INTELLIO VIDEO SYSTEM 5

Intellio Video SDK

Table of Contents

1 Introduction 5

1.1 Components 5

1.2 Features 5

1.3 System Requirements 6

2 Getting Started 6

2.1 Installing the Intellio Video SDK 6

3 Using the Intellio Video SDK 7

3.1 Connecting to a Site 7

3.2 Getting Cameras 7

3.3 Get a VideoSource 7

3.4 PTZ-Control and Presets 8

3.5 Getting Still Image 8

3.6 Live View 8

3.7 Simple Video Playback 9

3.8 Advanced Video Playback 9

3.9 Expert Video Playback with Events 9

3.10 Getting the Image of an Event 10

3.11 Export Video to a File 10

3.12 Using Devices 10

3.13 Alarm Detectors and Live Events 10

3.14 Signalling a TCPEventReceiver Event 11

3.15 Getting Views with ViewManager 11

3.16 Controlling Monitor Wall 11

3.17 ANPR Groups and List 11

3.18 Using Visual Component 11

4 Intellio Video SDK C# Sample Application 12

5 DownloadEventsImage 14

6 API Reference 15

6.1 Site 15

6.1.1 ISiteConnector 15

6.1.2 ISiteConnection 15

6.2 Device 16

6.2.1 IDeviceManager 16

6.2.2 IDevices 16

6.2.3 IDevice 16

6.3 VideoSource 17

6.3.1 IVideoSourceManager 17
6.3.2 IVideoSources 17
6.3.3 IVideoSource 17
6.3.4 IVideoSourceMonitoringCallback 18
6.3.5 IVideoSourceMonitoringCallback2 18
6.3.6 IVideoSourcePlayback 19
6.3.7 IVideoSourcePlaybackCallback 19
6.3.8 IVideoFrame 19
6.3.9 IImage 19
6.3.10 ICameraPTZControl 20
6.4 Playback 21
6.4.1 PlaybackManager 21
6.4.2 IPlaybackVideoCallback 22
6.4.3 IPlaybackVideoFrame 22
6.5 Export 22
6.5.1 IExportManager 22
6.5.2 IExportManagerCallback 23
6.6 Alarm 23
6.6.1 IAlarmManager 23
6.6.2 IAlarmManagerEventCallback 23
6.6.3 IAlarmManagerEvents_Event 23
6.6.4 IDetectors 23
6.6.5 IDetector 24
6.6.6 IEvents 24
6.6.7 IEvent 24
6.6.8 IEventInfo 25
6.6.9 IEventInfoLicensePlate 25
6.6.10 IEventInfoLicensePlate2 25
6.6.11 IEventInfoLicensePlate3 26
6.6.12 IEventInfoFaceTracker 26
6.6.13 IEventReceiver 27
6.7 ViewManager 27
6.8 MonitorModuleManager 27
6.9 ANPR 28
6.9.1 IANPRManager 28
6.9.2 IANPRGroups 28
6.9.3 IANPRList 28
6.10 Visual component 29
6.10.1 AxIntellioVideoMonitor 29

7 Enums and Structs 30

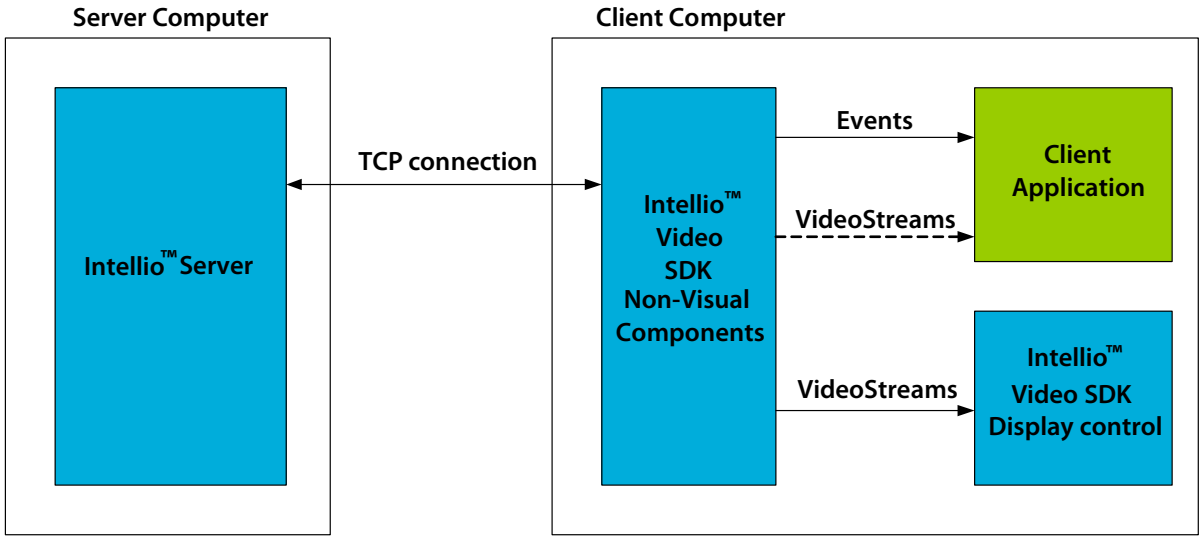
7.1 ANPRColorEnum 30
7.2 ANPRDirectionEnum 30
7.3 ANPRGroupItem 30
7.4 ANPRItem 31
7.5 BandwidthEnum 31
7.6 DetectorClassType 31
7.7 DetectorLocationType 31
7.8 DetectorState 31
7.9 DeviceState 32
7.10 EventInfoTypeEnum 32
7.11 EventState 32
7.12 EventType 33
7.13 EventTypeEnum2 33
7.14 ExportManagerResultEnum 33
7.15 ExportManagerStateEnum 34
7.16 FaceTrackerGenderEnum 34
7.17 MMRData 34
7.18 MMRViewType 35
7.19 PixelFormat 35
7.20 PlaybackSpeedEnum 35
7.21 VideoFrameLossType 36
7.22 VideoFrameType 36

1 Introduction

The Intellio Video Software Development Kit enables the developer to integrate the Intellio System into their application.

