DATASTEAD SOFTWARE
Multipurpose Directshow Encoder
version 2.8

Copyright (©) Datastead 2024
www.datastead.com

Overview **3**
Features 3
System requirements 4
Download 4
License 4
Contact 4
Limitations of the evaluation version 4
FAQ **5**
Licensing 5
Evaluation version 5
Troubleshooting 5
Filter install/Uninstall **7**
installing the package automatically from the command line 7
uninstalling the package automatically from the command line 7
installing the package manually 7
uninstalling the package manually 7
Filter configuration **8**
If the source is a video file (or audio file) 8
If the source is a live DirectShow source (e.g. a webcam) 9
Command line syntax 10
Recording 10
a) example of H264->MP4 recording using the OpenH264 software encoder 10
b) example of H264+AAC->MP4 recording using the NVidia hardware encoder 10
c) example of H265(HEVC)+AAC->MP4 recording using the NVidia hardware encoder 11
Streaming **11**
a) Streaming at 1Mb/s in UDP multicast to the address 239.255.0.1 on the port 12801 11
b) Pushing to Youtube at 2Mb/s with the audio input pin connected: 11
c) Pushing to Youtube at 4Mb/s with the audio input pin NOT connected: 11
d) Pushing to an Ant Media Server 11
e) Pushing to a Wowza server 12
Filter GUID and interface **13**

Filter CLSID....13
IDatasteadMultipurposeDirectShowEncoder interface....13
SetCommandLine....13
GetCommandLine....14
Stop_PauseWhenStop....14
GetCurrentLog....14
IsCurrentLogUpdated....15
GetExitCode....15
GetInputsTotalDurationMs....15
GetProgress_FrameCount....15
GetProgress_TimeMs....15
GetProgress_DuplicatedCount....15
GetProgress_DroppedCount....15
GetProgress_Fps....15
GetProgress_Quality....15
GetProgress_SizeWrittenKb....15
GetProgress_BitRateKbps....15
GetConnectedVideoPinInfo....15
GetConnectedAudioPinInfo....16
SetMediaEventSinkNotifyID....16
Is64BitWindows....16
Is64BitApplication....16

Reserved command-line keywords....17

PAUSEWHENSTOP....17
SHOWCONSOLE....17
NOLOG....17
LOWLATENCY....17
NOABORTONERROR....17
DONTOVERWRITE....17
TOPDOWN....18
COMPRESS....18

Graph event notification through IMediaEventEx....19

IMediaEventEx events....19
Retrieving the sender instance when several instances of the Multipurpose Encoder are used in the same graph....19

Quick start from GraphEdit.exe....21

How to debug....22

Using the filter for a normal processing, without DirectShow input.....23

Screenshots....24

graphedit:....24
graphedit with debug console:....25

Overview

Features

The Datastead Multipurpose Encoder can be used in 2 different ways:

- from the TVideoGrabber SDK as compression/recording/streaming codecs to encode in various formats (c.f. "Datastead Encoder" in the TVideoGrabber SDK help),

- as a DirectShow Sink filter to compress/encode audio/video streams by invoking in the background through the FFmpeg command line.

Note: if the Multirpurpose Encoder is used as plugin from TVideoGrabber, the codec parameters can be set programmatically without requiring the FFMpeg command line syntax.

A LGPL build of FFmpeg named ffmpegLGPL.exe is included in the filter, allowing to record in most of the encoding formats, as well as :

- H264 software encoding (through the OpenH264 codec)

- H264/HEVC GPU encoding through Intel Quicksync (" -c:v h264_qsv") or NVidia NVENC (" -c:v h264_nvenc")

The filter multiplexes the uncompressed DirectShow video and audio streams into an ASF transport stream and writes this transport stream to a named pipe. This named pipe is taken as input by the transcoder, that is invoked as a child process in a non-visible background process from the command line.

There is no binding or C++ linking between the filter and the transcoder, all the settings are passed by the command-line, and the audio/video stream is passed through the named pipe.

To configure the filter, just invoke SetCommandLine and specify the desired command line, along with some reserved keywords for the filter control.

When the graph runs, the transcoder process starts in the background, and stops when the graph is stopped.

Additionally to this default DirectShow behavior, the "PAUSEWHENSTOP" feature allows to pause the process while the graph is stopped, and to resume it when the graph is ran again, allowing to build in real time a single audio/video clip from start/stop sequences separated in time.

It is possible to make the transcoder console visible for debugging or control purpose.

System requirements

Windows 11, Windows 10, Windows 8.1, Windows 8, Windows 7

32bit or 64bit operating system supported.

Download

The evaluation package can be downloaded here:

https://www.datastead.com/downloads/

License

Royalty-free developer license:
https://www.datastead.com/purchase/

Contact

Email:

contact@datastead.com

support@datastead.com.

sales@datastead.com

Alternative email address:

datastead@hotmail.com

Web:
http://www.datastead.com/contact/

Limitations of the evaluation version

- if the video pin is connected in RGB24 or RGB32: the filter overlays a "Datastead" logo over the video frames,

- if the video pin is connected in YUY2, UYVY or RGB555: the video frames show a green flash one time and 2 times alternatively, and so on periodically

These limitations are removed in the licensed version.

