const GUIDUser: TRVMAnsiString(244);
  MediaType: TRVMediaType(244)) of object;
property OnUserConnect: TRVMSUserEvent;
property OnUserDisconnect: TRVMSUserEvent;
```

The events occur when a channel⁽²¹⁾ to client's TRVCamReceiver⁽¹²⁰⁾ is created/closed.

GUIDUser identifies the client, it is equal to TRVCamReceiver.GUIDMy⁽¹²³⁾.

MediaType is a data type for the channel.

These events are called in a thread context. Do not update user interface (or make any other operation that requires a context of the main process) in these events.

See also:

- OnConnectionCountChanged¹⁷³

2.3.4 TRVTrafficMeter

TRVTrafficMeter shows traffic of a camera, a receiver, and a sender.

This component can be used for debugging or finding optimal settings.

Unit [VCL and LCL] MRVTrafficMeter;

Unit [FMX] fmxMRVTrafficMeter;

Syntax

```
TRVTrafficMeter = class(TCustomControl)
```

▼ Hierarchy**VCL and LCL:**

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl

FMX:

TObject
TPersistent
TComponent
TFmxObject
TControl
TStyledControl
TTextControl

Description

This component displays charts and summary for the following sources:

- Camera¹⁷⁶
- Sender¹⁷⁶
- Receiver¹⁷⁶

Examples:

1. A sender gets data from a camera and sends them to the network; you can compare the amount of data received from the camera and send by the sender. A difference in traffic may be because of the following reasons: compression, reduced frame dimensions, lost frames (the sender may skip frames if it is too busy).
2. A sender sends data to the network, a receiver receives them; UDP or TRVMediaServer¹⁶⁵ connection. You can see how much data are received (and lost), and estimate a bandwidth.

2.3.4.1 Properties

In TRVTrafficMeter

- Camera⁽¹⁷⁶⁾
- Language⁽¹⁷⁶⁾
- Receiver⁽¹⁷⁶⁾
- Sender⁽¹⁷⁶⁾

2.3.4.1.1 TRVTrafficMeter.Camera

Specifies the camera to show network traffic for.

property Camera: TRVCamera⁽⁴²⁾;

The component shows traffic for received video data and commands. It shows only a network traffic (no activity is shown for web cameras).

2.3.4.1.2 TRVTrafficMeter.Language

Specifies the language for user interface.

property Language: TRVMLanguage⁽²⁴⁵⁾;

Default value

rvmlEnglish

2.3.4.1.3 TRVTrafficMeter.Receiver

Specifies the receiver to show network traffic for.

property Receiver: TCustomRVReceiver⁽¹⁸⁴⁾;

The component shows amount of data received by the **Receiver** via the network from TRVCamSender⁽¹³⁹⁾ or TRVMediaServer⁽¹⁶⁵⁾.

2.3.4.1.4 TRVTrafficMeter.Sender

Specifies the sender to show network traffic for.

property Sender: TCustomRVSender⁽¹⁸⁵⁾;

The component shows amount of data sent by **Sender** via the network to TRVCamReceiver⁽¹²⁰⁾ or TRVMediaServer⁽¹⁶⁵⁾.

2.4 Ancestor classes

Media Sources

TRVMediaSource⁽¹⁸⁹⁾ is a parent class of TRVAudioSource⁽¹⁸⁵⁾ and TRVVideoSource⁽¹⁹⁰⁾.

TRVAudioSource⁽¹⁸⁵⁾ and TRVAudioSourceWithOutput⁽¹⁸⁷⁾ are parent classes of TRVMicrophone⁽¹¹⁷⁾ and TRVCamSound⁽¹¹⁵⁾ components.

TRVVideoSource⁽¹⁹⁰⁾ is a parent class of TRVCamera⁽⁴²⁾ component and TCustomRVReceiver⁽¹⁸⁴⁾.

TCustomRVReceiver⁽¹⁸⁴⁾ is a parent class of TRVCamReceiver⁽¹²⁰⁾ component.

Audio Player

TCustomRVAudioOutput⁽¹⁹¹⁾ is a parent class of TCustomRVAudioPlayer⁽¹⁹²⁾.

TCustomRVAudioPlayer⁽¹⁹²⁾ is a parent class of TRVAudioPlayer⁽¹¹⁰⁾ component.

Audio Viewer

TRVAudioViewer⁽¹⁸⁸⁾ is a parent class of TRVMicrophoneView⁽¹¹⁷⁾ component.

2.4.1 TCustomRVMicrophone

TCustomRVMicrophone reads sound from a microphone (or other audio capture device) or an uncompressed WAV file.

Unit [VCL and LCL] MRVCustomMic;

Unit [FMX] fmxMRVCustomMic;

Syntax

```
TCustomRVMicrophone = class(TRVAudioSource(185))
```

▼ Hierarchy

TObject

TPersistent

TComponent

TRVAudioSource⁽¹⁸⁵⁾

Description

Use

Objects of this class are not used directly. TRVMicrophone⁽¹¹⁷⁾ components are used instead.

Choosing a microphone (or other audio input device)

By default, the component reads sound from the default audio input device. You can choose another device.

A list of available devices is returned in AudioInputDeviceList⁽¹⁷⁹⁾ array property, the count of devices is returned in AudioInputDeviceCount⁽¹⁷⁹⁾ property.

You can choose the device by assigning to AudioInputDeviceIndex⁽¹⁷⁹⁾ property (you can assign an index in AudioInputDeviceList⁽¹⁷⁹⁾, or -1 to choose the default device).

Sound from microphone

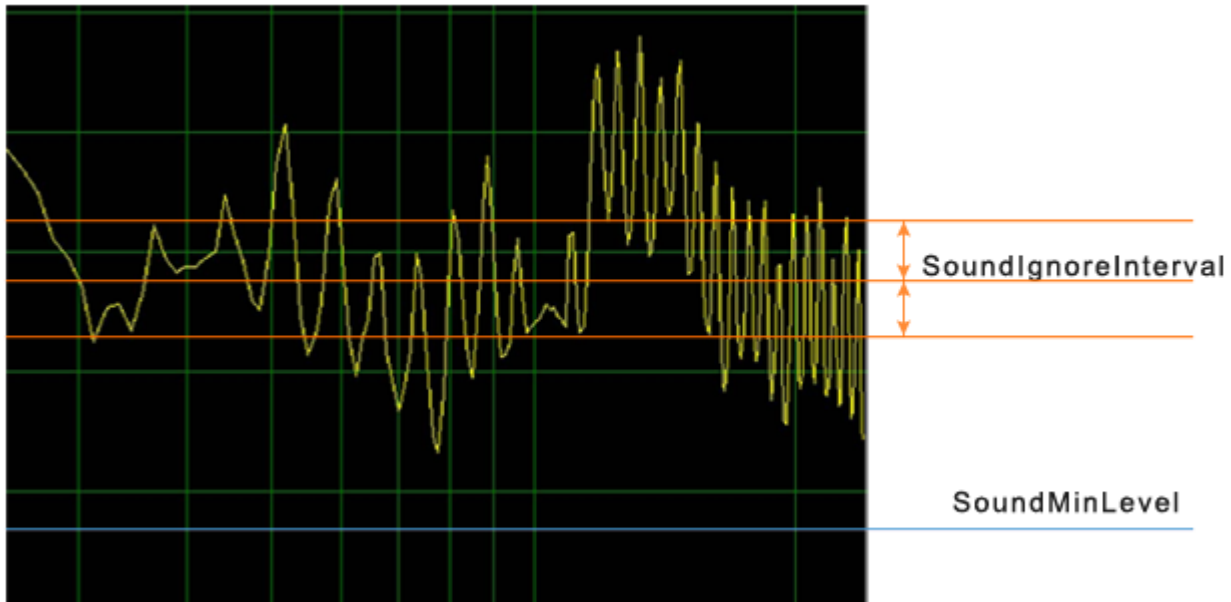
If Active⁽¹⁸⁶⁾=True, the component reads sound from a microphone.

The following properties allow changing the system properties of the microphone (affect all applications): Volume⁽¹⁸⁶⁾.

The following properties allow changing the sound read from a microphone before playing/sending it: VolumeMultiplier⁽¹⁸²⁾, NoiseReduction⁽¹⁸⁰⁾ (and related properties), Pitch⁽¹⁸¹⁾.

Mute⁽¹⁸⁰⁾ turns off reading from the microphone.

The following properties allow to cut off non-informative sound: SoundMinLevel⁽¹⁸¹⁾, SoundIgnoreInterval⁽¹⁸¹⁾. The chart below shows how they work. The yellow curve shows an absolute value of a sound amplitude changing over time. The component ignores sound if: (1) it lies below SoundMinLevel, (2) it lies inside the interval defined by SoundIgnoreInterval.



Sound quality is specified in BitsPerSample and SamplesPerSec properties

Sound from WAV-files

To read sound from a file, assign SourceType⁽¹⁸²⁾=rvsstWAV, assign file name to WAVFileName⁽¹⁸³⁾, assign Active⁽¹⁸⁶⁾=True.

Only uncompressed WAV files are supported.

To apply modification properties to sound read from a file, assign WAVUseOptions⁽¹⁸³⁾=True.

While processing a file, OnOpenWavFile, OnReadWavFile, OnCloseWavFile⁽¹⁸³⁾ events occur.

2.4.1.1 Properties

In TCustomRVMicrophone

- ▶ AudiInputDeviceCount⁽¹⁷⁹⁾
- AudiInputDeviceIndex⁽¹⁷⁹⁾
- AudiInputDeviceList⁽¹⁷⁹⁾
- AudioOutput⁽¹⁸⁸⁾
- BitsPerSample⁽¹⁷⁹⁾
- BufferDuration⁽¹⁷⁹⁾
- Mute⁽¹⁸⁰⁾
- NoiseReduction⁽¹⁸⁰⁾
- NoiseReductionLevel⁽¹⁸⁰⁾
- Pitch⁽¹⁸¹⁾
- SamplesPerSec⁽¹⁷⁹⁾

SoundIgnoreInterval⁽¹⁸¹⁾
 SoundMinLevel⁽¹⁸¹⁾
 SourceType⁽¹⁸²⁾
 UseRNNoise⁽¹⁸⁰⁾
 VolumeMultiplier⁽¹⁸²⁾
 WAVFileName⁽¹⁸³⁾
 WAVUseOptions⁽¹⁸³⁾

Inherited from TRVAudioSource⁽¹⁸⁵⁾

Active⁽¹⁸⁶⁾
 Volume⁽¹⁸⁶⁾

2.4.1.1.1 TCustomRVMicrophone.AudioInputDeviceIndex, AudioInputDeviceCount, AudioInputDeviceList

Properties allowing to choose a microphone (or another audio capture device)

```

property AudioInputDeviceIndex: Integer;
property AudioInputDeviceCount: Integer;
property AudioInputDeviceList[Index: Integer]: String;
  
```

AudioInputDeviceCount returns the count of audio capture devices.

AudioInputDeviceList[Index] (where Index is in the range 0..**AudioInputDeviceCount**-1) returns names of devices. These names can be shown to users in the application UI.

Assign a value in the range 0..**AudioInputDeviceCount**-1 to **AudioInputDeviceIndex** to choose the device. Alternatively, you can assign -1 to use the default device.

2.4.1.1.2 TCustomRVMicrophone.BitsPerSample, SamplesPerSec

The properties define quality of sound read from a microphone.

```

property SamplesPerSec: TRVSamplesPerSec(250);
property BitsPerSample: TRVBitsPerSample(237);
  
```

Higher values may increase sound quality, but they increase traffic and lag as well.

"Samples per second" is often called "sample rate", "bits per sample" is often called "sample size".

Default values

- SamplesPerSec: *rvsps8000*
- BitsPerSample: *rvbps8*

See also

- WAVFileName⁽¹⁸³⁾

2.4.1.1.3 TCustomRVMicrophone.BufferDuration

Specifies the buffer size, in milliseconds

```

property BufferDuration: Word;
  
```

The component sends sound data when the buffer is completely filled.

Smaller value means lesser lag, however, it may reduce sound quality.

Default value

1000

2.4.1.1.4 TCustomRVMicrophone.Mute

Turns the microphone on/off

property Mute: Boolean;

This property does not change the system "mute" property of the audio device, it mutes only sound read by this component.

Default value:*False***See also:**

- Volume¹⁸⁶
- VolumeMultiplier¹⁸²

2.4.1.1.5 TCustomRVMicrophone.NoiseReduction, NoiseReductionLevel, UseRNNoise

Noise reduction.

property NoiseReduction: Boolean;**property** NoiseReductionLevel: Integer;**property** UseRNNoise: Boolean;**NoiseReduction** turns noise reduction on/off.

This property is applied to sound read from a microphone. If WAVUseOptions¹⁸³ = *True*, it is applied to sound from WAV-files as well.

The component supports two ways of noise reduction.

Built-in noise reduction

RVMedia implements a noise reduction algorithm that performs a fast Fourier transform and filtering results.

NoiseReductionLevel is used if **NoiseReduction** = *True*. It defines the level of noise reduction. The recommended range of values is 0..100. 0 means no reduction, higher values apply higher noise reduction (but may lead to higher sound distortion).

RNNoise

If RNNoise²⁴ library is available, and **UseRNNoise** = *True*, the component can use RNNoise, a library that implements a noise reduction based on a recurrent neural network. Usually, RNNoise's results are better than RVMedia's own results.

NoiseReductionLevel is not used in this mode.

Note: this mode works much better for 16-bit samples than for 8-bit samples (see BitsPerSample¹⁷⁹).

Default value:

- NoiseReduction: *True*
- NoiseReductionLevel: 20
- UseRNNoise: *True*

See also

- TRVAudioPlayer⁽¹¹⁰⁾.NoiseReduction and NoiseReductionLevel⁽¹⁹⁴⁾

2.4.1.1.6 TCustomRVMicrophone.Pitch

Allows changing a pitch of the sound.

property Pitch: Shortint;

- Negative values (-127..-1): high pitch
- 0: unchanged sound
- Positive values (1..127): low pitch

This property is applied to sound read from a microphone. If WAVUseOptions⁽¹⁸³⁾=*True*, it is applied to sound from WAV-files as well.

Default value:

0

2.4.1.1.7 TCustomRVMicrophone.SoundIgnoreInterval

Defines the minimal acceptable interval of changes of the sound amplitude.

property SoundIgnoreInterval: Byte;

For the given period of time, the component calculates the average value of the sound amplitude (Av). If sound amplitude varies in the interval [Av-SoundInterval, Av+SoundInterval], the sound is ignored.

This property allows to cut off a monotonous hum.

This property is applied to sound read from a microphone. If WAVUseOptions⁽¹⁸³⁾=*True*, it is applied to sound from WAV-files as well.

Note: value of this property corresponds to 8-bit sound audio samples (BitsPerSample⁽¹⁷⁹⁾ = *rvbps8*). If BitsPerSample⁽¹⁷⁹⁾ = *rvbps16*, the value of this property is multiplied by 256 before the comparison with samples.

Default value:

5

2.4.1.1.8 TCustomRVMicrophone.SoundMinLevel

The minimal acceptable value of the sound amplitude.

property SoundMinLevel: Byte;

Sound fragments having lower average amplitude will be ignored.

0 means that all sound is played.

This property allows to cut off too quiet sound.

This property is applied to sound read from a microphone. If `WAVUseOptions(183) = True`, it is applied to sound from WAV-files as well.

Note: value of this property corresponds to 8-bit sound audio samples (`BitsPerSample(179) = rvbps8`). If `BitsPerSample(179) = rvbps16`, the value of this property is multiplied by 256 before the comparison with samples.

Default value:

10

2.4.1.1.9 TCustomRVMicrophone.SourceType

Specifies the sound source.

type

```
// defined in MRVType unit
TRVSoundSourceType = (rvsstMicrophone, rvsstWAV);
```

property SourceType: TRVSoundSourceType;

Value	Meaning
<i>rvsstMicrophone</i>	Microphone
<i>rvsstWAV</i>	WAV-file specified in WAVFileName ⁽¹⁸³⁾

2.4.1.1.10 TCustomRVMicrophone.VolumeMultiplier

A multiplier for the microphone sound volume.

property VolumeMultiplier: Double;

The component multiplies the sound signal read from the microphone by this value.

If the value = 1, the sound is not changed.

If the value = 0, the sound is turned off.

If the value > 1, the volume is increased (but the quality of the sound may be decreased).

If the value < 1, the volume is decreased.

This property is applied to sound read from a microphone. If `WAVUseOptions(183) = True`, it is applied to sound from WAV-files as well.

Default value:

1

See also:

- Volume⁽¹⁸⁶⁾
- Mute⁽¹⁸⁰⁾

2.4.1.1.11 TCustomRVMicrophone.WAVFileName

Specifies the name of a WAV-file.

property WAVFileName: **String**;

The component reads sound from this file if SourceType⁽¹⁸²⁾=rvsstWAV

Assigning a value to this property does not start processing a file. After assigning this property, assign Active⁽¹⁸⁶⁾=True (even if it is already True).

The component supports only uncompressed WAV files.

See also

- WAVUseOptions⁽¹⁸³⁾

2.4.1.1.12 TCustomRVMicrophone.WAVUseOptions

Specifies whether the component applies sound options to WAV files.

property WAVUseOptions : **Boolean**;

The component reads sound from a WAV-file if SourceType⁽¹⁸²⁾=rvsstWAV

If True, the following properties are applied when playing a file: NoiseReduction⁽¹⁸⁰⁾, SoundMinLevel⁽¹⁸¹⁾, SoundIgnoreInterval⁽¹⁸¹⁾, Pitch⁽¹⁸¹⁾, VolumeMultiplier⁽¹⁸²⁾.

See also

- WAVFileName⁽¹⁸³⁾

2.4.1.2 Events

In TCustomRVMicrophone

OnCloseWavFile⁽¹⁸³⁾

OnReadWavFile⁽¹⁸³⁾

OnOpenWavFile⁽¹⁸³⁾

2.4.1.2.1 TCustomRVMicrophone.OnOpenWavFile, OnReadWavFile, OnCloseWavFile

The events occur when reading sound from a WAV file.

type // defined in MRVType unit

```
TRVOpenWavFileEvent = procedure (Sender: TObject;
  WavSampleCount, WavSamplesPerSec,
  WavBitsPerSample, WavChanneles : Integer) of object;
TRVReadWavFileEvent = procedure (Sender: TObject;
  CurSample, Samples: Integer) of object;
TRVCloseWavFileEvent = procedure (Sender: TObject;
  CurSample, SampleCount: Integer) of object;
```

property OnOpenWavFile: TRVOpenWavFileEvent;

property OnReadWavFile: TRVReadWavFileEvent;

property OnCloseWavFile: TRVCloseWavFileEvent;

OnOpenWavFile occurs when WAV file is opened.

Parameters:

- WavSampleCount – count of sound samples in the file
- WavSamplesPerSec – sample rate
- WavBitsPerSample – bit depth
- WavChanneles – count of channels

OnReadWavFile occurs while WAV file is read.

Parameters:

- CurSample – number of the current sound sample
- SampleCount – total count of sound samples in the file

OnCloseWavFile occurs when WAV file is closed.

Parameters:

- CurSample – number of the last read sound sample
- SampleCount – total count of sound samples in the file

If CurSample<SampleCount, processing was aborted. If CurSample=SampleCount, the file is processed completely.

See also:

- SourceType ⁽¹⁸²⁾
- WAVFileName ⁽¹⁸³⁾

2.4.2 TCustomRVReceiver

This is the ancestor class for TRVCamReceiver ⁽¹²⁰⁾ component.

Unit [VCL and LCL] MRVCustomReceiver;

Unit [FMX] fmxMRVCustomReceiver;

Syntax

```
TCustomRVReceiver = class (TRVVideoSource (190))
```

▼ Hierarchy

TObject
TPersistent
TComponent
TRVVideoSource ⁽¹⁹⁰⁾

Description

The class introduces OnDataRead ⁽¹⁸⁵⁾ event.

2.4.2.1 Properties

In TCustomRVReceiver

AudioOutput ⁽¹⁸⁵⁾

Inherited from TRVVideoSource¹⁹⁰

- ▶ Aborting¹⁹⁰
- ▶ FramePerSec¹⁹⁰
- ▶ FramePerSecInt¹⁹⁰

2.4.2.1.1 TCustomRVReceiver.AudioOutput

Links this receiver component to TRVAudioPlayer¹¹⁰ component.

property AudioOutput: TCustomRVAudioOutput¹⁹¹;

A receiver component can play sound itself. However, it does it using default sound settings, and only on the default audio device.

Assign TRVAudioPlayer¹¹⁰ component to play sound in a more configurable way.

2.4.2.2 Events

In TCustomRVReceiver

OnDataRead¹⁸⁵

2.4.2.2.1 TCustomRVReceiver.OnDataRead

Occurs when new data are received.

property OnDataRead: TRVDataReadEvent²³⁴;

This is a low level event. It is rarely needs to be processed.

This event is called in a thread context. Do not update user interface (or make any other operation that requires a context of the main process) in this event.

2.4.3 TCustomRVSender

This is the ancestor class for TRVCamSender¹³⁹ component.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

Syntax

TCustomRVSender = **class** (TComponent)

▼ Hierarchy

TObject

TPersistent

TComponent

2.4.4 TRVAudioSource

This is the ancestor class for TRVMicrophone¹¹⁷ and TRVCamSound¹¹⁵ components.

Unit [VCL and LCL] MRVMediaSource;

Unit [FMX] fmxMRVMediaSource;

Syntax

```
TRVAudioSource = class (TRVMediaSource189)
```

▼ Hierarchy

TObject
TPersistent
TComponent
*TRVMediaSource*¹⁸⁹

This class introduces properties:

- Active¹⁸⁶ – turn on/off the audio source (if this audio source supports turning on/off)
- Volume¹⁸⁶ – changes sound volume (if this audio source supports changing volume)

Do not use this class directly, use its inherited components:

- TRVMicrophone¹¹⁷
- TRVCamSound¹¹⁵

2.4.4.1 Properties

In TRVAudioSource

- Active¹⁸⁶
- Volume¹⁸⁶

2.4.4.1.1 TRVAudioSource.Active

Enables reading audio from the device.

property Active: Boolean;

Note: this property is available in TRVMicrophone¹¹⁷ but not available in TRVCamSound¹¹⁵.

Default value

False

2.4.4.1.2 TRVAudioSource.Volume

Returns or changes the volume of the audio device (microphone).

property Volume: Word;

Note: this property is available in TRVMicrophone¹¹⁷ but not available in TRVCamSound¹¹⁵.

The range of values is 0..65535.

This property changes the system volume of the audio device. This is a system global setting.

The component may work with several audio devices (for example, TRVMicrophone allows choosing the device using *AudioInputDeviceIndex*¹⁷⁹ property). In this case, this property gets/sets a value for the chosen device.

See also:

- TCustomRVMicrophone.VolumeMultiplier¹⁸²
- TCustomRVMicrophone.Mute¹⁸⁰

2.4.4.2 Events

In TRVAudioSource

OnGetAudio⁽¹⁸⁷⁾

2.4.4.2.1 TRVAudioSource.OnGetAudio

property OnGetAudio: TRVAudioEvent⁽²³¹⁾;

A low level event.

2.4.5 TRVAudioSourceWithOutput

This is the ancestor class for TRVMicrophone⁽¹¹⁷⁾ and TRVCamSound⁽¹¹⁵⁾ components. This is an audio source component that can be linked to an audio output component.

Unit [VCL and LCL] MRVAudioSourceEx;

Unit [FMX] fmxMRVAudioSourceEx;

Syntax

TRVAudioSource = **class** (TRVMediaSource⁽¹⁸⁹⁾)

▼ Hierarchy

TObject

TPersistent

TComponent

TRVMediaSource⁽¹⁸⁹⁾

TRVAudioSource⁽¹⁸⁵⁾

This class introduces a new property: AudioOutput⁽¹⁸⁸⁾.

Do not use this class directly, use its inherited components:

- TRVMicrophone⁽¹¹⁷⁾
- TRVCamSound⁽¹¹⁵⁾

2.4.5.1 Properties

In TRVAudioSourceWithOutput

AudioOutput⁽¹⁸⁸⁾

Inherited from TRVAudioSource⁽¹⁸⁵⁾

Active⁽¹⁸⁶⁾

Volume⁽¹⁸⁶⁾

2.4.5.1.1 TRVAudioSourceWithOutput.AudioOutput

Links this audio source component to TRVAudioPlayer⁽¹¹⁰⁾ component.

property AudioOutput: TCustomRVAudioOutput⁽¹⁹¹⁾;

Assign TRVAudioPlayer⁽¹¹⁰⁾ component to this property to play or record sound.

For TRVCamSound⁽¹¹⁵⁾, the playback speed of **AudioOutput** is also used to synchronize video to audio (Windows-only; synchronization in Linux will be implemented in future updates)

2.4.6 TRVAudioViewer

This is the ancestor class for TRVMicrophoneView⁽¹¹⁷⁾ component.

Unit [VCL and LCL] MRVAudioViewer;

Unit [FMX] fmxMRVAudioViewer;

Syntax