While it's intended to be used by software developers with experience and knowledge of ActiveX Controls and COM objects; the SDK also comes with various simple and advanced example.

Typical usage architecture is the following:



1.1 Components

The Intellio Video Software Development Kit is based on Microsoft COM technology and it consists of some visual ActiveX controls and non-visual COM components that provide the access to Intellio Video Server and its resources.

1.2 Features

- Connecting to an Intellio Server
- Getting information about and modify Devices *(e.g. cameras)*
- Getting live video footages, still images
- Getting Event informations
- Exporting VideoSources to AVI
- Playback video with stepping and speed controls
- Modifying Views
- Getting Monitoring users and setting their Views
- Modifying ANPR List and Groups
- Interface to display the live video footage
- Sample applications

1.3 System Requirements

For End-Users:

Operating System: Windows 7 or higher

.NET Framework 3.5 or higher

For Developers:

Operating System: Windows 7 or higher

Visual Studio Community 2012 or later

.NET Framework 4.5 or higher

The C# sample application uses ***Dispatcher.InvokeAsync*** which requires .NET Framework 4.5 at least. Technically the minimally required version would be .NET Framework 3.5.

2 Getting Started

2.1 Installing the Intellio Video SDK

The ***IntellioVideoSDKRedistributable.exe*** file is required for the end-user. It's recommended to bundle with your application.

For developers, run the ***IntellioVideoSDKSetup.exe*** file to install the SDK. This also includes documentation and sample application on top of what the *IntellioVideoSDKRedistributable* offers.

3 Using the Intellio Video SDK

The following components have global unique identifiers:

DetectorID for Detector
DeviceID for Device
SiteID for the Site
VideoSourceID for VideoSource

3.1 Connecting to a Site

The ***SiteConnector*** is the first component required by a client application. The client can use the ***Connect*** method to connect to an Intellio Site and get the ***SiteConnection*** which will return all available informations.

The ***SiteConnection*** represents a connected Site. ***DeviceManager***, ***VideoSourceManager*** and other managers of that Site can be reached from here.

Instantiate a ***SiteConnector*** object then store the result of ***ISiteConnector.Connect(string*** *IP*, ***int*** *TCPPort*, ***string*** *LoginName*, ***string*** *Password)* into a ***SiteConnection*** variable.

```
ISiteConnector m_siteConnector = new SiteConnector();
m_siteConnection = m_siteConnector.Connect(IP, TCPPort, LoginName, Password);
```

To disconnect from a Site, use your ***SiteConnection*** variable and call ***Disconnect***.

```
m_siteConnection.Disconnect();
```

3.2 Getting Cameras

A ***VideoSource*** is a video input in a ***Device***. This is equivalent to ***Cameras*** in the Intellio Video Client.

VideoSourceManager is responsible for managing the ***VideoSources*** of the Site's own Devices.

You can use a ***VideoSource*** to get a still image, start video monitoring, video playback, export to AVI or control their PTZ.

3.3 Get a VideoSource

To get a single VideoSource, use ***VideoSourceManager.VideoSources.VideoSource[int*** *index of VideoSource]* or ***VideoSourceManager.VideoSources.VideoSourceByID[Guid*** *of VideoSource]*.

VideoSourceManager.VideoSources.Count returns the number of VideoSources.

3.4 PTZ-Control and Presets

PTZControl can be used to get number of presets with ***PresetCount*** and all the presets with ***Presets[int*** *index of preset]* like: ***VideoSource.PTZControl.Presets[int*** *index of Preset].*

PTZ actions will continue until you call the action's method with 0 as parameters: ***Zoom(0)*** and/or ***PanTilt(0,0)***.

Panning and Tilting with ***PanTilt(int*** *PanSpeed****, int*** *TiltSpeed****)****:*

Tilt down:	*PanTilt(**0**, **-100**)*
Tilt up:	*PanTilt(**0**, **100**)*
Pan left:	*PanTilt(-**100**, **0**)*
Pan right:	*PanTilt(**100**, **0**)*

Zooming with ***Zoom(int*** *Speed****)****:*

Zoom in:	*Zoom(**100**)*
Zoom out:	*Zoom(**-100**)*

Go to Preset by Name:	*int **GotoPreset(string** Name**)***
Set Preset to Name:	*int **SetPreset(string** Name**)***
Clear Preset with Name:	*int **ClearPreset(string** Name**)***

3.5 Getting Still Image

You can retrieve a still image with ***GetStillImage*** method or obtain the entire video stream via ***VideoReceived*** event.