FAQ

Licensing

Should I buy one license for each one of my customers?

No, it's a per-developer, royalty-free license. After purchasing the developer license you can distribute the filter along with the end-user application you have developed on as many PCs as needed, without having to pay anything else.

Does the Multipurpose DirectShow encoder include GPL code?

No, the Multipurpose DirectShow encoder includes only a LGPL build of ffmpeg.exe. There is no GPL code installed by the package.

Evaluation version

What are the limitations of the evaluation version?

- if the video pin is connected in RGB32 or RGB24, the filter overlays a logo over the video frames

- if the video is connected in another format (YUY2, etc...), the filter flashes alternatively one time then 2 times, this flashing occurs periodically.

Troubleshooting

The recorded file or output stream stops after 1 minute

The filter is running in evaluation mode. In this mode the recording is limited to 1 minute.

The frame rate is too slow

Check the codec's settings options, and choose a faster compression setting, and/or reduce the output video size.

In real time compression, if the time to compress 2 frames becomes longer than the time between 2 frames, frames are dropped.

The transcoder does not not recognize the pipe input stream

Verify in the syntax help of the transcoder if you can specify additionally the pipe input format (asf). If this does not help retry after specifying the COMPRESS keyword, however this will increase the CPU load.

When I run the graph I get a "The graph could not change state error"

The origin of the problem is usually a wrong FFmpeg command line parameter.

If you are using FFmpeg add " -report" to the command line, so you will get the FFmpeg log file in the current folder.

When I run the graph it just hangs

. Usually the problem is wrong transcoder parameter syntax, a missing space, etc.... The filter verifies if the transcoder process starts, but it does not check the syntax itself.

- if you are using FFmpeg, remove the"DONTOVERWRITE" keyword if you have added it.

If specified and the file exists, FFmpeg waits indefinitely for the manual confirmation to overwrite the file. Using DONTOVERWRITE is not recommended for this reason.

- if FFmpeg is connecting to a remote URL (e.g. when pushing the FFmpeg output to a Wowza server), this can takes a few seconds, and even wait until the FFmpeg connection timeout if the remote URL does not respond.

Filter install/Uninstall

The single .exe installer included in the package installs and registers automatically:

- the x86 version of the filter on Windows 32bit,
- both the x86 and x64 versions of the filter on Windows 64bit.

installing the package automatically from the command line

The command line to run the installer is:

Datastead_MultipurposeEncoder.exe **/silent**

or

Datastead_MultipurposeEncoder.exe **/verysilent**

uninstalling the package automatically from the command line

The command line to run the uninstaller is:

"C:\Program Files\Datastead\MultipurposeEncoder\unins000.exe"

installing the package manually

Double-click on the .exe installer, and accept each confirmation dialog.

uninstalling the package manually

Control panel → Add/Remove program → uninstall the Datastead Multipurpose DirectShow Encoder

Filter configuration

If the source is a video file (or audio file)

1. create the filter instance
2. add the **“Datastead Multipurpose Encoder (From File)”** to the graph
3. query the filter's IDatasteadMultipurposeDirectShowEncoder interface
4. invoke SetCommandLine on this interface.
6. run the graph

```
... create the filter graph...
IBaseFilter *DatasteadMPE = NULL;
HRESULT hr = CoCreateInstance(CLSID_DatasteadMultipurposeDirectShowEncoderFromFile,
NULL, CLSCTX_INPROC_SERVER, (void**) &DatasteadMPE);
if (SUCCEEDED (hr)) {
  hr = pGraph->AddFilter(DatasteadMPE, "Datastead Multipurpose Encoder (From
File)");
  if (SUCCEEDED (hr) {
    IDatasteadMultipurposeDirectShowEncoder *MPEConfig;
    hr = Filter.QueryInterface (IID_IDatasteadMultipurposeDirectShowEncoder,
(void**) &MPEConfig);
    if (SUCCEEDED (hr) {
       hr = MPEConfig.SetCommandLine (...transcoder's ".exe" command line...);
       if (SUCCEEDED (hr)) {
          ... run the graph
       }
       MPEConfig->Release();
    }
  }
  DatasteadMPE->Release();
}
```

If the source is a live DirectShow source (e.g. a webcam)

1. create the filter instance
2. add the camera video device to the graph
3. add the camera audio device to the graph (if audio needed)
4. add the Datastead Multipurpose Encoder to the graph
5. query the filter's IDatasteadMultipurposeDirectShowEncoder interface, set the command line
6. connect the video and/or audio pins
7. run the graph.

E.g.:

```
... create filter graph...
... add video and/or audio capture devices ...
IBaseFilter *DatasteadMPE = NULL;
HRESULT hr = CoCreateInstance(CLSID_DatasteadMultipurposeDirectShowEncoder, NULL,
CLSCTX_INPROC_SERVER, (void**) &DatasteadMPE);
if (SUCCEEDED (hr)) {
  hr = pGraph->AddFilter(DatasteadMPE, "Datastead Multipurpose Encoder");
  if (SUCCEEDED (hr) {
    IDatasteadMultipurposeDirectShowEncoder *MPEConfig;
    hr = Filter.QueryInterface (IID_IDatasteadMultipurposeDirectShowEncoder,
(void**) &MPEConfig);
    if (SUCCEEDED (hr) {
       hr = MPEConfig.SetCommandLine (...transcoder's ".exe" command line...);
       if (SUCCEEDED (hr)) {
          ... from render the video and audio output pins, and run the graph
       }
       MPEConfig->Release();
    }
  }
  DatasteadMPE->Release();
}
```

Command line syntax

The command-line is based on the FFmpeg configuration syntax, the documentation can be found here. At the left part of the command line (input part), "***fmpegLGPL.exe -i %PIPE%***" is required. This tells FFmpeg to take the Multipurpose encoder ouput as input.