```
TRVAudioViewer = class(TCustomControl)
```

▼ Hierarchy

VCL and LCL:

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl

FMX:

TObject
TPersistent
TComponent
TFmxObject
TControl

Description

The component visualizes an activity of AudioSource⁽¹⁸⁹⁾ or ReceiverSource⁽¹⁸⁹⁾.

2.4.6.1 Properties

In TRVAudioViewer

AudioSource⁽¹⁸⁹⁾
 GUIDFrom⁽¹⁸⁹⁾
 ReceiverSource⁽¹⁸⁹⁾

2.4.6.1.1 TRVAudioViewer.AudioSource

An audio source to visualize audio data from.

property AudioSource: TRVAudioSource⁽¹⁸⁵⁾;

If you assign this property, ReceiverSource⁽¹⁸⁹⁾ is reset to *nil*.

You can assign the following components to this property:

- TRVMicrophone⁽¹¹⁷⁾ (sound from a microphone or a WAV file)
- TRVCamSound⁽¹¹⁵⁾ (sound from video received by TRVCamera⁽⁴²⁾ component)

2.4.6.1.2 TRVAudioViewer.ReceiverSource, GUIDFrom

ReceiverSource is an audio source to visualize audio data received from the network, for example, by TRVCamReceiver⁽¹²⁰⁾ component.

property ReceiverSource: TCustomRVReceiver⁽¹⁸⁴⁾;

property GUIDFrom: TRVMAnsiString⁽²⁴⁴⁾;

If you assign value to **ReceiverSource**, AudioSource⁽¹⁸⁹⁾ is reset to *nil*.

If **GUIDFrom** is empty, this component shows sound activity from all senders.

You can specify GUID of specific sender to indicate only sound received from this sender.

See also:

- TRVCamSender⁽¹³⁹⁾.GUIDFrom⁽¹⁴⁷⁾
- TRVCamReceiver⁽¹²⁰⁾.Senders⁽¹²⁵⁾

Default value

GUIDFrom: '' (empty string)

2.4.7 TRVMediaSource

This is the ancestor class for audio- and video-source components.

Unit [VCL and LCL] MRVMediaSource;

Unit [FMX] fmxMRVMediaSource;

Syntax

TRVMediaSource = **class** (TComponent)

▼ **Hierarchy**

TObject

TPersistent

TComponent

This class is not used directly.

The following classes are inherited from it:

- TRVAudioSource⁽¹⁸⁵⁾ - TRVAudioSourceWithOutput⁽¹⁸⁷⁾ - TRVMicrophone⁽¹¹⁷⁾, TRVCamSound⁽¹¹⁵⁾
- TRVVideoSource⁽¹⁹⁰⁾ - TRVCamera⁽⁴²⁾, TRVCamReceiver⁽¹²⁰⁾

2.4.8 TRVVideoSource

This is the ancestor class for TRVCamera⁽⁴²⁾ and TRVCamReceiver⁽¹²⁰⁾ components.

Unit [VCL and LCL] MRVMediaSource;

Unit [FMX] fmxMRVMediaSource;

Syntax

```
TRVVideoSource = class (TRVMediaSource(189))
```

▼ Hierarchy

TObject
TPersistent
TComponent
TRVMediaSource⁽¹⁸⁹⁾

2.4.8.1 Properties

In TRVVideoSource

- ▶ Aborting⁽¹⁹⁰⁾
- ▶ FramePerSec⁽¹⁹⁰⁾
- ▶ FramePerSecInt⁽¹⁹⁰⁾

2.4.8.1.1 TRVVideoSource.Aborting

Returns *True* if the component is aborting the current operation

property Aborting: Boolean; // read-only

See also

Abort⁽¹⁹¹⁾

2.4.8.1.2 TRVVideoSource.FramePerSec

FramePerSec changes a frame per second value (frame rate), if supported by the source (e.g. IP camera).

property FramePerSec: Double;

property FramePerSecInt: Integer; // read-only

Additionally, TRVCamera⁽⁴²⁾ uses this property when playing MJPEG file⁽⁶⁷⁾.

Fractional values can be used in TRVCamera, if DeviceType⁽⁴⁹⁾ = *rvdtWebCamera*, *rvdtDesktop*, or *rvdtUserData*, or when playing MJPEG files. Otherwise, it is rounded to the integer value (minimum 1). The integer value is returned in **FramePerSecInt** property.

This property is ignored in TRVCamReceiver⁽¹²⁰⁾.

Default value

25

2.4.8.2 Methods

In TRVVideoSource

Abort⁽¹⁹¹⁾

GetOptimalVideoResolution⁽¹⁹¹⁾

2.4.8.2.1 TRVVideoSource.Abort

Aborts the current operation.

procedure Abort;

For TRVCamera⁽⁴²⁾: In a no-wait⁽⁴⁷⁾ mode, the method only sends a command for aborting the current operation to the camera. You can use events or WaitFor***⁽⁶⁹⁾ methods to determine when the operation is actually aborted.

See also

Aborting⁽¹⁹⁰⁾

2.4.8.2.2 TRVVideoSource.GetOptimalVideoResolution

Returns the optimal video resolution for the specified Width and Height.

type

```
TRVVideoResolution = (rv160_120, rv320_240, rv640_480, rv1024_768);
```

function GetOptimalVideoResolution(Width, Height : Integer) : TRVVideoResoluti

2.4.9 TCustomRVAudioOutput

This is the ancestor class for TRVAudioPlayer⁽¹¹⁰⁾ component.

Unit [VCL and LCL] MRVAudioOutput;

Unit [FMX] fmxMRVAudioOutput;

Syntax

```
TCustomRVAudioOutput = class (TComponent)
```

▼ **Hierarchy**

TObject

TPersistent

TComponent

Description

This class is not used directly. It is a parent class for TCustomRVAudioPlayer⁽¹⁹²⁾, which, in its order, is a parent class for TRVAudioPlayer⁽¹¹⁰⁾ component.

TCustomRVAudioOutput introduces properties:

- Active⁽¹⁹²⁾
- Volume⁽¹⁹²⁾

2.4.9.1 Properties

In TCustomRVAudioOutput

Active ⁽¹⁹²⁾
Volume ⁽¹⁹²⁾

2.4.9.1.1 TCustomRVAudioOutput.Active

Turns the audio player on/off

property Volume: Word;

Assign *True* to start playing sound.

Default value

False

2.4.9.1.2 TCustomRVAudioOutput.Volume

Returns or changes the volume of the audio player.

property Volume: Word;

The range of values is 0..65535.

This property changes the system volume of the audio device. This is a system global setting.

The component may work with several audio devices. In this case, this property gets/sets a value for the chosen device.

See also:

- TCustomRVMicrophone.VolumeMultiplier ⁽¹⁸²⁾
- TCustomRVMicrophone.Mute ⁽¹⁸⁰⁾

2.4.9.2 Events

In TCustomRVAudioOutput

OnGetAudio ⁽¹⁹²⁾

2.4.9.2.1 TCustomRVAudioOutput.OnGetAudio

property OnGetAudio: TRVAudioEvent ⁽²³¹⁾;

A low level event.

2.4.10 TCustomRVAudioPlayer

This is the ancestor class for TRVAudioPlayer ⁽¹¹⁰⁾ component.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

Syntax

```
TCustomRVAudioPlayer = class (TCustomRVAudioOutput191)
```

▼ Hierarchy

TObject

TPersistent

TComponent

*TCustomRVAudioOutput*¹⁹¹

Description

Use

Objects of this class are not used directly. TRVAudioPlayer¹¹⁰ components are used instead.

Choosing an audio output device

By default, the component plays sound on the default audio output device. You can choose another device (speakers, headphones, etc.)

A list of available devices is returned in AudioOutputDeviceList¹⁹⁴ array property, the count of devices is returned in AudioOutputDeviceCount¹⁹⁴ property.

You can choose the device by assigning to AudioOutputDeviceIndex¹⁹⁴ property (you can assign an index in AudioOutputDeviceList¹⁹⁴, or -1 to choose the default device).

Properties

The component starts playing sound when *True* is assigned to Active¹⁹².

The following properties allow changing the system properties of the device (affect all applications): Volume¹⁹².

The following properties allow changing the sound: VolumeMultiplier¹⁹⁵, NoiseReduction¹⁹⁴, Pitch¹⁹⁵.

Mute¹⁹⁴ turns off sound.

2.4.10.1 Properties

In TCustomRVAudioPlayer

- ▶ AudioOutputDeviceCount¹⁹⁴
- AudioOutputDeviceIndex¹⁹⁴
- ▶ AudioOutputDeviceList¹⁹⁴
- BufferDuration¹⁹⁴
- Mute¹⁹⁴
- NoiseReduction¹⁹⁴
- Pitch¹⁹⁵
- VolumeMultiplier¹⁹⁵

Inherited from TCustomRVAudioOutput¹⁹¹

- Active¹⁹²
- Volume¹⁹²

2.4.10.1.1 TCustomRVAudioPlayer.AudioInputDeviceIndex, AudioInputDeviceCount, AudioInputDeviceList

Properties allowing to choose an audio output device (such as speakers or headphones)

```
property AudioOutputDeviceIndex: Integer;
property AudioOutputDeviceCount: Integer;
property AudioOutputDeviceList[Index: Integer]: String;
```

AudioOutputDeviceCount returns the count of available audio output devices.

AudioOutputDeviceList[Index] (where Index is in the range 0..**AudioOutputDeviceCount**-1) returns names of devices. These names can be shown to users in the application UI.

Assign a value in the range 0..**AudioOutputDeviceCount**-1 to **AudioOutputDeviceIndex** to choose the device. Alternatively, you can assign -1 to use the default device.

2.4.10.1.2 TCustomRVAudioPlayer.BufferDuration

Specifies the buffer size, in milliseconds

```
property BufferDuration: Word;
```

Default value

1000

2.4.10.1.3 TCustomRVAudioPlayer.Mute

Turns the sound on/off

```
property Mute: Boolean;
```

This property does not change the system "mute" property of the audio device, it mutes only sound played by this component.

Important: do not assign *True* to this property if the component plays sound from TRVCamSound⁽¹¹⁵⁾ ! When muted, this audio player does not play sound, so video and audio will be out of sync. To mute, assign VolumeMultiplier⁽¹⁹⁵⁾ = 0.

Default value:

False

See also:

- Volume⁽¹⁹²⁾
- VolumeMultiplier⁽¹⁹⁵⁾

2.4.10.1.4 TCustomRVAudioPlayer.NoiseReduction

Activates/deactivates a noise reduction and set its level.

```
property NoiseReduction: Boolean;
property NoiseReductionLevel: Integer;
```

When playing sound from TRVMicrophone⁽¹¹⁷⁾, you can use RVMicrophone.NoiseReduction and NoiseReductionLevel⁽¹⁸⁰⁾ instead.

Default value:

- NoiseReduction: False
- NoiseReductionLevel: 20

2.4.10.1.5 TCustomRVAudioPlayer.Pitch

Allows changing a pitch of the sound.

property Pitch: Shortint;

- Negative values (-127..-1): high pitch
- 0: unchanged sound
- Positive values (1..127): low pitch

When playing sound from TRVMicrophone⁽¹¹⁷⁾, you can use RVMicrophone.Pitch⁽¹⁸¹⁾ instead.

Default value:

0

2.4.10.1.6 TCustomRVAudioPlayer.VolumeMultiplier

A multiplier for the sound volume.

property VolumeMultiplier: Double;

The component multiplies the sound signal by this value.

If the value = 1, the sound is not changed.

If the value = 0, the sound is turned off.

If the value > 1, the volume is increased (but the quality of the sound may be decreased).

If the value < 1, the volume is decreased.

When playing sound from TRVCamSound⁽¹¹⁵⁾, this assigning 0 to this property is the correct way to mute sound (instead of Mute⁽¹⁹⁴⁾ property).

When playing sound from TRVMicrophone⁽¹¹⁷⁾, this property is applied after applying RVMicrophone.VolumeMultiplier⁽¹⁸²⁾.

Default value:

1

See also:

- Mute⁽¹⁹⁴⁾

3 Classes

Camera⁽⁴²⁾ Properties

- TRVGStreamerProperty⁽²⁰⁴⁾ contains properties configuring GStreamer; a type of GStreamerProperty⁽⁵²⁾ property.
- TRVFFmpegProperty⁽²⁰¹⁾ contains properties configuring FFmpeg; a type of FFmpegProperty⁽⁵⁰⁾ property.

Commands

- TRVCmd⁽¹⁹⁷⁾ – a command that can be sent from a sender to a receiver (or a server).

- TRVCmdParamCollection⁽¹⁹⁸⁾ – a collection of TRVCmdParamItem⁽¹⁹⁸⁾ items; a type of TRVCmd⁽¹⁹⁷⁾.Params property; a collection of command parameters.

Graphics

- TRVImageWrapper⁽²⁰⁶⁾ encapsulates a graphic object.
- TRVMBitmap⁽²⁰⁶⁾ contains a raster image.

Sender⁽¹³⁹⁾ Properties

- TRVMediaSourceCollection⁽²⁰⁷⁾ – a collection of TRVMediaSourceItem⁽²⁰⁸⁾ items; a type of ExtraMediaSources⁽¹⁴⁶⁾ property; media sources for additional media channels.
- TRVCompressionOptions⁽¹⁹⁹⁾ – media compression options; a type of CompressionOptions⁽¹⁴⁴⁾ property.
- TRVConnectionProperties⁽²⁰⁰⁾ – connection properties; a type of ConnectionProperties⁽¹⁴⁵⁾ property.
- TRVSendOptions⁽²¹⁴⁾ – sending options; a type of SendOptions⁽¹⁵¹⁾ property.

Receiver⁽¹²⁰⁾ Properties

- TRVConnectionProperties⁽²⁰⁰⁾ – connection properties; a type of ConnectionProperties⁽¹²³⁾ property.
- TRVSenderCollectionEx⁽²¹³⁾ – a collection of TRVSenderItemEx⁽²¹⁴⁾ items; a type of Senders⁽¹²⁵⁾ property; describes a list of allowed senders for this receiver.

Server⁽¹⁶⁵⁾ Properties

- TRVBufferOptions⁽¹⁹⁶⁾ – buffering options; a type of BufferOptions⁽¹⁶⁷⁾ property.
- TRVConnectionProperties⁽²⁰⁰⁾ – connection properties; a type of SenderConnectionProperties and ReceiverConnectionProperties⁽¹⁷⁰⁾ properties.

Other

- TRVMotionDetector⁽²⁰⁹⁾ allows to find changed areas in video frames.

3.1 TRVBufferOptions

Buffering options for TRVMediaServer⁽¹⁶⁵⁾.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

Syntax

```
TRVBufferOptions = class(TPersistent);
```

▼ Hierarchy

TObject
TPersistent

Description

This is a type of TRVMediaServer.BufferOptions⁽¹⁶⁷⁾ property.

This class has two sets of properties:

- properties defining where the server stores temporal data (in memory or files):

- Video: TRVStreamType – buffering options for video streams (default value = rvstMemory)
- Audio: TRVStreamType – buffering options for audio streams (default value = rvstMemory)
- UserData: TRVStreamType – buffering options for custom data (default value = rvstMemory)
- Cmd: TRVStreamType – buffering options for commands (default value = rvstMemory)
- FileData: TRVStreamType – buffering options for files (default value = rvstFile)
- properties defining buffer sizes:
 - VideoBufferSize: Cardinal (default value=MAX_VIDEO_BUFFER)
 - AudioBufferSize: Cardinal (default value=MAX_AUDIO_BUFFER)
 - UserDataBufferSize: Cardinal (default value=MAX_USER_BUFFER)
 - CmdBufferSize: Cardinal (default value=MAX_CMD_BUFFER)
 - FileDataBufferSize: Cardinal (default value=MAX_FILE_BUFFER)

and LimitType: TRVBufferLimitType (default value rvbltLimitMemory) property.

where

type

```
TRVStreamType = (rvstMemory, rvstFile);
TRVBufferLimitType = (rvbltNo, rvbltLimitAll,
  rvbltLimitMemory);
```

const

```
BUFFER_SIZE = 4096 * 2;
MAX_VIDEO_BUFFER = BUFFER_SIZE * 100;
MAX_AUDIO_BUFFER = BUFFER_SIZE * 200;
MAX_CMD_BUFFER = BUFFER_SIZE * 100;
MAX_USER_BUFFER = BUFFER_SIZE * 1000;
MAX_FILE_BUFFER = 1024*1024 * 10; // 10 Mb
```

When a buffer on the server exceeds the specified value, data in this buffer is dropped.

This behavior is controlled by LimitType property

Value	Meaning
<i>rvbltNo</i>	All buffers are unlimited, *BufferSize properties are ignored
<i>rvbltLimitAll</i>	All buffers are limited by *BufferSize properties
<i>rvbltLimitMemory</i>	Only memory buffers are limited

3.2 TRVCmd

A command that can be sent⁽¹⁶⁰⁾ by TRVCamSender⁽¹³⁹⁾ component.

Unit [VCL and LCL] MRVCmd;

Unit [FMX] fmxMRVCmd;

Syntax

```
TRVCMD = class(TPersistent)
```

▼ **Hierarchy**

TObject

TPersistent

Description

This class has the following properties:

- Cmd: TRVMAncsiString⁽²⁴⁴⁾ – name of the command
- Params : TRVCMDParamCollection⁽¹⁹⁸⁾ – parameters, a collection of TRVCmdParamItem⁽¹⁹⁸⁾ items.
- Sessionkey : TRVSessionkey⁽²⁵⁰⁾ identifies the session where this command was received.

When implementing your own commands, do not start their names with 'RV_' and 'RVS_'. These prefixes are reserved for internal commands.

3.3 TRVCmdParamCollection

A collection of command parameters⁽¹⁹⁸⁾.

Unit [VCL and LCL] MRVCmd;

Unit [FMX] fmxMRVCmd;

Syntax

```
TRVCmdParamCollection = class (TCollection)
```

▼ Hierarchy

TObject
TPersistent
TCollection

Description

This collections has the following properties and methods:

- function Add : TRVCmdParamItem⁽¹⁹⁸⁾;
- property Items[Index: Integer]: TRVCmdParamItem⁽¹⁹⁸⁾.

This is the type of TRVCmd⁽¹⁹⁷⁾.Params property.

3.4 TRVCmdParamItem

A class of item in TRVCmdParamCollection⁽¹⁹⁸⁾.

Unit [VCL and LCL] MRVCmd;

Unit [FMX] fmxMRVCmd;

Syntax

```
TRVCmdParamItem = class (TCollectionItem)
```

▼ Hierarchy

TObject
TPersistent
TCollectionItem

Description

This class represents a pair ("name", "value"). "Name" is a string and it can be accessed directly as a property. "Value" may have different type and it is accessed using multiple methods.

This class has the following properties:

- Name: TRVMAnsiString⁽²⁴⁴⁾ – name of the parameter.
- ParamType: TRVParamType⁽²⁴⁸⁾ – type of the parameter's value.
- ValueSize: Integer (read-only) returns the size of the value (a count of bytes)

This class has the following methods:

- reading the parameter's value:
 - AsString returns the value as a string.
 - AsInteger returns the value as an integer number.
 - AsFloat returns the value as a floating point number.
 - AsDateTime returns the value as a date.
 - ReadValue reads the value to Buffer.
- assigning the parameter's value:
 - SetString assigns a string to the value.
 - SetInteger assigns an integer number to the value.
 - SetFloat assigns a floating point number to the value.
 - SetDateTime assigns a date to the value.
 - WriteValue writes the value of the parameter from Buffer.

```
function AsString      : TRVMAnsiString(244);
function AsInteger    : Int64;
function AsFloat      : Extended;
function AsDateTime   : TDateTime;
procedure SetString(Value : TRVMAnsiString(244));
procedure SetInteger(Value : Int64);
procedure SetFloat(Value : Extended);
procedure SetDateTime(Value : TDateTime);
function ReadValue(var Buffer; Count: Integer) : Integer;
procedure WriteValue(var Buffer; Count: Integer);
```

3.5 TRVCompressionOptions

This class defines compression options for different types of media data.

Unit [VCL and LCL] MRVConnectionProp;

Unit [FMX] fmxMRVConnectionProp;

Syntax

```
TRVCompressionOptions = class(TPersistent)
```

▼ Hierarchy

TObject

TPersistent

Description

This class has the following properties:

- Video: TRVCompressionType⁽²⁴¹⁾ – compression for video streams
- Audio: TRVCompressionType⁽²⁴¹⁾ – compression for audio streams
- UserData: TRVCompressionType⁽²⁴¹⁾ – compression for data sent by TRVCamSender.SendUserData⁽¹⁶¹⁾
- Cmd: TRVCompressionType⁽²⁴¹⁾ – compression for commands sent by TRVCamSender.SendCmd⁽¹⁶⁰⁾. Commands having names starting from 'RVS_' or 'RVSU_' are never compressed, regardless of this option (these commands are usually sent to TRVMediaServer⁽¹⁶⁵⁾).
- FileData: TRVCompressionType⁽²⁴¹⁾ – compression for files sent by TRVCamSender.SendFile⁽¹⁶¹⁾

This is a class of TRVCamSender.CompressionOptions⁽¹⁴⁴⁾ property.

3.6 TRVConnectionProperties

Contains connection properties

Unit [VCL and LCL] MRVConnectionProp;

Unit [FMX] fmxMRVConnectionProp;

Syntax

```
TRVConnectionProperties = class (TPersistent);
```

▼ Hierarchy

TObject

TPersistent

Description

This class has the following properties:

- VideoTimeout: Integer (default value = DefaultWaitForVideo)
- AudioTimeout: Integer (default value = DefaultWaitForAudio)
- CmdTimeout: Integer (default value = DefaultWaitForCmd)
- DataTimeout: Integer (default value = DefaultWaitForData)
- FileTimeout: Integer read (default value = DefaultWaitForFile)
- UseBlockingSocketsForVideo: Boolean (default value = *False*)
- UseBlockingSocketsForAudio: Boolean (default value = *False*)
- UseBlockingSocketsForCmd: Boolean (default value = *True*)
- UseBlockingSocketsForData: Boolean (default value = *False*)
- UseBlockingSocketsForFile: Boolean (default value = *True*)

where

const

```
DefaultWaitForVideo = 3000;
DefaultWaitForAudio = 3000;
DefaultWaitForCmd    = 15000;
DefaultWaitForData   = 10000;
DefaultWaitForFile   = 10000;
```

Timeout properties define a maximal waiting time (in ms) for non-blocking sockets for data of the specified type. As you can see, the largest waiting time is for commands, because they are the most important (and may be even critical for a stable working).

If UseBlockingSockets*=True

The component makes a request and waits for other party to respond (or to break the connection). There are no disconnects on timeout.

Pluses: all sent data will be received.

Minuses: stability; if one application hangs, other application hangs too.

If UseBlockingSockets*=False

The component makes a request and waits for other party to respond, or to break the connection, or for the time-out interval to elapse.

Pluses: stability; if one application hangs, or it is too busy to process requests, other application breaks the connection after a time out, and so it can continue working.

Minuses: data could be lost if other application is busy, connection will be broken.

Use

The following properties have this type:

- TRVCamSender.ConnectionProperties ⁽¹⁴⁵⁾
- TRVCamReceiver.ConnectionProperties ⁽¹²³⁾
- TRVMediaServer.SenderConnectionProperties and ReceiverConnectionProperties ⁽¹⁷⁰⁾

3.7 TRVFFMpegProperty

A class of TRVCamera.FFMpegProperty ⁽⁵⁰⁾. Contains properties configuring FFMpeg ⁽²⁴⁾.

Unit [VCL and LCL] MRVCamera;

Unit [FMX] fmxMRVCamera;

Syntax

