www.bytescout.com

How to capture video and add time stamp in C++ (unmanaged) with ByteScout Screen Capturing SDK

The tutorial shows how to capture video and add time stamp in C++ (unmanaged)

On this page you will learn from code samples for programming in C++ (unmanaged).Writing of the code to capture video and add time stamp in C++ (unmanaged) can be done by developers of any level using ByteScout Screen Capturing SDK. ByteScout Screen Capturing SDK is the tool for developers who want to add screen capturing in their application. Can record screen into video and into single screenshots. Output formats are WMV, AVI, WebM for video and PNG for screenshots. You can adjust output video size, quality, resolution, framerate, video and audio codecs. Includes special privacy features for blacking out sensitive information on screen. Can also capture video from web camera, can add overlays with text or images. It can be used to capture video and add time stamp using C++ (unmanaged).

You will save a lot of time on writing and testing code as you may just take the C++ (unmanaged) code from ByteScout Screen Capturing SDK for capture video and add time stamp below and use it in your application. In your C++ (unmanaged) project or application you may simply copy & paste the code and then run your app! You can use these C++ (unmanaged) sample examples in one or many applications.

Download free trial version of ByteScout Screen Capturing SDK from our website with this and other source code samples for C++ (unmanaged).

Source Code Files:

CaptureFromEntireScreen.cpp

```
// CaptureFromEntireScreen.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#import "BytescoutScreenCapturing.dll"

using namespace BytescoutScreenCapturingLib;
using namespace std;


void usage(ICapturer* capturer);
void setParams(int argc, _TCHAR* argv[], ICapturer* capturer);


int _tmain(int argc, _TCHAR* argv[])
{
	::CoInitialize(0);

	// Create Capturer instance

	CLSID clsid_ScreenCapturer;
	CLSIDFromProgID(OLESTR("BytescoutScreenCapturing.Capturer"), &clsid_ScreenCaptu

	ICapturer* capturer = NULL;
	::CoCreateInstance(clsid_ScreenCapturer, NULL, CLSCTX_ALL, __uuidof(ICapturer)

	if (!capturer)
	{
		_ftprintf(stdout, _T("Screen Capturer is not installed properly."));
		::CoUninitialize();
		return 1;
	}

	capturer->put_RegistrationName(_T("demo"));
	capturer->put_RegistrationKey(_T("demo"));

	// Set capturing type
	capturer->put_CapturingType(catScreen);

	// Set output video width and height
	capturer->put_OutputWidth(640);
	capturer->put_OutputHeight(480);

	    // WMV and WEBM output use WMVVideoBitrate property to control output vide
	    // so try to increase it by x2 or x3 times if you think the output video a
	    // capturer->put_WMVVideoBitrate(capturer->WMVVideoBitrate * 2);


	// set the text overlay with autochanging time stamp macros to indicate playing
	capturer->OverlayingRedTextCaption = _T("Recording: {RUNNINGMIN}:{RUNNINGSEC}:-

	// uncomment to enable recording of semitransparent or layered windows (Warning
```

```
        // capturer->CaptureTransparentControls = true;


        // Set output file name
        capturer->OutputFileName = _T("Output.wmv");

        // Start capturing
        HRESULT hr = capturer->Run();

        // IMPORTANT: if you want to check for some code if need to stop the recording
        // using Thread.Sleep(1) inside the checking loop, so you have the loop like
        // Do
        // Thread.Sleep(1)
        // While StopButtonNotClicked


        if (FAILED(hr))
        {
                // Error handling
                CComBSTR s;
                capturer->get_LastError(&s);
                _ftprintf(stdout, _T("Capture failed: %s\n"), CString(s));
        }
        else
        {
                _tprintf(_T("Starting capture - Hit a key to stop ...\n"));

                int i = 0;
                TCHAR *spin = _T("|/-\\");

                // Show some progress
                while (!_kbhit())
                {
                        _tprintf(_T("\rEncoding %c"), spin[i++]);
                        i %= 4;
                        Sleep(50);
                }

                // Stop after key press
                capturer->Stop();

                _tprintf(_T("\nDone."));
                getchar();
        }

        // Release Capturer
        capturer->Release();
        capturer = NULL;

        ::CoUninitialize();

        return 0;
}
```

stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
// CaptureFromEntireScreen.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

// TODO: reference any additional headers you need in STDAFX.H
// and not in this file
```

stdafx.h

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#pragma once

#ifndef _WIN32_WINNT            // Allow use of features specific to Windows XP or late
#define _WIN32_WINNT 0x0501     // Change this to the appropriate value to target other
#endif

#include <stdio.h>
#include <tchar.h>

#include <atlbase.h>
#include <atlstr.h>
#include <conio.h>
```

VIDEO

https://www.youtube.com/watch?v=fujkvtWUVCw