There was a task to implement the WPF application “Photobooth”. This app should take photographs and video from Canon EOS cameras. Canon provides developers EOS Digital SDK (EDSDK) for this purpose. We used version 2.11. The full list of supported cameras is shown below:

- EOS-1D Mark III
- EOS 40D
- EOS-1Ds Mark III
- EOS DIGITAL REBEL Xsi/450D/ Kiss X2
- EOS DIGITAL REBEL XS/ 1000D/ KISS F
- EOS 50D
- EOS 5D Mark II
- EOS Kiss X3/EOS REBEL T1i /EOS 500D
- EOS 7D

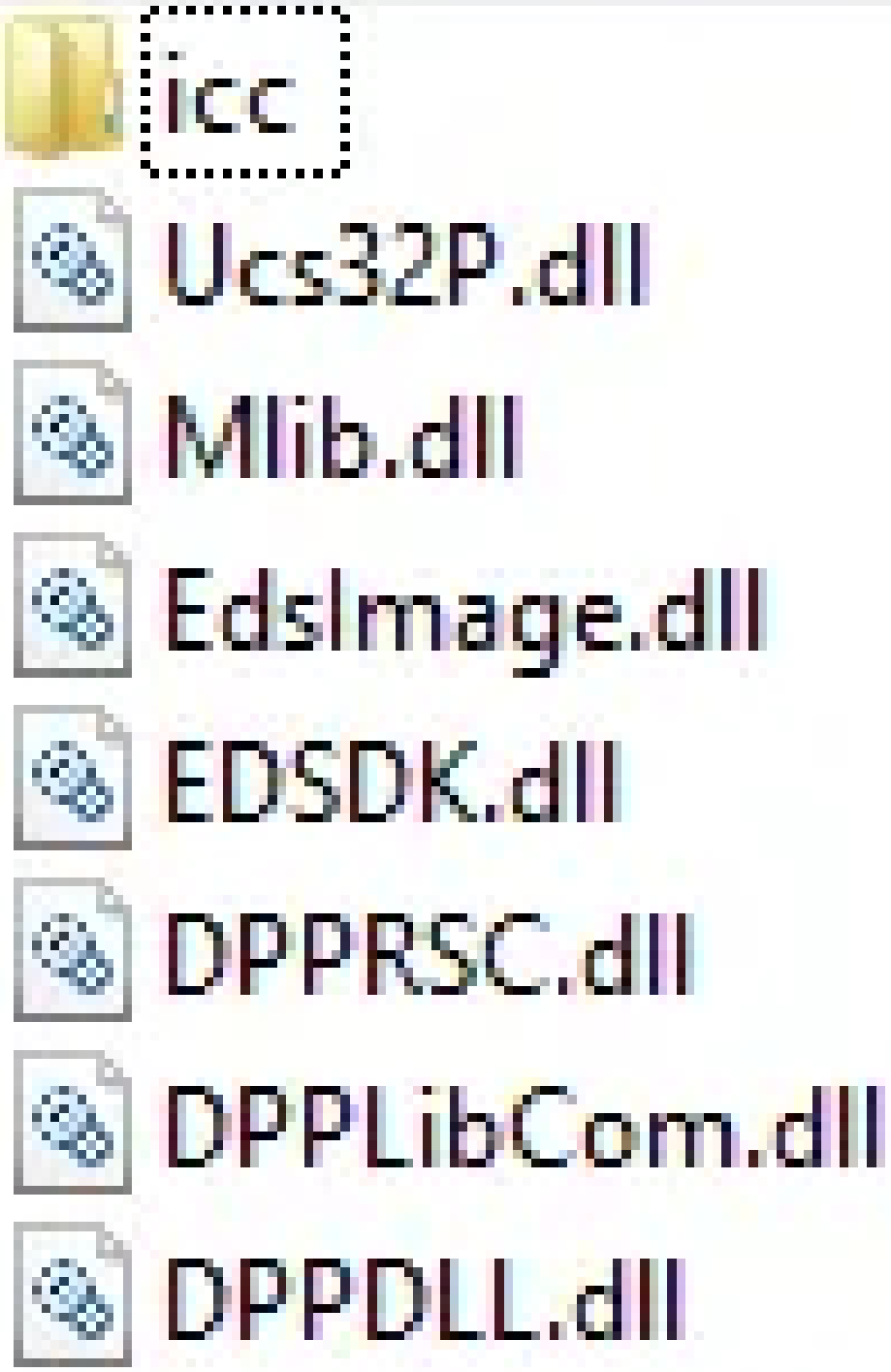
+44 20 7183 5820

info@magora.co.uk
sales@magora.co.uk

- EOS-1D Mark IV
- EOS Kiss X4/EOS REBEL T2i /EOS 550D
- EOS 60D
- EOS Kiss X5/EOS REBEL T3i /EOS 600D
- EOS Kiss X50/EOS REBEL T3 /EOS 1100D
- EOS 5D Mark III
- EOS 1D X
- EOS Kiss X6i/EOS 650D/EOS REBEL T4i