```
TRVFFMpegProperty = class (TPersistent)
```

▼ Hierarchy

TObject

TPersistent

Description

If the value of any property is changed during video playback, the playback will restart. The only exception is the Remuxing properties.

Property for turning FFmpeg support on/off:

- UseFFMpeg: Boolean – if *True*, and FFmpeg is available, the component uses it (default value = *True*). If both **GStreamer** and FFmpeg are available and turned on, the component uses FFmpeg (see TRVCamera.GStreamerProperty⁽⁵²⁾.UseGStreamer).

Property allowing/disallowing using FFmpeg parameters specified in this class:

- UseFFMpegProperty: Boolean – if *True*, the parameters listed below are passed to FFmpeg (default value = *True*). Otherwise, FFmpeg is used with the default parameters.

Properties defining video processing by TRVCamera⁽⁴²⁾:

- FrameDrop: Boolean - if *True*, TRVCamera drops frames that are received too late (default value= *False*).
- NoDelay: Boolean - if *True*, received video frames will be displayed as soon as possible. If *False*, RVMedia adds delays between frames using information specified in the video. This option is useful for displaying online video streams, and should not be used to display video files (they will be displayed too fast). TRVCamera uses this option only for videos without sound, and not from local files (default value= *False*).

Properties to resize video:

- UseVideoScale: Boolean – if *True*, video is requested in the size specified in VideoWidth, VideoHeight properties. Otherwise (default), it has the original size.
- VideoWidth, VideoHeight: Integer are used only if UseVideoScale = *True*. At least one of them must be a positive value to apply scaling (if value of one of these properties is zero or negative, this side of video is auto-calculated to keep aspect ratio).
- VideoFilter: TRVFFMpegFilter⁽²⁴³⁾ – method used to scale video frames, *rvffBilinear* by default.

Remuxing (file saving) properties:

Remuxing is saving to a file/streaming without changing video and audio format.

- Remuxing: TRVFFMpegRemuxProperty⁽²⁰³⁾ contains properties for remuxing.
- Remuxing2: TRVFFMpegRemuxProperty⁽²⁰³⁾ contains properties for remuxing.

Remuxing and Remuxing2 are independent. You can start and stop any of them at any time. For example, you can use one of them for recording in a file, and another one for UDP streaming.

Input format (support of special video sources):

- VideoInputFormat: String. If not empty, specifies the input format for video. See GetListOfVideoInputFormats⁽²²³⁾.

Other FFmpeg parameters:

- Video: Boolean allows receiving video, *True* by default.
- Audio: Boolean allows receiving audio, *True* by default. If *True*, sound is processed by the linked TRVCamSound⁽¹¹⁵⁾ component.
- CustomProperties: TStringsList – additional FFmpeg properties. It allows providing parameters in addition to the parameters listed below. Each string must have the format 'param_name=param_value'. For the list of supported parameters, see the documentation about **ffplay command line**.
- Threads: Integer – count of video processing threads; 0 (default) for auto-calculation.
- TimeOut: Integer – maximum timeout (in seconds) to wait for incoming connections, -1 for infinite waiting. Default value is -1. Note: in any case, RVMedia does not allow timeouts more than 20 seconds.

- RTSPTransport: TRVProtocolsEx⁽²⁴⁹⁾ – RTSP transport network protocol, [*rvpeTCP*, *rvpeUDP*] by default. Multiple lower transport protocols may be specified, in that case they are tried one at a time.
- RTSPFlags: TRVRTSPFlags⁽²⁴⁹⁾ sets RTSP flags, [] by default.
- MaxDelay: Integer – maximum muxing or demuxing delay in microseconds (0 or -1 for using defaults), 0 by default.
- STimeOut: Integer – timeout (in microseconds) of socket TCP I/O operations (0 for using defaults), 0 by default.
- RecvBufferSize: Integer – UDP receive buffer size in bytes, 65535 by default.
- UdpFifoBufferSize: Integer – packet buffer size in bytes, 1000000 by default. May be set to 0.
- TcpNoDelay: Boolean – *True* to disable Nagle’s algorithm, *False* to enable it; *False* by default.
- OverrunNonfatal: Boolean allows to survive in case of UDP receiving circular buffer overrun, *True* by default.
- VideoFilter: TRVFFMpegFilter (one of *rvffNone*, *rvffFastBilinear*, *rvffBilinear*, *rvffBicubic*, *rvffX*, *rvffPoint*, *rvffArea*, *rvffBicublin*, *rvffGauss*, *rvffSinc*, *rvffLaczos*, *rvffSpline*) defines video scaling method, *rvffBilinear* by default.
- ProbeSize: Int64 sets probing size in bytes, i.e. the size of the data to analyze to get stream information. A higher value will enable detecting more information in case it is dispersed into the stream, but will increase latency. Must be an integer not lesser than 32. It is 5000000 by default.
- DecodeAudioCodecName, DecodeVideoCodecName: String – name of codecs for decoding audio and video. If empty, codecs is auto-detected by content. This is a low-level property. Incorrect codec names may cause decoding to fail. You can use *GetListOfAudioDecoders* and *GetListOfVideoDecoders*⁽²²⁰⁾ functions to get possible values of these properties.

3.8 TRVFFMpegRemuxProperty

A class of TRVCamera.FFMpegProperty⁽⁵⁰⁾.Remuxing and Remuxing2⁽²⁰¹⁾. Contains properties configuring remuxing (saving to file/streaming without changing video and audio format) using FFMpeg⁽²⁴⁾.

Unit [VCL and LCL] MRVCamera;

Unit [FMX] fmxMRVCamera;

Syntax

```
TRVFFMpegProperty = class (TPersistent)
```

▼ Hierarchy

TObject

TPersistent

Description

Remuxing, short for *re-multiplexing*, is a video processing technique that transfers the contents of a multimedia file or stream into a different file format without altering the original encoding settings. It preserves the audio and video as they are, simply re-wrapping them from one container format to another. This differs from recoding with the TRVCamRecorder⁽⁸⁵⁾ component, which generates video files using specified video and audio formats along with custom settings.

In RVMedia, remuxing is only available for videos played via FFmpeg, while TRVCamRecorder⁽⁸⁵⁾ works with all video sources.

Remuxing requires FFmpeg version 4 or newer.

This class has the following properties:

- **Active:** Boolean turns remuxing on/off (default value = False)
- **FileName:** String is an output file name (default value = 'video.mp4') or UDP/SRT/RTMP/RTSP URL address for streaming (note: for RTMP and RTSP stream formats, you need additional server software).

Unlike other properties of TRVCamera.FFMpegProperty⁽⁵⁰⁾, changing values of the remuxing properties does not restart video playback. You can enable or disable remuxing, or change the output file name, while the video is playing.

The output video file format is determined by the extension of the FileName. If UDP or SRT URL address is specified, MPEG-TS is used.

Only the played video stream and, optionally, the audio stream (if TRVCamera.FFMpegProperty⁽⁵⁰⁾.Audio⁽²⁰¹⁾ = *True* and TRVCamSound⁽¹¹⁵⁾ is linked to this TRVCamera⁽⁴²⁾) are saved.

3.9 TRVGStreamerProperty

A class of TRVCamera.GStreamerProperty⁽⁵²⁾. Contains properties configuring **GStreamer**.

Unit [VCL and LCL] MRVCamera;

Unit [FMX] fmxMRVCamera;

Syntax

```
TRVGStreamerProperty = class (TPersistent)
```

▼ Hierarchy

TObject
TPersistent

Description

This class has the properties defined below.

General properties

- **UseGStreamer:** Boolean – if *True*, and GStreamer is available, the component uses it (default value = *True*). If both GStreamer and **FFmpeg** are available and turned on, the component uses FFmpeg (see TRVCamera.FFMpegProperty⁽⁵⁰⁾).
- **LaunchString:** String. This property allows specifying your own pipeline string for GStreamer 1.0, overriding RVMedia settings. See the details at the end of this topic.
- **LaunchStringMiddle:** String. This property allows adding additional elements to GStreamer 1.9 pipeline string.

Video scale properties:

- **UseVideoScale:** Boolean – use GStreamer to scale video (default value = *False*). Other properties in this group are applied only if *UseVideoScale = True*.
- **AddBorders:** Boolean adds black borders if necessary to keep the display aspect ratio (default value = *False*), only used if *UseVideoScale = True*
- **Dither:** Boolean adds dither; only used for Lanczos method (default value = *False*); see also **Method**
- **Envelope:** Integer – size of filter envelope (default value = 200)
- **Method:** TRVGVideoScaleMethod – video scaling method (default value = *rvgvfBilinear*)
- **Sharpen:** Integer – sharpening; allowed values: 0..100 (default value=0)
- **Sharpness:** Integer – sharpness of filter; allowed values:50..150 (default value = 100)
- **VideoHeight:** Integer – video output height; 0 - use the original height (default value = 0)
- **VideoWidth:** Integer – video output width; 0 - use the original width (default value = 0)

RTSP properties:

- **BufferSize:** Integer – size of UDP buffer used by GStreamer (default value = 524288)
- **Latency:** Integer – amount of ms to buffer for RTSP (default value = 2000)

Other properties:

- **UseQueue:** Boolean – if *True*, additional "queue2" element is added in GStreamer 1.0 pipeline, between videoscale elements and RVMedia video sink. It provides additional buffering. In some cases (e.g. reading UDP stream) it helps to remove artifacts in video frames.

Properties reserved for future use (encoding properties):

- **KBitrate:** Integer – bitrate in kbit/sec (default value = 2048)
- **SlicedThreads:** Boolean – low latency but lower efficiency threading (default value = *False*)
- **Threads:** Integer – number of threads used by the codec, 0 for automatic; allowed values: <= 4 (default value = 0)

Types used in the properties:

type

```
TRVGVideoScaleMethod = (rvgvfNearest, rvgvfBilinear, rvgvf4Tap,
    rvgvfLanzos);
```

Types of video scaling method (see **Method** property)

- *rvgvfNearest* – use nearest neighbor scaling (fast and ugly)
- *rvgvfBilinear* – use bi-linear scaling (slower but prettier)
- *rvgvf4Tap* – use a 4-tap filter for scaling (slow)
- *rvgvfLanzos* – use a multitap Lanczos filter for scaling (slow)

About LaunchStrings

LaunchString and LaunchStringMiddle properties are used if GStreamer version is 1.0 is available. They are ignored for GStreamer 0.1. These are low-level properties, they require knowledge of GStreamer pipelines.

LaunchString

If LaunchString is not empty, it is used instead of video source and video decoding elements constructed by TRVCamera⁽⁴²⁾.

In this case, it must define a pipeline for video source and video format (otherwise, TRVCamera⁽⁴²⁾ builds a pipeline string itself, using VideoFormat⁽⁵⁹⁾, URL⁽⁵⁷⁾, UserName, UserPassword⁽⁵⁷⁾, CameraPort, RTSPPort⁽⁴⁶⁾, ProxyProperty⁽⁵⁵⁾ properties).

LaunchString must not include video scale and video sink entries, they will be added by RVMedia automatically (RVMedia adds: videoconvert, videoscale, capsfilter, and its own video sink).

Example of LaunchString:

```
'souphttpsrc location="https://www.trichview.com/videotest/h264.avi " ! avidemux ! h264parse !
avdec_h264'
```

LaunchStringMiddle

Like LaunchString, it defines a part of GStreamer pipeline. However, it does not override TRVCamera pipeline, but adds new entries in it.

The complete pipeline consists of:

- if LaunchString is not empty: LaunchString, then LaunchStringMiddle, then video scale elements, then RVMedia video sink
- if LaunchString is empty: video source element, video decoding elements, then LaunchStringMiddle, then video scale elements, then RVMedia video sink.

You can use this property to split the pipeline (using "tee" element) in two or more branches. One branch, like before, is used by TRVCamera to display video. Other branches can be used to record or to stream video.

3.10 TRVImageWrapper

A wrapper for a graphic object.

Unit [VCL and LCL] MRVImg;

Unit [FMX] fmxMRVImg;

Syntax

```
TRVImageWrapper = class (TObject);
```

▼ Hierarchy

TObject

Description

This class encapsulate a graphic object. It allows supporting different image libraries without implementing a full-functional TGraphic descendant.

The main property is Bitmap: TRVMBitmap²⁰⁶, it returns the image as a bitmap.

3.11 TRVMBitmap

A class representing a raster image.

Unit [VCL and LCL] MRVBitmap;

Unit [FMX] fmxMRVBitmap;

Syntax

```
TRVMBitmap = class (TPersistent);
```

▼ **Hierarchy****VCL:**

TObject
TPersistent
TRVMCustomBitmap
TRVMBitmapVCL

LCL:

TObject
TPersistent
TRVMCustomBitmap
TRVMBitmapLaz

FMX:

TObject
TPersistent
TRVMCustomBitmap
TRVMBitmapFM

Description

An object of this class stores a bitmap image.

The image size is returned in **Width** and **Height** properties. The image uses 32 bits per pixel.

Use **GetBitmap** method to receives this image as TBitmap. You must not free a TBitmap object returned by this method. If you painted something on this bitmap, call **UpdateData** method to apply changes.

You can use **Assign** method to assign objects of this class to each other, TGraphic object to TRVMBitmap object, TRVMBitmap object to TBitmap.

Image data can be accessed using **Data** field, or using **ScanLine** method:

```
Data: array of Cardinal; // one item represents one pixel (RGBA)
function ScanLine(y : Integer) : PByteArray;
```

3.12 TRVMediaSourceCollection

A collection of audio/video sources ²⁰⁸.

Unit [VCL and LCL] MRVSender;

Unit [FMX] fmxMRVSender;

Syntax

```
TRVMediaSourceCollection = class (TCollection)
```

▼ **Hierarchy**

TObject

*TPersistent**TCollection*

Description

This collections has the following properties and methods:

- function Add : TRVMediaSourceItem⁽²⁰⁸⁾;
- property Items[Index: Integer]: TRVMediaSourceItem⁽²⁰⁸⁾.

This is the type of TRVCamSender⁽¹³⁹⁾.ExtraMediaSources⁽¹⁴⁶⁾ property.

3.13 TRVMediaSourceItem

A class of items in TRVMediaSourceCollection⁽²⁰⁷⁾.

Unit [VCL and LCL] MRVSender;

Unit [FMX] fmxMRVSender;

Syntax

```
TRVMediaSourceItem = class (TCollectionItem)
```

▼ Hierarchy

*TObject**TPersistent**TCollectionItem*

Description

This class allows to define sources of video- and audio-data for sending.

It has the following properties:

- Name: TRVMAnsiString⁽²⁴⁴⁾
- VideoSource: TRVVideoSource⁽¹⁹⁰⁾
- AudioSource: TRVAudioSource⁽¹⁸⁵⁾
- SourceGUID: String
- SourceAudioIndex: Integer
- SourceVideoIndex: Integer.

You can assign any **Name** to the item.

The meaning of other properties are identical to meaning of properties of TRVCamSender⁽¹³⁹⁾:

- AudioSource⁽¹⁴³⁾
- VideoSource⁽¹⁵⁵⁾
- SourceGUID⁽¹⁵⁵⁾
- SourceAudioIndex⁽¹⁵¹⁾
- SourceVideoIndex⁽¹⁵²⁾

Properties of TRVCamSender define the 0th media channel. Properties of the collection items define the 1st, 2nd media channels and so on.

See also:

- TRVCamSender⁽¹³⁹⁾.ExtraMediaSources⁽¹⁴⁶⁾

3.14 TRVMotionDetector

TRVMotionDetector finds differences between two images, and returns them as a collection of rectangular areas.

Unit [VCL and LCL] MRVFuncImg;

Unit [FMX] fmxMRVFuncImg;

Syntax

```
TRVMotionDetector = class;
```

▼ Hierarchy

TObject

Description

This class can be used to detect motion, i.e. changes between two video frames. It returns the result as a collection of rectangles covering all changed areas.

The class has the following properties defining detection options:

- ChangedAreaProcessingMode⁽²⁰⁹⁾
- MinChangeAreaSize,⁽²¹⁰⁾
- PixelColorThreshold⁽²¹⁰⁾

The changes are detected by calling DetectChanges⁽²¹¹⁾. The results are returned by GetRect and IsRectValid⁽²¹¹⁾.

The example of using this class can be found in the demo projects, **Cameras\MotionDetect**

3.14.1 Properties

In TRVMotionDetector

ChangedAreaProcessingMode⁽²⁰⁹⁾
MinChangeAreaSize⁽²¹⁰⁾
PixelColorThreshold⁽²¹⁰⁾
PixelColorThresholdB⁽²¹⁰⁾
PixelColorThresholdG⁽²¹⁰⁾
PixelColorThresholdR⁽²¹⁰⁾

3.14.1.1 TRVMotionDetector.ChangedAreaProcessingMode

Specifies how video frames are preprocessed before detecting changed areas

property ChangedAreaProcessingMode: TRVChangedAreaProcessingMode⁽²⁴⁰⁾;

Additional information about this process can be found in the topic about MinChangeAreaSize and PixelColorThreshold⁽²¹⁰⁾ properties.

Default value:*rvcapmAuto***See also**

- TRVCamSender⁽¹³⁹⁾.ChangedAreaProcessingMode⁽¹⁴⁴⁾

3.14.1.2 TRVMotionDetector.TestMode

Allows turning on a test mode.

property TestMode: TRVBoundsTestMode⁽²³⁷⁾;

If the test mode is turned on, DetectChanges returns an additional image showing changed areas.

Default value:**Default value***rvstmNone***3.14.1.3 TRVMotionDetector.MinChangeAreaSize, PixelColorThreshold**

The properties specify the parameters used to compare video frames.

property MinChangeAreaSize: Integer;

property PixelColorThreshold: Integer;

property PixelColorThresholdR: Integer;

property PixelColorThresholdG: Integer;

property PixelColorThresholdB: Integer;

The motion detector compares two images (usually video frames).

Let the first pixel is (R1 G1 B1), the second pixel is (R2 G2 B2). If **PixelColorThreshold** >= 0, they are treated as different if $(|R1-R2| + |G1-G2| + |B1-B2|)/3 > \text{PixelColorThreshold}$. Otherwise, they are treated as different if $(|R1-R2| > \text{PixelColorThresholdR})$ or $(|G1-G2| > \text{PixelColorThresholdG})$ or $(|B1-B2| > \text{PixelColorThresholdB})$.

The component calculates rectangles covering changed pixels optimally. Rectangles containing less than **MinChangeAreaSize** changed pixels are ignored.

Default values

- MinChangeAreaSize: 10
- PixelColorThreshold -R, -G -B: 8

Default value

15

See also

- TRVCamSender⁽¹³⁹⁾.MinChangeAreaSize, PixelColorThreshold⁽¹⁴⁸⁾

3.14.2 Methods**In TRVMotionDetector**

DetectChanges⁽²¹¹⁾

GetChangedRect⁽²¹¹⁾

GetRect⁽²¹¹⁾

3.14.2.1 TRVMotionDetector.DetectChanges

Compares two video frames.

```
function DetectChanges(nImg, oImg: TRVMBitmap(206); var Mask: TRVMBitmap(206)): Integer
```

Parameters:

nImg – the current video frame.

oImg – previous video frame.

Mask – if TestMode⁽²¹⁰⁾ is turned on, receives an image showing changed areas, otherwise this parameter receives *nil*. The method always assigns a new value to this parameter, input value is ignored.

Return value:

count of rectangles covering changed areas.

These rectangles are returned by GetRect and IsRectValid⁽²¹¹⁾ methods.

3.14.2.2 TRVMotionDetector.GetRect, IsRectValid

The methods return rectangles around changed areas.

```
function IsRectValid(Index: Integer): Boolean;  
function GetRect(Index: Integer): TRVMRect(246);
```

These methods can be called after calling DetectChanges⁽²¹¹⁾.

DetectChanges⁽²¹¹⁾ returns the count of these rectangles. **Index** must be in the range from 0 to count - 1. Not all of these rectangles are valid, ignore rectangles with the indexes where **IsRectValid** returns *False*.

3.15 TRVProxyProperty

Contains properties for proxy configuration.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

Syntax

```
TRVProxyProperty = class(TPersistent);
```

▼ **Hierarchy**

TObject

TPersistent

Description

This property is useful for users who use a proxy server for internet connections.

This property is used when TRVCamera⁽⁴²⁾, TRVCamSender⁽¹³⁹⁾, TRVCamReceiver⁽¹²⁰⁾ components send and receive data using HTTP. This property is ignored when TCP or UDP is used.

This class has the following properties:

- **ProxyMode:** TRVMFProxyMode, one of the following values:

Value	Meaning
<i>rvmNoProxy</i>	Proxy is not used. Other properties of this class are ignored.
<i>rvmManualSettings</i>	Proxy is used, its properties are specified in this class.

rvmNoProxy by default.

- **Host:** String – IP address or host name of the proxy server.
- **Port:** Integer – port number that is used by the proxy server for client connections.

The following parameters are used only in TRVCamera⁽⁴²⁾ for its own connections. They are not used by FFmpeg and GStreamer connections, they are not used in TRVCamSender⁽¹³⁹⁾ and TRVCamReceiver⁽¹²⁰⁾:

- **UserName:** String – user name used by the proxy server for client connections.
- **Password:** String – password used by the proxy server for client connections.

TRVProxyProperty is a class of the following properties:

- TRVCamera⁽⁴²⁾.ProxyProperty⁽⁵⁵⁾
- TRVCamSender⁽¹³⁹⁾.ProxyProperty⁽¹⁴⁹⁾
- TRVCamReceiver⁽¹²⁰⁾.ProxyProperty⁽¹²⁵⁾

3.16 TRVSenderCollection

A collection of senders.

Unit [VCL and LCL] MRVReceiver;

Unit [FMX] fmxMRVReceiver;

Syntax

```
TRVSenderCollection = class (TCollection)
```

▼ Hierarchy

TObject
TPersistent
TCollection

Description

The type of items of this collection: TRVSenderItem⁽²¹³⁾.

This is a type of TRVSenderItemEx⁽²¹⁴⁾'s properties: AudioSenders, VideoSenders, CmdSenders, FileSenders, UserDataSenders.

The class has the following methods:


```

function FindGUID(const GUID: TRVMansiString244): Integer;
function AddGUID(const GUID: TRVMansiString244): Integer;
function DeleteGUID(const GUID: TRVMansiString244): Boolean;

```

FindGUID return the index of item having GUIDFrom property equal to GUID parameter, or -1 if not found.

AddGUID adds a new item and assigns the GUID parameter to its GUIDFrom property, then returns its index; if such an item already exists, it returns its index instead, to prevent duplicates.

DeleteGUID deletes an item having GUIDFrom property equal to GUID (if it exists).

3.17 TRVSenderCollectionEx

A collection describing senders allowing to send data to TRVCamReceiver¹²⁰.

Unit [VCL and LCL] MRVReceiver;

Unit [FMX] fmxMRVReceiver;

Syntax

```
TRVSenderCollectionEx = class (TCollection)
```

▼ Hierarchy

TObject

TPersistent

TCollection

Description

The type of items of this collection: TRVSenderItemEx²¹⁴.

This is the type of TRVCamReceiver¹²⁰.Senders¹²⁵.

This property is used in two cases:

- 1) when the receiver initiates connection to sender¹³⁹(s)
- 2) when the receiver filters data received from a media server¹⁶⁵.

3.18 TRVSenderItem

A class of item in TRVSenderCollection²¹².

Unit [VCL and LCL] MRVReceiver;

Unit [FMX] fmxMRVReceiver;

Syntax

```
TRVSenderItem = class (TCollectionItem)
```

▼ Hierarchy

TObject

TPersistent

TCollectionItem

Description

This class has the following properties:

- GUIDFrom: TRVMAnsiString⁽²⁴⁴⁾ – identifier of the sender

3.19 TRVSenderItemEx

A class of item in TRVSenderCollectionEx⁽²¹³⁾.

Unit [VCL and LCL] MRVReceiver;

Unit [FMX] fmxMRVReceiver;

Syntax

```
TRVSenderItem = class (TCollectionItem)
```

▼ Hierarchy

TObject

TPersistent

TCollectionItem

Description

This class describe sender(s) allowing to send data to TRVCamReceiver⁽¹²⁰⁾.

This class has the following properties:

- GUIDFrom: TRVMAnsiString⁽²⁴⁴⁾ – identifier of a sender.
- SenderHost: String – host of a sender
- SenderPort: Word – port of a sender
- VideoSenders: TRVSenderCollection⁽²¹²⁾ – a collection of identifiers of senders which can send video data to this receiver
- AudioSenders: TRVSenderCollection⁽²¹²⁾ – the same for audio data
- UserDataSenders: TRVSenderCollection⁽²¹²⁾ – the same for arbitrary data
- FileSenders: TRVSenderCollection⁽²¹²⁾ – the same for files
- CmdSenders: TRVSenderCollection⁽²¹²⁾ – the same for commands
- VideoDefaultAcceptAll: Boolean (default: *True*) instructs to accept all connections if VideoSenders is empty (if *False*, all are rejected)
- AudioDefaultAcceptAll: Boolean (default: *True*) – the same for audio data
- UserDefaultAcceptAll: Boolean (default: *True*) – the same for arbitrary data
- FileDefaultAcceptAll: Boolean (default: *True*) – the same for files
- CmdDefaultAcceptAll: Boolean (default: *True*) – the same for commands.

3.20 TRVSendOptions

Sending options for TRVCamSender⁽¹³⁹⁾.

Unit [VCL and LCL] MRVSender;

Unit [FMX] fmxMRVSender;

Syntax

