RVMedia

Table of Contents

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

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**	**SMInter net**	**VideoL ab**	**VisioFor ge**	**RVMedi a**
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

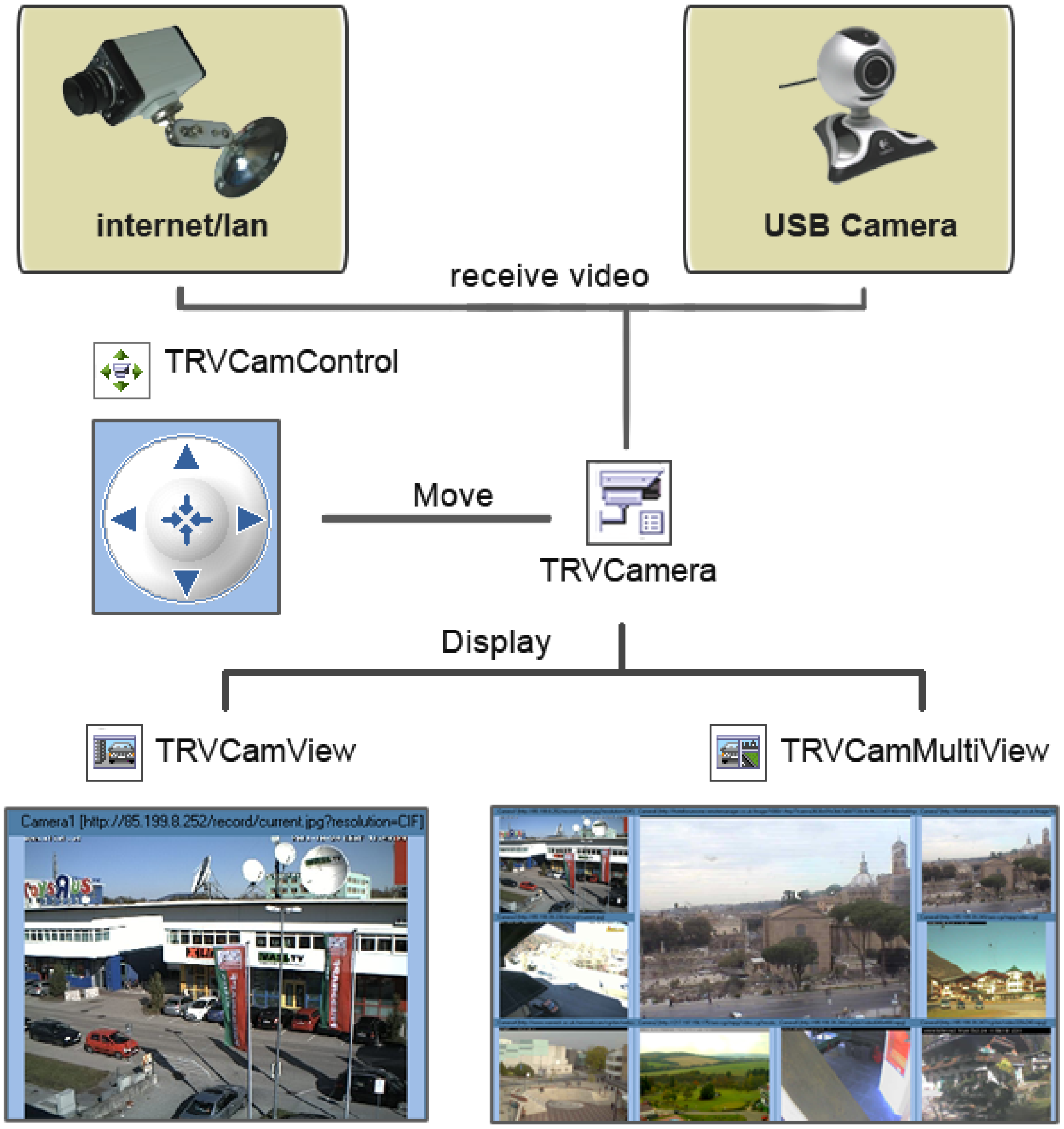
○ supported partially (requires an additional code)

1.4 How it works: cameras

Overview

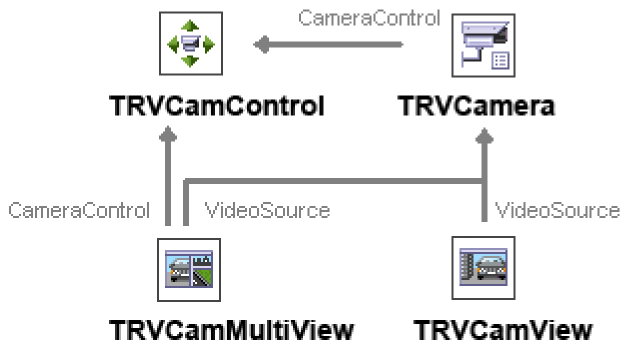
TRVCamera[42] component works with cameras: searches, configures, receives a video stream, saves or plays a video file.

A video received by TRVCamera can be displayed in TRVCamView[93] or TRVCamMultiView[74] components. The camera movement is controlled by TRVCamControl[39] component or by the viewer itself.

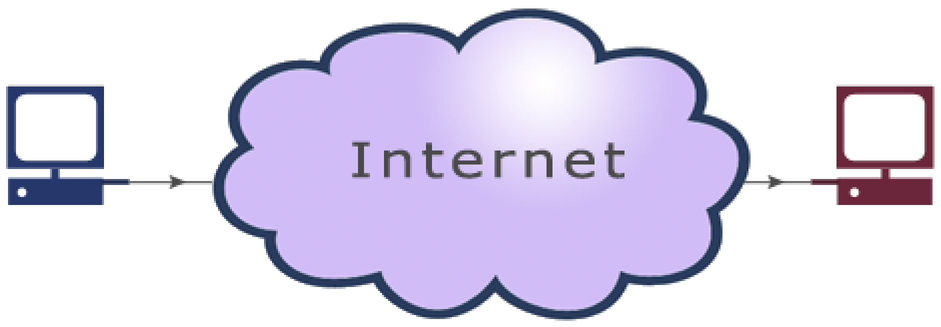


How the components are linked

The components are linked using CameraControl and VideoSource properties:



1.5 How it works: video chat without a server



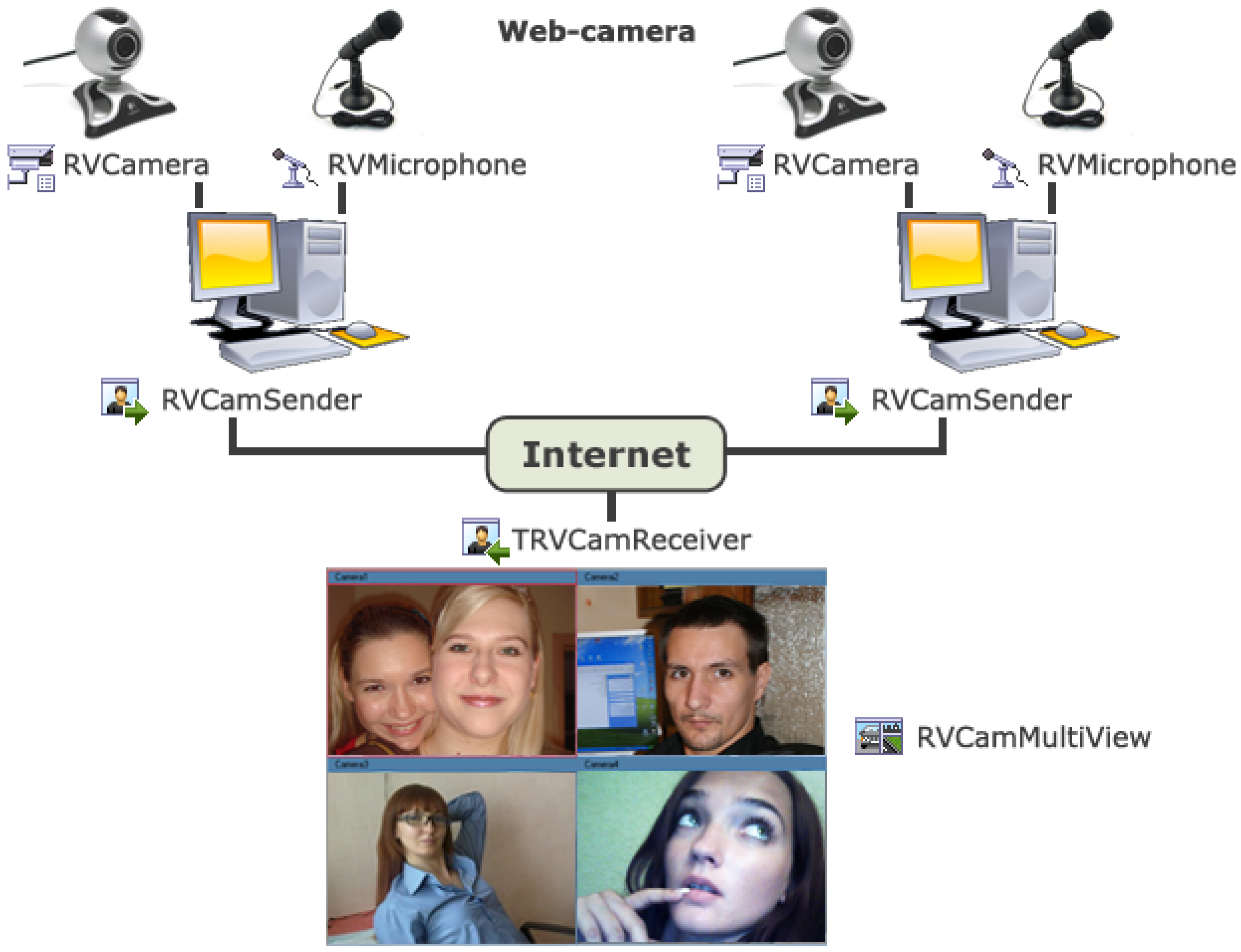
Overview

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.