Canon EDSDK is native C/C++ library. This library provides manipulation of a digital camera connected to the host PC, digital images and video created in the camera and downloaded to the PC. Since we used C# and WPF, we found EDSDK .NET wrapper – https://edsdkwrapper.codeplex.com/. This wrapper uses COM Interop. We just added video mode support. EDSDK contains multiple native DLLs:

+44 20 7183 5820
info@magora.co.uk
sales@magora.co.uk



To use EDSDK in your .NET project, put these DLLs and folder **“icc”** into your **Debug** or **Release** folder and add a reference to the **EDSDKWrapper.Framework.dll**. The main class of this wrapper is FrameworkManager located in **EDSDKWrapper.Framework.Managers** namespace. This class implements logic for initialising and terminating of the EDSDK and access to a list of cameras. To access the camera connected to a PC you can use the following code: using EDSDKWrapper.Framework.Managers; using EDSDKWrapper.Framework.Objects; var frameworkManager = new FrameworkManager(); var camera = frameworkManager.GetCameras().FirstOrDefault(); **Camera** is the second important class in EDSDK. It

+44 20 7183 5820
info@magora.co.uk
sales@magora.co.uk

implements access to such camera settings as AE mode, ISO sensitivity, aperture value, shutter speed, white balance, etc. You can get the full list of settings in the EDSDK API reference. Also this class implements logic for live view mode, taking photographs and recording video. All camera settings are wrapped in enumerations in **EDSDKWrapper.Framework.Enums** namespace. To set some settings and start video recording with getting a live image view stream you can use the following code:

```
using EDSDKWrapper.Framework.Managers;
using EDSDKWrapper.Framework.Objects;
using EDSDKWrapper.Framework.Enums;
```

In this code we turn on live view mode and video mode, set transferring of live view and recorded video to the host PC, and add event handler to VideoDownloaded event. Then we start recording for some time and stop after this time is over with resources disposing.
As we have said above, the current wrapper does not implement video support. How can we get it to work? When we set any camera parameter, we call the native **EDSDK function EdsSetPropertyData**. This function has the following signature:

```
EdsError EdsSetPropertyData(
                        EdsBaseRef inRef,
                    EdsPropertyID inPropertyID,
```

Parameters description:

- **inRef** - designate the object for which the properties are to set. Designate either EdsCameraRef or EdsImageRef.
- **inPropertyID** - designate the property ID.
- **inParam** - designate additional property information. Use additional property information if multiple items of information such as picture styles can be set or retrieved for a property. For descriptions of values that can be designated for each property, see the description of inParam for EdsGetPropertyData.
- **inPropertySize** - designate the size of the property data in bytes. The data size of each property can be retrieved by means of EdsGetPropertySize.
- **inPropertyData** - designate the propertydata to set.

The parameter **inPropertyID** defines what operation or camera parameter we want to manipulate. For video recording this parameter equals 0x00000510. Therefore, we added the following line to PropertyId enumeration in wrapper:

```
namespace EDSDKWrapper.Framework.Enums
{
   public enum PropertyId : uint
```


+44 20 7183 5820
info@magora.co.uk
sales@magora.co.uk

```
        ...
        Record                          =                          0x00000510,
        ...
```

Then we created new **enumeration Record** with video recording states:

```
namespace                                                EDSDKWrapper.Framework.Enums
{
   public              enum              Record              :              uint
```

Then we added some properties to **Camera class**:

```
public                                Record                                Record
{
     get   {   return   (Record)   GetUInt32Property(PropertyId.Record);   }
```

Then we defined our delegate:

```
public                static                class                UserDelegates
{
    public delegate void DownloadedVideoHandler(Stream videoStream, string
```

After that, we changed the **objectEventHandler** method and added the new **DownloadVideo** method in **Camera** class:

```
if          (inEvent          ==          (uint)ObjectEvent.DirItemCreated)
{
                                                                DownloadVideo(inRef);
```


+44 20 7183 5820

info@magora.co.uk
sales@magora.co.uk

```
                ReturnValueManager.HandleFunctionReturnValue(returnValue);

                                                uint              videoLength;
```

This method passes a recorded video stream to the **VideoDownloaded** event handler. We can save it to file in this way:

```
private void CameraVideoDownloaded(Stream videoStream, string filename)
{
        using (var fileStream = File.Create(filename))
```

Therefore, we can manipulate the Canon EOS cameras via EDSDK in many ways. You can find the full list of parameters and commands in the EDSDK API.

+44 20 7183 5820

info@magora.co.uk
sales@magora.co.uk