```
TRVSendOptions = class(TPersistent);
```

▼ Hierarchy

TObject

TPersistent

Description

This is a type of TRVCamSender.SendOptions⁽¹⁵¹⁾ property.

This class has the properties:

- Video: TRVMSendType
- Audio: TRVMSendType;
- UserData: TRVMSendType
- Cmd: TRVMSendType
- FileData: TRVMSendType

where

type

```
TRVMSendType = (rvmvstOnlyCurrentData, rvmvstStream);
```

Value	Meaning
<i>rvmvstOnlyCurrentData</i>	A new connection is created when data are available, and is closed when data are sent
<i>rvmvstStream</i>	Continuous sending, connection is kept

Default value for all the properties above is *rvmvstStream*.

4 Global variables and constants

RVMedia Global Variables and Constants

*_DATA⁽²¹⁵⁾ constant identify data types.

glRVInetMaxConnect⁽²¹⁶⁾ defines maximal possible count of simultaneous connection to the same host.

4.1 *_DATA

Constants identifying a data type

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

```
RVVIDEO_DATA = 1;  
RVAUDIO_DATA = 2;  
RVUSER_DATA  = 4;  
RVCMD_DATA   = 8;  
RVFILE_DATA  = 16;  
RVMEDIA_DATA = RVVIDEO_DATA or RVAUDIO_DATA;
```

4.2 glRVInetMaxConnect

Maximal possible count of simultaneous connections to the same host.

Unit [VCL and LCL] MRVInetTypes;

Unit [FMX] fmxMRVInetTypes;

const

RVINET_DEFAULT_MAX_CONNECT = 8

var

glRVInetMaxConnect : Integer = RVINET_DEFAULT_MAX_CONNECT;

5 Global procedures

MRVCore Unit

MRVCore [VCL and LCL] (or **fmxMRVCore [FMX]**) includes the following functions:

- RVMGetWindowRect⁽²¹⁷⁾ returns coordinates of the specified top-level window

MRVDesktop Unit

MRVDesktop [VCL and LCL] (or **fmxMRVDesktop [FMX]**) includes the following procedures:

- GetVisibleWindowHandles⁽²¹⁸⁾ returns a list of top-level visible windows
- GetWindowTitleByHandle⁽²¹⁸⁾ returns a window title

MRVFFmpeg Unit

MRVFFmpeg [VCL and LCL] (or **fmxMRVFFmpeg [FMX]**) includes the following procedures:

- LoadFFmpegLibraries⁽²¹⁹⁾ [re]loads **FFmpeg** libraries from the specified path
- IsSupportedFFmpeg⁽²¹⁹⁾ checks if FFmpeg is available for the application.

MRVGStreamer Unit

MRVGStreamer [VCL and LCL] (or **fmxMRVGStreamer [FMX]**) includes the following procedures:

- LoadGStreamerLibraries⁽²²⁵⁾ [re]loads **GStreamer** libraries from the specified path
- IsSupportedGStreamer⁽²²⁶⁾ checks if GStreamer is available for the application.

MRVFFMpegLists Unit

MRVFFMpegLists [VCL and LCL] (or **fmxMRVFFMpegLists [FMX]**) includes the following procedures:

- GetListOfAvailableFFmpegFileFormats⁽²²²⁾ fills a list with file formats available for encoding
- GetListOfAvailableFFmpegAudioCodecs⁽²²¹⁾ fills a list with available audio codecs for encoding
- GetListOfAvailableFFmpegVideoCodecs⁽²²²⁾ fills a list with available video codecs for encoding
- GetListOfAvailableSampleFormats⁽²²³⁾ fills a list with sample formats available for the specified audio codec
- GetListOfAvailableSampleRates⁽²²³⁾ fills a list with sample rates available for the specified audio codec
- GetListOfAudioDecoders⁽²²⁰⁾ fills a list with available audio codecs for decoding (a low-level version)

- `GetListOfVideoDecoders`⁽²²⁰⁾ fills a list with available video codecs for decoding (a low-level version)
- `GetListOfAudioEncoders`⁽²²⁰⁾ fills a list with available audio codecs for encoding (a low-level version)
- `GetListOfVideoEncoders`⁽²²⁰⁾ fills a list with available video codecs for encoding (a low-level version)
- `GetListOfVideoInputFormats`⁽²²³⁾ fills list with available video input formats

MRVFormatInfo Unit

MRVFormatInfo [VCL and LCL] (or **fmxMRVFormatInfo [FMX]**) includes the following functions:

- `GetAudioCodecName`, `GetVideoCodecName`⁽²²⁵⁾ return name of the specified audio/video codec
- `GetAudioCodecFileExt`, `GetVideoCodecFileExt`⁽²²⁴⁾ return file extension for the specified audio/video codec
- `GetSampleFormatName`⁽²²⁵⁾ return name of the specified audio sample format

MRVRNNoise Unit

MRVRNNoise [VCL and LCL] (or **fmxMRVRNNoise [FMX]**) includes the following procedures:

- `LoadRNNoise`⁽²²⁷⁾ [re]loads **RNNoise**⁽²⁴⁾ library from the specified path
- `IsRNNoiseLoaded`⁽²²⁷⁾ checks if RNNoise library has been loaded

MRVWebCamFuncs Unit

MRVWebCamFuncs [VCL and LCL] (or **fmxMRVWebCamFuncs [FMX]**) includes the following functions:

- `DescribeVideoModePixelFormat`⁽²²⁸⁾ returns text describing a pixel format of a video mode
- `DescribeVideoMode`⁽²²⁸⁾ returns text describing a pixel format of a video mode

5.1 MRVCore Unit

MRVCore Unit

MRVCore [VCL and LCL] (or **fmxMRVCore [FMX]**) includes the following functions:

- `RVMGetWindowRect`⁽²¹⁷⁾ returns coordinates of the specified top-level window

5.1.1 RVMGetWindowRect

Returns the coordinates of the specified window (relative to the origin of the primary monitor).

Unit [VCL and LCL] `MRVCore`⁽²¹⁷⁾;

Unit [FMX] `fmxMRVCore`⁽²¹⁷⁾;

Syntax

```
function RVMGetWindowRect (AHWnd: TRVMWindowHandle(247);  
    out ARect: TRVMRect(246)) : Boolean;
```

The function returns *True* on success. In this case, coordinates are returned in **ARect** parameter.

See also:

- `GetVisibleWindowsHandles` ⁽²¹⁸⁾

5.2 MRVDesktop Unit

MRVDesktop Unit

MRVDesktop [VCL and LCL] (or **fmxMRVDesktop [FMX]**) includes the following procedures:

- `GetVisibleWindowHandles` ⁽²¹⁸⁾ returns a list of top-level visible windows
- `GetWindowTitleByHandle` ⁽²¹⁸⁾ returns a window title

5.2.1 GetVisibleWindowsHandles

Returns a list of handles of visible top-level windows.

Unit [VCL and LCL] `MRVDesktop` ⁽²¹⁸⁾;

Unit [FMX] `fmxMRVDesktop` ⁽²¹⁸⁾;

Syntax

```
function GetVisibleWindowHandles: TRVMWindowHandleArray (247);
```

This function is implemented for Windows and macOS platforms. For Linux, it always returns an empty list.

Returned handles can be used:

- as a parameter of `GetWindowTitleByHandle` ⁽²¹⁸⁾ function (to get a window title)
- as a parameter of `RVMGetWindowRect` ⁽²¹⁷⁾ function (to get a window coordinates)
- as a value of `TRVCamera` ⁽⁴²⁾.`DesktopWindowHandle` ⁽⁴⁷⁾ property

5.2.2 GetWindowTitleByHandle

Returns the title of the specified window.

Unit [VCL and LCL] `MRVDesktop` ⁽²¹⁸⁾;

Unit [FMX] `fmxMRVDesktop` ⁽²¹⁸⁾;

Syntax

```
function GetWindowTitleByHandle (
    Handle: TRVMWindowHandle (247)): String;
```

See also:

- `GetVisibleWindowHandles` ⁽²¹⁸⁾

5.3 MRVFFmpeg Unit

MRVFFmpeg Unit

MRVFFmpeg [VCL and LCL] (or fmxMRVFFmpeg [FMX]) includes the following procedures:

- LoadFFmpegLibraries⁽²¹⁹⁾ [re]loads **FFmpeg** libraries from the specified path
- IsSupportedFFmpeg⁽²¹⁹⁾ checks if FFmpeg is available for the application.

5.3.1 LoadFFmpegLibraries

[Re]Loads **FFmpeg** libraries from the specified **Path**.

Unit [VCL and LCL] MRVFFmpeg⁽²¹⁹⁾;

Unit [FMX] fmxMRVFFmpeg⁽²¹⁹⁾;

Syntax

```
procedure LoadFFmpegLibraries(Path : string);
```

By default, FFmpeg libraries are loaded from default locations (i.e. the current application directory and directories specified in PATH environment variable).

You can use this procedure for loading (or reloading) FFmpeg libraries located in a specific directory.

Note: by default, RVMedia loads *avdevice* library (that provides RVCamera⁽⁴²⁾.FFmpegProperty⁽⁵⁰⁾.VideoInputDevice⁽²⁰¹⁾ functionality) for all FFmpeg version except for 6.x (due to known errors in this version). If you have problems with other version of FFmpeg, you can disable avdevice loading by assigning *False* to the global variable **UseAVDevice**.

Windows

You can download compiled Win64 version of FFmpeg here:

<https://www.ffmpeg.org/download.html#build-windows> ("shared" versions are necessary).

macOS

You cannot install FFmpeg from the official website, because it does not provide "shared" builds.

You can use Homebrew package manager to install FFmpeg (type "brew install ffmpeg" in Terminal).

5.3.2 IsSupportedFFmpeg

Returns *True* if **FFmpeg** is available

Unit [VCL and LCL] MRVFFmpeg⁽²¹⁹⁾;

Unit [FMX] fmxMRVFFmpeg⁽²¹⁹⁾;

Syntax

function IsSupportedFFmpeg: Boolean;

If FFmpeg libraries were not loaded, the function loads them from the default location.

5.4 MRVFFMpegLists Unit

MRVFFMpegLists Unit

MRVFFMpegLists [VCL and LCL] (or **fmxMRVFFMpegLists [FMX]**) includes the following procedures:

- **GetListOfAvailableFFmpegFileFormats** ⁽²²²⁾ fills a list with file formats available for encoding
- **GetListOfAvailableFFmpegAudioCodecs** ⁽²²¹⁾ fills a list with available audio codecs for encoding
- **GetListOfAvailableFFmpegVideoCodecs** ⁽²²²⁾ fills a list with available video codecs for encoding
- **GetListOfAvailableSampleFormats** ⁽²²³⁾ fills a list with sample formats available for the specified audio codec
- **GetListOfAvailableSampleRates** ⁽²²³⁾ fills a list with sample rates available for the specified audio codec
- **GetListOfAudioDecoders** ⁽²²⁰⁾ fills a list with available audio codecs for decoding (a low-level version)
- **GetListOfVideoDecoders** ⁽²²⁰⁾ fills a list with available video codecs for decoding (a low-level version)
- **GetListOfAudioEncoders** ⁽²²⁰⁾ fills a list with available audio codecs for encoding (a low-level version)
- **GetListOfVideoEncoders** ⁽²²⁰⁾ fills a list with available video codecs for encoding (a low-level version)
- **GetListOfVideoInputFormats** ⁽²²³⁾ fills list with available video input formats

5.4.1 GetListOfAvailable-Audio/Video-Decoders/Encoders

The function fill lists of FFmpeg codecs of the specified type

Unit [VCL and LCL] MRVFFMpegLists ⁽²²⁰⁾;

Unit [FMX] fmxMRVFFMpegLists ⁽²²⁰⁾;

function GetListOfVideoDecoders (CodecNames,
CodecDescr: TStrings): Boolean;

function GetListOfAudioDecoders (CodecNames,
CodecDescr: TStrings): Boolean;

function GetListOfVideoEncoders (CodecNames,
CodecDescr: TStrings): Boolean;

function GetListOfAudioEncoders (CodecNames,
CodecDescr: TStrings): Boolean;

The functions fill the lists of **CodecNames** and **CodecDescr** (**CodecDescr** is optional, pass nil if you do not need codec descriptions).

CodecNames is a list of unique codec names that identify codecs.

CodecDescr is a list of codec names that can be shown to a user. Items of this list correspond to items of **CodecNames**.

You rarely need to use these functions. Normally:

- codecs for video and audio decoding are auto-detected, you do not need to specify them explicitly
- codecs for video and audio encoding are assigned as values of `TRVAudioCodec`⁽²³⁶⁾ and `TRVVideoCodec`⁽²⁵¹⁾, and you can check their availability using `GetListOfAvailableFFmpegAudioCodecs`⁽²²¹⁾ and `GetListOfAvailableFFmpegVideoCodecs`⁽²²²⁾ procedures.

GetListOfVideoDecoders returns a list of available codecs for video decoding. You can use values in **CodecNames** to assign `TRVCamera`⁽⁴²⁾.`FFMpegProperty`⁽⁵⁰⁾.`DecodeVideoCodecName`⁽²⁰¹⁾.

GetListOfAudioDecoders returns a list of available codecs for sound decoding. You can use values in **CodecNames** to assign `TRVCamera`⁽⁴²⁾.`FFMpegProperty`⁽⁵⁰⁾.`DecodeAudioCodecName`⁽²⁰¹⁾.

GetListOfVideoEncoders returns a list of available codecs for video encoding. You can use values in **CodecNames** to assign `TRVCamRecorder`⁽⁸⁵⁾.`VideoCodecName`⁽⁸⁸⁾.

GetListOfAudioEncoders returns a list of available codecs for sound encoding. You can use values in **CodecNames** to assign `TRVCamRecorder`⁽⁸⁵⁾.`AudioCodecName`⁽⁸⁸⁾ and `TRVAudioPlayer`⁽¹¹⁰⁾.`EncodeAudioCodecName`⁽¹¹²⁾.

The functions return *True* if the list of codes is read successfully.

5.4.2 GetListOfAvailableFFmpegAudioCodecs

Fills **AList** with audio formats available for encoding.

Unit [VCL and LCL] `MRVFFMpegLists`⁽²²⁰⁾;

Unit [FMX] `fmvMRVFFMpegLists`⁽²²⁰⁾;

```
procedure GetListOfAvailableFFmpegAudioCodecs(AList: TStrings;
  const DefaultCaption: String = '');;
```

This function works only if **FFmpeg** is available. Otherwise, it simply clears **AList**.

Each added item has a text describing a codec, and an object equal to the corresponding codec identifier (a `TRVAudioCodec`⁽²³⁶⁾ value type-casted to `TObject`).

If **DefaultCaption** is not empty, the procedure uses it to add the first item with the object equal to *rvacDefault*. Otherwise, *rvacDefault* is not added.

Internally, this function uses `GetAudioCodecName`⁽²²⁵⁾ to assign text to items.

After calling this function, you can use **AList** to choose a value for

- `TRVCamRecorder`⁽⁸⁵⁾.`AudioCodec`⁽⁸⁸⁾
- `TRVAudioPlayer`⁽¹¹⁰⁾.`EncodeCodec`⁽¹¹²⁾

See also:

- `GetListOfAudioEncoders`⁽²²⁰⁾

5.4.3 GetListOfAvailableFFmpegFileFormats

Fills **AList** with video file formats available for encoding.

Unit [VCL and LCL] MRVFFMpegLists⁽²²⁰⁾;

Unit [FMX] fmxMRVFFMpegLists⁽²²⁰⁾;

```
procedure GetListOfAvailableFFmpegFileFormats(AList : TStrings);
```

This function works only if **FFmpeg** is available. Otherwise, it simply clears **AList**.

The function checks availability of encoders of the following file formats:

- MP4
- AVI
- MPEG
- 3GP
- ASF (if MRVCODEC_MSMPEG4v3 is \$defined)
- FLV
- MKV
- MOV
- RM
- MJPEG

Available formats are added to **AList**.

After calling this function, you can use **AList** to choose a file extension for OutputFileName⁽⁸⁹⁾ property of TRVCamRecorder⁽⁸⁵⁾ component.

5.4.4 GetListOfAvailableFFmpegVideoCodecs

Fills **AList** with video formats available for encoding.

Unit [VCL and LCL] MRVFFMpegLists⁽²²⁰⁾;

Unit [FMX] fmxMRVFFMpegLists⁽²²⁰⁾;

```
procedure GetListOfAvailableFFmpegVideoCodecs(AList: TStrings;  
  const DefaultCaption: String = ''; AddExt: Boolean = False);
```

This function works only if **FFmpeg** is available. Otherwise, it simply clears **AList**.

Each added item has a text describing a codec, and an object equal to the corresponding codec identifier (a TRVVideoCodec⁽²⁵¹⁾ value type-casted to TObject).

If **DefaultCaption** is not empty, the procedure uses it to add the first item with the object equal to *rvvcDefault*. Otherwise, *rvvcDefault* is not added.

If **AddExt** = *True*, the procedure adds possible file extensions to text of each item, in square brackets.

Internally, this function uses GetVideoCodecName⁽²²⁵⁾ and GetVideoFileExts⁽²²⁴⁾ to assign text to items.

After calling this function, you can use **AList** to choose a value for TRVCamRecorder⁽⁸⁵⁾.VideoCodec⁽⁸⁸⁾.

See also:

- GetListOfVideoEncoders⁽²²⁰⁾

5.4.5 GetListOfAvailableSampleFormats

Fills **AList** with audio sample formats acceptable for the **Codec**.

Unit [VCL and LCL] MRVFFMpegLists⁽²²⁰⁾;

Unit [FMX] fmxMRVFFMpegLists⁽²²⁰⁾;

procedure GetListOfAvailableSampleFormats(Codec: TRVAudioCodec⁽²³⁶⁾; AList : TStrings);

This function works only if **FFmpeg** is available. Otherwise, it simply clears **AList**.

Each added item has a text describing the sample format, and an object equal to the corresponding sample format (a TRVSampleFormat⁽²⁴⁹⁾ value type-casted to TObject).

Internally, this function uses GetSampleFormatName⁽²²⁵⁾ to assign text to items.

5.4.6 GetListOfAvailableSampleRates

Fills **AList** with audio sample rates acceptable for the **Codec**.

Unit [VCL and LCL] MRVFFMpegLists⁽²²⁰⁾;

Unit [FMX] fmxMRVFFMpegLists⁽²²⁰⁾;

procedure GetListOfAvailableSampleRates(Codec: TRVAudioCodec⁽²³⁶⁾; SampleFormat: TRVSampleFormat⁽²⁴⁹⁾; AList : TStrings);

Sample rate is a number of audio samples played in 1 second.

This function works only if **FFmpeg** is available. Otherwise, it simply clears **AList**.

For most codecs, **AList** is filled basing on **Codec** parameter, **SampleFormat** parameter is ignored.

Only for **Codec** = *rvacWAV*, **SampleFormat** is used (because RVMedia chooses different WAV codec depending on sample format).

Each added item has a text equal to a text representation of a sample rate, and an object equal to a sample rate (an integer value type-casted to TObject).

Not all codecs provide a list of acceptable sample rates.

After calling this function, you can use **AList** to choose possible value of:

- AudioSampleRate⁽⁸⁷⁾ property of TRVCamRecorder⁽⁸⁵⁾ component;
- EncodeSampleRate⁽¹¹²⁾ property of TRVAudioPlayer⁽¹¹⁰⁾ component.

5.4.7 GetListOfVideoInputFormats

Fills **ShortNames** and **LongNames** with available input formats.

Unit [VCL and LCL] MRVFFMpegLists⁽²²⁰⁾;

Unit [FMX] fmxMRVFFMpegLists⁽²²⁰⁾;

```
function GetListOfVideoInputFormats (ShortNames,
    LongNames: TStrings): Boolean;
```

ShortNames or **LongNames** can be nil. Only non-nil lists are filled.

LongNames are for descriptions.

ShortNames contain comma-delimited values (but usually a single value) that can be assigned to TRVCamera⁽⁴²⁾.FFmpegProperty⁽⁵⁰⁾.VideoInputFormat⁽²⁰¹⁾.

Note: due to known errors, input formats are disabled for FFmpeg 6.x (avdevice-60.dll), but enabled for older and newer versions.

Input formats identify various platform-specific video sources (such as video grabbing devices). Filename/URL for these devices often does not refer to an actually existing file or a network resource, but has some specific meaning.

Some values that can be returned:

- dshow: DirectShow capture
- gdigrab: GDI API Windows frame grabber

Additional information: <https://ffmpeg.org/ffmpeg-devices.html>

5.5 MRVFormatInfo Unit

MRVFormatInfo Unit

MRVFormatInfo [VCL and LCL] (or **fmxMRVFormatInfo [FMX]**) includes the following functions:

- GetAudioCodecName, GetVideoCodecName⁽²²⁵⁾ return name of the specified audio/video codec
- GetAudioCodecFileExt, GetVideoCodecFileExts⁽²²⁴⁾ return file extension for the specified audio/video codec
- GetSampleFormatName⁽²²⁵⁾ return name of the specified audio sample format

5.5.1 GetAudioCodecFileExt, GetVideoCodecFileExts

The functions return file extension for codecs.

Unit [VCL and LCL] MRVFormatInfo⁽²²⁴⁾;

Unit [FMX] fmxMRVFormatInfo⁽²²⁴⁾;

```
function GetAudioCodecFileExt (Codec: TRVAudioCodec(236)): String;
```

```
function GetVideoCodecFileExts (Codec: TRVVideoCodec(251)): String;
```

Extensions are hard-coded and are not localizable. You can see them in "Recommended file extension" column of the tables in the topics about the codec types.

GetAudioCodecFileExt returns a single extension for each codec. It may help to assign extension to TRVAudioPlayer⁽¹¹⁰⁾.OutputFileName⁽¹¹³⁾.

GetVideoCodecFileExts may return several extensions separated by space character. It may help to find possible values of `TRVCamRecorder(85).VideoCodec(88)` corresponding to the extension of `TRVCamRecorder.OutputFileName(89)`.

5.5.2 GetAudioCodecName, GetVideoCodecName

The functions return names of codecs.

Unit [VCL and LCL] `MRVFormatInfo(224)`;

Unit [FMX] `fmxMRVFormatInfo(224)`;

function `GetAudioCodecName (Codec: TRVAudioCodec(236)) : String;`

function `GetVideoCodecName (Codec: TRVVideoCodec(251)) : String;`

Names are hard-coded and are not localizable. You can see them in "Format name" column of the tables in the topics about the codec types.

These names are intended for displaying to users, not for identifying codecs.

5.5.3 GetSampleFormatName

Returns name of the specified audio sample format.

Unit [VCL and LCL] `MRVFormatInfo(224)`;

Unit [FMX] `fmxMRVFormatInfo(224)`;

function `GetSampleFormatName (Value: TRVSampleFormat(249)) : String;`

Names are hard-coded and are not localizable.

These names are intended for displaying to users, not for identifying sample formats.

5.6 MRVGStreamer Unit

MRVGStreamer Unit

MRVGStreamer [VCL and LCL] (or **fmxMRVGStreamer [FMX]**) includes the following procedures:

- `LoadGStreamerLibraries(225)` [re]loads **GStreamer** libraries from the specified path
- `IsSupportedGStreamer(226)` checks if GStreamer is available for the application.

5.6.1 LoadGStreamerLibraries

[Re]Loads **GStreamer** libraries from the specified **Path**.

Unit [VCL and LCL] `MRVGStreamer(225)`;

Unit [FMX] `fmxMRVGStreamer(225)`;

Syntax

procedure `LoadGStreamerLibraries (const Path: string);`

You can use this procedure for loading (or reloading) GStreamer libraries located in a specific directory. If **Path** is empty, GStreamer libraries are loaded from the default location.

If you do not call this procedure, GStreamer will be loaded from the default location when necessary.

Windows

On Windows, the default location is stored by the GStreamer installer in the following system environment variables:

- GSTREAMER_1_0_ROOT_X86_64 or GSTREAMER_1_0_ROOT_MSVC_X86_64 or GSTREAMER_1_0_ROOT_MINGW_X86_64: GStreamer 1.0 for Win64
- GSTREAMER_SDK_ROOT_X86_64: GStreamer 0.1 for Win64
- GSTREAMER_1_0_ROOT_X86 or GSTREAMER_1_0_ROOT_MSVC_X86 or GSTREAMER_1_0_ROOT_MINGW_X86: GStreamer 1.0 for Win32
- GSTREAMER_SDK_ROOT_X86: GStreamer 0.1 for Win32

RVMedia uses these variables to find the default GStreamer location.

If the both versions (0.1 and 1.0) of GStreamer are available, GStreamer 1.0 is loaded.

You can specify **Path** to load GStreamer libraries from the specified location. **Path** may be a root GStreamer folder or its "bin" subfolder. Ending "\" is optional.

macOS

For macOS, the recommended way of installing GStreamer is the runtime installer from <https://gstreamer.freedesktop.org/download/#macos>.

(an alternative way, using Homebrew package, is not recommended).

The default location for GStreamer is

'/Library/Frameworks/GStreamer.framework/Versions/1.0/lib/'. If **Path** is empty, RVMedia loads GStreamer from this location.

5.6.2 IsSupportedGStreamer

Returns *True* if **GStreamer** is available.

Unit [VCL and LCL] MRVGGStreamer⁽²²⁵⁾;

Unit [FMX] fmxxMRVGGStreamer⁽²²⁵⁾;

Syntax