3.6 Live View

Using the ***IVideoSourceMonitoringCallback2*** and a ***VideoSource***, you can start video monitoring. You will handle the incoming frames inside the callback. The handle is returned with ***StartVideoMonitoring2***. Uses ***VideoFrame***.

```
m_videoSource.StartVideoMonitoring2(Callback, UserData)

m_videoSource.StopVideoMonitoring(MonitoringID);
```

3.7 Simple Video Playback

Using the ***IVideoSourcePlaybackCallback*** and a ***VideoSource*** you can start video playback from a given time. Uses ***PlaybackVideoFrame***.

```
m_videoSource.StartVideoPlayback(StartTime, Callback, UserData);

m_videoSource.StopVideoPlayback();
```

You can also make a looping playback from a given time until another time.

```
m_videoSource.StartVideoPlaybackLoop(StartTime, EndTime, Callback, UserData);

m_videoSource.StopVideoPlayback();
```

3.8 Advanced Video Playback

The ***VideoSourcePlayback*** let's you change the playback speed. Uses ***PlaybackVideoFrame***.

Using the ***IVideoSourcePlaybackCallback*** and ***VideoSource****(s):*

```
VideoSourcePlayback playback = m_videoSource.CreateVideoPlayback(this, 0);

m_playback.Speed = (int)PlaybackSpeedEnum.pbs1x;

m_playback.Start(m_startDateTime.Value);

m_playback.Stop();
```

3.9 Expert Video Playback with Events

Using the ***PlaybackManager***,the ***IPlaybackVideoCallback*** and ***PlaybackVideoFrame****,* you can add multiple ***Detectors*** and ***VideoSources***, to playback multiple cameras along with detector events. You can also change the playback speed, bandwidth limit.

PlaybackManager.AddDetector(Detector) to add a ***Detector***. ***PlaybackManager.AddVideoSource(VideoSource)*** to add a ***VideoSource***. The detector events' can be retrieved with ***PlaybackManager.GetEventListForward(DateTime*** *startTime,* ***int*** *count,* ***out Events*** *events)*.

```
m_playbackManager.SetVideoCallback(Callback, UserData);

m_playbackManager.BandwidthInBitPerSec = (int)BandwidthEnum.bwLAN;

// Add Detector, VideoSources, etc.

m_playbackManager.Play(StartTime);

m_playbackManager.Stop();
```

3.10 Getting the Image of an Event

Using the previously explained multiple ***VideoSources*** and ***Detectors*** playback, you can use ***Locate(Time)*** to seek to target time to be able to get an event's single image.

3.11 Export Video to a File

The ***ExportManager*** let's you export multiple video sources to an AVI file. You will use the ***IExportManagerCallback*** to handle the status of the exporting.

```
m_exportManager.AddVideoSource(m_videoSource);

result = m_exportManager.ExportToAviFileAsync(StartTime, EndTime, Path);
```

A ***VideoSource*** can be also exported without the ***ExportManager***.

```
result = m_videoSource.ExportToAviFileAsync(StartTime, EndTime, Path, Callback);
```

3.12 Using Devices

The ***DeviceManager*** contains all ***Device****s* available and responsible for managing them.

Represents a physical device on the network. It has a ***MAC address***, ***DisplayName***, ***Description*** and it can control multiple ***VideoSources***.

3.13 Alarm Detectors and Live Events

The ***AlarmManager*** contains all ***Alarms*** available and it's responsible for managing the Site's own ***Detectors*** *(system and on-board).* A client can obtain alarm ***Events*** via ***EventReceived*** event.

Subscribe to AlarmManager's ***OnEventReceieved*** event to handle ***IEvents***.

Events with a not null ***EventInfo*** have an ***Event.EventInfo.EventInfoType*** with a ***EventInfoTypeEnum*** value.

To get License Plate informations, cast ***Event.EventInfo*** as ***EventInfoLicensePlate*** for older SDK and ***EventInfoLicensePlate2*** for newer SDK.

To get MMR Data, first cast as ***EventInfoLicensePlate3*** and then call ***GetMMRData*** from there.

```
IEventInfoLicensePlate3 lprInfo3 = lprInfo2 as IEventInfoLicensePlate3;

lprInfo3.GetMMRData(out MMRData mmrData);
```

To get face tracker informations, cast ***Event.EventInfo*** as ***EventInfoFaceTracker***.

3.14 Signalling a TCPEventReceiver Event

Use ***AlarmManager*** to get all available detectors. To filter out non TCP Event Receiver detectors, try to cast it as ***EventReceiver*** and check for nullability.

SendEvent(int EventCode***, EventTypeEnum2*** EventType***)***

3.15 Getting Views with ViewManager

The ***ViewManager*** contains all ***Views***. Use ***GetViewName(Index, out ViewID)*** *for Views,* ***GetViewCount*** *for number of Views.*

```
m_viewManager.GetViewName(i, out m_viewID)
```

3.16 Controlling Monitor Wall

The ***MonitorManager*** contains the currently logged in users using a monitoring shortcut. The ***ViewManager*** contains the ***Views***.

You can change their current views with a compatible view. To change a monitoring user's 2nd monitor, use a multi monitor view.

```
m_monitorModuleManager.SetView(ref m_monitorID, ref m_viewID);
```

To get a ***Monitor****'s* name, use:

```
m_monitorModuleManager.GetMonitorName(Index, out Guid MonitorID);
```

3.17 ANPR Groups and List

Use ***ANPRManager*** to get all License Plate groups and list.

- 32 groups maximum.
- ANPR Group Colors are in low endian.
- Refer to the API details for more information.

3.18 Using Visual Component

Drag and drop ***IntellioVisualVideoSDK.ocx*** from your file system to your Visual Studio Toolbox, then drag and drop ***IntellioVideoMonitor Control*** component from the Toolbox to your form.