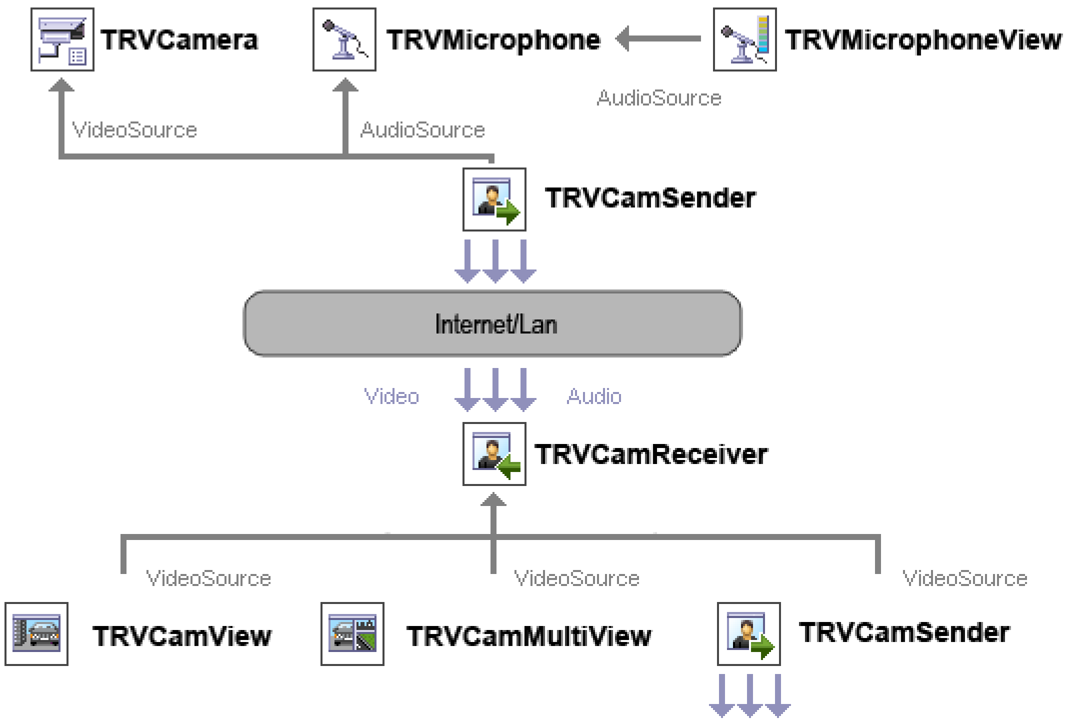
In addition to video and audio data, the components can send files, commands and arbitrary data.



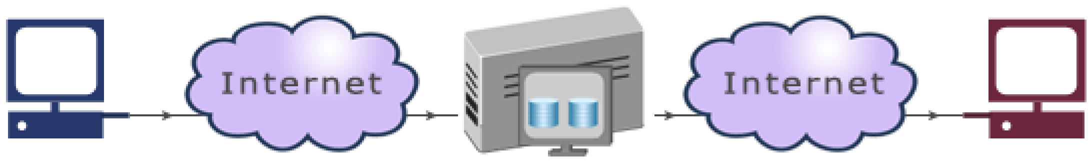
To create a video chat, you need a sender and a receiver at the both sides.

How the components are linked

The components are linked using VideoSource and AudioSource properties



1.6 How it works: video chat with a server



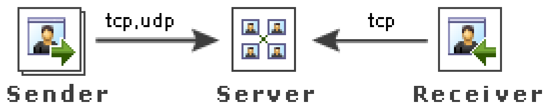
Overview

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

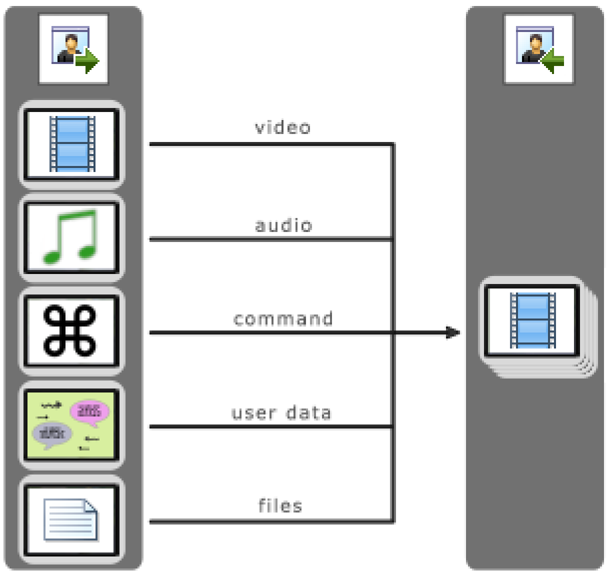
In the Option 1, there are two possible modes for transferring data from Client1 to Client2. The both modes requires that at least one side has a "white IP address" (available for another side)

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.

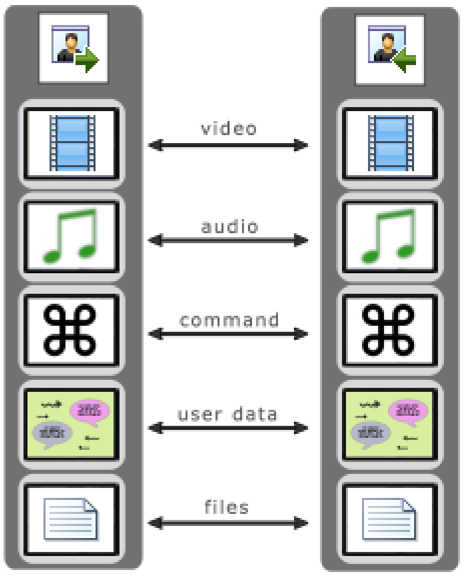
In this mode, a connection session is not created, because there are no permanent connections between the clients, a connection is created only when data are being sent, and it is terminated when the transfer is finished.



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).

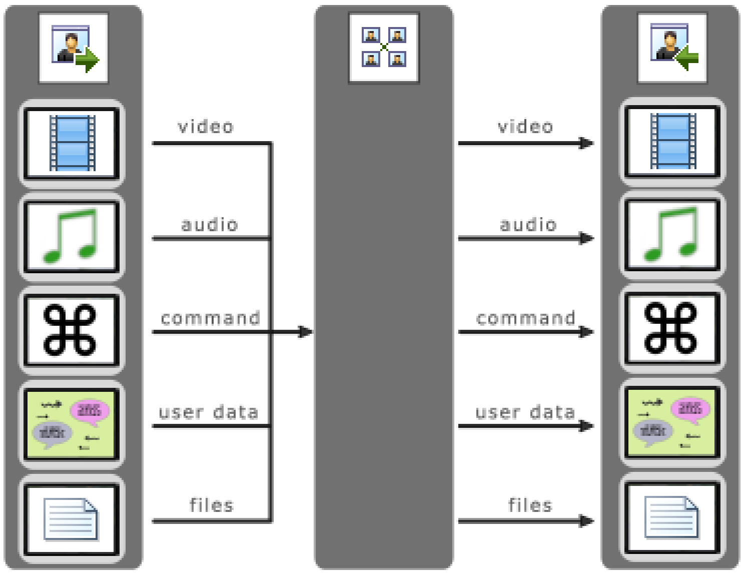


Option 2: Client-Server-Client

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 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 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 URL to either TRVCamRecorder[85].OutputFileName[89] or TRVCamera[42].FFmpegProperty[50].RemuxProperty[201].FileName[203].

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: 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 fond 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 uniit (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]: rvvcHEVC.

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].OnEncodeAudiio[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
- RTFPFlags, ProbeSize properties
- FrameDrop: Boolean allows/disallows dropping frames if they are received too late.
- CustomProperties: TStringList: additional parameters for FFMpeg.
- RTSPTransport 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[208].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] = *rvdtUserData*. 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 recieved (TRVCamReceiver[120].OnDecodeAudio[130]).

Changes in local USB cameras

- RVMedia can decode different formats of video from local web cameras (YV12, YUYV, YUY2, YVYU, 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[208].
- 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 designtime 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 [208]) 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 SynchonizedReceiveUserData [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

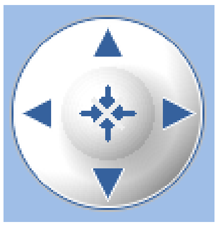
FMX:

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 TRVCamView [93] /TRVCamMultiView [74] components, or implement your own user interface by calling TRVCamera [42] .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 TRVCamera [42] component.