```
function IsSupportedGStreamer: Boolean;
```

If GStreamer libraries were not loaded, the function loads them from the default location.

5.7 MRVRNNoise Unit

MRVRNNoise Unit

MRVRNNoise [VCL and LCL] (or **fmxMRVRNNoise [FMX]**) includes the following procedures:

- **LoadRNNoise**⁽²²⁷⁾ [re]loads **RNNoise**⁽²⁴⁾ library from the specified path
- **IsRNNoiseLoaded**⁽²²⁷⁾ checks if RNNoise library has been loaded

5.7.1 LoadRNNoise

[Re]Loads **RNNoise**⁽²⁴⁾ libraries.

Unit [VCL and LCL] **MRVRNNoise**⁽²²⁷⁾;

Unit [FMX] **fmxMRVRNNoise**⁽²²⁷⁾;

Syntax

```
function LoadRNNoise(Path: String = '';  
    LibName: String = ''): Boolean;
```

By default, RNNoise libraries are loaded from from:

- Windows: application folder (or folder from the PATH environment variable)
- Linux: application folder
- macOS: bundle folder, or "Contents/Resources/Data" folder in the bundle.

By default, the library has the following name:

- Windows 32-bit: librnnoise-windows-x86.dll
- Windows 64-bit: librnnoise-windows-x86-64.dll
- Linux: librnnoise-linux-x86-64.so
- macOS 64-bit ARM: librnnoise-macos-aarch64.dylib
- macOS Intel: librnnoise-macos-x86-64.dylib

These values are used when **Path** or **LibName** are empty. You can call this function with the specified **Path** and/or **LibName**.

If RNNoise is already loaded, and **Path** or **LibName** are empty, this function does nothing and returns *True*. Otherwise, it tries to load RNNoise libraries and returns *True* on success.

5.7.2 IsRNNoiseLoaded

Returns *True* if **RNNoise**⁽²⁴⁾ library has been loaded.

Unit [VCL and LCL] **MRVRNNoise**⁽²²⁷⁾;

Unit [FMX] **fmxMRVRNNoise**⁽²²⁷⁾;

Syntax

```
function IsRNNoiseLoaded: Boolean;
```

This function does not load RVNoise.

5.8 MRVWebCamFuncs Unit

MRVWebCamFuncs Unit

MRVWebCamFuncs [VCL and LCL] (or **fmxMRVWebCamFuncs [FMX]**) includes the following functions:

- **DescribeVideoModePixelFormat**⁽²²⁸⁾ returns text describing a pixel format of a video mode
- **DescribeVideoMode**⁽²²⁸⁾ returns text describing a pixel format of a video mode

5.8.1 DescribeVideoMode

Returns text describing the specified video mode of a local camera.

Unit [VCL and LCL] MRVWebCamFuncs⁽²²⁸⁾;

Unit [FMX] fmxMRVWebCamFuncs⁽²²⁸⁾;

Syntax

```
function DescribeVideoMode (
  const Mode: TRVCamVideoMode(239)): String;
```

5.8.2 DescribeVideoModePixelFormat

Returns text describing pixel format of the specified video mode of a local camera.

Unit [VCL and LCL] MRVWebCamFuncs⁽²²⁸⁾;

Unit [FMX] fmxMRVWebCamFuncs⁽²²⁸⁾;

Syntax

```
function DescribeVideoModePixelFormat (
  const Mode: TRVCamVideoMode(239)): String;
```

6 Examples

Examples

1. How to play a video from an IP camera in a "wait" mode⁽²²⁸⁾
2. How to play a video from an IP camera in a "no-wait" mode⁽²²⁹⁾.
3. How to play a video stream⁽²³⁰⁾

6.1 Example 1: playing a camera video (wait mode)

This example shows how to play a video from an IP camera in a "wait" mode: the component waits for each operation to complete. This mode is simpler but it is not recommended: an application freezes while waiting.

Example

1. Place `RVCamera1:TRVCamera`⁽⁴²⁾ and `RVCamView1:TRVCamView`⁽⁹³⁾ on the form. Assign `RVCamView.VideoSource = RVCamera1`.
2. If you need a separate component for controlling the camera movement, place `RVCamControl1:TRVCamControl`⁽³⁹⁾ on the form. Assign `RVCamera1.CameraControl`⁽⁴⁶⁾ = `RVCamControl`.
3. Assign `RVCamera1`'s properties: the camera address and the user name and password. For example:
 - `CameraHost`⁽⁴⁶⁾ = 'novatron.dyndns.tv',
 - `CameraPort` = 8888
 - `UserName`⁽⁵⁷⁾ = 'demo15'
 - `UserPassword` = 'demo15'
4. Assign `RVCamera.CommandMode`⁽⁴⁷⁾ = `rvsmWait`.
5. Add in `TForm1.FormCreate`:

```
if RVCamera1.SearchCamera(68) then
  RVCamera1.PlayVideoStream(67);
```

6.2 Example 2: playing a camera video (no wait mode)

This example shows how to play a video from an IP camera in a "no wait" mode: operations are performed in background threads, without waiting. When an operation is finished, an event is called. This is a recommended mode, you can provide a visual feedback while waiting, and the user can interrupt a long operation.

Example

1. Place `RVCamera1:TRVCamera`⁽⁴²⁾ and `RVCamView1:TRVCamView`⁽⁹³⁾ on the form. Assign `RVCamView.VideoSource = RVCamera1`.
2. If you need a separate component for controlling the camera movement, place `RVCamControl1:TRVCamControl`⁽³⁹⁾ on the form. Assign `RVCamera1.CameraControl`⁽⁴⁶⁾ = `RVCamControl`.
3. Assign `RVCamera1`'s properties: the camera address and the user name and password. For example:
 - `CameraHost`⁽⁴⁶⁾ = 'novatron.dyndns.tv',
 - `CameraPort` = 8888
 - `UserName`⁽⁵⁷⁾ = 'demo15'
 - `UserPassword` = 'demo15'
4. Assign `RVCamera.CommandMode`⁽⁴⁷⁾ = `rvsmNoWait`.
5. Add in `TForm1.FormCreate`:

```
RVCamera1.SearchCamera(68);
```

6. Create the event handler for `RVCamera1.OnSearchComplete`⁽⁷³⁾ event:

```
procedure Form1.RVCamera1SearchComplete(Sender: TObject;
  Status: Integer);
begin
  if Status = 0 then
    RVCamera1.PlayVideoStream(67);
```

```
end;
```

6.3 Example 3: playing a video stream

If the IP camera does not have a direct Internet access, but its video steam (or video frames) can be read from the specified address, you can use `RVCamera.URL` property.

Example

1. Place `RVCamera1:TRVCamera`⁽⁴²⁾ and `RVCamView1:TRVCamView`⁽⁹³⁾ on the form. Assign `RVCamView.VideoSource = RVCamera1`.

2. Assign `RVCamera1.URL` property. For example: 'http://85.199.8.252/record/current.jpg?resolution=CIF').

3. Add in `TForm1.FormCreate`:

```
RVCamera1.PlayVideoStream(67);
```

7 Types

Audio and Video Decoding/Encoding

- `TRVAudioCodec`⁽²³⁶⁾ – a codec for sound recording.
- `TRVVideoCodec`⁽²⁵¹⁾ – a codec for video recording.
- `TRVBitsPerSample`⁽²³⁷⁾ – a format of sound samples (simple version).
- `TRVSampleFormat`⁽²⁴⁹⁾ – a format of sound samples (advanced version).
- `TRVSamplesPerSec`⁽²⁵⁰⁾ – a sound sample rate.
- `TRVFFmpegFilter`⁽²⁴³⁾ – image scaling method

Data Transfer

- `TRVChangedAreaProcessingMode`⁽²⁴⁰⁾ affect processing of changed areas before sending.
- `TRVCompressionType`⁽²⁴¹⁾ – a compression of data before sending.
- `TRVBoundsTestMode`⁽²³⁷⁾ activates a test sending mode that shows changed areas of frames.
- `TRVEncodingType`⁽²⁴²⁾ – a video encoding type.
- `TRVMediaType`⁽²⁴⁴⁾ – a media type.
- `TRVParamType`⁽²⁴⁸⁾ – a type of a command parameter.
- `TRVProtocol`⁽²⁴⁸⁾ – a data transfer protocol.
- `TRVSessionKey`⁽²⁵⁰⁾ – a session key.
- `TRVSocket`⁽²⁵¹⁾ – a socket.
- `TRVTCPCConnectionType`⁽²⁵¹⁾ specifies how sender and receiver initiate connections.
- `TRVVideoResolution`⁽²⁵²⁾ – a resolution of video when sending.

Types for TRVCamera

- `TRVCameraType`, `TRVCameraTypes`⁽²³⁸⁾ – types of IP cameras.
- `TRVCamVideoMode`, `TRVColorModel`⁽²³⁹⁾ – a video mode of a USB web camera.
- `TRVColorControlProperty`⁽²⁴⁰⁾ – a color control property (brightness, contrast, etc.)
- `TRVDesktopVideoMode`⁽²⁴¹⁾ – a video mode of a desktop.
- `TRVJpegIntegrity`⁽²⁴³⁾ specifies how JPEG images are checked when receiving JPEG or MJPEG streams.

- TRVVideoResolution⁽²⁵²⁾ – a desired video resolution.

Other Types

- TRVMAnsiString⁽²⁴⁴⁾ – a text string.
- TRVCamMoveMode⁽²³⁸⁾ – camera movement control mode for video viewers.
- TRVMLanguage⁽²⁴⁵⁾ – a UI language.
- TRVMRect, TRVMPoint, TRVUnitSize⁽²⁴⁶⁾ – coordinates.
- TRVMRenderMode⁽²⁴⁶⁾ – a video rendering mode.

Types of Events

These types are used for several events in different components:

- TRVAudioEvent⁽²³¹⁾
- TRVCamDoneEvent⁽²³²⁾
- TRVCamErrorEvent⁽²³²⁾
- TRVImageEvent⁽²³⁵⁾
- TRVCmdEvent⁽²³³⁾
- TRVDataReadEvent⁽²³⁴⁾
- TRVFullScreenEvent⁽²³⁴⁾
- TRVSocketEvent⁽²³⁵⁾

7.1 Events

Types of Events

These types are used for several events in different components:

- TRVAudioEvent⁽²³¹⁾
- TRVCamDoneEvent⁽²³²⁾
- TRVCamErrorEvent⁽²³²⁾
- TRVImageEvent⁽²³⁵⁾
- TRVCmdEvent⁽²³³⁾
- TRVDataReadEvent⁽²³⁴⁾
- TRVFullScreenEvent⁽²³⁴⁾
- TRVSocketEvent⁽²³⁵⁾

7.1.1 TRVAudioEvent

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVAudioEvent = procedure (Sender: TObject; AStream: TMemoryStream;
    var ADataSize : Integer; const AAudioIndex : Word;
    var AStartTime: Int64; var ADuration: Cardinal;
    var ASamplesPerSec: Integer; var ABitsPerSample : TRVBitsPerSample(237);
    var AChannels: Integer) of object;
```

AStream contains sound data, **ASamplesPerSec**, **ABitsPerSample**, **AChannels** contain sound parameters.

Only starting **ADataSize** bytes in **AStream** contain sound, other content is undefined.

AAudioIndex may be non-zero if this sound is received from TRVCamReceiver⁽¹²⁰⁾. In this case, this is an index of media channel of TRVCamSender⁽¹³⁹⁾ which sent these audio data.

AStartTime is a time from the beginning of sound recording/playing to the beginning of this sound fragment, in milliseconds.

ADuration is a length of this sound fragment, in milliseconds.

This is the type of the following events:

- TCustomRVAudioOutput⁽¹⁹¹⁾.OnGetAudio⁽¹⁹²⁾
- TCustomRVMicrophone⁽¹⁷⁷⁾.OnGetAudio⁽¹⁸⁷⁾
- TRVCamSender⁽¹³⁹⁾.OnEncodeAudio⁽¹⁶⁴⁾

You can use this event, for example, for writing sound to a file, or to modify sound before processing.

7.1.2 TRVCamDoneEvent

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVCamDoneEvent = procedure (Sender : TObject;  
    Status : Integer) of object;
```

TRVCamDoneEvent is used for TRVCamera⁽⁴²⁾ events:

- OnEndVideoFile, OnEndVideoStream⁽⁷⁰⁾
- OnDownloaded⁽⁷⁰⁾
- OnMoved⁽⁷²⁾
- OnSearchComplete⁽⁷³⁾
- OnPrepared⁽⁷²⁾
- OnSetProperty⁽⁷³⁾
- OnEndMove⁽¹⁰⁷⁾

These events occur when some operation is completed. The Status parameter contains 0 if the operation is performed successfully, or nonzero value otherwise.

7.1.3 TRVCamErrorEvent

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```

TRVCamErrorSource = (rvcesFFmpeg, rvcesGStreamer,
    rvcesWebCamera, rvcesFilePlayer);

TRVCamErrorEvent = procedure (Sender: TObject;
    Source: TRVCamErrorSource;
    const ErrorCode: Integer; const ErrorString: String;
    var IsCritical: Boolean) of object;

```

This is the type of the following events:

- TRVAudioPlayer.OnError⁽¹¹⁴⁾ (on recording audio)
- TRVCamera.OnError⁽⁷⁰⁾ (on receiving or remuxing video)
- TRVCamRecorder.OnError⁽⁹²⁾ (on recording video).

These events allows handling errors and warnings.

Parameters

Source – error source (FFmpeg decoding or encoding, or GStreamer decoding, web camera (video capture device), local file player).

ErrorCode – numeric error code. Error codes are platform-dependent. For example, RVMedia uses different methods to access video capture devices on Windows, macOS, and Linux, each with its own set of possible error codes.

ErrorString – human readable error message (in English)

If **IsCritical** = *True*, the error is critical, and the operation will be interrupted. Any value assigned to **IsCritical** in this event will be ignored.

If **IsCritical** = *False*, the error is considered a warning, and the operation will continue. However, you can set **IsCritical** := *True* in the event handler to stop the current operation.

7.1.4 TRVCmdEvent

Unit [VCL and LCL] MRVReceiver;

Unit [FMX] fmxMRVReceiver;

type

```

TRVCmdEvent = procedure (Sender: TRVCamReceiver(120);
    SessionKey: TRVSessionKey(250);
    const GUIDGroup, GUIDUser: TRVMAnsiString(244);
    ACmd : TRVCmd(197)) of object;

```

Parameters:

GUIDGroup – identifier of the group

GUIDUser – identifier of the client which sent

This is the type of the following events of TRVCamReceiver⁽¹²⁰⁾:

- OnGetAllGroups⁽¹³³⁾
- OnGetAllUsers, OnGetAllOnlineUsers⁽¹³³⁾

7.1.5 TRVDataReadEvent

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVDataReadEvent = procedure (Sender: TObject; AData: TStream;
    ASocket: TRVSocket(251);
    GUIDFrom, GUIDTo, GUIDGroup : TGUID) of object;
```

AData – received data.

ASocket – a socket from which data are read. 0 if data are received from a camera.

All GUID parameters are available only for TCP connections. For UDP connections, they are equal to 0.

GUIDFrom – identifier of the data sender (if available)

GUIDTo – identifier of the data receiver (if available)

GUIDGroup – identifier of the group in TRVMediaServer⁽¹⁶⁵⁾ (if available)

This is the type of the following events:

- TRVRecvSource.OnDataRead⁽¹⁸⁵⁾

7.1.6 TRVFullScreenEvent

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVFullScreenEvent = procedure (Sender: TObject;
    const AShow: Boolean) of object;
```

This is the type of the following events:

- TRVCamView.OnFullScreen⁽¹⁰⁷⁾
- TRVCamMultiView.OnFullScreen⁽⁸⁵⁾

These events is called before showing a full-screen window, and when closing a full-screen window.

Parameters

AShow = *True* when a full-screen window is about to show.

7.1.7 TRVGetMediaStreamIndexEvent

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVGetMediaStreamIndexEvent = procedure (Sender: TObject;
    StreamCount: Integer; var StreamIndex: Integer) of object;
```

This is the type of the following events:

- TRVCamSender.OnConnected, OnConnecting, OnDisconnect, OnConnectError⁽¹⁶³⁾
- TRVCamReceiver.OnConnected, OnConnecting, OnDisconnect, OnConnectError⁽¹³⁰⁾

This event allows choosing a video or an audio stream to play.

Parameters

StreamCount – the count of video or audio streams in a video file

StreamIndex – index of the media stream to play

7.1.8 TRVImageEvent

Unit [VCL and LCL] MRVBitmap;

Unit [FMX] fmxMRVBitmap;

type

```
TRVImageEvent = procedure (Sender: TObject;  
    img: TRVMBitmap(206)) of object;
```

TRVImageEvent is used for the events:

- TRVCamera⁽⁴²⁾.OnNewImage⁽⁷²⁾
- TRVCamera⁽⁴²⁾.OnGetImage⁽⁷¹⁾
- TRVCamReceiver⁽¹²⁰⁾.OnGetImage⁽¹³⁴⁾
- TRVCamRecorder⁽⁸⁵⁾.OnGetImage⁽⁹³⁾

This event allows receiving an image (a video frame or a snap shot), or provide a video frame for a video.

See also

- GetSnapShot⁽⁶⁵⁾

7.1.9 TRVSocketEvent

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVSocketEvent = procedure (Sender : TObject;  
    SessionKey: TRVSessionKey(250); MediaTypes: TRVMediaTypes(244);  
    RemoteSessionKey: TRVSessionKey(250)) of object;
```

This is the type of the following events:

- TRVCamSender.OnConnected, OnConnecting, OnDisconnect, OnConnectError⁽¹⁶³⁾
- TRVCamReceiver.OnConnected, OnConnecting, OnDisconnect, OnConnectError⁽¹³⁰⁾

Parameters

SessionKey equals to the **Sender**'s SessionKey (depending on the component, it is TRVCamSender⁽¹³⁹⁾.SessionKey⁽¹⁵¹⁾ or TRVCamReceiver⁽¹²⁰⁾.SessionKey⁽¹²⁶⁾). If you perform time-consuming operations inside an event, it makes sense to compare values of **SessionKey** parameter and SessionKey property, to make sure that a connection was not closed or reopened.

RemoteSessionKey can be used when a remote side initiates the connection, and this side accepts it (i.e., for events in TRVCamSender, its TCPConnectionType⁽¹⁵³⁾ = *rvtcpReceiverToSender*; for events in TRVCamReceiver, its TCPConnectionType⁽¹²⁸⁾ = *rvtcpSenderToReceiver*). In this case, **RemoteSessionKey** is a SessionKey property of the remote side, and you can use it to detect reconnections. In the opposite connection mode (when this side initiates the connection, and a remote side accepts it), **RemoteSessionKey** = 0.

MediaTypes identifies a data type for this connection. It can be either empty or contain a single data type, see the topics about the events.

7.2 TRVAudioCodec

Specifies a codec used to encode sound in TRVAudioPlayer⁽¹¹⁰⁾ and TRVCamRecorder⁽⁸⁵⁾ components.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVAudioCodec = (rvacDefault,
  rvacWAV, rvacMP2, rvacMP3, rvacAC3,
  rvacVorbis, rvacFLAC, rvacOpus, rvacWMAv1, rvacWMAv2,
  rvacAAC, rvacRA1, rvacG723_1, rvacSpeex,
  {$IFDEF MRV_CODEC_DTS} // still experimental in FFmpeg 4
  rvacDTS,
  {$ENDIF}
  rvacWavPack, rvacALAC, rvacAMR);
```

Value	Format name (see GetAudioCodecName ⁽²²⁵⁾)	Recommended file extension (see GetAudioFileExt ⁽²²⁴⁾)
<i>rvacWAV</i>	WAV (Waveform Audio)	wav
<i>rvacMP2</i>	MP2 (MPEG-1 Audio Layer II)	mp2
<i>rvacMP3</i>	MP3 (MPEG-1 Audio Layer III)	mp3
<i>rvacAC3</i>	AC3 (Dolby Audio Codec 3)	ac3
<i>rvacVorbis</i>	Vorbis	ogg
<i>rvacFLAC</i>	FLAC (Free Lossless Audio Codec)	flac
<i>rvacOpus</i>	Opus	opus
<i>rvacWMAv1</i>	WMA (Windows Media Audio) v.1	wma
<i>rvacWMAv2</i>	WMA (Windows Media Audio) v.2	wma

<i>rvacAAC</i>	AAC (Advanced Audio Coding)	m4a
<i>rvacRA1</i>	RealAudio 1	ra
<i>rvacG723_1</i>	G.723.1	wav
<i>rvacSpeex</i>	Speex	spx
<i>rvacDTS</i>	DTS	dts
<i>rvacWavPack</i>	WavPack	wv
<i>rvacALAC</i>	ALAC (Apple Lossless Audio Codec)	m4a
<i>rvacAMR</i>	AMR (Adaptive Multi-Rate audio codec)	amr

Warning: Some audio formats may be patent-protected in some countries, and supporting these formats will require from you obtaining licenses from the patent owners.

This type is used for:

- `TRVAudioPlayer.EncodeAudioCodec`⁽¹¹²⁾ property.
- `TRVCamRecorder.AudioCodec`⁽⁸⁸⁾ property

See also:

- `TRVVideoCodec`⁽²⁵¹⁾

7.3 TRVBitsPerSample

Specifies a bit depth (a number of bits in a sound sample)

Unit [VCL and LCL] `MRVType`;

Unit [FMX] `fmxMRVType`;

type

```
TRVBitsPerSample = (rvbps8, rvbps16);
```

Value	Meaning
<i>rvbps8</i>	8 bits, unsigned integer
<i>rvbps16</i>	16 bits, signed integer

See also

- `TRVSampleFormat`⁽²⁴⁹⁾ (an advanced version of this type, used for recording)
- `TCustomRVMicrophone`⁽¹⁷⁷⁾.`BitsPerSample`⁽¹⁷⁹⁾ property

7.4 TRVBoundsTestMode

Allows to visualize changes in a video frame (comparing to the previous video frame).

Unit [VCL and LCL] `MRVType`;

Unit [FMX] fmxMRVType;

type

```
TRVBoundsTestMode = (rvstmNone, rvstmChangedFragments.  
rvstmRectangles);
```

Value	Meaning
<i>rvstmNone</i>	Test mode is turned off
<i>rvstmChangedFragments</i>	An image showing changed areas is created. In this image: unchanged pixels are black, changed pixels are highlighted (greater changes - brighter pixels); changed areas are outlined.
<i>rvstmRectangles</i>	An image showing changed areas is created. In this image, changed areas are outlined.

This type is used for properties:

- TRVCamSender⁽¹³⁹⁾.TestMode⁽¹⁵⁴⁾
- TRVMotionDetector⁽²⁰⁹⁾.TestMode⁽²¹⁰⁾

7.5 TRVCameraType, TRVCameraTypes

The types list camera models.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVCameraType = (rvctFoscam, rvctPanasonic, rvctAviosys,  
rvctACTi, rvctArcVision, rvctAxis, rvctDLink, rvctBeward,  
rvctGenius, rvctPlanet, rvctSamsung, rvctTrendnet,  
rvctVivotek, rvctMobotix, rvctUnknown);  
TRVCameraTypes = set of TRVCameraType;
```

This is used in:

- TRVCamera⁽⁴²⁾.IPCameraTypes⁽⁵²⁾
- TRVCamera⁽⁴²⁾.SearchCamera⁽⁶⁸⁾

7.6 TRVCamMoveMode

A camera movement control modes for video viewers.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVCamMoveMode = (vcmmNone, vcmmToCursor, vcmmDrag);
```

Move Mode	Meaning
<i>vcmmNone</i>	The component does not move the camera
<i>vcmmToCursor</i>	Click to rotate the camera to the mouse pointer
<i>vcmmDrag</i>	Click and drag (holding the mouse button) to rotate the camera to the drag direction

See also

- TRVCamView⁽⁹³⁾.CamMoveMode⁽⁹⁵⁾
- TRVCamMultiView⁽⁷⁴⁾.CamMoveMode⁽⁷⁷⁾

7.7 TRVCamVideoMode, TRVColorModel

Contains information about webcam video mode.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

For Windows**type**

```
TRVColorModel = (rvcmARGB, rvcmRGB, rvcmYV, rvcmOther);
TRVCamVideoMode = packed record
  Index:      Integer;
  Width:      Integer;
  Height:     Integer;
  ColorDepth: byte;
  ColorModel: TRVColorModel;
  VideoFormat: String;
  majortype:  TGUID;
  subtype:    TGUID;
  formattype: TGUID;
end;
```

For Linux and macOS**type**