To start monitoring, first call `axIntellioVideoMonitor1.SetVideoSource(m_videoSource)` where `m_videoSource` is a VideoSource from the `VideoSourceManager` for example `m_videoSourceManager.VideoSources.VideoSource[0]` and then call `m_axIntellioVideoMonitor1.Start();`

4Intellio Video SDK C# Sample Application

The C# sample application shows various functionalities of the SDK.

MainForm.cs

Connecting to a Site
Uses ISiteConnector and ISiteConnection

Visual.cs

Demonstrates IntellioVideoMonitor.ocx
Uses IVideoSourceManager and VideoSource

NonVisual.cs

Demonstrates IVideoSourceMonitoringCallback2
Uses IVideoSourceManager, VideoSource and VideoFrame

SimplePlayback.cs

Demonstrates IVideoSourcePlaybackCallback
Uses IVideoSourceManager, VideoSource and PlaybackVideoFrame

CreatePlayback.cs

Demonstrates CreateVideoPlayback and Loop
Uses IVideoSourceManager, IVideoSourcePlaybackCallback, VideoSourcePlayback and PlaybackVideoFrame

ExpertPlayback.cs

Demonstrates IPlaybackVideoCallback
Uses IVideoSourceManager, PlaybackManager and PlaybackVideoFrame

Events.cs

Demonstrates Events
Uses AlarmManager and IEvent

ANPRList.cs

Demonstrates ANPR List
Uses IANPRManager and ANPRItem

ANPRGroups.cs

Demonstrates ANPR Groups
Uses IANPRManager and ANPRGroupItem

Export.cs

Demonstrates Exporting VideoSources
Uses IVideoSourceManager, IExportManager and IExportManagerCallback

Monitor.cs

Demonstrates Monitor module
Uses ViewManager and MonitorModuleManager

EventReceiver.cs

Demonstrates TCPEventReceiver
Uses IAlarmManager and IeventReceiver

5DownloadEventsImage

- Connects to a ***Site***
- Iterates through ***Detectors*** and gets detectorName (Detector.DisplayName)
- Gets the ***VideoSourceID*** of the Detector
- ***Adds Detector*** to PlaybackManager
- ***Adds VideoSource*** to PlaybackManager
- ***Gets the next 100 events*** from startTime (PlaybackManager.GetEventListForward)
- Increases startTime until endTime reached
- ***Saves images*** to the directory where the program started (VideoFrame.GetStillImage)
- Disconnects from the Site

Format
<siteIp> <sitePort> <username> <password> <detectorName> <startTime> <endTime>

Example
192.168.1.1 53540 admin adminpw "Motion detector" "2019-02-20 14:00:00" "2019-02-20 14:10:10"

6 API Reference

6.1 Site

6.1.1 ISiteConnector

```
SiteConnection Connect(string IP, int TCPPort, string LoginName, string Password);
```

Connects to a Site

Returns a SiteConnection

6.1.2 ISiteConnection

```
void Disconnect();
```

Disconnects from a Site

```
Guid SiteID { get; }
```

Gets the ID of the Site

```
string LoginName { get; }
```

Gets the Login name

```
AlarmManager AlarmManager { get; }
```

Gets the AlarmManager

```
ANPRManager ANPRManager { get; }
```

Gets the ANPRManager

```
DeviceManager DeviceManager { get; }
```

Gets the DeviceManager

```
ExportManager ExportManager { get; }
```

Gets the ExportManager

```
MonitorModuleManager MonitorModuleManager { get; }
```

Gets the MonitorModuleManager

```
PlaybackManager PlaybackManager { get; }
```

Gets the PlaybackManager

```
VideoSourceManager VideoSourceManager { get; }
```

Gets the VideoSourceManager

```
ViewManager ViewManager { get; }
```

Gets the ViewManager

6.2 Device

6.2.1 IDeviceManager

```
int RefreshDevices();
```
Refreshes Devices
Returns 1 when successful; 0 otherwise

```
Devices Devices { get; }
```
Gets the Devices

6.2.2 IDevices

```
Device get_Device(int Index);
```
Returns the Device at Index
Same as `Devices[int Index]`

```
Device get_DeviceByID(ref Guid DeviceID);
```
Returns the Device with DeviceID
Same as `DevicesByID[ref Guid DeviceID]`

```
int Count { get; }
```
Gets the number of Devices

6.2.3 IDevice

```
Guid DeviceID { get; }
```
Gets the ID of the Device

```
string Description { get; }
```
Gets the description of the Device

```
Detectors Detectors { get; }
```
Gets the Detectors of the Device

```
DeviceState DeviceState { get; }
```
Gets the state of the Device

```
string DisplayName { get; }
```
Gets the display name of the Device

```
bool Enabled { get; }
```
Gets the availability of the Device

```
string MACAddress { get; }
```
Gets the MAC address of the Device

```
VideoSources VideoSources { get; }
```
Gets the VideoSources of the Device

6.3 VideoSource

6.3.1 IVideoSourceManager

```
VideoSources VideoSources { get; }
```
Gets VideoSources

6.3.2 IVideoSources

```
VideoSource get_VideoSource(int Index);
```
Returns the VideoSource at Index
Same as `VideoSource[int Index]`

```
VideoSource get_VideoSourceByID(ref Guid VideoSourceID)
```
Returns the VideoSource with VideoSourceID
Same as `VideoSource[ref Guid VideoSourceID]`

```
int Count { get; }
```
Gets the number of VideoSources