Option 2: Create TRVCamControl component and assign it to CameraControl property of TRVCamMultiView [74] component. In this mode, the component controls the movement of the camera from the selected view.

2.1.1.1 Properties

In TRVCamControl

- ArrowColor [41]
- ArrowColorClicked [41]
- ArrowColorHot [41]
- ArrowLineColor [41]
- BorderColor [41]
- DiskBrightness [41]
- DiskColor [41]
- ShowFocus [42]

Inherited from TCustomControl

- Color [41]
- ParentColor [39]

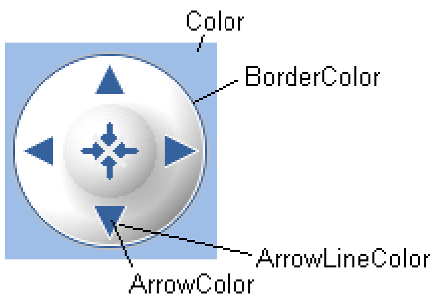
2.1.1.1.1 TRVCamControl's colors

Colors

```
property Color: TRVMColor 244;
property BorderColor: TRVMColor 244;
property ArrowLineColor: TRVMColor 244;
property ArrowColor: TRVMColor 244;
property ArrowColorClicked: TRVMColor 244;
property ArrowColorHot: TRVMColor 244;
property DiskColor: TRVMColor 244;
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 **DiskBrigtness** define the color of the control itself (disk). **DiskColor** defines hue, **DiskBrigtness** 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: *clWhite*
- 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(TRVVideoSource 190)
```

Hierarchy

TObject
TPersistent
TComponent
TRVMediaSource 189
TRVVideoSource 190

Description

A video received by TRVCamera can be displayed in TRVCamView 93 or TRVCamMultiView 74 components. The camera movement is controlled by TRVCamControl 39 component or by the viewer itself.

TRVCamera can be assigned as a VideoSource to TRVCamSender 139.

You can record video from TRVCamera using TRVCamRecorder 85.

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 file[50] and played from a MJPEG file[67].

——

* 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] = *rvdtRTSP* and *rvdtHTTP*, assigns GStreamer's bitrate, range 0..2097152 (0 means autocalculated bitrate) DeviceType [49] = *rvdtIPCamera*, 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 SearchCamera [68].

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 GetColorControlPropertyRange [65] method.

Property	DeviceType [49]			
	rvdtIPCamera	***rvdtWebCamera***	***rvdtDesktop***	**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 Foscam cameras (or compatible)			
• Hue • Saturation • Sharpness			–	

See also

- ResetImageSetting [68]

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)
- RTSPort: 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] founds 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
rvsmWait	The component waits for the commands to complete
rvsmNoWait	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
rvdmFull	A whole desktop or a whole monitor is used as a source, see VideoDevice*** [58] properties.
rvdmRect	A rectangle defined in **DesktopRect** is used
rvdmWindow	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
rvdtIPCamera	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
rvdtWebCamera	USB webcam. VideoDeviceIndex[58] property is used.	not used	
rvdtDesktop	Screen or window. DesktopMode[47] property is used.	not used	
rvdtFile	Video from a file. SourceFileName[56] property is used.	not used	used optionally**
rvdtRTSP	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	
rvdtHTTP	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	
rvdtUserData	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

rvdtIPCamera

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 SearchCamera[68].

Default values:

False

2.1.2.1.14 TRVCamera.FocusType, FocusDistance

The properties control the camera focus, it 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 SearchCamera[68]). These are write-only properties: RVMedia cannot read these properties from the camera, but can assign them.

FocusType:

Value	Meaning
rvcfFocusAuto	automatic focus
rvcfFocusNear	focus near
rvcfFocusFar	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 Latency: TRVMLanguage[245];
```

This dialog is displayed if LoginPromt[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(**MaxCameraSearchThreadCount**, 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
qtStandard	standard quality
qtFavorMotion	optimizing for a fast motion
qtFavorClarity	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
rvvfMJPEG	***MJPEG*** video stream	optional, if DeviceType 49 =*rvdtHTTP* required, if DeviceType 49 =*rvdtRTSP*	optional, if DeviceType 49 =*rvdtHTTP* or *rvdtIPCamera,* required otherwise;

rvvfH264	***H.264*** video stream	required	the specified video format is ignored: it is autodetected
rvvfAVI_H264	Video from AVI file. Video must be encoded as H.264		
rvvfMP4_H26	Video from ***MP4*** (or QuickTime) file Video must be encoded as H.264		
rvvfAVI_MPEG	Video from AVI file. Video must be encoded as ***MPEG-4 Part 2***		
rvvfMP4_MPEG	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
vm50HZ	50 Hz, typically when the camera is used indoor
vm60HZ	60 Hz, typically when the camera is used indoor
vmOutdoor	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;
```

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.

GetDesktopVideoModeIndex 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 commends 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 LoginPromt [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:

AUserName, **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: TRVCamDoneEvent(232);
```

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

A search is started in SearchCamera(68) method.

2.1.2.3.13 TRVCamera.OnSetProperty

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

```
property OnSetProperty: TRVCamDoneEvent(232);
```

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 PlayVideoFile(67) method.

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

See also

- PlayVideoStream, PlayVideoFile(67)
- OnEndVideoStream, OnEndVideoFile(70)
- OnVideoStart(74)

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] [].ViewMode [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;
```

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

Default value

vcmmDrag

See also:

- TRVCamView.CamMoveMode
- TRVCamera.Move*** methods
- TRVCamControl component

2.1.3.1.5 TRVCamMultiView.CaptionColor, CaptionFont, CaptionHeight

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

```
property CaptionColor: TRVMColor;
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[].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: 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

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: *clRed*

Default values [FMX]

- FocusLineColor: *TAlphaColorRec.Red*

Assign *clNone* (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 *clNone* (VCL and LCL) or *TAlphaColorRec.Null* (FMX) to 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]

clNone

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 *clNone*. 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 lager 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 multivewer 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
rvaavTop	Audio viewer above video viewer
rvaavBottom	Audio viewer below video viewer
rvaavLeft	Audio viewer to the left side of video viewer
rvaavRight	Audio viewer to the right side of video viewer
rvaavClient	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 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;
property VideoCodec: TRVVideoCodec;
```

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
rvmfscLow	Low quality, quick, low CPU (or GPU) usage
rvmfscNormal	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;
```

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;
```

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 *clNone/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]