```
TRVCamVideoMode = record
  Width, Height: Integer;
  PixelFormat: Cardinal;
end;
```

You can use the functions from MRVWebCamFuncs⁽²²⁸⁾ unit to display description of the video mode to the user.

See also

- TRVCamera⁽⁴²⁾'s webcam video mode methods⁽⁶³⁾

7.8 TRVChangedAreaProcessingMode

Specifies how video frames are preprocessed before detecting changed areas

Unit [VCL and LCL] MRVFuncImg;

Unit [FMX] fmxMRVFuncImg;

type

```
TRVChangedAreaProcessingMode = (rvcapmOriginalSize, rvcapmAuto);
```

Value	Meaning
<i>rvcapmOriginalSize</i>	Changed areas are searched in the original video frame
<i>rvcapmAuto</i>	A video frame is shrunk before processing. The largest side of the frame is reduced by half until it becomes ≥ 320 pixels. The smallest side is reduced accordingly.

Searching for changed areas may be a time consuming process in large frames. Working with reduced images allows to make processing fast.

This type is used for the properties:

- TRVCamSender⁽¹³⁹⁾.ChangedAreaProcessingMode⁽¹⁴⁴⁾
- TRVMotionDetector⁽²⁰⁹⁾.ChangedAreaProcessingMode⁽²⁰⁹⁾

7.9 TRVColorControlProperty

Identifies a property for image settings.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVColorControlProperty = (rvccpBrightness, rvccpContrast,  
rvccpSaturation, rvccpHue, rvccpSharpness);
```

Value	Meaning
<i>rvccpBrightness</i>	Brightness
<i>rvccpContrast</i>	Contrast
<i>rvccpSaturation</i>	Saturation
<i>rvccpHue</i>	Hue
<i>rvccpSharpness</i>	Sharpness

See also

- TRVCamera.GetColorControlPropertyRange⁽⁶⁵⁾ method

7.10 TRVCompressionType

Specifies compression of media data.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVCompressionType = (rvtcNone, rvtcParts, rvtcFully);
```

Value	Meaning
<i>rvtcNone</i>	No compression. If some part of data is damaged on sending, the error does not affect other parts.
<i>rvtcParts</i>	Data are separated into packets, each packet is sent compressed. Data are compressed while sending, without significant delays. A medium compression quality. If some compressed packet is damaged on sending, the error does not affect other packets.
<i>rvtcFully</i>	Data is compressed as a whole. It may require some time to start sending. A high compression quality. If some part of data is damaged, the whole data are lost.

See also:

- TRVCompressionOptions ⁽¹⁹⁹⁾

7.11 TRVDesktopVideoMode

Contains information about desktop video mode.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVDesktopVideoMode = packed record
    Width      : Integer;
    Height     : Integer;
    ColorDepth : Byte;
end;
```

Fields:

- Width, Height – screen size
- ColorDepth – bytes per pixel

See also

- TRVCamera ⁽⁴²⁾'s desktop video mode methods ⁽⁶³⁾

7.12 TRVEncodingType

Specifies the video encoding

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVEncodingType = (rvetJPEG, rvetJPEGChange,
  rvetHWL, rvetHWLChange,
  rvetBMP, rvetBMPChange,
  rvetPNG, rvetPNGChange,
  rvetMixFormat, rvetMixFormatChange;
```

Value	Meaning
<i>rvetJPEG</i>	A video frame is sent as a Jpeg image
<i>rvetJPEGChange</i>	A video frame is separated into fragments, changed fragments are sent as Jpeg images
<i>rvetHWL (BETA!)</i>	A video frame is sent compressed using the Haar Wavelet Transform
<i>rvetHWLChange (BETA!)</i>	A video frame is separated into fragments, changed fragments are sent compressed using the Haar Wavelet Transform
<i>rvetBMP</i>	A video frame is sent as a bitmap image
<i>rvetBMPChange</i>	A video frame is separated into fragments, changed fragments are sent as bitmap images
<i>rvetPNG</i>	A video frame is sent as a Png image. This option requires Delphi 2009 or newer.
<i>rvetPNGChange</i>	A video frame is separated into fragments, changed fragments are sent as Png images. This option requires Delphi 2009 or newer.
<i>rvetMixFormat</i>	A video frame is sent as an image. The component chooses an image format providing a smallest size for this frame. This mode requires excessive computations.
<i>rvetMixFormatChange</i>	A video frame is separated into fragments, changed fragments are sent as images. The component chooses an image format providing a smallest size for this frame. This mode requires excessive computations.

The Haar Wavelet Transform implementation is based on the code by <http://ainc.de>. The current version of HWL implementation is not stable and not recommended to use.

PNG encoding may be useful for sending lossless images from TRVCamera having DeviceType⁽⁴⁹⁾ =rvdtDesktop.

This type is used by the following properties:

- TRVCamSender⁽¹³⁹⁾.Encoding⁽¹⁴⁵⁾

7.13 TRVFFMpegFilter

This type defines the method for scaling video frames.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVFFMpegFilter = (rvffNone, rvffFastBilinear, rvffBilinear, rvffBicubic,
    rvffX, rvffPoint, rvffArea, rvffBicublin, rvffGauss, rvffSinc, rvffLaczos,
    rvffSpline);
```

Value	Method
<i>rvffNone</i>	(auto-selection)
<i>rvffFastBilinear</i>	fast bilinear
<i>rvffBilinear</i>	bilinear
<i>rvffBicubic</i>	bicubic
<i>rvffX</i>	experimental
<i>rvffPoint</i>	nearest neighbor / point
<i>rvffArea</i>	area averaging (downscale only, for upscale it is bilinear)
<i>rvffBicublin</i>	luma bicubic / chroma bilinear
<i>rvffGauss</i>	Gaussian
<i>rvffSinc</i>	sinc
<i>rvffLaczos</i>	Lanczos
<i>rvffSpline</i>	bicubic spline

See also:

- TRVFFMpegProperty⁽²⁰¹⁾.VideoFilter

7.14 TRVJpegIntegrity

Specifies how received JPEG images are checked for correctness.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVJpegIntegrity = (rvjiNone, rvjiFast, rvjiFull);
```

Value	Meaning
<i>rvjiNone</i>	No checking
<i>rvjiFast</i>	Structure checking
<i>rvjiFull</i>	Structure and content checking

This is a type of the following properties:

- TRVCamera⁽⁴²⁾.JpegIntegrity⁽⁵²⁾
- TRVCamReceiver⁽¹²⁰⁾.JpegIntegrity⁽¹²⁴⁾

7.15 TRVMAnsiString

ANSI string type

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVMAnsiString = type AnsiString;
```

7.16 TRVMColor

A type of data that can be sent between TRVCamSender⁽¹³⁹⁾, TRVCamReceiver⁽¹²⁰⁾ and TRVMediaServer⁽¹⁶⁵⁾.

Unit [VCL and LCL] MRVCore;

Unit [FMX] fmxMRVCore;

VCL and LCL:

type

```
TRVMColor = TColor;
```

FMX:

type

```
TRVMColor = TAlphaColor;
```

7.17 TRVMediaType

A type of data that can be sent between TRVCamSender⁽¹³⁹⁾, TRVCamReceiver⁽¹²⁰⁾ and TRVMediaServer⁽¹⁶⁵⁾.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVMediaType = (rvmtVideo, rvmtAudio, rvmtUserData, rvmtFileData,
```



```
rvmtCmdData);  
TRVMediaTypes = set of TRVMediaType;
```

Value	Meaning
<i>rvmtVideo</i>	Video
<i>rvmtAudio</i>	Audio
<i>rvmtUserData</i>	Arbitrary binary data
<i>rvmtFileData</i>	Files
<i>rvmtCmdData</i>	Commands

7.18 TRVMIconStyle



Defines a type of animation displaying while the component searches for an IP camera.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVMIconStyle = (rvisClassic, rvisModern);
```

Value	Animation
<i>rvisClassic</i>	
<i>rvisModern</i>	

rvisModern is supported in Lazarus or in Delphi 2009 and newer.

This property changes not only an animation, it also changes colors of a search panel and its text (if they are not specified explicitly).

7.19 TRVMLanguage

A language of user interface.

Unit [VCL and LCL] MRVLocalize;

Unit [FMX] fmxMRVLocalize;

type

```
TRVMLanguage = (rvmlEnglish, rvmlFrench, rvmlGerman, rvmlRussian,  
rvmlSpanish, rvmlPortugueseBr, rvmlChineseSimplified,  
rvmlChineseTraditional);
```

Value	Meaning
<i>rvmlEnglish</i>	English (US)

<i>rvmlFrench</i>	French
<i>rvmlGerman</i>	German
<i>rvmlRussian</i>	Russian
<i>rvmlSpanish</i>	Spanish
<i>rvmlPortugueseBr</i>	Portuguese (Brazil)
<i>rvmlChineseSimplified</i>	Chinese (Simplified)
<i>rvmlChineseTraditional</i>	Chinese (Traditional)

This type is used for the properties:

- TRVCamView.Language⁹⁹
- TRVCamMultiView.Language⁸⁰
- TRVTrafficMeter.Language¹⁷⁶
- TRVWebCamDialog.Language¹¹⁰

7.20 TRVMRect, TRVMPoint, TRVUnitSize

The types representing coordinates.

Unit [VCL and LCL] MRVCore;

Unit [FMX] fmxMRVCore;

type

```
TRVMRect      = TRect;
TRVMPoint     = TPoint;
TRVUnitSize   = Integer;
```

TRVMRect represents the dimensions of a rectangle.

TRVMPoint represents a point (X and Y)

TRVUnitSize represents a linear coordinate.

7.21 TRVMRenderMode

Specifies a video rendering method.

Unit [VCL and LCL] MRVType;

type

```
TRVMRenderMode = (rvmrmSoftware, rvmrmOpenGL,
  rvmrmSkia, rvmrmAuto);
```

Value	Meaning
<i>rvmrmSoftware</i>	Standard drawing (GDI in Windows)
<i>rvmrmOpenGL</i>	OpenGL drawing.

	<p>In <code>TRVCamView</code>⁽⁹³⁾: if a video frame is not visible, the viewer is drawn by the standard drawing. If a video frame is visible, the whole viewer is drawn by OpenGL.</p> <p>in <code>TRVCamMultiView</code>⁽⁷⁴⁾: the whole viewer is drawn by OpenGL.</p> <p>OpenGL uses GPU for scaling images, so less CPU resources are used.</p> <p>Requirements:</p> <ul style="list-style-type: none"> • Delphi • Lazarus • C++Builder XE or newer
<i>rvmrmSkia</i>	<p>Skia4Delphi drawing.</p> <p>MRVCamViewSkia unit must be included in your project, otherwise this mode will not be initialized (and the component will fall back to <i>rvmrmSoftware</i>).</p> <p><code>TRVCamView</code>⁽⁹³⁾ and <code>TRVCamMultiView</code>⁽⁷⁴⁾ use Skia4Delphi only for drawing video frames. All other parts of video viewers are drawn by the standard drawing.</p> <p>GPU is not used, but frame drawing sometimes may be more efficient than in the standard drawing.</p> <p>Requirements:</p> <ul style="list-style-type: none"> • Delphi or C++Builder XE7 or newer
<i>rvmrmAuto</i>	<p>Auto selection:</p> <ul style="list-style-type: none"> • if OpenGL is available, uses it; otherwise • if Skia4Delphi is available, uses it; otherwise • uses the standard rendering method

This is a type of the following properties:

- `TRVCamView`⁽⁹³⁾.RenderMode, CurRenderMode⁽⁹⁶⁾
- `TRVCamMultiView`⁽⁷⁴⁾.RenderMode, CurRenderMode⁽⁷⁸⁾

7.22 TRVMWindowHandle

The type representing a window handle.

Unit [VCL and LCL] MRVCore;

Unit [FMX] fmxMRVCore;

For macOS:

type

```
TRVMWindowHandle = CGWindowID;
```

For other platforms:

type

```
TRVMWindowHandle = THandle;
```

type

```
TRVMWindowHandleArray = array of TRVMWindowHandle;
```

7.23 TRVParamType

Type of a command parameter⁽¹⁹⁸⁾.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVParamType = (rvptNone, rvptString, rvptInteger, rvptFloat,  
rvptDateTime, rvptBin);
```

Value	Meaning
<i>rvptNone</i>	Unknown command type
<i>rvptString</i>	String
<i>rvptInteger</i>	Integer value
<i>rvptFloat</i>	Floating point value
<i>rvptDateTime</i>	Date
<i>rvptBin</i>	Binary data

7.24 TRVProtocol

A protocol used to send data between TRVCamSender⁽¹³⁹⁾ and TRVCamReceiver⁽¹²⁰⁾, or between TRVCamSender⁽¹³⁹⁾ and TRVMediaServer⁽¹⁶⁵⁾.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVProtocol = (rvpTCP, rvpUDP, rvpHTTP);
```

UDP uses a simple transmission model with a minimum of protocol mechanism. UDP is fast but unreliable: there is no guarantee of delivery, ordering or duplicate protection. Recommended for sending video and audio, especially video. Highly not recommended for sending other types of data (binary data, files, commands).

TCP and **HTTP** provide a reliable, ordered delivery. **HTTP** is necessary to connect using a proxy server, or when a server has a symbolic name (URL). Otherwise, you can use **TCP**.

See also:

- TRVCamSender.Protocol⁽¹⁴⁹⁾
- TRVCamReceiver.Protocol⁽¹²⁴⁾

7.25 TRVProtocolEx, TRVProtocolsEx

Protocols.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVProtocolEx = (rvpeTCP, rvpeUDP, rvpeHTTP, rvpeUDPMulticast);  
TRVProtocolsEx = set of TRVProtocolEx;
```

See also:

- TRVFFMpegProperty⁽²⁰¹⁾.RTSPTransport

7.26 TRVRTSPFlag, TRVRTSPFlags

FFmpeg RTSP options.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVRTSPFlag = (rvfPreferTCP, rvfListen, rvfFilterSrc);  
TRVRTSPFlags = set of TRVRTSPFlag;
```

Value	Meaning
<i>rvfPreferTCP</i>	Instructs to try TCP for RTP transport first, if TCP is available as RTSP RTP transport (see TRVFFMpegProperty ⁽²⁰¹⁾ .RTSPTransport)
<i>rvfListen</i>	Instructs to act as a server, listening for an incoming connection (not supported in RVMedia)
<i>rvfFilterSrc</i>	Instructs to accept packets only from negotiated peer address and port (not supported in RVMedia)

See also:

- TRVFFMpegProperty⁽²⁰¹⁾.RTSPFlags

7.27 TRVSampleFormat

Specifies a format of sound samples.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVSampleFormat = (rvsf8, rvsf16, rvsf32, rvsfFloat, rvsfDouble);
```

Value	Meaning
-------	---------

<i>rvsf8</i>	integer value, 8 bits
<i>rvsf16</i>	integer value, 16 bits
<i>rvsf32</i>	integer value, 32 bits
<i>rvsfFloat</i>	floating point value, 32 bits
<i>rvsfDouble</i>	floating point value, 64 bits

See also

- `TRVBitsPerSample`⁽²³⁷⁾ (a simplified version of this type, used for input audio devices)
- `TRVAudioPlayer`⁽¹¹⁰⁾.`EncodeSampleFormat`⁽¹¹²⁾ property
- `GetSampleFormatName`⁽²²⁵⁾

7.28 TRVSamplesPerSec

Specifies a sample rate (a number of sound samples read in a second).

Unit [VCL and LCL] `MRVType`;

Unit [FMX] `fmvMRVType`;

type

```
TRVSamplesPerSec = (rvsps8000, rvsps11025, rvsps22050, rvsps44100,
    rvsps48000, rvsps96000, rvsps192000);
```

Value	Meaning
<i>rvsps8000</i>	8000 Hz
<i>rvsps11025</i>	11025 Hz
<i>rvsps22050</i>	22050 Hz
<i>rvsps44100</i>	44100 Hz
<i>rvsps96000</i>	96000 Hz
<i>rvsps192000</i>	192000 Hz

See also

- `TCustomRVMicrophone`⁽¹⁷⁷⁾.`SamplesPerSec`⁽¹⁷⁹⁾ property

7.29 TRVSessionKey

A type of a session identifier.

Unit [VCL and LCL] `MRVType`;

Unit [FMX] `fmvMRVType`;

type

```
TRVSessionKey = type Cardinal;
```

This type is used for properties:

- TRVCamReceiver.SessionKey¹²⁶
- TRVCamSender.SessionKey¹⁵¹
- TRVMediaServer.SessionKey¹⁷⁰

7.30 TRVSocket

A type of a socket identifier.

Unit [VCL and LCL] MRVSocket;

Unit [FMX] fmxMRVSocket;

type

```
TRVSocket = type TSocket;
```

7.31 TRVTCPConnectionType

Defines how TRVCamSender¹³⁹ and TRVCamReceiver¹²⁰ are connected, if TCP/HTTP protocols are used.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVTCPConnectionType = (rvtcpSenderToReceiver, rvtcpReceiverToSender);
```

Value	Meaning
<i>rvtcpSenderToReceiver</i>	A sender initiates a connection to a receiver
<i>rvtcpReceiverToSender</i>	A receiver initiates a connection to a sender

See also:

- TRVCamSender.TCPConnectionType¹⁵³
- TRVCamReceiver.TCPConnectionType¹²⁸

7.32 TRVVideoCodec

Specifies a codec used to encode video TRVCamRecorder⁸⁵ component.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

type

```
TRVVideoCodec = (rvvcDefault, rvvcMPEG1, rvvcMPEG2, rvvcMPEG4,  
    rvvcH263P, rvvcH264,  
    rvvcMJPEG, rvvcWMV1, rvvcWMV2,  
    // allowed only in Windows Media Technology application or SDK  
    {$IFDEF MRV_CODEC_MSMPEG4v3}  
    rvvcMSMPEG4v3  
    {$ENDIF}  
    rvvcFLV1, rvvcRV1, rvvcRV2, rvvcHEVC);
```

Value	Format name (see <code>GetVideoCodecName</code> ⁽²²⁵⁾)	Recommended file extensions (see <code>GetVideoFileExts</code> ⁽²²⁴⁾)
<code>rvvcMPEG1</code>	MPEG-1	mpeg
<code>rvvcMPEG2</code>	MPEG-2	mpeg
<code>rvvcMPEG4</code>	MPEG-4	mp4 3gp
<code>rvvcH263P</code>	H.263+	avi
<code>rvvcH264</code>	H.264	mp4 avi mov
<code>rvvcMJPEG</code>	MJPEG	mjpeg
<code>rvvcWMV1</code>	Windows Media Video 7	avi
<code>rvvcWMV2</code>	Windows Media Video 8	avi
<code>rvvcMSMPEG4v3</code>	MPEG-4 part 2 Microsoft variant v.3	asf avi
<code>rvvcFLV1</code>	FLV (Sorenson Spark)	flv
<code>rvvcRV1</code>	RV10 (RealVideo 1)	rm
<code>rvvcRV2</code>	RV20 (RealVideo 2)	rm
<code>rvvcHEVC*</code>	H.265 (High Efficiency Video Coding)	mp4 mkv

* FFmpeg 3 or newer is required

Warning: Some video formats may be patent-protected in some countries, and supporting these formats will require from you obtaining licenses from the patent owners.

This type is used for `TRVCamRecorder.VideoCodec` ⁽⁸⁸⁾ property.

See also:

- `TRVAudioCodec` ⁽²³⁶⁾

7.33 TRVVideoResolution

Video resolution.

Unit [VCL and LCL] `MRVType`;

Unit [FMX] `fmxMRVType`;

type

```
TRVVideoResolution = (rvDefault, rv160_120, rv320_240, rv640_480,
    rv1024_768, rv1280_720, rv1900_1280);
```

Value	Meaning
-------	---------

<i>rvDefault</i>	The default video resolution is used, no scaling occurs
<i>rv160_120</i>	160 x 120
<i>rv320_240</i>	320 x 240
<i>rv640_480</i>	640 x 480
<i>rv1024_768</i>	1024 x 768
<i>rv1280_720</i>	1280 x 720
<i>rv1900_1280</i>	1900 x 1280

This is a type of the following properties:

- TRVCamera⁴².VideoResolution⁶¹
- TRVCamSender¹³⁹.VideoResolution¹⁵⁴

Index

- A -

- Abort 191
 - TRVVideoSource 191
- Aborting 190
 - TRVVideoSource 190
- Access 58
 - TRVCamUser 58
- Active 87, 122, 143, 186, 192
 - TCustomRVAudioOutput 192
 - TRVAudioSource 186
 - TRVCamReceiver 122
 - TRVCamRecorder 87
 - TRVCamSender 143
- AddAllowedSender 156
 - TRVCamSender 156
- AddAllowedSenders 156
 - TRVCamSender 156
- AddDefaultReceiver 157
 - TRVCamSender 157
- AddUser 62
 - TRVCamera 62
- Agent 44
 - TRVCamera 44
- AllowFullScreen 76, 95
 - TRVCamMultiView 76
 - TRVCamView 95
- AllowMediaAccess 158
 - TRVCamSender 158
- ArrowColor 41
 - TRVCamControl 41
- ArrowColorClicked 41
 - TRVCamControl 41
- ArrowColorHot 41
 - TRVCamControl 41
- ArrowLineColor 41
 - TRVCamControl 41
- AudioBitrate 87
 - TRVCamRecorder 87
- AudioChannels 87
 - TRVCamRecorder 87
- AudioCodec 88
 - TRVCamRecorder 88
- AudioCodecName 88
 - TRVCamRecorder 88
- AudioInputDeviceCount 179
 - TCustomRVMicrophone 179
- AudioInputDeviceIndex 179
 - TCustomRVMicrophone 179
- AudioInputDeviceList 179
 - TCustomRVMicrophone 179
- AudioLatency 122
 - TRVCamReceiver 122
- AudioOutput 185, 188
 - TCustomRVReceiver 185
 - TRVAudioSourceWithOutput 188
- AudioOutputDeviceCount 194
 - TCustomRVAudioPlayer 194
- AudioOutputDeviceIndex 194
 - TCustomRVAudioPlayer 194
- AudioOutputDeviceList 194
 - TCustomRVAudioPlayer 194
- AudioSampleFormat 87
 - TRVCamRecorder 87
- AudioSampleRate 87
 - TRVCamRecorder 87
- AudioSource 76, 89, 143, 189, 208
 - TRVAudioViewer 189
 - TRVCamMultiView 76
 - TRVCamRecorder 89
 - TRVCamSender 143
 - TRVMediaSourceItem 208
- AutoSize 95
 - TRVCamView 95

- B -

- BeginCmd 160
 - TRVCamSender 160
- Bitrate 45
 - TRVCamera 45
- BitsPerSample 179
 - TCustomRVMicrophone 179
- BorderColor 41
 - TRVCamControl 41
- Brightness 45
 - TRVCamera 45
- BufferDuration 122, 179, 194
 - TCustomRVAudioPlayer 194
 - TCustomRVMicrophone 179
 - TRVCamReceiver 122
- BufferOptions 167
 - TRVMediaServer 167
- BufferSize 122, 144
 - TRVCamReceiver 122

BufferSize 122, 144
TRVCamSender 144

- C -

Camera 103, 109, 116, 176
TRVCamSound 116
TRVCamView 103
TRVTrafficMeter 176
TRVWebCamDialog 109
CameraControl 46, 76
TRVCamera 46
TRVCamMultiView 76
CameraHost 46
TRVCamera 46
CameraPort 46
TRVCamera 46
CameraSearchTimeOut 46
TRVCamera 46
CamMoveMode 77, 95
TRVCamMultiView 77
TRVCamView 95
CancelMediaAccess 158
TRVCamSender 158
CaptionColor 77, 101
TRVCamMultiView 77
TRVCamView 101
CaptionFont 77, 101
TRVCamMultiView 77
TRVCamView 101
CaptionHeight 77, 101
TRVCamMultiView 77
TRVCamView 101
CaptionParts 101
TRVCamView 101
CaptionTextSettings 77
TRVCamMultiView 77
ChangedAreaProcessingMode 144, 209
TRVCamSender 144
TRVMotionDetector 209
ClearAllowedSenders 156
TRVCamSender 156
CmdOptions 168
TRVMediaServer 168
Color 41, 78, 96, 122
TRVCamControl 41
TRVCamMultiView 78
TRVCamReceiver 122
TRVCamView 96
CommandMode 47

TRVCamera 47
Comparison (RVMedia) 15
Components 38
Ancestor classes 176
Audio 38, 110
Network 38, 120
Video 38, 39
CompressionOptions 144
TRVCamSender 144
CompressionQuality 145
TRVCamSender 145
ConnectionProperties 123, 145
TRVCamReceiver 123
TRVCamSender 145
Contrast 45
TRVCamera 45
CurRenderMode 78, 96
TRVCamMultiView 78
TRVCamView 96
CycleVideoImages 47
TRVCamera 47