So the encoding settings that configure the video compression + audio compression + multiplexing, are located **after** "*%PIPE%*".

Recording

Feel free to contact us at contact@datastead.com if a command-line is failing or not working properly!

a) example of H264->MP4 recording using the OpenH264 software encoder

```
MPEConfig.SetCommandLine ("ffmpegLGPL.exe -threads 8 -i %PIPE% -c:v h264
-profile:v baseline -allow_skip_frames 1 -loopfilter 0 -b:v 1M -an -f mp4
myoutputfile.mp4");
```

"-c:v h264" means "*use the H264 sofware encoder as video codec*", followed by the specific OpenH264 codec settings (b:v 1M = video bitrate 1Mbps/s)

"-an" *means "no audio"*

b) example of H264+AAC->MP4 recording using the NVidia hardware encoder

```
MPEConfig.SetCommandLine ("ffmpegLGPL.exe -i %PIPE% -c:v h264_nvenc -preset
fast -rc 1 -cbr 1 -b:v 200M -c:a aac -ab 128k -ac 2 -ar 44100 -f mp4
myoutputfile.mp4);
```

"-c:v h264_nvenc" means *"compress video using Nvidia H264 harware sofware encoder",* followed by the specific OpenH264 codec settings (cbr = costant bitrate, b:v 200M means video bitrate 200Mbps)

"-c:a aac" means *"compress audio using the AAC encoder",* followed by the audio encoding settings (ab 128K = audio bitrate 128kbps, ac 2 = 2 audio channels, ar 44100 = sample rate 44100hz)

ffmpegLGPL.exe is the name of the LGPL build of FFmpeg included in the package ans located in the installation folder of the filter.

%PIPE% is mandatory for the transcoder to take the filter's output as input. I must be written exactly as is. It corresponds to the pipe generated by the filter and used as input by the transcoder. At runtime, %PIPE% is replaced by the real name of the pipe, generated automatically by the filter.

c) example of H265(HEVC)+AAC->MP4 recording using the NVidia hardware encoder

```
MPEConfig.SetCommandLine ("ffmpegLGPL.exe -i %PIPE% -c:v hevc_nvenc -preset
fast -rc 1 -cbr 1 -b:v 200M -c:a aac -ab 128k -ac 2 -ar 44100 -f mp4
myoutputfile.mp4);
```

Streaming

Feel free to contact us at contact@datastead.com if a command-line is failing or not working properly!

Note: to stream without audio, replace the e.g. "-c:a aac -ac 2 -ab 128k" sequence by "-an"

a) Streaming at 1Mb/s in UDP multicast to the address 239.255.0.1 on the port 12801

```
MPEConfig.SetCommandLine ("ffmpegLGPL.exe -threads 8 -i %PIPE% -c:v h264
-profile:v constrained_baseline -allow_skip_frames 1 -loopfilter 0 -b:v 1M
-c:a aac -ac 2 -ab 128k -f mpegts udp://239.255.0.1:12801?pkt_size=1316");
```

b) Pushing to Youtube at 2Mb/s with the audio input pin connected:

```
MPEConfig.SetCommandLine ("ffmpegLGPL.exe -i %PIPE% -c:v h264 -profile:v
constrained_baseline -allow_skip_frames 1 -loopfilter 0 -g 90 -b:v 2M -c:a
aac -ab 64k -ac 2 -f flv rtmps://a.rtmp.youtube.com:443/live2/te42-648f-uj55-
fjt9-chz5");
```

c) Pushing to Youtube at 4Mb/s with the audio input pin NOT connected:

(in this case a dummy silent audio stream is generated because Youtube requires the audio)

```
MPEConfig.SetCommandLine ("ffmpegLGPL.exe -i %PIPE% -f lavfi -i anullsrc
-c:v h264 -profile:v constrained_baseline -allow_skip_frames 1 -loopfilter 0
-g 90 -b:v 4M -f flv rtmps://a.rtmp.youtube.com:443/live2/te42-648f-uj55-
fjt9-chz5");
```

d) Pushing to an Ant Media Server

Same syntax as the b) example, the end of the command line will look like:

```
-f flv rtmp://192.168.1.199/WebRTCApp/899895991642161748961446

-f flv rtmp://192.168.1.199/LiveApp/722968383194925013459884
```