clNone

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;
```

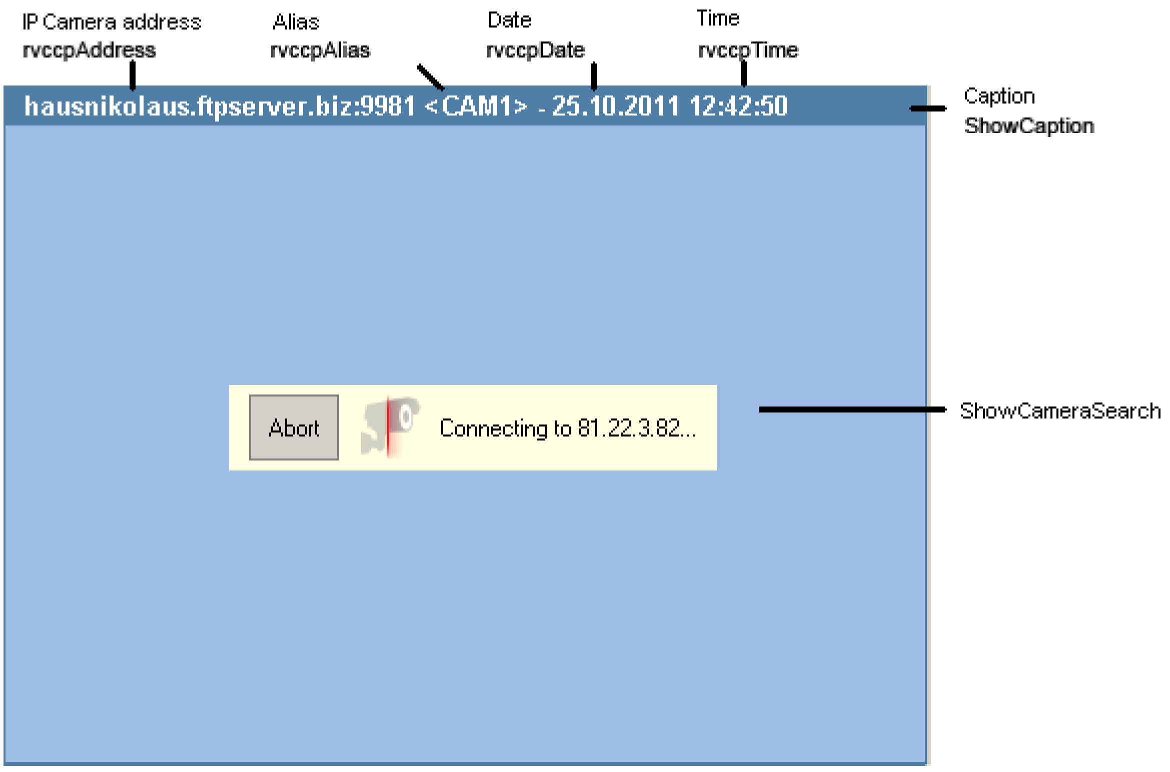
The panel contains some text, animation and “Abort” button.

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.

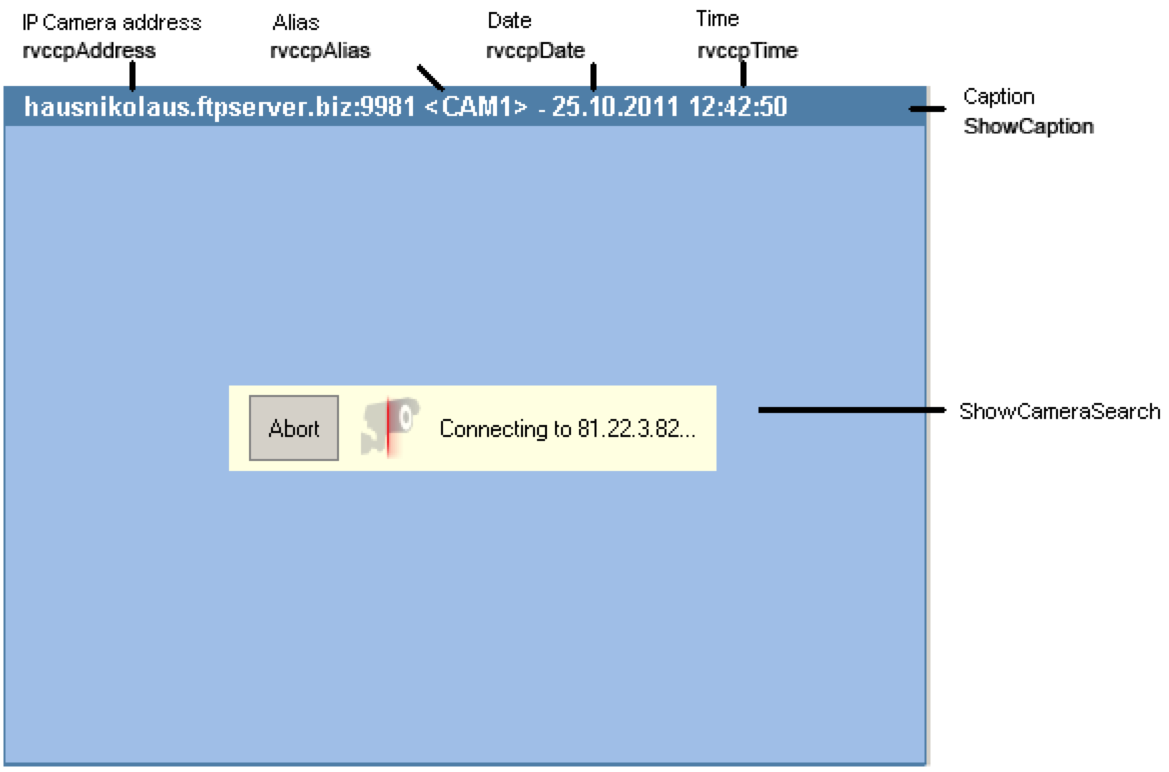
CaptionColor is also used for drawing a frame around the whole window.

[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
rvccpAddress	IP camera address
rvccpAlias	RVCamera.Parameters[54].Alias (if playing a video stream, not a file played by PlayVideoFile[67])
rvccpDate	the current date
rvccpTime	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*.

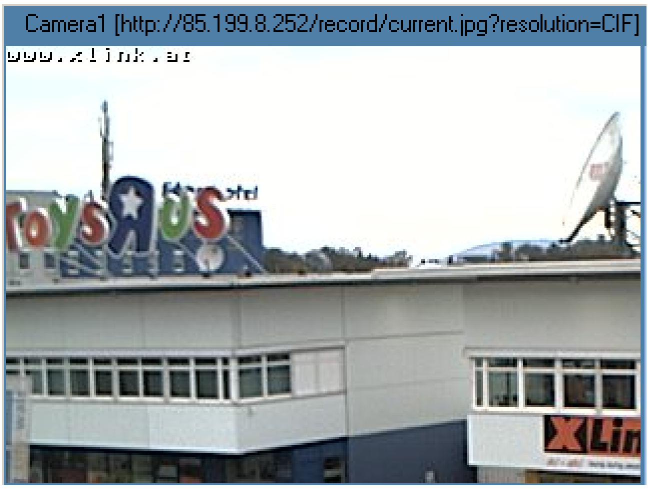
See also

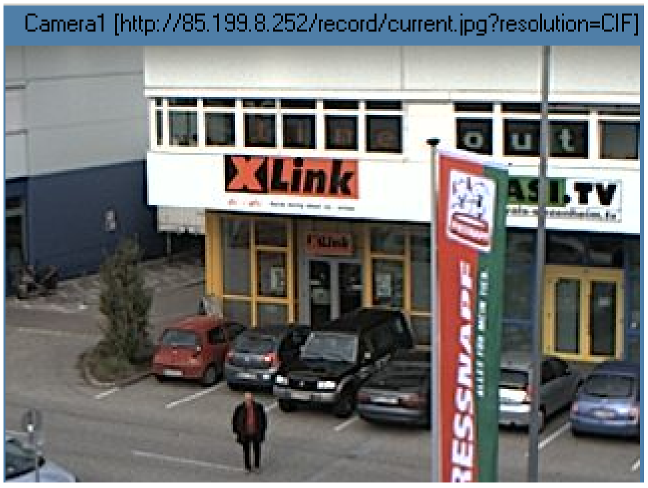
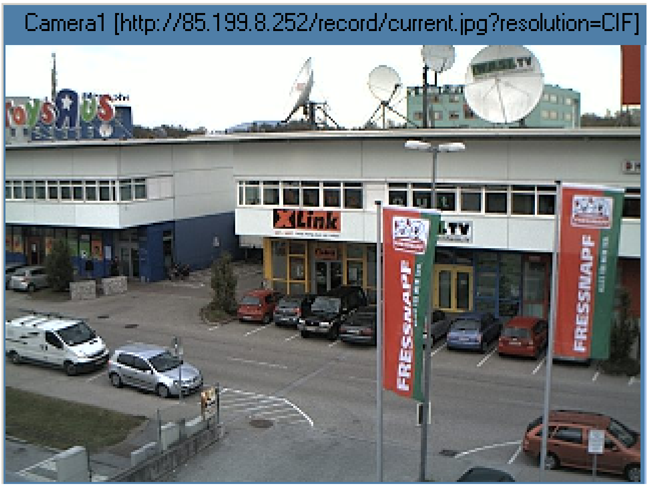
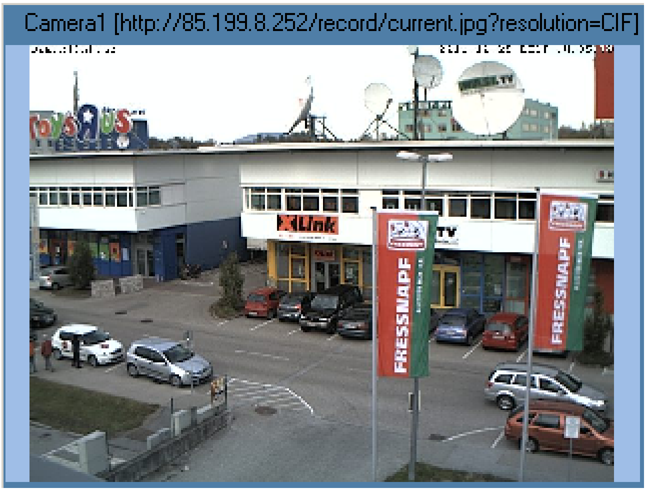
GUIDFrom [98]

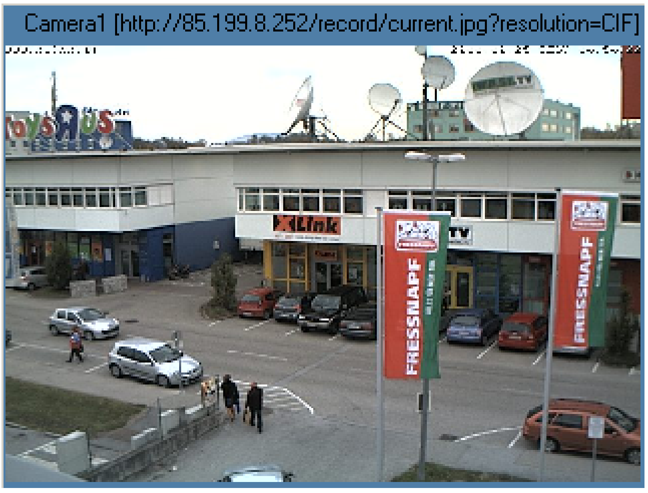
2.1.5.1.21 TRVCamView.ViewMode

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
vvmNormal A video is displayed at the top left corner of the window, without stretching	 Camera1 [http://85.199.8.252/record/current.jpg?resolution=CIF] www.xlink.at

vvmCenter A video is displayed in the center of the window, without stretching	 Camera1 [http://85.199.8.252/record/current.jpg?resolution=CIF]
vvmCut A video is stretched proportionally so that the smallest side fits the window, the largest side is truncated	 Camera1 [http://85.199.8.252/record/current.jpg?resolution=CIF]
vvmAspect (default) A video is stretched proportionally to fit the window, keeping the side proportions.	 Camera1 [http://85.199.8.252/record/current.jpg?resolution=CIF]

vvmStretch A video is stretched to fit the window.	 Camera1 [http://85.199.8.252/record/current.jpg?resolution=CIF]

[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: TRVCamDoneEvent 232;
```