6.3.3 IVideoSource

```
int StartVideoMonitoring2(IVideoSourceMonitoringCallback2 Callback, int UserData);
```
Starts video monitoring
Returns the handle
Callback used to handle incoming frames
UserData can be used to distinguish

```
void StartVideoPlayback(DateTime StartTime, IVideoSourcePlaybackCallback Callback, int UserData);
```
Starts video playback from StartTime
Callback used to handle incoming frames
UserData can be used to distinguish

```
void StartVideoPlaybackLoop(DateTime StartTime, DateTime EndTime, IVideoSourcePlaybackCallback Callback, int UserData);
```
Starts video playback loop from StartTime to EndTime
Callback used to handle incoming frames
UserData can be used to distinguish

```
void StopVideoMonitoring(int MonitoringID);
```
Stops video monitoring with MonitoringID

```
void StopVideoPlayback();
```
Stops video playback

```
VideoSourcePlayback CreateVideoPlayback(IVideoSourcePlaybackCallback Callback,
int UserData);
```
Creates a video source playback
Returns a VideoSourcePlayback
Callback used to handle incoming frames
UserData can be used to distinguish

```
ExportManagerResultEnum ExportToAviFileAsync(DateTime StartTime, DateTime
EndTime, string TargetFileName, IExportManagerCallback Callback);
```
Exports to AVI file asynchronously from StartTime to EndTime to file TargetFileName
Returns an ExportManagerResultEnum
Callback used to handle ExportManagerStateEnum and ExportManagerResultEnum

```
Guid DeviceID { get; }
```
Gets the ID of the Device

```
Guid VideoSourceID { get; }
```
Gets the ID of the VideoSource

```
string Description { get; }
```
Gets the description

```
Device Device { get; }
```
Gets the Device

```
string DisplayName { get; }
```
Gets the display name

```
bool Enabled { get; }
```
Gets the availiability

```
VideoFrame GetStillImage();
```
Returns a still image

```
CameraPTZControl PTZControl { get; }
```
Gets the CameraPTZControl

6.3.4 IVideoSourceMonitoringCallback

```
void OnVideoReceived(VideoFrame VideoFrame);
```
Event callback for OnVideoReceived

6.3.5 IVideoSourceMonitoringCallback2

```
void OnVideoReceived(VideoFrame VideoFrame, int UserData);
```
Event callback for OnVideoReceived
UserData can be used to distinguish

6.3.6 IVideoSourcePlayback

```
void Start(DateTime StartTime);
```
Starts playback from StartTime

```
void Stop();
```
Stops playback

```
bool Playing { get; }
```
Gets the state of playing

```
float Speed { get; set; }
```
Gets or sets the speed of playback

```
DateTime Time { get; }
```
Gets the time of playback

6.3.7 IVideoSourcePlaybackCallback

```
void OnPlaybackVideoReceived(PlaybackVideoFrame VideoFrame, int UserData);
```
Event callback for OnPlaybackVideoReceived
UserData can be used to distinguish

6.3.8 IVideoFrame

```
uint SaveToBitmap();
```
Saves the bitmap decoded from the frame
Returns the handle for the image

```
DateTime FrameTime { get; }
```
Gets the frame time of the VideoFrame

```
int FrameType { get; }
```
Gets the frame type of the VideoFrame

```
Image GetImage(int PixelFmt);
```
Returns an Image in PixelFmt pixel format

```
int LossType { get; }
```
Gets the loss type of the VideoFrame

```
VideoSource VideoSource { get; }
```
Gets the VideoSource of the VideoFrame

6.3.9 IImage

```
IntPtr get_ScanLine(int Y);
```
Returns a horizontal row of the Image at Y
Same as `ScanLine[int Y]`

```
IntPtr Ptr { get; }
```
Gets the pointer of the Image

```
int PixelFormat { get; }
```
Gets the pixel format of the Image

```
int ScanLineSize { get; }
```
Gets the size of scan line

```
int Width { get; }
```
Gets the width of the Image

```
int Height { get; }
```
Gets the height of the Image

6.3.10 ICameraPTZControl

```
void PanTilt(int PanSpeed, int TiltSpeed);
```

Tilt down	PanTilt(0, -100)
Tilt up	PanTilt(0, 100)
Pan left	PanTilt(-100, 0)
Pan right	PanTilt(100, 0)

```
void Zoom(int Speed);
```

Zoom in	Zoom(100)
Zoom out	Zoom(-100)

```
int GotoPreset(string Name);
```
Goes to Preset by Name
Returns 0 when successful

```
int SetPreset(string Name);
```
Sets Preset to Name
Returns 0 when successful

```
int ClearPreset(string Name);
```
Clear Preset with Name
Returns 0 when successful

```
int PresetCount { get; }
```
Gets the number of Presets