E.g. to push video without audio:

```
MPEConfig.SetCommandLine ("ffmpegLGPL.exe -i %PIPE% -c:v h264 -profile:v
constrained_baseline -allow_skip_frames 1 -loopfilter 0 -g 90 -b:v 2M -an -f
flv rtmp://192.168.1.199/LiveApp/722968383194925013459884");
```

e) Pushing to a Wowza server

Same syntax as the b) example, the end of the command line will look like:

```
-f rtsp rtsp://36197f.entrypoint.cloud.wowza.com/app-bcae
-fflv rtmp://314f9a.entrypoint.cloud.wowza.com/app-f6de/3ef94773
flashver=FMLE/3.0\20(compatible;\20FMSc/1.0) live=true

-f flv "rtmp://192.168.1.167:1935/live/myStream
flashver=FMLE/3.0\20(compatible;\20FMSc/1.0) live=true pubUser=live
pubPasswd=live
```

Filter GUID and interface

Filter CLSID

```
CLSID_DatasteadMultipurposeDirectShowEncoder:
   TGUID = '{39E3007B-6185-47FC-9839-C4B8621AC065}';
```

IDatasteadMultipurposeDirectShowEncoder interface

The filter exposes the following interface:

```
DECLARE_INTERFACE_(IDatasteadMultipurposeDirectShowEncoder, IUnknown)
{
 virtual HRESULT      STDMETHODCALLTYPE SetCommandLine(/*in*/ LPWSTR CommandLine) PURE;
 virtual HRESULT      STDMETHODCALLTYPE GetCommandLine(/*out*/ LPWSTR *CommandLine) PURE;
 virtual HRESULT      STDMETHODCALLTYPE Stop_PauseWhenStop() PURE;
 virtual HRESULT      STDMETHODCALLTYPE GetCurrentLog(
                                        /*in*/ bool OnlyIfUpdated,
                                        /*in*/ PCHAR pBuffer,
                                        /*in*/ int MaxBufferSize,
                                        /*out*/ int *CurrentLogSize,
                                        /*out*/ int *CurrentLogFlag) PURE;
 virtual HRESULT      STDMETHODCALLTYPE IsCurrentLogUpdated() PURE;
 virtual int          STDMETHODCALLTYPE GetExitCode() PURE;
 virtual unsigned int STDMETHODCALLTYPE GetInputsTotalDurationMs() PURE;
 virtual unsigned int STDMETHODCALLTYPE GetProgress_FrameCount() PURE;
 virtual unsigned int STDMETHODCALLTYPE GetProgress_TimeMs() PURE;
 virtual unsigned int STDMETHODCALLTYPE GetProgress_DuplicatedCount() PURE;
 virtual unsigned int STDMETHODCALLTYPE GetProgress_DroppedCount() PURE;
 virtual double       STDMETHODCALLTYPE GetProgress_Fps() PURE;
 virtual double       STDMETHODCALLTYPE GetProgress_Quality() PURE;
 virtual double       STDMETHODCALLTYPE GetProgress_SizeWrittenKb() PURE;
 virtual double       STDMETHODCALLTYPE GetProgress_BitRateKbps() PURE;
 virtual HRESULT      STDMETHODCALLTYPE GetConnectedVideoPinInfo(
                                        /*out*/ int *VideoWidth,
                                        /*out*/ int *VideoHeight,
                                        /*out*/ int *VideoAvgTimePerFrame) PURE;
 virtual HRESULT      STDMETHODCALLTYPE GetConnectedAudioPinInfo(
                                        /*out*/ int *AudioChannels,
                                        /*out*/ int *AudioSampleRate,
                                        /*out*/ int *AudioBitsPerSample) PURE;
 virtual HRESULT       STDMETHODCALLTYPE SetMediaEventSinkNotifyID (LONG_PTR Value) PURE;
 virtual BOOL          STDMETHODCALLTYPE Is64BitWindows() PURE;
 virtual BOOL          STDMETHODCALLTYPE Is64BitApplication() PURE;
};
```

SetCommandLine

Sets the FFmpeg command line, with eventual extra keywords destined to the filter. The syntax is described here.

GetCommandLine

Retrieves the current command line. Pass LPWSTR pointer to the function. The function allocates memory and copy the string. If the function succeeds, the LPWSTR pointer must be freed by invoking CoTaskMemFree.
E.g.:

```
LPWSTR pCommandLine;
if (SUCCEEDED (MPEConfig.GetCommandLine (&pCommandLine))) {
   ... do what you need with the pCommandLine string returned
   CoTaskMemFree (pCommandLine);
}
```

Stop_PauseWhenStop

See the PAUSEWHENSTOP keyword in the reserved keywords chapter.

GetCurrentLog

Copies the current FFmpeg log string in a buffer. If the string contains several lines, they are separated by CR/LF characters: char(13)/char(10).
Note that this function does not work if SHOWCONSOLE has been specified.