See also

- CamMoveMode 95

2.1.5.2.2 TRVCamView.OnFullScreen

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

```
property OnFullScreen : TRVFullScreenEvent 234;
```

See also properties:

- AllowFullScreen 95
- FullScreen 97

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 *rvdtWebCamera*, 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 (TCustomRVAudioPlayer)
```

Hierarchy

TObject
TPersistent
TComponent
TCustomRVAudioOutput [191]
TCustomRVAudioPlayer [192]

Description

This component must be linked to TRVMicrophone [117], TRVCamSound [115] or TRVCamReceiver [120] components to play sound.

Without **TRVAudioPlayer**, TRVMicrophone [117] cannot play or record sound.

Without **TRVAudioPlayer**, TRVCamSound [115] cannot play or record sound, and cannot synchronize video playback speed to audio playback speed.

Without **TRVAudioPlayer**, TRVCamReceiver [120] 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 TRVMicrophone [117].AudioOutput [188], TRVCamSound [115].AudioOutput [188], or TRVCamReceiver [120].AudioOutput [185] properties.

Playing sound

See the topic on TCustomRVAudioPlayer [192] for properties controlling sound playing.

Recoding

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 EncodeAudioCodec [112] 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 Recording [113] property. Sound is recorded to OutputFileName [113].

Recording is stopped when you assign *False* to Recording [113] property, or change any of Encode* [112] properties. When it is stopped, OnStopRecording [114] occurs.

Alternatively, you can use TRVCamRecorder [85] 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(TCustomRVMicrophone[177])
```

Hierarchy

TObject
TPersistent
TComponent
TRVMediaSource[189]
TRVAudioSource[185]
TRVAudioSourceWithOutput[187]
TCustomRVMicrophone[177]

Description

This component publishes properties inhertied from TCustomRVMicrophone[177].

How to use

TRVMicrophone can be assigned as an AudioSource[143] to TRVCamSender[139] to send sound to the network.

TRVMicrophone can be assigned as an AudioSource[189] to TRVMicrophoneView[117] to visualize its activity.

This component does not play sound itself. To play sound, assign TRVAudioPlayer[110] component to AudioOutput[188] property.

Other audio source components

- TRVCamSound[115] (sound from videos received by TRVCamera[42] 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(TRVAudioViewer[188])
```

Hierarchy

VCL and LCL:

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl[188]
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 TRVAudioViewer[188]

- 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
rvmpsSimple	
rvmpsHistogram	
rvmpsGradient	
rvmpsOwnerDraw	(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 chose 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
- 🟩 SynchonizedReceiveUserData 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]:

clNone (transparent)

Default value [FMX]:

TAlphaColorRec.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]
rvmsDisconnect	The receiver is disconnected	*False*
rvmsConnecting	The receiver is connecting	*True*
rvmsConnect	The receiver is connected	*True*
rvmsDisconnecting	The receiver is disconnecting	*True*

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.SynchonizedReceiveUserData

Defines the thread context for OnReceiveUserData[137] event

```
property SynchonizedReceiveUserData: 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: TRVTCPConnectionType(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;
    SessionKey: TRVSessionKey; const GUIDGroup: TRVMAnsiString;
    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;
```

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;
  SessionKey: TRVSessionKey; const GUIDGroup,
  GUIDUser: TRVMAnsiString; ACmd: TRVCmd);
var
  i, Count : Integer;
  GUIDGroup   : TRVMAnsiString;
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;
property OnGetAllOnlineUsers: TRVCmdEvent;
```

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;
    FileName: String; FileOffs, TotalFileSize: Int64;
    AData: TStream; ASocket: TRVSocket;
    GUIDFrom, GUIDTo, GUIDGroup: TGUID; AMediaIndex : Word) of object;
  TRVFileEvent = procedure(Sender: TObject; SessionKey : TRVSessionKey;
    FileName: String; TotalFileSize: Int64; ASocket: TRVSocket;
    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 SynchonizedReceiveUserData[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 of TRVCamSender[150] does not send 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

TRVCamSenser sends video and audio to the network.

Unit [VCL and LCL] MRVSender;

Unit [FMX] fmxMRVSender;

Syntax

```
TRVCamSender = class(TCustomRVSender[185])
```

Hierarchy

TObject
TPersistent
TComponent
TCustomRVSender[185]

Description

If Active[143]=*True*, the component gets the video from VideoSource[155] and audio from AudioSource[143], and sends them to the network. These video and audio streams can be received by TRVCamReceiver[120] and TRVMediaServer[165]. In addition to audio and video, a sender can send commands[160] and files[161].

A video format is specified in Encoding[145].

Media channels

Data may be organized in multiple media channels.

Media channels are useful when you send information via TRVMediaServer[165]; for example, they allow for clients to send video from multiple cameras. For direct connections (to TRVCamReceiver[120]), you can use multiple TRVCamSender components instead.

Audio and video for the default (0th) channel are defined in properties VideoSource[155] and AudioSource[143].

Audio and video for the 1st, 2nd channels and so on are defined in VideoSource and AudioSource properties of items in ExtraMediaSource[146] 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 TRVCamReceiver[120] - 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:

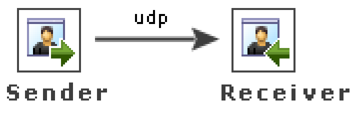
Settings for TRVCamSender:

- Active[143]=*True*

Settings for TRVCamReceiver[120]:

- Active[122]=*True*

Option 1: UDP connection from Sender to Receiver



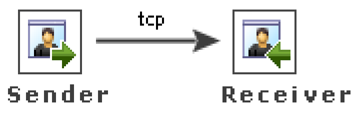
Settings for TRVCamSender:

- 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



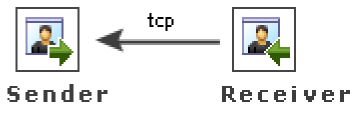
Settings for TRVCamSender:

- 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[208].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 CompressionQuality	Large values of Compression quiality
Jpeg, HWL	Small data size, low image quality, low CPU usage	Large data size, high image quality, high CPU usage
Png	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: TRVMediaSourceCollection207;
```

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 TRVMediaServer [165]

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 group [158] identified by **GUIDGroup**.

Otherwise, the sender sends data to default receivers [157]. If this list is empty, data go to nowhere.

Default values

'' (empty string)

See also:

- GUIDFrom [147]

2.3.2.1.15 TRVCamSender.MinChangeAreaSize, PixelColorThreshold

The properties specify how the previous and the current video frames are compared, if Encoding [145] = *rvet*Change*

```
property MinChangeAreaSize: Integer;
property PixelColorThreshold: Integer;
property PixelColorThresholdR: Integer;
property PixelColorThresholdG: Integer;
property PixelColorThresholdB: Integer;
```

if Encoding [145] = *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>**PixelColorThreshold**. Otherwise, they are treated as different if (|R1-R2| > **PixelColorThresholdR**) or (|G1-G2| > **PixelColorThresholdG**) or (|B1-B2| > **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

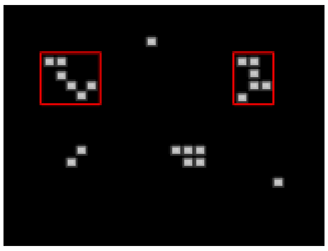
See also

- TestMode [154]
- ChangedAreaProcessingMode [144]

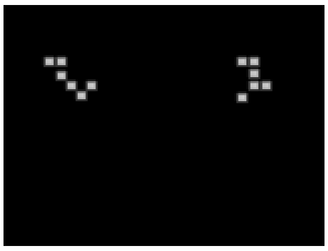
Example

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[208].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;
```

2.3.2.1.22 TRVCamSender.SessionKey

Returns the identifier of the current session.

```
property SessionKey: TRVSessionKey;
```