6.4 Playback

6.4.1 PlaybackManager

```
void Play(DateTime StartTime);
```
Plays playback from StartTime

```
void Stop();
```
Stops playback

```
void AddVideoSource(VideoSource VideoSource);
```
Adds the VideoSource to the PlaybackManager

```
void RemoveVideoSource(VideoSource VideoSource);
```
Removes the VideoSource from the PlaybackManager

```
void Locate(DateTime TargetTime);
```
Locates to TargetTime

```
void StepForward();
```
Steps one frame forward

```
void StepBackward();
```
Steps one frame backward

```
void AddDetector(IDetector Detector);
```
Adds Detector to the PlaybackManager

```
void RemoveDetector(IDetector Detector);
```
Removes Detector from the PlaybackManager

```
void GetEventListForward(DateTime StartTime, int EventCount, out Events Events);
```
Returns EventCount number of events from StartTime to Events in forwards
Events as out parameter will store the Events

```
void GetEventListBackward(DateTime StartTime, int EventCount, out Events Events);
```
Returns EventCount number of events from StartTime to Events in backwards
Events as out parameter will store the Events

```
void SetVideoCallback(IPlaybackVideoCallback Callback, int UserData);
```
Sets video callback
UserData can be used to distinguish

```
int BandwidthInBitPerSec { get; set; }
```
Gets or sets the bandwidth in bit per sec

```
PlaybackVideoFrame GetFrameByTime(Guid ACameraID, DateTime ATime);
```
Returns a frame of ACameraID from ATime

```
DateTime Time { get; }
```
Gets the time of playback

```
PlaybackSpeedEnum PlaybackSpeed { get; set; }
```
Gets or sets the speed of playback

```
bool Playing { get; }
```
Gets the state of playing

6.4.2 IPlaybackVideoCallback

```
void OnPlaybackVideoReceived(PlaybackVideoFrame VideoFrame, int UserData);
```
Event callback for OnPlaybackVideoReceived
UserData can be used to distinguish

6.4.3 IPlaybackVideoFrame

```
bool LossOfSignal { get; }
```
Gets the state of being a loss of signal

```
DateTime Time { get; }
```
Gets the time of PlaybackVideoFrame

```
VideoSource VideoSource { get; }
```
Gets the VideoSource of PlaybackVideoFrame

```
VideoFrame VideoFrame { get; }
```
Gets the VideoFrame of PlaybackVideoFrame

6.5 Export

6.5.1 IExportManager

```
void AddVideoSource(VideoSource VideoSource);
```
Adds VideoSource to the ExportManager

```
void Cancel();
```
Cancels the export

```
void ClearSources();
```
Clears VideoSources

```
ExportManagerResultEnum ExportToAviFile(DateTime StartTime, DateTime EndTime,
string TargetFileName);
```
Export to AVI file from StartTime to EndTime to file TargetFileName
Returns an ExportManagerResultEnum

```
ExportManagerResultEnum ExportToAviFileAsync(DateTime StartTime, DateTime
EndTime, string TargetFileName);
```
Exports to AVI file asynchronously from StartTime to EndTime to file TargetFileName
Returns an ExportManagerResultEnum

```
void SetCallback(IExportManagerCallback Callback);
```
Sets callback

```
bool ExportInProgress { get; }
```
Gets the progress of the exporting

6.5.2 IExportManagerCallback

```
void OnStateChanged(ExportManagerStateEnum State, ExportManagerResultEnum ErrorCode);
```
Event callback for OnStateChanged

```
void OnProgressChanged(Guid SourceID, DateTime CurrentTime, float ProgressPercent);
```
Event callback for OnProgressChanged

6.6 Alarm

6.6.1 IAlarmManager

```
int RefreshDetectors();
```
Refreshes detectors
Returns 1 when successful; 0 otherwise

```
void SetEventCallback(IAlarmManagerEventCallback Callback, int UserData);
```
Sets event callback to Callback
UserData can be used to distinguish

```
Detectors Detectors { get; }
```
Gets the Detectors

6.6.2 IAlarmManagerEventCallback

```
void OnEventReceived(Event Event, int UserData);
```
Event callback for OnEventReceived
UserData can be used to distinguish

6.6.3 IAlarmManagerEvents_Event

```
event IAlarmManagerEvents_OnEventReceivedEventHandler OnEventReceived;
```
Event handler for OnEventReceived

6.6.4 IDetectors

```
IDetector get_Detector(int Index);
```
Returns Detector at Index
Same as `Detector[int Index]`

```
IDetector get_DetectorByID(ref Guid DetectorID);
```
Returns Detector with DetectorID
Same as `DetectorByID[ref Guid DetectorID]`

```
int Count { get; }
```
Gets the number of Detectors

6.6.5 IDetector

```
Guid DetectorID { get; }
```
Gets the ID of the Detector

```
Guid DeviceID { get; }
```
Gets the ID of the Device

```
Guid VideoSourceID { get; }
```
Gets the ID of the VideoSource

```
string Description { get; }
```
Gets the description

```
string DetectorClass { get; }
```
Gets the Detector class

```
int DetectorClassType { get; }
```
Gets the Detector class type

```
int DetectorState { get; }
```
Gets the Detector state

```
Device Device { get; }
```
Gets the Device

```
string DisplayName { get; }
```
Gets the display name

```
bool Enabled { get; }
```
Gets the availability

```
bool Hidden { get; }
```
Gets the visibility

```
int Location { get; }
```
Gets the location

6.6.6 IEvents

```
Event get_Event(int Index);
```
Returns Event at Index

Same as `Events[int Index]`

```
int Count { get; }
```
Gets the number of Events

6.6.7 IEvent

```
Guid SourceID { get; }
```
Gets the ID of the Source

```
string SourceName { get; }
```
Gets the name of the Source

```
IDetector Detector { get; }
```
Gets the Detector

```
int EventCode { get; }
```
Gets the code of the event

```
EventInfo EventInfo { get; }
```
Gets the EventInfo

```
string EventMessage { get; }
```
Gets the message of the event

```
int EventState { get; }
```
Gets the EventState

```
DateTime EventTime { get; }
```
Gets the time of the event

```
int EventType { get; }
```
Gets the type of the event