You must allocate the buffer and pass its pointer to the function.
To determine the required maximum buffer size and allocate the buffer, invoke GetGurrentLog with all parameters to 0 / NULL excepted CurrentLogSize that will return the size required to allocate the buffer, e.g.:

```
int MaxBufferSize;
CHAR *pBuffer = NULL;
...
if (pBuffer == NULL) {
   if SUCCEEDED (MPEConfig.GetCurrentLog (false, NULL, 0, &MaxBufferSize, NULL)) {
      pBuffer = new CHAR[MaxBufferSize];
   }
}
```

Then to read the current log:

```
int BufferSizeRead = 0;
if SUCCEEDED (MPEConfig.GetCurrentLog (true, pBuffer, BufferSize, &BufferSizeRead, NULL)) {
    ... do anything with the string in the buffer...
}
```

Notes:

OnlyIfUpdated parameter:
. if false, the buffer returns its content, even if this content has already been returned by the function
- if true and the buffer content has changed, the function fills the buffer and returns S_OK, otherwise it returns E_NOT_SET

the buffer is "string safe": GetCurrentLog writes a NULL character at the end of the text read.

IsCurrentLogUpdated

return S_OK if the buffer content has changed or E_NOT_SET. It is useless to invoke it to determine if GetBufferLog must be invoked just after, in this case invoke directly with the OnlyIfUpdated parameter = true, e.g. GetBufferLog (true, ...) so it will return directly E_NOT_SET if no new log is available.

GetExitCode

returns the exit code, 0 on success or another value if the encoding failed for any reason (usually a wrong command line syntax or a non-writeable output file).

GetInputsTotalDurationMs

(*)
when the input file has a fixed size, it returns the duration reported by FFmpeg in milliseconds. If several input files are used (e.g. when concatening clips, the value returned is the total duration of final clip (the sum of the durations of the concatened clips).

GetProgress_FrameCount

returns the current frame count (*)

GetProgress_TimeMs

returns the current stream time in milliseconds (*)

GetProgress_DuplicatedCount

returns the number of duplicated frames, if any (*)

GetProgress_DroppedCount

returns the number of dropped frames, if any (*)

GetProgress_Fps

returns the average number of frames per second (*)

GetProgress_Quality

returns the average quality (the meaning depends on the codec) (*)

GetProgress_SizeWrittenKb

return the size written to disk in KiloBytes (*)

GetProgress_BitRateKbps

returns the average bit rate in KiloBytes per second (*)

(*) these functions are currently supported only with FFmpeg, and only if SHOWCONSOLE has not been specified.

GetConnectedVideoPinInfo

returns the video size of the input pin, and average time per frame of the video input stream (expressed in 100 nanoseconds units, 1 second = 10000000, e.g. at 25 fps the average time per frame returns 400000)

GetConnectedAudioPinInfo

return the number of audio channels, the sample rate and the number of bits per sample of the audio input pin

SetMediaEventSinkNotifyID

This is useful only:
- when processing the event notifications sent by the filter to the graph (when implementing IMediaEventSink),
- **and** when using more than one instance of the Multipurpose Encoder in the same graph

Is64BitWindows

This helper function lets you know if your app is running on a Windows 64bit PC, or on a Windows 32bit PC.

Is64BitApplication

This helper function lets you know if the application compiled in 32bit or 64bit.

Reserved command-line keywords

The following keywords can be specified before the command line. They are interpreted by the filter and removed from the final command line passed to FFmpeg. E.g.:

```
MPEConfig.SetCommandLine (SHOWCONSOLE NOLOG ffmpegLGPL.exe -i %PIPE% myclip.webm");
```

PAUSEWHENSTOP

If specified, when the graph is stopped, the FFmpeg recording is paused (instead of being stopped as well). When the graph is ran again, the FFmpeg recording is resumed. So this lets record a single, continuous clip without gap, although the graph has been stopped and restarted, whatever the interval of time between the stops and starts.
Note that when this feature is enabled, and you want to FINALLY stop the FFmpeg recording, invoke **MPEConfig.Stop_PauseWhenStop()**.
If Stop_PauseWhenStop is never invoked, the recorded clip will be closed when exiting the application.

Notes:
- while the graph is stopped, the video and audio pins must not be disconnected / reconnected.

SHOWCONSOLE

makes the console visible. Mainly for debugging purpose. In this mode the FFmpeg information (e.g. current time, frame count) is not available through the the filter interface (see the interface below)

NOLOG

Can be specified to save CPU. if specified, the filter does not analyze the FFmpeg progress information (e.g. current time, frame count), so the filter does not report them.

LOWLATENCY

Specify it if the filter is used for a live streaming in real time (e.g. to send the stream to a Wowza server).

NOABORTONERROR

If there is an error (e.g. if the syntax is not correct and/or if FFmpeg exits by itself with an error code), the filter sends an EC_ERRORABORT notification to the graph, that stops the graph.
If NOABORTONERROR is specified, the filter sends instead an "neutral" EC_MPE_TERMINATED_ERROR notification to the graph that can be handled, but the graph does not stop.

DONTOVERWRITE

by default, if the command line contains "FFmpeg", the filter passes the "-y" option to FFmpeg to overwrite a previous clip, if any.
Set this option to prevent overwriting an existing file, however it is **NOT RECOMMENDED** to use it, because if SHOWCONSOLE is disabled and FFmpeg waits for the overwrite confirmation, the graph will just hang.

TOPDOWN

If specified, the image is flipped vertically without extra CPU load. Can be required by some encoders if the image appears inverted.

COMPRESS

If specified, the video stream passed through the pipe, compressed in MPEG-4. If the command line transcoder does not accept the pipe uncompressed, enabling this option could fix the problem.
Note that enabling this option will significantly increase the CPU load.