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

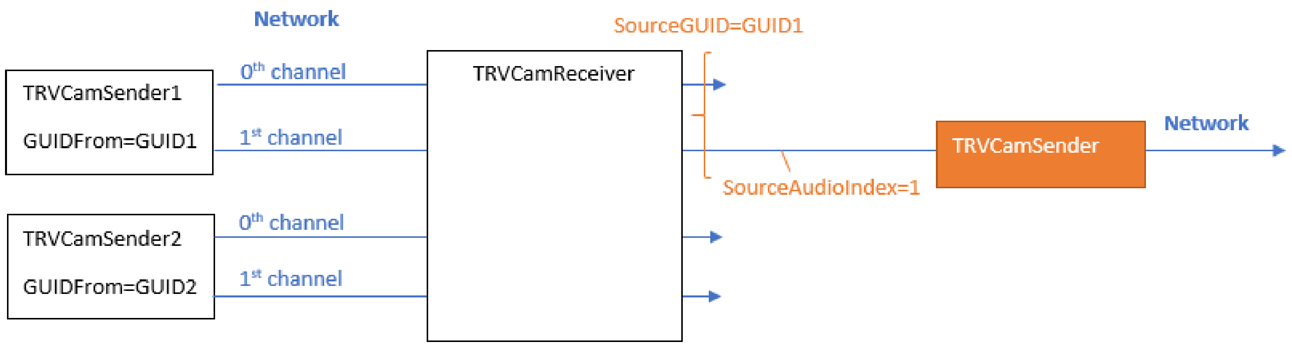
Example

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

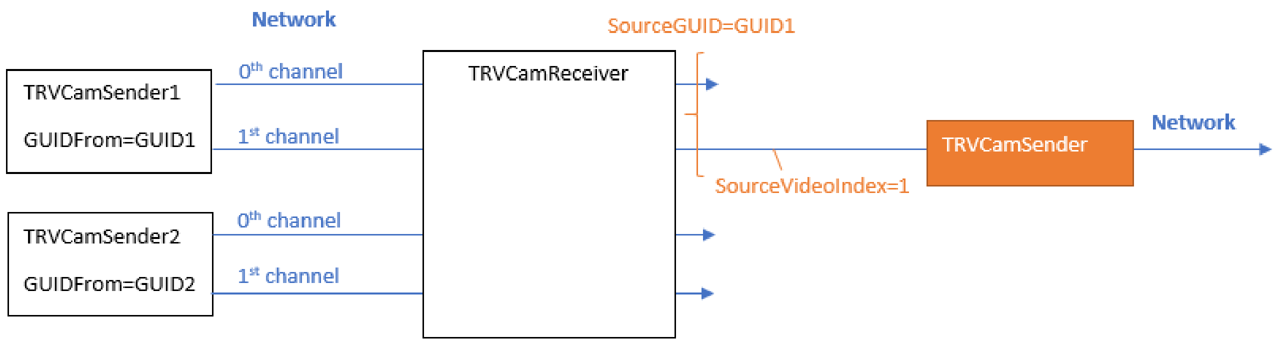
Example

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: TRVTCPConnectionType [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;
```

rvstmChangedFragments may be used if Encoding=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 property) are framed with rectangles. This mode can be useful to find the optimal values of MinChangeAreaSize and PixelColorThreshold 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;
```

if **VideoResolution** <> *rvDefault*, and **VideoResolution** is less than the video resolution of VideoSource, the sender sends video in **VideoResolution**.

Default value

rvDefault

See also

TRVCamera.VideoResolution

2.3.2.1.29 TRVCamSender.VideoSendType

Specifies how video is sent.

```
type
  TRVMVideoSendType = (rvmvstImageStream, rvmvstVideoStream);
property VideoSendType : TRVMVideoSendType;
```

Value	Meaning
rvmvstImageStream	Frames are sent separately. A connection is established for sending each frame. Several frames can be sent in parallel.
rvmvstVideoStream	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;
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]

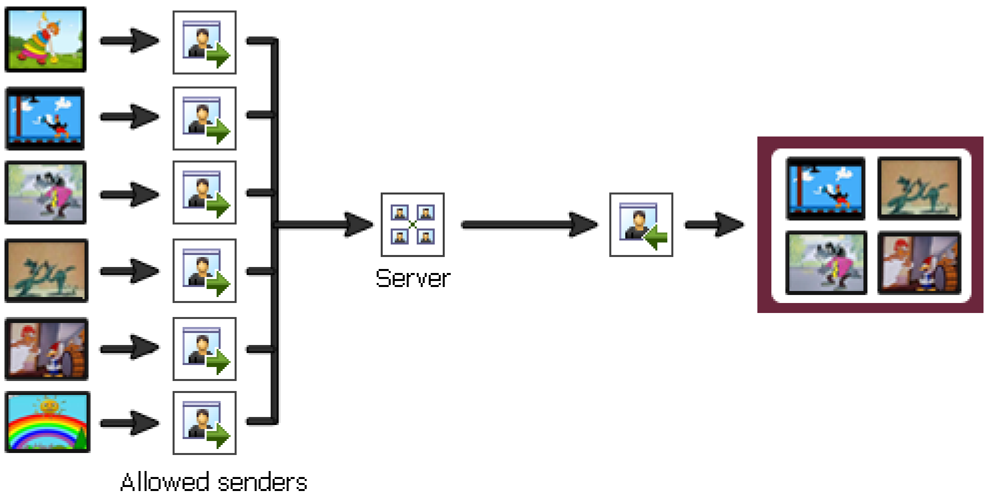
NeedSendFullFrame 160
Reconnect 160
RemoveAllowedSender 156
RemoveDefaultReceiver 157
RestartServer 160
SendCmd 160
SendFile 160
SendMediaAccessRequest 162
SendMediaAccessCancelRequest 162
SendUserData 160
WaitForSendCmd 160

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].

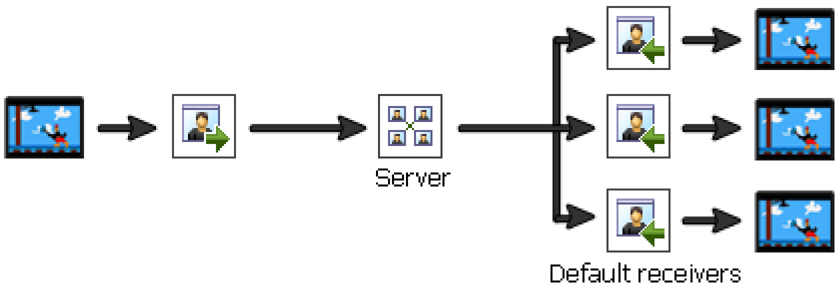
2.3.2.2.2 TRVCamSender.AddDefaultReceiver and others

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.

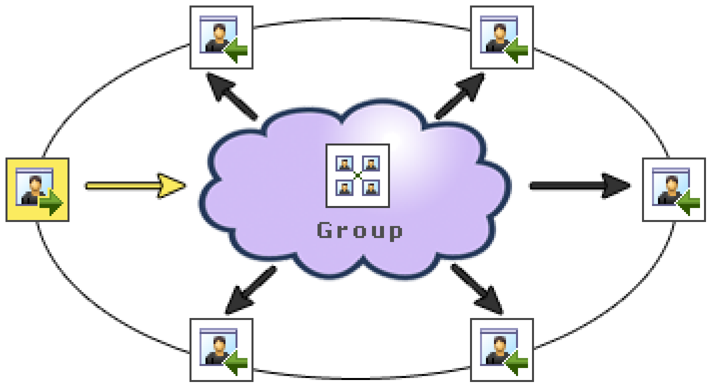
2.3.2.2.5 TRVCamSender.JoinGroup, LeaveGroup, etc.

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 empy, 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 empy, 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 untill 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, SendUseData

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 empy, 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 empy, 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;
  ADataType: Word = RVMEDIA_DATA);
procedure SendMediaAccessCancelRequest(const GUID: TRVMAnsiString;
  ADataType: Word = RVMEDIA_DATA);
```

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 us 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)
```

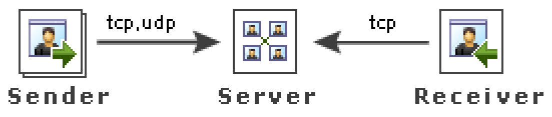
Hierarchy

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.*

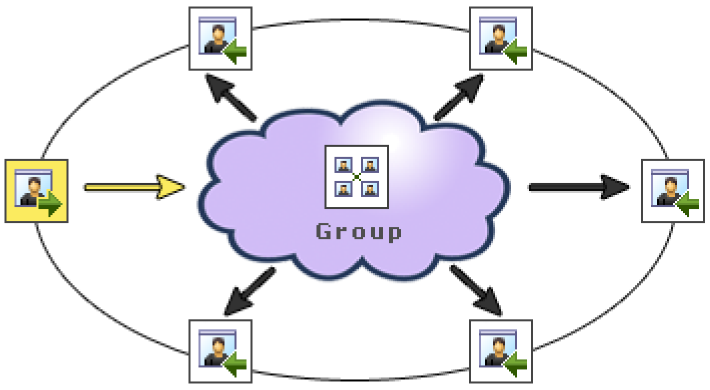
Client to client

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.

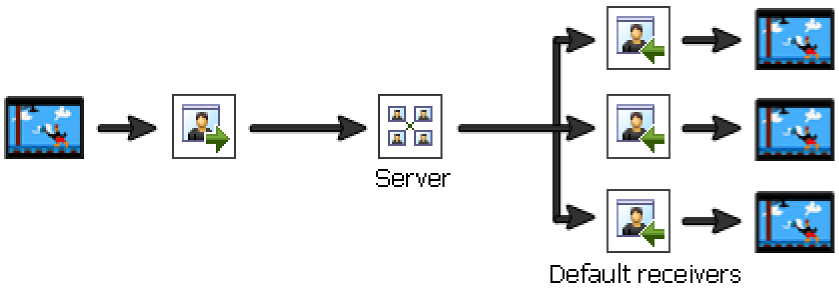
The target group for data can be specified in TRVCamSender.GUIDTo[147] (or similar parameters of methods sending commands[160], files[161], etc.)