6.6.8 IEventInfo

```
EventInfoTypeEnum EventInfoType { get; }
```
Gets the EventInfoType

6.6.9 IEventInfoLicensePlate

```
EventInfoTypeEnum EventInfoType { get; }
```
Gets the EventInfoType of the event

```
int FontColor { get; }
```
Gets the font color

```
int BackgroundColor { get; }
```
Gets the background color

```
string CountryCode { get; }
```
Gets the country code

```
string CountrySubCode { get; }
```
Gets the country sub code

```
string LicensePlate { get; }
```
Gets the license plate

6.6.10 IEventInfoLicensePlate2

```
ANPRDirectionEnum Direction { get; }
```
Gets the Direction detected

```
EventInfoTypeEnum EventInfoType { get; }
```
Gets the EventInfoType of the event

```
int FontColor { get; }
```
Gets the font color

```
int BackgroundColor { get; }
```
Gets the background color

```
string CountryCode { get; }
```
Gets the country code

```
string CountrySubCode { get; }
```
Gets the country sub code

```
string LicensePlate { get; }
```
Gets the License Plate

6.6.11 IEventInfoLicensePlate3

```
ANPRDirectionEnum Direction { get; }
```
Gets the Direction detected

```
EventInfoTypeEnum EventInfoType { get; }
```
Gets the EventInfoType of the event

```
MMRData GetMMRData(out MMRData mmrData)
```
Gets the MMRData

```
int FontColor { get; }
```
Gets the font color

```
int BackgroundColor { get; }
```
Gets the background color

```
string CountryCode { get; }
```
Gets the country code

```
string CountrySubCode { get; }
```
Gets the country sub code

```
string LicensePlate { get; }
```
Gets the License Plate

6.6.12 IEventInfoFaceTracker

```
int TrackID { get; }
```
Gets the Face Tracker ID

```
int Age { get; }
```
Gets the age

```
EventInfoTypeEnum EventInfoType { get; }
```
Gets the EventInfoType of the event

```
FaceTrackerGenderEnum Gender { get; }
```
Gets the gender

```
int FrontalTimeInMS { get; }
```
Gets the frontal time in milliseconds

```
int LeftViewTimeInMS { get; }
```
Gets the left view time in milliseconds

```
int RightViewTimeInMS { get; }
```
Gets the right view time in milliseconds

```
int TotalTimeInMS { get; }
```
Gets the total time in milliseconds

6.6.13 IEventReceiver

```
void SendEvent(int EventCode, EventTypeEnum2 EventType);
```
Sends event with EventCode and EventType

6.7 ViewManager

```
void Refresh();
```
Refreshes the ViewManager

```
int GetViewCount();
```
Returns the number of Views

```
string GetViewName(int Index, out Guid ViewID);
```
Returns the name of the View at Index
ViewID as out parameter will store it's Guid

6.8 MonitorModuleManager

```
void Refresh();
```
Refreshes the MonitorModuleManager

```
void SetView(ref Guid MonitorID, ref Guid ViewID);
```
Sets the ViewID to a MonitorID

```
int GetMonitorCount();
```
Returns the number of Monitors

```
string GetMonitorName(int Index, out Guid MonitorID);
```
Returns the name of the Monitor at Index
MonitorID as out parameter will store it's Guid

6.9 ANPR

6.9.1 IANPRManager

```
ANPRGroups ANPRGroups { get; }
```
Gets ANPRGroups

```
ANPRList ANPRList { get; }
```
Gets ANPRList

6.9.2 IANPRGroups

```
ANPRGroupItem GetGroup(int ID);
```
Returns an ANPRGroupItem with ID

```
void SetGroup(ANPRGroupItem Group);
```
Sets Group

```
void Refresh();
```
Refreshes ANPRGroups

```
int FirstID { get; }
```
Gets the FirstID

```
int LastID { get; }
```
Gets the LastID

6.9.3 IANPRList

```
ANPRItem get_Items(int AIndex);
```
Returns the ANPRItem at AIndex
Same as `Items[int]`

```
void Add(ANPRItem AItem);
```
Adds an ANPRItem to ANPRList

```
void Delete(string ANumberPlate);
```
Deletes an ANPRItem with ANumberPlate

```
void DeleteColorANPR(string ANumberPlate, ANPRColorEnum ATextColor, ANPRColorEnum
ABackGroundColor);
```
Deletes an ANPRItem with ANumberPlate, ATextColor and ABackGroundColor

```
void DeleteFromGroup(string ANumberPlate, int AANPRGroupID);
```
Deletes an ANPRItem with ANumberPlate and AANPRGroupID

```
void DeleteColorANPRFromGroup(string ANumberPlate, ANPRColorEnum ATextColor,
ANPRColorEnum ABackGroundColor, int AANPRGroupID);
```
Deletes an ANPRItem with ANumberPlate, ATextColor, ABackGroundColor

```
void Refresh();
```
Refreshes ANPRList

```
int Count { get; }
```
Gets the number of ANPRList items

6.10 Visual component

6.10.1 AxIntellioVideoMonitor

```
bool ShowFPS { get; set; }
```
Gets or sets the state of showing FPS
FPS will be drawn on top left corner of the stream

```
bool ShowMagnifier { get; set; }
```
Gets or sets the state of showing magnifier
Magnifies the area hovered over with the mouse

```
int SetVideoSource(VideoSource videoSource);
```
Setting the VideoSource
Returns the handle

```
int Start();
```
Start monitoring a VideoSource
Returns the handle

```
void Stop();
```
Stops monitoring

