

Re: WMR9, dual-screen, EC_PAUSED and delay in video playback

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.directx.video/2007-04/msg00>

- *From:* Mehdi <vioccc@xxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 23 Apr 2007 10:45:18 +0100
-

Hi Alessandro,

On Fri, 20 Apr 2007 12:45:13 -0400, Alessandro Angeli wrote:

The documentation is very clear on this: when you run the graph, the filter graph manager first pauses the graph, if it is not already paused, so that data can be cued, then actually runs it.

[...]

Check whether the graph changed (either different filters or different connection media types) after it starts running on the secondary display. It is likely that the secondary display hardware has different requirements from the primary one, so the renderer may need to be reconnected and it may take some time for IntelligentConnect to build a suitable filter chain.

Thanks for your help. This makes sense. I'll try to write some code to see what exactly happens to the graph when I run it on the secondary screen. What still worries me though is that when I run it on the primary screen, it buffers 2 frames before starting the playback while when I run it on the secondary screen it seems to want to buffer 9 frames initially which causes a huge delay in playback. I don't quite know how to solve this problem. In case anybody else is having the same problem, I've overridden the Pause, Stop and Run methods of my source filter in order to see when they are being called. The sequence of events goes like this:

*** When run on the primary screen ***

```
Graph event: EC_VMR_RENDERDEVICE_SET
Pause()
FillBuffer()
FillBuffer()
Graph event: EC_PAUSED
```

Run()
FillBuffer()
.... (calls to FillBuffer())
=> Video plays with almost no delay

*** When run on the secondary screen ***

Graph event: EC_VMR_RENDERDEVICE_SET
Pause()
FillBuffer()
FillBuffer()
Stop()
Graph event: EC_VMR_RENDERDEVICE_SET
Pause()
FillBuffer()
FillBuffer()
FillBuffer()
FillBuffer()
FillBuffer()
FillBuffer()
FillBuffer()
FillBuffer()
FillBuffer()
FillBuffer()
Run()
Graph event: EC_PAUSED
FillBuffer() called
.... (calls to FillBuffer())
=> Video plays with a 1 second delay

2 observations:

– the EC_PAUSED event is traced long after Pause() has been called. It might be because I'm retrieving graph events using IMediaEvents.GetEvent() in a separate thread which is maybe not scheduled to run before some time after Pause() has been called. However, the fact that EC_PAUSED always gets traced after Run() has been called is a bit strange. If thread scheduling timing was involved, I would have expected to see some randomness here but the sequence of events is always exactly the same as above.

– For some reason, I only get one EC_PAUSED event in the second case even though Pause() has been called twice. I'm not sure why. It might be that the renderer somehow cancels the first pause but I'd need to do some more reading on how these things work.