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
rvcpUseSystemCmd	Allows processing system commands on the server. All the options below works only if *rvcpUseSystemCmd* is included.
rvcpCmdAllGroups	Allows clients to receive a list of groups on the server. See: • TRVCamSender[139].GetAllGroups[158] • TRVCamReceiver[120].OnGetAllGroups[133]
rvcpCmdAllUsers	Allows clients to receive a list of all clients on the server. See: • TRVCamSender[139].GetAllUsers[158] • TRVCamReceiver[120].OnGetAllUsers[133]
rvcpCmdAllOnline	Allows clients to receive a list of all online clients on the server. See: • TRVCamSender[139].GetAllOnlineUsers[158] • TRVCamReceiver[120].OnGetAllOnlineUsers[133]
rvcpCmdResetServer	Allows clients to restart the server. See: • 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;
```

This property is not used directly, but it recommended to pass it as a parameter AGUIDFrom to SendCommandToGUID.

Initial value:

'{00000000-0000-0000-0000-000000000001}'

2.3.3.1.5 TRVMediaServer.HTTPActive, HTTPPort

HTTPActive turns on/offs an HTTP server. HTTPPort defines the server port.

```
property HTTPActive: Boolean;
property HTTPPort: Word;
```

HTTPPort and UDPPort 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
rvkclmWhileOnline	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.
rvkclmAlways	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
- list of default receivers

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.UDPActive, UDPPort

UDPActive turns on/offs a UDP server. UDPPort defines the server port.

```
property UDPActive: Boolean;
property UDPPort: Word;
```

HTTPPort(169) and UDPPort must be different.

Default values

- UDPActive: *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:

- OnConnectionCountChanged[173]

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:

- Camera[176]
- Sender[176]
- Receiver[176]

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 TRVMediaServer[165] 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

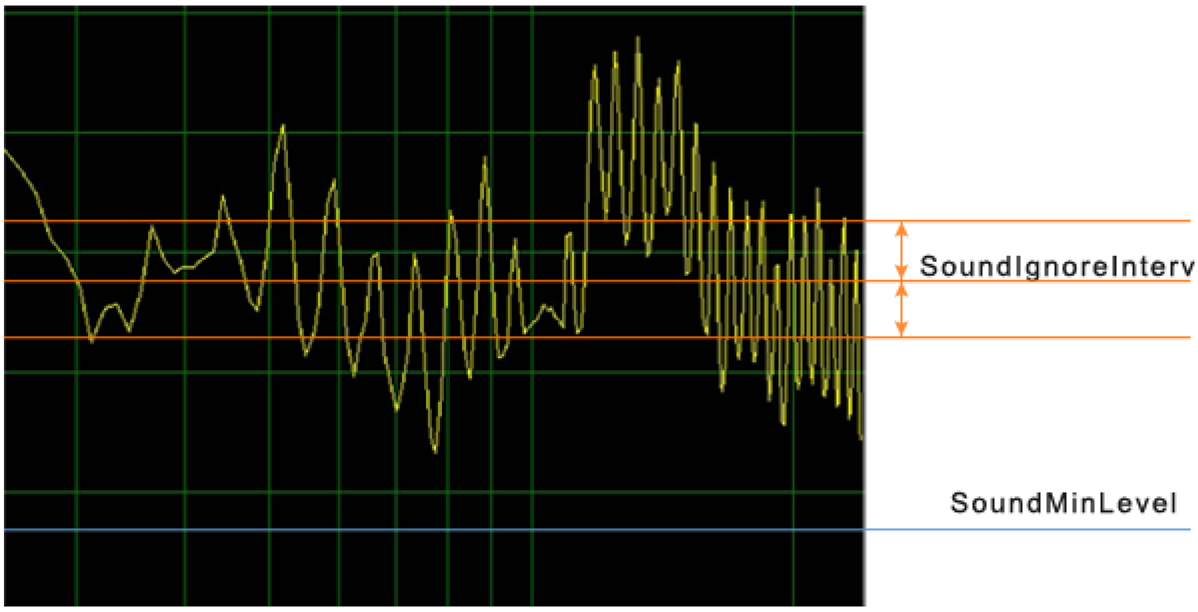
If Active[186]=*True*, the component reads sound from a microphone.

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

- AudioInputDeviceCount [179]
- AudioInputDeviceIndex [179]
- AudioInputDeviceList [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:

- Volume[186]
- VolumeMultiplier[182]

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 WAVUseOptions[183]=*True*, it is applied to sound from WAV-files as well.

The component supports two ways of noise reduction.

Built-in noise reduction

RVMediia implements a noise reduction algorithm that performs a fast Fourier transform and filtering results.

NoiseReductionLevel is used if **NoiseReduction** = True. It defined 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 RNNoise[24] 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 BiitsPerSample[179]).

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
rvsstMicrophone	Microphone
rvsstWAV	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 competely.

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 TRVVideoSource 190

- ▶ Aborting 190
- FramePerSec 190
- ▶ FramePerSecInt 190

2.4.2.1.1 TCustomRVReceiver.AudioOutput

Links this receiver component to TRVAudioPlayer 110 component.

```
property AudioOutput: TCustomRVAudioOutput 191;
```

A receiver component can play sound itself. However, it does it using default sound settings, and only on the default audio device.

Assign TRVAudioPlayer 110 component to play sound in a more configurable way.

2.4.2.2 Events

In TCustomRVReceiver

OnDataRead 185

2.4.2.2.1 TCustomRVReceiver.OnDataRead

Occurs when new data are received.

```
property OnDataRead: TRVDataReadEvent 234;
```

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 TRVCamSender 139 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 TRVMicrophone 117 and TRVCamSound 115 components.

Unit [VCL and LCL] MRVMediaSource;

Unit [FMX] fmxMRVMediaSource;

Syntax

```
TRVAudioSource = class (TRVMediaSource 189 )
```

Hierarchy

TObject
TPersistent
TComponent
TRVMediaSource 189

This class introduces properties:

- Active 186 – turn on/off the audio source (if this audio source supports turning on/off)
- Volume 186 – changes sound volume (if this audio source supports changing volume)

Do not use this class directly, use its inherited components:

- TRVMicrophone 117
- TRVCamSound 115

2.4.4.1 Properties

In TRVAudioSource

- Active 186
- Volume 186

2.4.4.1.1 TRVAudioSource.Active

Enables reading audio from the device.

```
property Active: Boolean;
```

Note: this property is available in TRVMicrophone 117 but not available in TRVCamSound 115 .

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 TRVMicrophone 117 but not available in TRVCamSound 115 .

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.VolumeMultiplier 182
- TCustomRVMicrophone.Mute 180

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* is 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 (TCustomRVAudioOutput 191)
```

Hierarchy

TObject
TPersistent
TComponent
TCustomRVAudioOutput 191

Description

Use

Objects of this class are not used directly. TRVAudioPlayer 110 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 AudioOutputDeviceList 194 array property, the count of devices is returned in AudioOutputDeviceCount 194 property.

You can choose the device by assigning to AudioOutputDeviceIndex 194 property (you can assign an index in AudioOutputDeviceList 194, or -1 to choose the default device).

Properties

The component starts playing sound when *True* is assigned to Active 192.

The following properties allow changing the system properties of the device (affect all applications): Volume 192.

The following properties allow changing the sound: VolumeMultiplier 195, NoiseReduction 194, Pitch 195.

Mute 194 turns off sound.

2.4.10.1 Properties

In TCustomRVAudioPlayer

▶ AudioOutputDeviceCount 194
AudioOutputDeviceIndex 194
▶ AudioOutputDeviceList 194
BufferDuration 194
Mute 194
NoiseReduction 194
Pitch 195
VolumeMultiplier 195

Inherited from TCustomRVAudioOutput 191

Active 192
Volume 192

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 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[208] 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
rvbltNo	All buffers are unlimited, *BufferSize properties are ignored
rvbltLimitAll	All buffers are limited by *BufferSize properties
rvbltLimitMemory	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: TRVMAnsiString[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 an 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 assign 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 if 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 applications 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: TRVFFmpegRemuxProperty[203] contains properties for remuxing.

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 [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 URL address for streaming.

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 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: Intege – 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: TRVMBitmap[206], 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: TRVBoundsTestMode237;
```

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 > **PixelColorThreshold**. Otherwise, they are treated as different if (|R1-R2| > **PixelColorThresholdR**) or (|G1-G2| > **PixelColorThresholdG**) or (|B1-B2| > **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

- TRVCamSender139.MinChangeAreaSize, PixelColorThreshold148

3.14.2 Methods

In TRVMotionDetector

DetectChanges211
GetChangedRect211
GetRect211

3.14.2.1 TRVMotionDetector.DetectChanges

Compares two video frames.