7 Enums and Structs

7.1 ANPRColorEnum

`anpcUnknown = 0,`
Unknown

`anpcBlack = 1,`
Black

`anpcWhite = 2,`
White

`anpcRed = 3,`
Red

`anpcGreen = 4,`
Green

`anpcBlue = 5,`
Blue

`anpcYellow = 6`
Yellow

7.2 ANPRDirectionEnum

`anpdUnknown = 0,`
Unknown

`anpdApproaching = 1,`
Approaching

`anpdMovingAway = 2`
Moving away

7.3 ANPRGroupItem

```
int ID;
```
ID of ANPRGroupItem

```
ANPRColorEnum Color;
```
Color of ANPRGroupItem

```
string Description;
```
Description of ANPRGroupItem

```
string Name;
```
Name of ANPRGroupItem

7.4 ANPRItem

```
int ANPRGroupID;
```
ANPRGoupID of ANPRItem

```
string Description;
```
Description of ANPRItem

```
short Enabled;
```
Enabled when 0, disabled if anything else.

```
string NumberPlate;
```
Number plate of ANPRItem

```
ANPRColorEnum TextColor;
```
Text color of ANPRItem

```
ANPRColorEnum BackgroundColor;
```
Background color of ANPRItem

7.5 BandwidthEnum

```
bwLAN = -1,
```
LAN

```
bwAuto = 0
```
Auto

7.6 DetectorClassType

```
ctSystem = 0,
```
System

```
ctOnBoard = 1,
```
On board

```
ctOnBoardCamera = 2
```
On board camera

7.7 DetectorLocationType

```
ltSystem = 0,
```
System

```
ltOnBoard = 1
```
On board

7.8 DetectorState

```
dsInitializing = 0,
```
Initializing

`dsOnline = 1,`
Online

`dsSignal = 2,`
Signal

`dsOffline = 3,`
Offline

`dsError = 4,`
Error

`dsInactive = 5`
Inactive

7.9 DeviceState

`dsDisconnected = 0,`
Disconnected

`dsConnecting = 1,`
Connecting

`dsConnected = 2`
Connected

`vftLossOfSignal = 1`
Loss of signal

7.10 EventInfoTypeEnum

`eitUnknown = 0,`
Unknown EventInfoType

`eitLicensePlate = 1,`
License Plate

`eitLoop = 2,`
Loop

`eitInputCounter = 3,`
Input Counter

`eitFaceTracker = 4,`
Face Tracker

7.11 EventState

`esInitBegin = 0,`
Start initializing

`esInitCompleted = 1,`

Initialized

`esInitFailed = 2,`
Initialization failed

`esRestored = 3,`
Restored

`esSignal = 4,`
Alarm

`esSimpleEvent = 5,`
Event

`esInactive = 6,`
Inactive

`esActive = 7`
Active

7.12 EventType

`etSignal = 0,`
Signal

`etAlarm = 1`
Alarm

7.13 EventTypeEnum2

`eteSignal = 0,`
Signal

`eteRestore = 1,`
Restore

`eteEvent = 2`
Event

7.14 ExportManagerResultEnum

`emrSuccess = 0,`
Successfully exported

`emrNoPermission = 1,`
You do not have right to export video

`emrCreateFileFailed = 2,`
Cannot create the specified file

`emrFail = 3,`
Exporting failed

`emrNoValidCamera = 4,`
No VideoSource(s) selected

`emrTooManyCameras = 5,`
Too many VideoSources selected (4 allowed)

`emrExportInProgress = 6,`
Export is already in progress

`emrInvalidInterval = 7`
Invalid interval selected

7.15 ExportManagerStateEnum

`emsExportStarted = 0,`
Export started

`emsExportEnded = 1,`
Export ended

`emsExportCancelled = 2,`
Export cancelled

`emsCameraExportStarted = 3,`
Camera export started

`emsCameraExportEnded = 4,`
Camera export ended

`emsError = 5`
Error

7.16 FaceTrackerGenderEnum

`ftgUnknown = 0,`
Unknown

`ftgMale = 1,`
Male

`ftgFemale = 2`
Female

7.17 MMRData

`vtNone = 0,`
None
`vtFrontal = 1,`
Frontal
`vtRear = 2`
Rear

7.18 MMRViewType

```
string Category;
```
Category

```
string Color;
```
Color

```
string Make;
```
Make

```
string Model;
```
Model

```
enum MMRViewType View;
```
MMRViewType

7.19 PixelFormat

```
pfGray = 0,
```
Gray

```
pfRGB32 = 1,
```
RGB 32-bit

```
pfYUV = 2,
```
YUV

```
pfYUYV = 3,
```
YUYV

```
pfUYVY = 4
```
UYVY

7.20 PlaybackSpeedEnum

```
pbs1x = 1,
```
Normal, 1x Speed

```
pbs2x = 2,
```
Double, 2x Speed

```
pbs4x = 4,
```
4x Speed

```
pbs10x = 10,
```
10x Speed

```
pbs20x = 20,
```
20x Speed

`pbs30x = 30`
: 30x Speed

7.21 VideoFrameLossType

`vflLossOfCamera = 0,`
: Loss of Camera

`vflLossOfServer = 1`
: Loss of Server

7.22 VideoFrameType

`vftImage = 0,`
: Image