- D -

DescribeVideoMode 228
DescribeVideoModePixelFormat 228
DesktopForm 47
TRVCamera 47
DesktopMode 47
TRVCamera 47
DesktopRect 47
TRVCamera 47
DesktopWindowHandle 47
TRVCamera 47
DesktopZoomPercent 47
TRVCamera 47
DetectChanges 211
TRVMotionDetector 211
DeviceType 49
TRVCamera 49
DiskBrightness 41
TRVCamControl 41
DiskColor 41
TRVCamControl 41

- E -

EncodeAudioCodec 112
TRVAudioPlayer 112
EncodeAudioCodecName 26, 112

EncodeAudioCodecName 26, 112
 TRVAudioPlayer 112
 EncodeBitrate 112
 TRVAudioPlayer 112
 EncodeChannels 112
 TRVAudioPlayer 112
 EncodeSampleRate 112
 TRVAudioPlayer 112
 Encoding 145
 TRVCamSender 145
 EndCmd 160
 TRVCamSender 160
 Example
 playing a camera video (no wait mode) 229
 playing a camera video (wait mode) 228
 playing a video stream 230
 Examples 228
 Execute 110
 TRVWebCamDialog 110
 ExtraMediaSources 146
 TRVCamSender 146

- F -

Features (RVMedia) 13
 FFMpegProperty 50
 TRVCamera 50
 FileName 50
 TRVCamera 50
 FillVideoDeviceList 63
 TRVCamera 63
 FilterBlur 146
 TRVCamSender 146
 FilterSystemCmd 123
 TRVCamReceiver 123
 FilterUserCmd 168
 TRVMediaServer 168
 FlipHorizontally 51
 TRVCamera 51
 FlipVertically 51
 TRVCamera 51
 FocusDistance 51
 TRVCamera 51
 FocusLineColor 78, 96
 TRVCamMultiView 78
 TRVCamView 96
 FocusType 51
 TRVCamera 51
 Font 79, 97
 TRVCamMultiView 79

 TRVCamView 97
 FrameDifferenceInterval 146
 TRVCamSender 146
 FramePerSec 190
 TRVVideoSource 190
 Framerate 45
 TRVCamera 45
 FrameScaleQuality 97
 TRVCamView 97
 FullFrameInterval 147
 TRVCamSender 147
 FullScreen 79, 97
 TRVCamMultiView 79
 TRVCamView 97
 FullScreenMultiView 79
 TRVCamMultiView 79
 FullScreenView 98
 TRVCamView 98

- G -

GetAccessibleCamMethods 64
 TRVCamera 64
 GetAccessibleCamProperties 64
 TRVCamera 64
 GetAllGroups 158
 TRVCamSender 158
 GetAllOnlineUsers 158
 TRVCamSender 158
 GetAllUsers 158
 TRVCamSender 158
 GetAudioCodecFileExt 224
 GetAudioCodecName 225
 GetAvailableCamMethods 64
 TRVCamera 64
 GetAvailableCamProperties 64
 TRVCamera 64
 GetCamCurrentVideoMode 63
 TRVCamera 63
 GetCamVideoMode 63
 TRVCamera 63
 GetCamVideoModeCount 63
 TRVCamera 63
 GetCamVideoModelIndex 63
 TRVCamera 63
 GetColorControlPropertyRange 65
 TRVCamera 65
 GetDesktopCurrentVideoMode 63
 TRVCamera 63
 GetDesktopVideoMode 63

GetDesktopVideoMode 63
 TRVCamera 63
 GetDesktopVideoModeCount 63
 TRVCamera 63
 GetDesktopVideoModelIndex 63
 TRVCamera 63
 GetGroupInfo 158
 TRVCamSender 158
 GetListOfAudioDecoders 220
 GetListOfAudioEncoders 220
 GetListOfAvailableFFmpegAudioCodecs 221
 GetListOfAvailableFFmpegFileFormats 222
 GetListOfAvailableFFmpegVideoCodecs 222
 GetListOfAvailableSampleFormats 223
 GetListOfAvailableSampleRates 223
 GetListOfVideoDecoders 220
 GetListOfVideoEncoders 220
 GetListOfVideoInputFormats 223
 GetMaxChannelCount 129
 TRVCamReceiver 129
 GetOpenChannelCount 129
 TRVCamReceiver 129
 GetOptimalVideoResolution 191
 TRVVideoSource 191
 GetRect 211
 TRVMotionDetector 211
 GetSampleFormatName 225
 GetSnapshot 65
 TRVCamera 65
 GetUsersFromGroup 158
 TRVCamSender 158
 GetVideoCodecFileExts 224
 GetVideoCodecName 225
 GetVisibleWindowsHandles 218
 GetWebCamPixelFormat 239
 GetWindowTitleByHandle 218
 glRVInetMaxConnect 216
 GoodbyeToAllowedSender 156
 TRVCamSender 156
 GoodbyeToDefaultReceivers 157
 TRVCamSender 157
 GStreamerProperty 52
 TRVCamera 52
 GUID 116, 189
 TRVAudioViewer 189
 TRVCamSound 116
 GUIDFrom 98, 147
 TRVCamSender 147
 TRVCamView 98
 GUIDGroup 147

TRVCamSender 147
 GUIDMy 123, 169
 TRVCamReceiver 123
 TRVMediaServer 169
 GUIDTo 147
 TRVCamSender 147

- H -

HelloToAllowedSenders 156
 TRVCamSender 156
 HelloToDefaultReceivers 157
 TRVCamSender 157
 HoverLineColor 80, 98
 TRVCamMultiView 80
 TRVCamView 98
 How it works - cameras (RVMedia) 16
 How it works - video chat with a server (RVMedia) 20
 How it works - video chat without a server (RVMedia) 18
 HTTPActive 169
 TRVMediaServer 169
 HTTPPort 169
 TRVMediaServer 169
 Hue 45
 TRVCamera 45

- I -

IconStyle 80, 99
 TRVCamMultiView 80
 TRVCamView 99
 IgnoreCorruptedFrames 123
 TRVCamReceiver 123
 IndexFrom 98
 TRVCamView 98
 IPCameraTypes 52
 TRVCamera 52
 IsRectValid 211
 TRVMotionDetector 211
 IsRNNoiseLoaded 227
 IsSupportedFFmpeg 65, 219
 TRVCamera 65
 IsSupportedGStreamer 66, 226
 TRVCamera 66

- J -

JoinGroup 158
 TRVCamSender 158

JpegIntegrity 52, 124
 TRVCamera 52
 TRVCamReceiver 124

- K -

KeepClientInfoMode 169
 TRVMediaServer 169

- L -

Language 52, 80, 99, 110, 176
 TRVCamera 52
 TRVCamMultiView 80
 TRVCamView 99
 TRVTrafficMeter 176
 TRVWebCamDialog 110

Latency 53
 TRVCamera 53

LeaveGroup 158
 TRVCamSender 158

LoadFFMpegLibraries 219

LoadGStreamerLibraries 225

LoadRNNoise 227

LockCommands 66
 TRVCamera 66

LoginPrompt 53
 TRVCamera 53

- M -

MAX_RETRY_COUNT 125
 MaxCameraSearchThreadCount 53
 TRVCamera 53

MaxGroupCount 170
 TRVMediaServer 170

MinChangeAreaSize 148, 210
 TRVCamSender 148
 TRVMotionDetector 210

Modes of connections (RVMedia) 21

ModifyUser 62
 TRVCamera 62

MoveCenter 66
 TRVCamera 66

MoveDown 66
 TRVCamera 66

MoveDownStop 66
 TRVCamera 66

MoveHPatrol 66
 TRVCamera 66

MoveHPatrolStop 66
 TRVCamera 66

MoveLeft 66
 TRVCamera 66

MoveLeftDown 66
 TRVCamera 66

MoveLeftStop 66
 TRVCamera 66

MoveLeftUp 66
 TRVCamera 66

MoveRight 66
 TRVCamera 66

MoveRightDown 66
 TRVCamera 66

MoveRightStop 66
 TRVCamera 66

MoveRightUp 66
 TRVCamera 66

MoveStop 66
 TRVCamera 66

MoveUp 66
 TRVCamera 66

MoveUpStop 66
 TRVCamera 66

MoveVPatrol 66
 TRVCamera 66

MoveVPatrolStop 66
 TRVCamera 66

MRVCore 216, 217

MRVDesktop 216, 218

MRVFFmpeg 216, 219

MRVFFMpegLists 216, 220

MRVFormatInfo 216, 224

MRVGStreamer 216, 225

MRVRNNoise 216, 227

MRVWebCamFuncs 216, 228

Mute 124, 180, 194
 TCustomRVAudioPlayer 194
 TCustomRVMicrophone 180
 TRVCamReceiver 124

- N -

Name 208
 TRVMediaSourceItem 208

NeedSendFullFrame 160
 TRVCamSender 160

NoiseReduction 180, 194
 TCustomRVAudioPlayer 194
 TCustomRVMicrophone 180

NoiseReduction and NoiseReductionLevel 180
 NoiseReductionLevel 180, 194
 TCustomRVAudioPlayer 194
 TCustomRVMicrophone 180

- O -

OnBeginMove 107
 TRVCamView 107
 OnCloseChannel 135
 TRVCamReceiver 135
 OnCloseWavFile 183
 TCustomRVMicrophone 183
 OnConnected 130, 163
 TRVCamReceiver 130
 TRVCamSender 163
 OnConnectError 130, 163
 TRVCamReceiver 130
 TRVCamSender 163
 OnConnecting 130, 163
 TRVCamReceiver 130
 TRVCamSender 163
 OnConnectionCountChanged 173
 TRVMediaServer 173
 OnDataRead 172, 185
 TCustomRVReceiver 185
 TRVMediaServer 172
 OnDecodeAudio 130
 TRVCamReceiver 130
 OnDecodeVideo 131
 TRVCamReceiver 131
 OnDisconnect 130, 163
 TRVCamReceiver 130
 TRVCamSender 163
 OnEncodeAudio 164
 TRVCamSender 164
 OnEncodeVideo 164
 TRVCamSender 164
 OnEndMove 107
 TRVCamView 107
 OnEndVideoFile 70
 TRVCamera 70
 OnEndVideoStream 70
 TRVCamera 70
 OnError 70, 92, 114, 174
 TRVAudioPlayer 114
 TRVCamera 70
 TRVCamRecorder 92
 TRVMediaServer 174
 OnFirstFrame 92
 TRVCamRecorder 92
 OnFullScreen 85, 107
 TRVCamMultiView 85
 TRVCamView 107
 OnFullScreen.OnPaint 107
 OnGetAllGroups 133
 TRVCamReceiver 133
 OnGetAllOnlineUsers 133
 TRVCamReceiver 133
 OnGetAllUsers 133
 TRVCamReceiver 133
 OnGetAudio 187, 192
 TCustomRVAudioOutput 192
 TRVAudioSource 187
 OnGetAudioStreamIndex 116
 TRVCamSound 116
 OnGetGroupInfo 131
 TRVCamReceiver 131
 OnGetGroupUsers 134
 TRVCamReceiver 134
 OnGetImage 71, 93, 134
 TRVCamera 71
 TRVCamReceiver 134
 TRVCamRecorder 93
 OnGetVideoStreamIndex 71
 TRVCamera 71
 OnLogin 71
 TRVCamera 71
 OnLoginFailed 72
 TRVCamera 72
 OnMediaAccessCancelRequest 139
 TRVCamReceiver 139
 OnMediaAccessRequest 139
 TRVCamReceiver 139
 OnMouseEnter 107
 TRVCamView 107
 OnMouseLeave 107
 TRVCamView 107
 OnMoved 72
 TRVCamera 72
 OnNewImage 72
 TRVCamera 72
 OnOpenChannel 135
 TRVCamReceiver 135
 OnOpenWavFile 183
 TCustomRVMicrophone 183
 OnPacketProcessing 173
 TRVMediaServer 173
 OnPaint 107, 119
 TRVCamView 107

OnPaint	107, 119	OnUserEnter	138
TRVMicrophoneView	119	TRVCamReceiver	138
OnPrepared	72	OnUserExit	138
TRVCamera	72	TRVCamReceiver	138
OnProgress	73	OnUserJoinsGroup	138
TRVCamera	73	TRVCamReceiver	138
OnReadWavFile	183	OnUserLeavesGroup	138
TCustomRVMicrophone	183	TRVCamReceiver	138
OnReceiveCmdData	135	OnVideoStart	74
TRVCamReceiver	135	TRVCamera	74
OnReceivedFile	136	OnViewerPaint	85
TRVCamReceiver	136	TRVCamMultiView	85
OnReceiveFileData	136	Orientation	118
TRVCamReceiver	136	TRVMicrophoneView	118
OnReceiveUserData	137	OutputFileName	89, 113
TRVCamReceiver	137	TRVAudioPlayer	113
OnReceivingFile	136	TRVCamRecorder	89
TRVCamReceiver	136		
OnRequestJoinGroup	132		
TRVCamReceiver	132		
OnSearchComplete	73		
TRVCamera	73		
OnSelectViewer	85		
TRVCamMultiView	85		
OnSendCmd	165		
TRVCamSender	165		
OnSentCmd	165		
TRVCamSender	165		
OnServerCmd	173		
TRVMediaServer	173		
OnSessionConnected	137		
TRVCamReceiver	137		
OnSessionDisconnected	137		
TRVCamReceiver	137		
OnSetProperty	73		
TRVCamera	73		
OnStart	174		
TRVMediaServer	174		
OnStartVideoFile	73		
TRVCamera	73		
OnStartVideoStream	73		
TRVCamera	73		
OnStop	174		
TRVMediaServer	174		
OnStopRecording	114		
TRVAudioPlayer	114		
OnUserConnect	174		
TRVMediaServer	174		
OnUserDisconnect	174		
TRVMediaServer	174		

- P -

Parameters	54
TRVCamera	54
ParentColor	41
TRVCamControl	41
ParentFont	79, 97
TRVCamMultiView	79
TRVCamView	97
Password	58
TRVCamUser	58
Paused	89
TRVCamRecorder	89
Pitch	181, 195
TCustomRVAudioPlayer	195
TCustomRVMicrophone	181
PixelColorThreshold	148, 210
TRVCamSender	148
TRVMotionDetector	210
PixelColorThresholdB	148, 210
TRVCamSender	148
TRVMotionDetector	210
PixelColorThresholdG	148, 210
TRVCamSender	148
TRVMotionDetector	210
PixelColorThresholdR	148, 210
TRVCamSender	148
TRVMotionDetector	210
PlayVideoFile	67
TRVCamera	67
PlayVideoStream	67
TRVCamera	67

Port 124
 TRVCamReceiver 124

Protocol 124, 149
 TRVCamReceiver 124
 TRVCamSender 149

ProxyProperty 55, 125, 149
 TRVCamera 55
 TRVCamReceiver 125
 TRVCamSender 149

- Q -

Quality 55
 TRVCamera 55

- R -

ReceiveMediaTypes 125
 TRVCamReceiver 125

Receiver 103, 176
 TRVCamView 103
 TRVTrafficMeter 176

ReceiverConnectionProperties 170
 RVMediaServer 170

ReceiverHost 150
 TRVCamSender 150

ReceiverPort 150
 TRVCamSender 150

ReceiverSource 189
 TRVAudioViewer 189

Reconnect 160
 TRVCamSender 160

Recording 113
 TRVAudioPlayer 113

RefreshRate 80
 TRVCamMultiView 80

RememberLastFrame 81, 99
 TRVCamMultiView 81
 TRVCamView 99

RemoveAllowedSender 156
 TRVCamSender 156

RemoveDefaultReceiver 157
 TRVCamSender 157

RenderMode 78, 96
 TRVCamMultiView 78
 TRVCamView 96

ResetImageSetting 68
 TRVCamera 68

RestartServer 160
 TRVCamSender 160

RetryCount 125
 TRVCamReceiver 125

Rotation 55
 TRVCamera 55

RTSPPort 46
 TRVCamera 46

RVAUDIO_DATA 215

RVCMD_DATA 215

RVFILE_DATA 215

RVMedia
 About IP cameras and USB cameras 12
 additional information 25
 Comparison 15
 Components 38
 Features 13
 How it works - cameras 16
 How it works - video chat with a server 20
 How it works - video chat without a server 18
 modes of connections 21
 Third-party libraries 24
 version history 26

RVMedia Overview 12

RVMEDIA_DATA 215

RVMGetWindowRect 217

RVUSER_DATA 215

RVVIDEO_DATA 215

- S -

SamplesPerSec 179
 TCustomRVMicrophone 179

Saturation 45
 TRVCamera 45

ScaleViewers 81
 TRVCamMultiView 81

SearchCamera 68
 TRVCamera 68

Searching 55
 TRVCamera 55

SearchPanelColor 81, 99
 TRVCamMultiView 81
 TRVCamView 99

SearchPanelTextColor 81, 99
 TRVCamMultiView 81
 TRVCamView 99

SendCmd 160
 TRVCamSender 160

SendCmdToGroup 171
 RVMediaServer 171

SendCmdToUser 171

SendCmdToUser 171
 TRVMediaServer 171
 SendCommandToGUID 172
 TRVMediaServer 172
 Sender 176
 TRVTrafficMeter 176
 SenderConnectionProperties 170
 TRVMediaServer 170
 SenderPort 150
 TRVCamSender 150
 Senders 125
 TRVCamReceiver 125
 SendFile 161
 TRVCamSender 161
 SendMediaAccessCancelRequest 162
 TRVCamSender 162
 SendMediaAccessRequest 162
 TRVCamSender 162
 SendMediaTypes 150
 TRVCamSender 150
 SendOptions 151
 TRVCamSender 151
 SendUserData 161
 TRVCamSender 161
 SessionKey 126, 151, 170
 TRVCamReceiver 126
 TRVCamSender 151
 TRVMediaServer 170
 SessionKey2 126
 TRVCamReceiver 126
 SetCamVideoMode 63
 TRVCamera 63
 SetDesktopVideoMode 63
 TRVCamera 63
 Sharpness 45
 TRVCamera 45
 ShowCameraSearch 100
 TRVCamView 100
 ShowCaption 101
 TRVCamView 101
 ShowCmd 151
 TRVCamSender 151
 ShowFocus 42
 TRVCamControl 42
 SmoothImage 56, 126
 TRVCamera 56
 TRVCamReceiver 126
 SoundIgnoreInterval 181
 TCustomRVMicrophone 181
 SoundMinLevel 181

TCustomRVMicrophone 181
 SourceAudioGUID 90
 TRVCamRecorder 90
 SourceAudioIndex 90, 151
 TRVCamRecorder 90
 TRVCamSender 151
 SourceFileName 56
 TRVCamera 56
 SourceGUID 155
 TRVCamSender 155
 SourceType 182
 TCustomRVMicrophone 182
 SourceVideoGUID 90
 TRVCamRecorder 90
 SourceVideoIndex 90, 152
 TRVCamRecorder 90
 TRVCamSender 152
 State 127
 TRVCamReceiver 127
 Style 119
 TRVMicrophoneView 119
 SwitchLEDOff 69
 TRVCamera 69
 SwitchLEDOOn 69
 TRVCamera 69
 SynchronizedReceiveUserData 127
 TRVCamReceiver 127

- T -

TCPConnectionType 128, 153
 TRVCamReceiver 128
 TRVCamSender 153
 TCustomRVAudioOutput 191
 TCustomRVAudioOutput.Active 192
 TCustomRVAudioOutput.OnGetAudio 192
 TCustomRVAudioOutput.Volume 192
 TCustomRVAudioPlayer 192
 TCustomRVAudioPlayer.AudioOutputDeviceCount 194
 TCustomRVAudioPlayer.AudioOutputDeviceIndex 194
 TCustomRVAudioPlayer.AudioOutputDeviceList 194
 TCustomRVAudioPlayer.BufferDuration 194
 TCustomRVAudioPlayer.Mute 194
 TCustomRVAudioPlayer.NoiseReduction 194
 TCustomRVAudioPlayer.NoiseReductionLevel 194
 TCustomRVAudioPlayer.Pitch 195
 TCustomRVAudioPlayer.VolumeMultiplier 195
 TCustomRVMicrophone 177
 TCustomRVMicrophone.AudioInputDeviceCount 179
 TCustomRVMicrophone.AudioInputDeviceIndex 179

- TCustomRVMicrophone.AudioInputDeviceList 179
- TCustomRVMicrophone.BitsPerSample 179
- TCustomRVMicrophone.BufferDuration 179
- TCustomRVMicrophone.Mute 180
- TCustomRVMicrophone.NoiseReduction 180
- TCustomRVMicrophone.NoiseReductionLevel 180
- TCustomRVMicrophone.OnCloseWavFile 183
- TCustomRVMicrophone.OnOpenWavFile 183
- TCustomRVMicrophone.OnReadWavFile 183
- TCustomRVMicrophone.Pitch 181
- TCustomRVMicrophone.SamplesPerSec 179
- TCustomRVMicrophone.SoundIgnoreInterval 181
- TCustomRVMicrophone.SoundMinLevel 181
- TCustomRVMicrophone.SourceType 182
- TCustomRVMicrophone.UseRNNNoise 180
- TCustomRVMicrophone.VolumeMultiplier 182
- TCustomRVMicrophone.WAVFileName 183
- TCustomRVMicrophone.WAVUseOptions 183
- TCustomRVReceiver 184
- TCustomRVReceiver.AudioOutput 185
- TCustomRVReceiver.OnDataRead 185
- TCustomRVSender 185
- TempFolder 167
 - TRVMediaServer 167
- TestMode 154, 210
 - TRVCamSender 154
 - TRVMotionDetector 210
- TextSettings 97
 - TRVCamView 97
- Third-party libraries used by RVMedia 24
- Title 101
 - TRVCamView 101
- ToFile 50
 - TRVCamera 50
- TRVAlignAudioViewer 82
- TRVAudioCodec 236
- TRVAudioEvent 164, 231
- TRVAudioPlayer 110
- TRVAudioPlayer.EncodeAudioCodec 112
- TRVAudioPlayer.EncodeAudioCodecName 112
- TRVAudioPlayer.EncodeBitrate 112
- TRVAudioPlayer.EncodeChannels 112
- TRVAudioPlayer.EncodeSampleRate 112
- TRVAudioPlayer.OnError 114
- TRVAudioPlayer.OnStopRecording 114
- TRVAudioPlayer.OutputFileName 113
- TRVAudioPlayer.Recording 113
- TRVAudioPlayer.UseFFmpeg 114
- TRVAudioSource 185
- TRVAudioSource.Active 186
- TRVAudioSource.OnGetAudio 187
- TRVAudioSource.Volume 186
- TRVAudioSourceWithOutput 187
- TRVAudioSourceWithOutput.AudioOutput 188
- TRVAudioViewer 188
- TRVAudioViewer.AudioSource 189
- TRVAudioViewer.GUID 189
- TRVBitsPerSample 237
- TRVBoundsTestMode 237
- TRVBufferLimitType 196
- TRVBufferOptions 196
- TRVCamControl 39
- TRVCamControl.ArrowColor 41
- TRVCamControl.ArrowColorClicked 41
- TRVCamControl.ArrowColorHot 41
- TRVCamControl.ArrowLineColor 41
- TRVCamControl.BorderColor 41
- TRVCamControl.Color 41
- TRVCamControl.DiskBrightness 41
- TRVCamControl.DiskColor 41
- TRVCamControl.ParentColor 41
- TRVCamControl.ShowFocus 42
- TRVCamDoneEvent 232
- TRVCamera 42
- TRVCamera.AddUser 62
- TRVCamera.Agent 44
- TRVCamera.Bitrate 45
- TRVCamera.Brightness 45
- TRVCamera.CameraControl 46
- TRVCamera.CameraHost 46
- TRVCamera.CameraPort 46
- TRVCamera.CameraSearchTimeout 46
- TRVCamera.CommandMode 47
- TRVCamera.Contrast 45
- TRVCamera.CycleVideoImages 47
- TRVCamera.DesktopForm 47
- TRVCamera.DesktopMode 47
- TRVCamera.DesktopRect 47
- TRVCamera.DesktopWindowHandle 47
- TRVCamera.DesktopZoomPercent 47
- TRVCamera.DeviceType 49
- TRVCamera.FFMpegProperty 50
- TRVCamera.FileName 50
- TRVCamera.FillVideoDeviceList 63
- TRVCamera.FlipHorizontally 51
- TRVCamera.FlipVertically 51
- TRVCamera.FocusDistance 51
- TRVCamera.FocusType 51