```
function DetectChanges(nImg, oImg: TRVMBitmap[206]; var Mask: TRVMBitmap[206]): Integ
```

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
rvmNoProxy	Proxy is not used. Other properties of this class are ignored.
rvmManualSettings	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: TRVMAnsiString 244): Integer;
function AddGUID(const GUID: TRVMAnsiString 244): Integer;
function DeleteGUID(const GUID: TRVMAnsiString 244): 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 TRVCamReceiver 120.

Unit [VCL and LCL] MRVReceiver;

Unit [FMX] fmxMRVReceiver;

Syntax

```
TRVSenderCollectionEx = class(TCollection)
```

Hierarchy

TObject
TPersistent
TCollection

Description

The type of items of this collection: TRVSenderItemEx 214.

This is the type of TRVCamReceiver 120.Senders 125.

This property is used in two cases:

1) when the receiver initiates connection to sender 139(s)
2) when the receiver filters data received from a media server 165.

3.18 TRVSenderItem

A class of item in TRVSenderCollection 212.

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: TRVMSendTyp;
- UserData: TRVMSendType
- Cmd: TRVMSendType
- FileData: TRVMSendType

where

```
type
  TRVMSendType = (rvmvstOnlyCurrentData, rvmvstStream);
```

Value	Meaning
rvmvstOnlyCurrentData	A new connection is created when data are available, and is closed when data are sent
rvmvstStream	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 ist 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, GetVideoCodecFileExts[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): String;
```
[247]

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 ist 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] fmxMRVFFMpegLists[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 **DefautCaption** 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 **DefautCaption** 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; AList : TStri
```

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;
  SampleFormat: TRVSampleFormat; 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 recommeded).

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] MRVGStreamer[225];

Unit [FMX] fmxMRVGStreamer[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.
- TRVTCPConnectionType [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].OnEncodeAudiio[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)
- TRVCamRecoder.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 idendifies 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,
    {$IFDEF MRVCODEC_DTS} // still experimental in FFmpeg 4
    rvacDTS,
    {$ENDIF}
    rvacWavPack, rvacALAC, rvacAMR);
```

Value	Format name (see GetAudioCodecName[225])	Recommended file extension (see GetAudioFileExt[224])
rvacWAV	WAV (Waveform Audio)	wav
rvacMP2	MP2 (MPEG-1 Audio Layer II)	mp2
rvacMP3	MP3 (MPEG-1 Audio Layer III)	mp3
rvacAC3	AC3 (Dolby Audio Codec 3)	ac3
rvacVorbis	Vorbis	ogg
rvacFLAC	FLAC (Free Lossless Audio Codec)	flac
rvacOpus	Opus	opus
rvacWMAv1	WMA (Windows Media Audio) v.1	wma
rvacWMAv2	WMA (Windows Media Audio) v.2	wma

rvacAAC	AAC (Advanced Audio Coding)	m4a
rvacRA1	RealAudio 1	ra
rvacG723_1	G.723.1	wav
rvacSpeex	Speex	spx
rvacDTS	DTS	dts
rvacWavPack	WavPack	wv
rvacALAC	ALAC (Apple Lossless Audio Codec)	m4a
rvacAMR	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
rvbps8	8 bits, unsigned integer
rvbps16	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
rvstmNone	Test mode is turned off
rvstmChangedFragments	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.
rvstmRectangles	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[208].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
vcmmNone	The component does not move the camera
vcmmToCursor	Click to rotate the camera to the mouse pointer
vcmmDrag	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
rvcapmOriginalSize	Changed areas are searched in the original video frame
rvcapmAuto	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[208].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
rvccpBrightness	Brightness
rvccpContrast	Contrast
rvccpSaturation	Saturation
rvccpHue	Hue
rvccpSharpness	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
rvtcNone	No compression. If some part of data is damaged on sending, the error does not affect other parts.
rvtcParts	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.
rvtcFully	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
rvetJPEG	A video frame is sent as a Jpeg image
rvetJPEGChange	A video frame is separated into fragments, changed fragments are sent as Jpeg images
rvetHWL (BETA!)	A video frame is sent compressed using the Haar Wavelet Transform
rvetHWLChange (BETA!)	A video frame is separated into fragments, changed fragments are sent compressed using the Haar Wavelet Transform
rvetBMP	A video frame is sent as a bitmap image
rvetBMPChange	A video frame is separated into fragments, changed fragments are sent as bitmap images
rvetPNG	A video frame is sent as a Png image. This option requires Delphi 2009 or newer.
rvetPNGChange	A video frame is separated into fragments, changed fragments are sent as Png images. This option requires Delphi 2009 or newer.
rvetMixFormat	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.
rvetMixFormatChange	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
rvffNone	(auto-selection)
rvffFastBilinear	fast bilinear
rvffBilinear	bilinear
rvffBicubic	bicubic
rvffX	experimental
rvffPoint	nearest neighbor / point
rvffArea	area averaging (downscale only, for upscale it is bilinear)
rvffBicublin	luma bicubic / chroma bilinear
rvffGauss	Gaussian
rvffSinc	sinc
rvffLaczos	Lanczos
rvffSpline	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
rvjiNone	No checking
rvjiFast	Structure checking
rvjiFull	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
rvmtVideo	Video
rvmtAudio	Audio
rvmtUserData	Arbitrary binary data
rvmtFileData	Files
rvmtCmdData	Commands

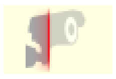
7.18 TRVMIconStyle

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
rvisClassic	
rvisModern	

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
rvmlEnglish	English (US)

rvmlFrench	French
rvmlGerman	German
rvmlRussian	Russian
rvmlSpanish	Spanish
rvmlPortugueseBr	Portuguese (Brazil)
rvmlChineseSimplified	Chinese (Simplified)
rvmlChineseTraditional	Chinese (Traditional)

This type is used for the properties:

- TRVCamView.Language 99
- TRVCamMultiView.Language 80
- TRVTrafficMeter.Language 176
- TRVWebCamDialog.Language 110

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
rvmrmSoftware	Standard drawing (GDI in Windows)
rvmrmOpenGL	OpenGL drawing.

	In TRVCamView[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. in TRVCamMultiView[74]: the whole viewer is drawn by OpenGL. OpenGL uses GPU for scaling images, so less CPU resources are used. **Requirements:** • Delphi • Lazarus • C++Builder XE or newer
rvmrmSkia	Skia4Delphi drawing. **MRVCamViewSkia** unit must be included in your project, otherwise this mode will not be initialized (and the component will fall back to *rvmrmSoftware*). TRVCamView[93] and TRVCamMultiView[74] use Skia4Delphi only for drawing video frames. All other parts of video viewers are drawn by the standard drawing. GPU is not used, but frame drawing sometimes may be more efficient that in the standard drawing. **Requirements:** • Delphi or C++Builder XE7 or newer
rvmrmAuto	Auto selection: • 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
rvptNone	Unknown command type
rvptString	String
rvptInteger	Integer value
rvptFloat	Floating point value
rvptDateTime	Date
rvptBin	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
rvfPreferTCP	Instructs to try TCP for RTP transport first, if TCP is available as RTSP RTP transport (see TRVFFMpegProperty[201].RTSPTransport)
rvfListen	Instructs to act as a server, listening for an incoming connection (not supported in RVMedia)
rvfFilterSrc	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

rvsf8	integer value, 8 bits
rvsf16	integer value, 16 bits
rvsf32	integer value, 32 bits
rvsfFloat	floating point value, 32 bits
rvsfDouble	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] fmxMRVType;

```
type
  TRVSamplesPerSec = (rvsps8000, rvsps11025, rvsps22050, rvsps44100,
    rvsps48000, rvsps96000, rvsps192000);
```

Value	Meaning
rvsps8000	8000 Hz
rvsps11025	11025 Hz
rvsps22050	22050 Hz
rvsps44100	44100 Hz
rvsps96000	96000 Hz
rvsps192000	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] fmxMRVType;

```
type
  TRVSessionKey = type Cardinal;
```

This type is used for properties:

- TRVCamReceiver.SessionKey[126]
- TRVCamSender.SessionKey[151]
- TRVMediaServer.SessionKey[170]

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 TRVCamSender[139] and TRVCamReceiver[120] are connected, if TCP/HTTP protocols are used.

Unit [VCL and LCL] MRVType;

Unit [FMX] fmxMRVType;

```
type
  TRVTCPConnectionType = (rvtcpSenderToReceiver, rvtcpReceiverToSender);
```

Value	Meaning
rvtcpSenderToReceiver	A sender initiates a connection to a receiver
rvtcpReceiverToSender	A receiver initiates a connection to a sender

See also:

- TRVCamSender.TCPConnectionType[153]
- TRVCamReceiver.TCPConnectionType[128]

7.32 TRVVideoCodec

Specifies a codec used to encode video TRVCamRecorder[85] 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
    {$IFDEF MRVCODEC_MSMPEG4v3}
    rvvcMSMPEG4v3
    {$ENDIF}
    rvvcFLV1, rvvcRV1, rvvcRV2, rvvcHEVC);
```

Value	Format name (see GetVideoCodecName[225])	Recommended file extensions (see GetVideoFileExts[224])
rvvcMPEG1	MPEG-1	mpeg
rvvcMPEG2	MPEG-2	mpeg
rvvcMPEG4	MPEG-4	mp4 3gp
rvvcH263P	H.263+	avi
rvvcH264	H.264	mp4 avi mov
rvvcMJPEG	MJPEG	mjpeg
rvvcWMV1	Windows Media Video 7	avi
rvvcWMV2	Windows Media Video 8	avi
rvvcMSMPEG4v3	MPEG-4 part 2 Microsoft variant v.3	asf avi
rvvcFLV1	FLV (Sorenson Spark)	flv
rvvcRV1	RV10 (RealVideo 1)	rm
rvvcRV2	RV20 (RealVideo 2)	rm
*rvvcHEVC**	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

rvDefault	The default video resolution is used, no scaling occurs
rv160_120	160 x 120
rv320_240	320 x 240
rv640_480	640 x 480
rv1024_768	1024 x 768
rv1280_720	1280 x 720
rv1900_1280	1900 x 1280

This is a type of the following properties:

- TRVCamera[42].VideoResolution[61]
- TRVCamSender[139].VideoResolution[154]

Index