Graph event notification through IMediaEventEx

IMediaEventEx events

(see https://msdn.microsoft.com/en-us/library/windows/desktop/dd377538%28v=vs.85%29.aspx)

```
#define EC_MPE_TERMINATED_SUCCESS EC_USER + 0x5501
#define EC_MPE_TERMINATED_ERROR   EC_USER + 0x5502
#define EC_MPE_LOGBUFFER_UPDATED  EC_USER + 0x5503
```

EC_MPE_TERMINATED_ERROR vs EC_ERRORABORT

By default, if an error occurs after the graph is ran, the filter notifies the graph with an EC_ERRORABORT event that stops the graph.

You can prevent the filter to stop the graph by specifying the NOABORTONERROR keyword at the beginning of the command line. In this case, if an error occurs the filter sends a EC_MPE_TERMINATED_ERROR notification message that you can handle but that does not stop the graph.

EC_MPE_TERMINATED_SUCCESS
If the FFmpeg processing ends successfully before the graph is stopped (e.g. if a maximal encoding duration has been specified to the command line), the filter sends a EC_MPE_TERMINATED_SUCCESS event to the graph.

EC_MPE_LOGBUFFER_UPDATED
Each time the log buffer has been updated, the filter sends a EC_MPE_LOGBUFFER_UPDATED event to the graph.
When this event is received you can successfully invoke GetCurrentLog to retrieve the log content, or read one of the GetProgress... functions to retrieve the corresponding value.

Retrieving the sender instance when several instances of the Multipurpose Encoder are used in the same graph

For all the events described above, Param1 and Param2 return the following values:

Param1:
returns the exit code of FFmpeg

Param2:

- by default Param2 returns the pointer to the instance of the filter (the same IBaseFilter pointer that has been returned when you invoked
CoCreateInstance(CLSID_DatasteadMultipurposeDirectShowEncoder...)

- when configuring the filter, if you have invoked SetMediaEventSinkNotifyID (Value) with a value of your

choice, Param2 returns this value (instead of the IBaseFilter pointer)

In the case where several Multipurpose Encoders are used within the same graph, any of these 2 mechanisms lets you easily retrieve what instance of the Multipurpose Encoder sent the event notification.

Quick start from GraphEdit.exe

- run GraphEdit -> Graph -> Insert Filters -> DirectShow Filters

- locate the "Datastead Multipurpose Encoder" filter, double-click on it to insert it,

- insert the video source filter (e.g. a Webcam) and/or the audio source filter (e.g. a microphone)

- connect video output pin (if any) to the Multipurpose video input pin

- connect audio output pin (if any) to the Multipurpose audio input pin

- right-click on the "Datastead Multipurpose Encoder" filter, and enter the command line, eventually preceded by reserved Multipurpose keywords.

Examples of a very minimalistic command line that can be tested "out the box":

- with console, for debugging purpose:

```
SHOWCONSOLE ffmpegLGPL.exe -i %PIPE% myclip.webm
```

- or not visible, like a normal DirectShow sink filter:

```
ffmpegLGPL.exe -i %PIPE% myclip.webm
```

How to debug

First verify if the solution is not in the the Support section of the FAQ. If not:

1. add "SHOWCONSOLE" at the beginning of the command line, to see if the FFmpeg process starts and if any red error report appears.

2. add " -report" or " -report -loglevel debug" to the command line, this will generate the FFmpeg log file in the current folder.
Verify/fix the FFmpeg syntax, then try again.

PS: don't forget to remove the report and loglevel statements because they slow-down the filter.

Using the filter for a normal processing, without DirectShow input.

To avoid having to implement specific code for editing or transcoding that do not use the pipe as input, you can use the filter for a normal batch transcoding as follows:

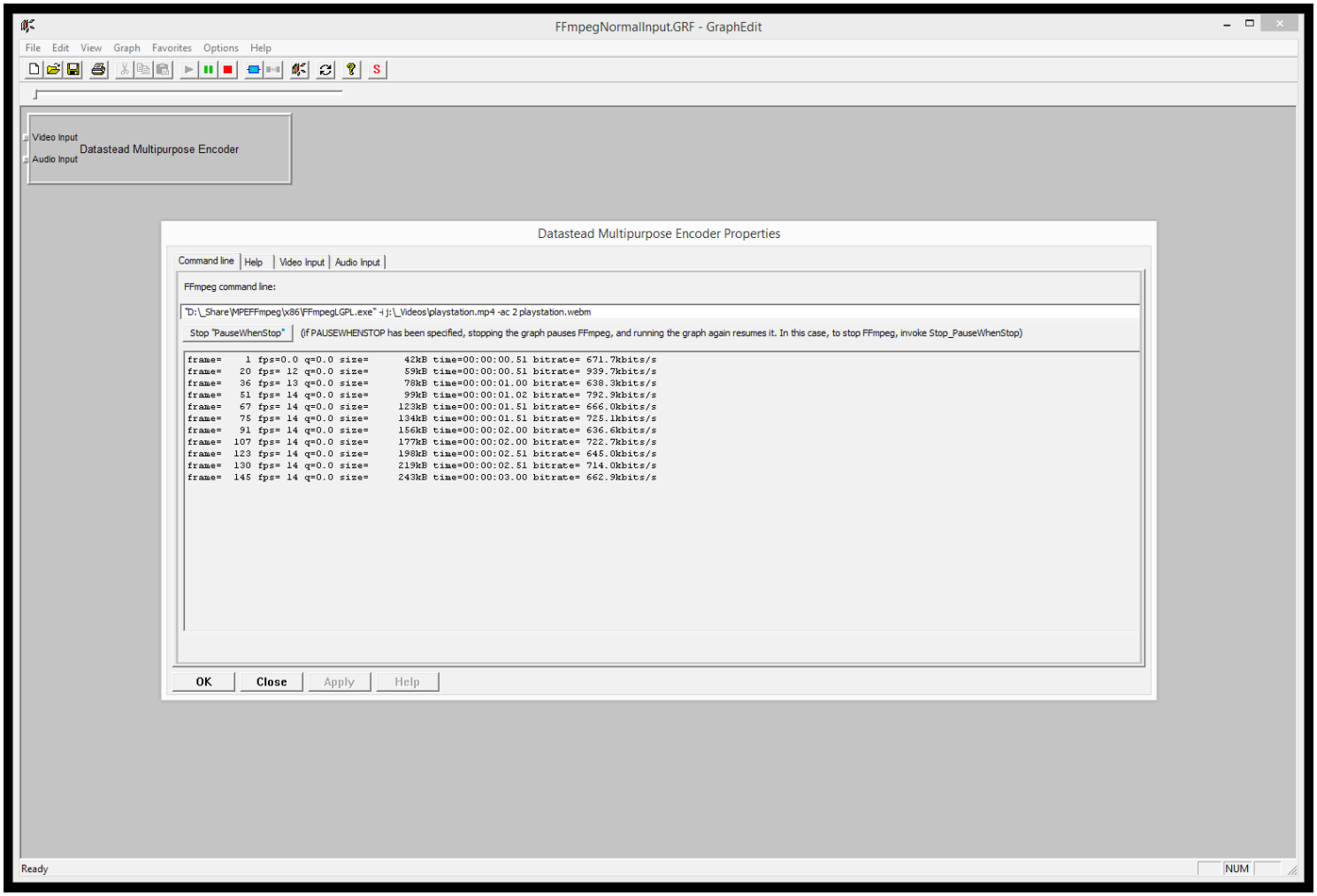
- add the filter to the graph (alone without connecting any pin)

- configure command line with a normal command line (without "%PIPE%")