TRVCamera.Framerate	45	TRVCamera.OnLogin	71
TRVCamera.GetAccessibleCamMethods	64	TRVCamera.OnLoginFailed	72
TRVCamera.GetAccessibleCamProperties	64	TRVCamera.OnMoved	72
TRVCamera.GetAvailableCamMethods	64	TRVCamera.OnNewImage	72
TRVCamera.GetAvailableCamProperties	64	TRVCamera.OnPrepared	72
TRVCamera.GetCamCurrentVideoMode	63	TRVCamera.OnProgress	73
TRVCamera.GetCamVideoMode	63	TRVCamera.OnSearchComplete	73
TRVCamera.GetCamVideoModeCount	63	TRVCamera.OnSetProperty	73
TRVCamera.GetCamVideoModelIndex	63	TRVCamera.OnStartVideoFile	73
TRVCamera.GetColorControlPropertyRange	65	TRVCamera.OnStartVideoStream	73
TRVCamera.GetDesktopCurrentVideoMode	63	TRVCamera.OnVideoStart	74
TRVCamera.GetDesktopVideoMode	63	TRVCamera.Parameters	54
TRVCamera.GetDesktopVideoModeCount	63	TRVCamera.PlayVideoFile	67
TRVCamera.GetDesktopVideoModelIndex	63	TRVCamera.PlayVideoStream	67
TRVCamera.GetSnapShot	65	TRVCamera.ProxyProperty	55
TRVCamera.GStreamerProperty	52	TRVCamera.Quality	55
TRVCamera.Hue	45	TRVCamera.ResetImageSetting	68
TRVCamera.IsSupportedFFMPEG	65	TRVCamera.Rotation	55
TRVCamera.IsSupportedGStreamer	66	TRVCamera.RTSPPort	46
TRVCamera.JpegIntegrity	52	TRVCamera.Saturation	45
TRVCamera.Language	52	TRVCamera.SearchCamera	68
TRVCamera.Latency	53	TRVCamera.Searching	55
TRVCamera.LockCommands	66	TRVCamera.SetCamVideoMode	63
TRVCamera.LoginPrompt	53	TRVCamera.SetDesktopVideoMode	63
TRVCamera.MaxCameraSearchThreadCount	53	TRVCamera.Sharpness	45
TRVCamera.ModifyUser	62	TRVCamera.SmoothImage	56
TRVCamera.MoveCenter	66	TRVCamera.SourceFileName	56
TRVCamera.MoveDown	66	TRVCamera.SwitchLEDOff	69
TRVCamera.MoveDownStop	66	TRVCamera.SwitchLEDOn	69
TRVCamera.MoveHPatrol	66	TRVCamera.ToFile	50
TRVCamera.MoveHPatrolStop	66	TRVCamera.UnlockCommands	66
TRVCamera.MoveLeft	66	TRVCamera.URL	57
TRVCamera.MoveLeftDown	66	TRVCamera.UserAccess	57
TRVCamera.MoveLeftStop	66	TRVCamera.UserName	57
TRVCamera.MoveLeftUp	66	TRVCamera.UserPassword	57
TRVCamera.MoveRight	66	TRVCamera.Users	58
TRVCamera.MoveRightDown	66	TRVCamera.VideoDeviceCount	58
TRVCamera.MoveRightStop	66	TRVCamera.VideoDeviceIdList	58
TRVCamera.MoveRightUp	66	TRVCamera.VideoDeviceIndex	58
TRVCamera.MoveStop	66	TRVCamera.VideoDeviceList	58
TRVCamera.MoveUp	66	TRVCamera.VideoFormat	59
TRVCamera.MoveUpStop	66	TRVCamera.VideoImagesURLs	47
TRVCamera.MoveVPatrol	66	TRVCamera.VideoMode	60
TRVCamera.MoveVPatrolStop	66	TRVCamera.VideoResolution	61
TRVCamera.OnEndVideoFile	70	TRVCamera.WaitForSearch	69
TRVCamera.OnEndVideoStream	70	TRVCamera.WaitForVideo	69
TRVCamera.OnError	70	TRVCamera.WaitForVideoFile	69
TRVCamera.OnGetImage	71	TRVCamera.WaitForVideoStream	69
TRVCamera.OnGetVideoStreamIndex	71	TRVCameras.IPCameraType	52

- TRVCameraType 238
- TRVCameraTypes 238
- TRVCamErrorEvent 232
- TRVCamFocus 51
- TRVCamMethod 64
- TRVCamMoveMode 238
- TRVCamMultiView 26, 74
- TRVCamMultiView.AllowFullScreen 76
- TRVCamMultiView.AudioSource 76
- TRVCamMultiView.CameraControl 76
- TRVCamMultiView.CamMoveMode 77
- TRVCamMultiView.CaptionColor 77
- TRVCamMultiView.CaptionFont 77
- TRVCamMultiView.CaptionHeight 77
- TRVCamMultiView.CaptionTextSettings 77
- TRVCamMultiView.Color 78
- TRVCamMultiView.CurRenderMode 78
- TRVCamMultiView.FocusLineColor 78
- TRVCamMultiView.Font 79
- TRVCamMultiView.FullScreen 79
- TRVCamMultiView.FullScreenMultiView 79
- TRVCamMultiView.HoverLineColor 80
- TRVCamMultiView.IconStyle 80
- TRVCamMultiView.Language 80
- TRVCamMultiView.OnFullScreen 85
- TRVCamMultiView.OnSelectViewer 85
- TRVCamMultiView.OnViewerPaint 85
- TRVCamMultiView.ParentFont 79
- TRVCamMultiView.RefreshRate 80
- TRVCamMultiView.RememberLastFrame 81
- TRVCamMultiView.RenderMode 78
- TRVCamMultiView.ScaleViewers 81
- TRVCamMultiView.SearchPanelColor 81
- TRVCamMultiView.SearchPanelTextColor 81
- TRVCamMultiView.ViewerColor 78
- TRVCamMultiView.ViewerIndex 82
- TRVCamMultiView.Viewers 82
- TRVCamMultiView.WaitAnimationDelay 84
- TRVCamPaintEvent 107
- TRVCamProgressEvent 73
- TRVCamProperty 64
- TRVCamReceiver 120
- TRVCamReceiver.Active 122
- TRVCamReceiver.AudioLatency 122
- TRVCamReceiver.BufferDuration 122
- TRVCamReceiver.BufferSize 122
- TRVCamReceiver.Color 122
- TRVCamReceiver.ConnectionProperties 123
- TRVCamReceiver.FilterSystemCmd 123
- TRVCamReceiver.GetMaxChannelCount 129
- TRVCamReceiver.GetOpenChannelCount 129
- TRVCamReceiver.GUIDMy 123
- TRVCamReceiver.IgnoreCorruptedFrames 123
- TRVCamReceiver.JpegIntegrity 124
- TRVCamReceiver.Mute 124
- TRVCamReceiver.OnCloseChannel 135
- TRVCamReceiver.OnConnect 130
- TRVCamReceiver.OnConnected 130
- TRVCamReceiver.OnConnectError 130
- TRVCamReceiver.OnDecodeAudio 130
- TRVCamReceiver.OnDecodeVideo 131
- TRVCamReceiver.OnDisconnect 130
- TRVCamReceiver.OnGetAllGroups 133
- TRVCamReceiver.OnGetAllOnlineUsers 133
- TRVCamReceiver.OnGetAllUsers 133
- TRVCamReceiver.OnGetGroupInfo 131
- TRVCamReceiver.OnGetGroupUsers 134
- TRVCamReceiver.OnGetImage 134
- TRVCamReceiver.OnMediaAccessCancelRequest 139
- TRVCamReceiver.OnMediaAccessRequest 139
- TRVCamReceiver.OnOpenChannel 135
- TRVCamReceiver.OnReceiveCmdData 135
- TRVCamReceiver.OnReceivedFile 136
- TRVCamReceiver.OnReceiveFileData 136
- TRVCamReceiver.OnReceiveUserData 137
- TRVCamReceiver.OnReceivingFile 136
- TRVCamReceiver.OnRequestJoinGroup 132
- TRVCamReceiver.OnSessionDisconnected 137
- TRVCamReceiver.OnUserEnter 138
- TRVCamReceiver.OnUserExit 138
- TRVCamReceiver.OnUserJoinsGroup 138
- TRVCamReceiver.OnUserLeavesGroup 138
- TRVCamReceiver.Port 124
- TRVCamReceiver.Protocol 124
- TRVCamReceiver.ProxyProperty 125
- TRVCamReceiver.ReceiveMediaTypes 125
- TRVCamReceiver.RetryCount 125
- TRVCamReceiver.Senders 125
- TRVCamReceiver.SessionKey 126
- TRVCamReceiver.SessionKey2 126
- TRVCamReceiver.SmoothImage 126
- TRVCamReceiver.State 127
- TRVCamReceiver.SynchronizedReceiveUserData 127
- TRVCamReceiver.TCPConnectionType 128
- TRVCamReceiver.UseTempFiles 128
- TRVCamReceiver.VideoLatency 122
- TRVCamReceiver.Volume 128
- TRVCamRecorder 85

TRVCamRecorder.Active	87	TRVCamSender.GetAllOnlineUsers	158
TRVCamRecorder.AudioBitrate	87	TRVCamSender.GetAllUsers	158
TRVCamRecorder.AudioChannels	87	TRVCamSender.GetGroupInfo	158
TRVCamRecorder.AudioCodec	88	TRVCamSender.GetUsersFromGroup	158
TRVCamRecorder.AudioCodecName	88	TRVCamSender.GoodbyeToAllowedSenders	156
TRVCamRecorder.AudioSampleFormat	87	TRVCamSender.GoodbyeToDefaultReceivers	157
TRVCamRecorder.AudioSampleRate	87	TRVCamSender.GUIDFrom	147
TRVCamRecorder.AudioSource	89	TRVCamSender.GUIDGroup	147
TRVCamRecorder.OnError	92	TRVCamSender.GUIDTo	147
TRVCamRecorder.OnFirstFrame	92	TRVCamSender.HelloToAllowedSenders	156
TRVCamRecorder.OnGetImage	93	TRVCamSender.HelloToDefaultReceivers	157
TRVCamRecorder.OutputFileName	89	TRVCamSender.JoinGroup	158
TRVCamRecorder.Paused	89	TRVCamSender.LeaveGroup	158
TRVCamRecorder.SourceAudioGUID	90	TRVCamSender.MinChangeAreaSize	148
TRVCamRecorder.SourceAudioIndex	90	TRVCamSender.NeedSendFullFrame	160
TRVCamRecorder.SourceVideoGUID	90	TRVCamSender.OnConnect	163
TRVCamRecorder.SourceVideoIndex	90	TRVCamSender.OnConnected	163
TRVCamRecorder.UseAudio	89	TRVCamSender.OnConnectError	163
TRVCamRecorder.VideoAutoSize	91	TRVCamSender.OnDisconnect	163
TRVCamRecorder.VideoBitrate	91	TRVCamSender.OnEncodeAudio	164
TRVCamRecorder.VideoCodec	88	TRVCamSender.OnEncodeVideo	164
TRVCamRecorder.VideoCodecName	88	TRVCamSender.OnSendCmd	165
TRVCamRecorder.VideoEncodingParameters	91	TRVCamSender.OnSentCmd	165
TRVCamRecorder.VideoFramePerSec	91	TRVCamSender.PixelColorThreshold	148
TRVCamRecorder.VideoHeight	91	TRVCamSender.PixelColorThresholdB	148
TRVCamRecorder.VideoSource	92	TRVCamSender.PixelColorThresholdG	148
TRVCamRecorder.VideoWidth	91	TRVCamSender.PixelColorThresholdR	148
TRVCamSender	139	TRVCamSender.Protocol	149
TRVCamSender.Active	143	TRVCamSender.ProxyProperty	149
TRVCamSender.AddAllowedSender	156	TRVCamSender.ReceiverHost	150
TRVCamSender.AddAllowedSenders	156	TRVCamSender.ReceiverPort	150
TRVCamSender.AddDefaultReceiver	157	TRVCamSender.Reconnect	160
TRVCamSender.AllowMediaAccess	158	TRVCamSender.RemoveAllowedSender	156
TRVCamSender.AudioSource	143	TRVCamSender.RemoveDefaultReceiver	157
TRVCamSender.BeginCmd	160	TRVCamSender.RestartServer	160
TRVCamSender.BufferSize	144	TRVCamSender.SendCmd	160
TRVCamSender.CancelMediaAccess	158	TRVCamSender.SenderPort	150
TRVCamSender.ChangedAreaProcessingMode	144	TRVCamSender.SendFile	161
TRVCamSender.ClearAllowedSenders	156	TRVCamSender.SendMediaAccessCancelRequest	162
TRVCamSender.CompressionOptions	144	TRVCamSender.SendMediaAccessRequest	162
TRVCamSender.CompressionQuality	145	TRVCamSender.SendMediaTypes	150
TRVCamSender.ConnectionProperties	145	TRVCamSender.SendOptions	151
TRVCamSender.Encoding	145	TRVCamSender.SendUserData	161
TRVCamSender.EndCmd	160	TRVCamSender.SessionKey	151
TRVCamSender.ExtraMediaSources	146	TRVCamSender.ShowCmd	151
TRVCamSender.FilterBlur	146	TRVCamSender.SourceAudioIndex	151
TRVCamSender.FrameDifferenceInterval	146	TRVCamSender.SourceGUID	155
TRVCamSender.FullFrameInterval	147	TRVCamSender.SourceVideoIndex	152
TRVCamSender.GetAllGroups	158	TRVCamSender.TCPConnectionType	153

- TRVCamSender.TestMode 154
- TRVCamSender.UseGUID 147
- TRVCamSender.VideoResolution 154
- TRVCamSender.VideoSendType 154
- TRVCamSender.VideoSource 155
- TRVCamSender.WaitForSendCmd 160
- TRVCamSound 115
- TRVCamSound.Camera 116
- TRVCamSound.GUID 116
- TRVCamSound.OnGetAudioStreamIndex 116
- TRVCamUser 58
 - Access 58
 - Password 58
 - UserName 58
- TRVCamUserCollection 58
- TRVCamVideoMode 239
- TRVCamView 93
- TRVCamView.AllowFullScreen 95
- TRVCamView.AutoSize 95
- TRVCamView.Camera 103
- TRVCamView.CamMoveMode 95
- TRVCamView.CaptionColor 101
- TRVCamView.CaptionFont 101
- TRVCamView.CaptionHeight 101
- TRVCamView.CaptionParts 101
- TRVCamView.Color 96
- TRVCamView.CurRenderMode 96
- TRVCamView.FocusLineColor 96
- TRVCamView.Font 97
- TRVCamView.FrameScaleQuality 97
- TRVCamView.FullScreen 97
- TRVCamView.FullScreenView 98
- TRVCamView.GUIDFrom 98
- TRVCamView.HoverLineColor 98
- TRVCamView.IconStyle 99
- TRVCamView.IndexFrom 98
- TRVCamView.Language 99
- TRVCamView.OnBeginMove 107
- TRVCamView.OnEndMove 107
- TRVCamView.OnMouseEnter 107
- TRVCamView.OnMouseLeave 107
- TRVCamView.OnPaint 107
- TRVCamView.ParentFont 97
- TRVCamView.Receiver 103
- TRVCamView.RememberLastFrame 99
- TRVCamView.RenderMode 96
- TRVCamView.SearchPanelColor 99
- TRVCamView.SearchPanelTextColor 99
- TRVCamView.ShowCameraSearch 100
- TRVCamView.ShowCaption 101
- TRVCamView.TextSettings 97
- TRVCamView.Title 101
- TRVCamView.UseOptimalVideoResolution 102
- TRVCamView.VideoSource 103
- TRVCamView.ViewMode 103
- TRVCamView.WaitAnimationDelay 105
- TRVCamViewCollection 82
- TRVCamViewerPaintEvent 85
- TRVCamViewItem 82
- TRVChangedAreaProcessingMode 144, 240
- TRVCloseWavFileEvent 183
- TRVCmd 197
- TRVCmdEvent 233
- TRVCmdOption 168
- TRVCmdOptions 168
- TRVCmdParamCollection 198
- TRVCmdParamItem 198
- TRVCmdWriteEvent 165
- TRVColorControlProperty 65, 240
- TRVColorModel 239
- TRVCompressionOptions 199
- TRVCompressionType 241
- TRVConnectionProperties 200
- TRVDataReadEvent 137, 234
- TRVDecodeAudioEvent 130
- TRVDecodeVideoEvent 131
- TRVDesktopMode 47
- TRVDesktopVideoMode 241
- TRVDeviceType 49
- TRVEncodeVideoEvent 164
- TRVEncodingType 242
- TRVFFMpegFilter 243
- TRVFFMpegProperty 201
- TRVFFMpegRemuxProperty 203
- TRVFileEvent 136
- TRVFileReadEvent 136
- TRVFullScreenEvent 234
- TRVGetGroupUsersEvent 134
- TRVGetImageEvent 134
- TRVGetMediaStreamIndexEvent 234
- TRVGroupEvent 138
- TRVGroupInfoEvent 131
- TRVGStreamerProperty 204
- TRVGVideoScaleMethod 204
- TRVImageEvent 235
- TRVImageWrapper 206
- TRVJoinGroupEvent 132
- TRVJpegIntegrity 243

TRVKeepClientInfoMode	169	TRVMicrophoneView	117
TRVMAnchor	82	TRVMicrophoneView.OnPaint	119
TRVMAnchors	82	TRVMicrophoneView.Orientation	118
TRVMAnsiString	244	TRVMicrophoneView.Style	119
TRVMBitmap	206	TRVMLanguage	245
TRVMColor	244	TRVMotionDetector	209
TRVMediaAccessCancelRequestEvent	139	TRVMotionDetector.ChangedAreaProcessingMode	209
TRVMediaAccessRequestEvent	139	TRVMotionDetector.DetectChanges	211
TRVMediaServer	165	TRVMotionDetector.GetRect	211
TRVMediaServer.BufferOptions	167	TRVMotionDetector.IsRectValid	211
TRVMediaServer.CmdOptions	168	TRVMotionDetector.MinChangeAreaSize	210
TRVMediaServer.FilterUserCmd	168	TRVMotionDetector.PixelColorThreshold	210
TRVMediaServer.GUIDMy	169	TRVMotionDetector.PixelColorThresholdB	210
TRVMediaServer.HTTPActive	169	TRVMotionDetector.PixelColorThresholdG	210
TRVMediaServer.HTTPPort	169	TRVMotionDetector.PixelColorThresholdR	210
TRVMediaServer.KeepClientInfoMode	169	TRVMotionDetector.TestMode	210
TRVMediaServer.MaxGroupCount	170	TRVMPoint	246
TRVMediaServer.OnConnectionCountChanged	173	TRVMRect	246
TRVMediaServer.OnDataRead	172	TRVMRenderMode	246
TRVMediaServer.OnError	174	TRVMSCountEvent	173
TRVMediaServer.OnPacketProcessing	173	TRVMSendType	214
TRVMediaServer.OnServerCmd	173	TRVMSServerCmdEvent	173
TRVMediaServer.OnStart	174	TRVMState	127
TRVMediaServer.OnStop	174	TRVMSUserEvent	174
TRVMediaServer.OnUserConnect	174	TRVMVideoSendType	154
TRVMediaServer.OnUserDisconnect	174	TRVMWindowHandle	247
TRVMediaServer.ReceiverConnectionProperties	170	TRVMWindowHandleArray	247
TRVMediaServer.SendCmdToGroup	171	TRVOnLoginEvent	71
TRVMediaServer.SendCmdToUser	171	TRVOnLoginFailedEvent	72
TRVMediaServer.SendCommandToGUID	172	TRVOpenWavFileEvent	183
TRVMediaServer.SenderConnectionProperties	170	TRVParamType	248
TRVMediaServer.SessionKey	170	TRVProtocol	248
TRVMediaServer.TempFolder	167	TRVProtocolEx	249
TRVMediaServer.UDPActive	170	TRVProtocolsEx	249
TRVMediaServer.UDPPort	170	TRVProxyProperty	211
TRVMediaSource	189	TRVQualityType	55
TRVMediaSourceCollection	207	TRVReadWavFileEvent	183
TRVMediaSourceItem	208	TRVRTSPFlag	249
TRVMediaSourceItem.AudioSource	208	TRVRTSPFlags	249
TRVMediaSourceItem.Name	208	TRVSampleFormat	249
TRVMediaSourceItem.VideoSource	208	TRVSamplesPerSec	250
TRVMediaTypes	244	TRVSelectCamViewEvent	85
TRVMFProxyMode	211	TRVSenderCollection	212
TRVMIconStyle	245	TRVSenderCollectionEx	213
TRVMicrophone	117	TRVSenderItem	213
TRVMicrophoneOrientation	118	TRVSenderItemEx	214
TRVMicrophonePaintEvent	119	TRVSenderTestMode	154
TRVMicrophoneStyle	119	TRVSendMode	47
		TRVSendOptions	214

TRVServerDataReadEvent 172
 TRVServerEvent event 174
 TRVSessionKey 250
 TRVSocket 251
 TRVSocketEvent 235
 TRVSoundSourceType 182
 TRVStreamType 196
 TRVTCPConnectionType 251
 TRVTrafficMeter 175
 TRVTrafficMeter.Camera 176
 TRVTrafficMeter.Language 176
 TRVTrafficMeter.Receiver 176
 TRVTrafficMeter.Sender 176
 TRVUnitSize 246
 TRVUserAccess 57, 58
 TRVUserEvent 138
 TRVVideoCodec 251
 TRVVideoEncodingParameters 91
 TRVVideoFormat 59
 TRVVideoMode 60
 TRVVideoResolution 191, 252
 TRVVideoSource 190
 TRVVideoSource.Abort 191
 TRVVideoSource.Aborting 190
 TRVVideoSource.FramePerSec 190
 TRVVideoSource.GetOptimalVideoResolution 191
 TRVWebCamDialog 107
 TRVWebCamDialog.Camera 109
 TRVWebCamDialog.Execute 110
 TRVWebCamDialog.Language 110

- U -

UDPActive 170
 TRVMediaServer 170
 UDPPort 170
 TRVMediaServer 170
 UnlockCommands 66
 TRVCamera 66
 URL 57
 TRVCamera 57
 UseAudio 89
 TRVCamRecorder 89
 UseFFMpeg 114
 TRVAudioPlayer 114
 UseGUID 147
 TRVCamSender 147
 UseOptimalVideoResolution 102
 TRVCamView 102

UserAccess 57
 TRVCamera 57
 UserName 57, 58
 TRVCamera 57
 TRVCamUser 58
 UseRNNoise 180
 TCustomRVMicrophone 180
 UserPassword 57
 TRVCamera 57
 Users 58
 TRVCamera 58
 UseTempFiles 128
 TRVCamReceiver 128

- V -

VideoAutoSize 91
 TRVCamRecorder 91
 VideoBitrate 91
 TRVCamRecorder 91
 VideoCodec 88
 TRVCamRecorder 88
 VideoCodecName 88
 TRVCamRecorder 88
 VideoDeviceCount 58
 TRVCamera 58
 VideoDeviceIdList 58
 TRVCamera 58
 VideoDeviceIndex 58
 TRVCamera 58
 VideoDeviceList 58
 TRVCamera 58
 VideoEncodingParameters 91
 TRVCamRecorder 91
 VideoFormat 59
 TRVCamera 59
 VideoFramePerSec 91
 TRVCamRecorder 91
 VideoHeight 91
 TRVCamRecorder 91
 VideoImagesURLs 47
 TRVCamera 47
 VideoLatency 122
 TRVCamReceiver 122
 VideoMode 60
 TRVCamera 60
 VideoResolution 61, 154
 TRVCamera 61
 TRVCamSender 154
 VideoSendType 154

VideoSendType 154
 TRVCamSender 154

VideoSource 92, 103, 155, 208
 TRVCamRecorder 92
 TRVCamSender 155
 TRVCamView 103
 TRVMediaSourceItem 208

VideoWidth 91
 TRVCamRecorder 91

ViewerColor 78
 TRVCamMultiView 78

ViewerIndex 82
 TRVCamMultiView 82

Viewers 82
 TRVCamMultiView 82

ViewMode 103
 TRVCamView 103

Volume 128, 186, 192
 TCustomRVAudioOutput 192
 TRVAudioSource 186
 TRVCamReceiver 128

VolumeMultiplier 182, 195
 TCustomRVAudioPlayer 195
 TCustomRVMicrophone 182

- W -

WaitAnimationDelay 84, 105
 TRVCamMultiView 84
 TRVCamView 105

WaitForSearch 69
 TRVCamera 69

WaitForSendCmd 160
 TRVCamSender 160

WaitForVideo 69
 TRVCamera 69

WaitForVideoFile 69
 TRVCamera 69

WaitForVideoStream 69
 TRVCamera 69

WAVFileName 183
 TCustomRVMicrophone 183

WAVUseOptions 183
 TCustomRVMicrophone 183