- run the graph

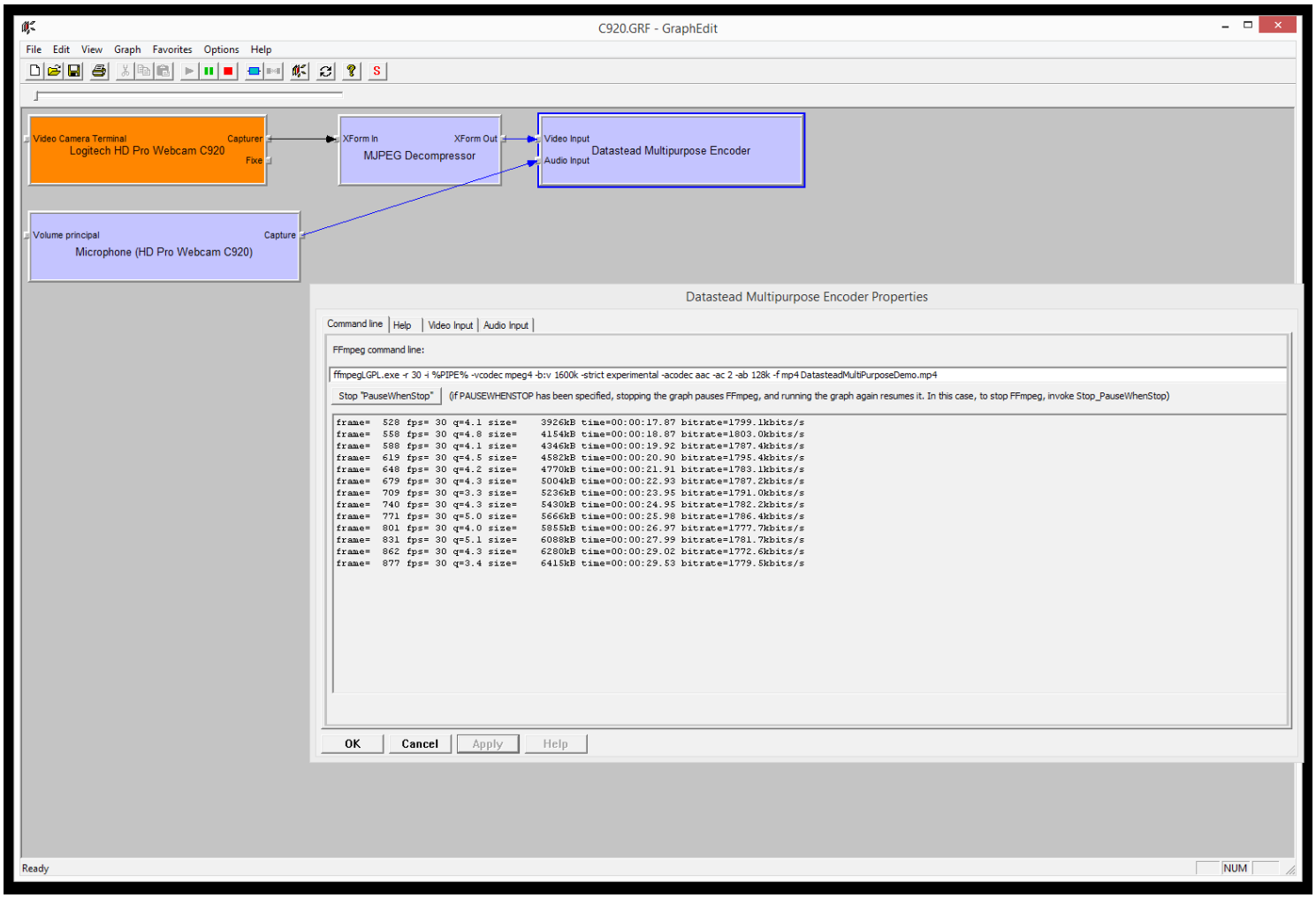
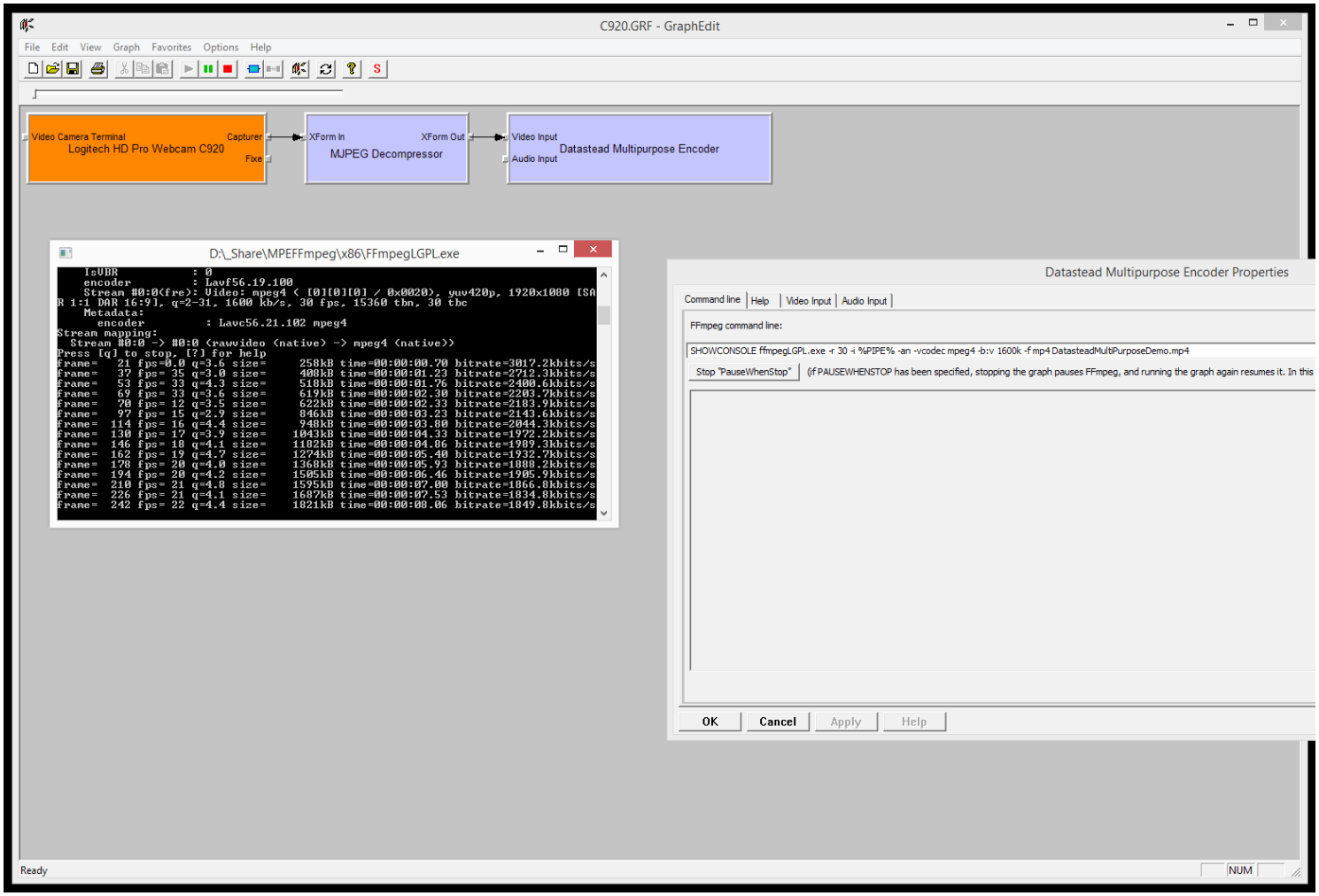
The progress log and completion notification events work the same way.

E.g.:



Screenshots

graphedit:



graphedit with debug console: