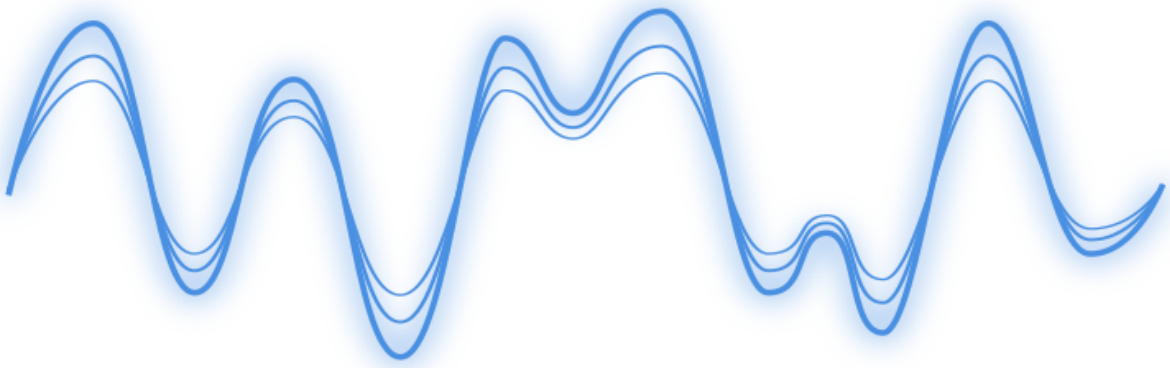
GLITCH FREE

An in-depth guide to tuning Windows
for reliable real-time audio performance



Brad Robinson

Cantabile

Contents

Chapter 1

Introduction

Are you a musician using a Windows PC for digital audio processing?

Perhaps you're composing music or perhaps you're performing on stage. In either case you've almost certainly encountered issues with audio "glitches" while using your software.

These glitches might manifest as audio dropouts, clicking, stuttering or other artefacts. Sometimes they occur often, sometimes rarely and seemingly completely at random.

If any of this sounds familiar to you then this guide is for you.

As the developer of music software designed specifically for live performance (Cantabile - see cantabilesoftware.com) I've spent over ten years helping customers to get their machines running reliably.

In this guide I hope to bring together all that information into an in-depth yet easy to follow set of steps that explain the things to check and how to fix the problems found. Along the way I'll also be going into reasonable detail about what's happening under the covers and why the topics being discussed matter.

Since Cantabile is aimed at musicians who perform on-stage, most of this guide takes the point of view that audio stability is absolutely paramount - glitches in front of a packed house are not a good look. If you're using your PC in other less critical ways you might want to ignore some of the suggestions if the occasional glitch is bearable and other factors like power saving or convenience are more important to you.

Either way, by the time you've read this guide you'll have all the tools and knowledge you need to get your machine running efficiently and reliably.

This book will be updated from time to time. To make sure you've got the latest edition please visit http://www.cantabilesoftware.com/glitchfree.

Chapter 2

Understanding Digital Audio

Before we get into diagnosing and fixing any problems it's important to have a good understanding of exactly what an audio glitch is. To understand this you need a fundamental understanding of how digital audio works.

Analog Audio

Analog audio uses a voltage on wire to move the cone of a speaker at particular frequencies to make sound.

The circuits driving these voltages are *continuous* and *instant*, that is, there are an infinite number of possible voltages and when the input signal changes they immediately produce a modified output signal.

You'll no doubt have seen audio signals represented as a wave on a graph:

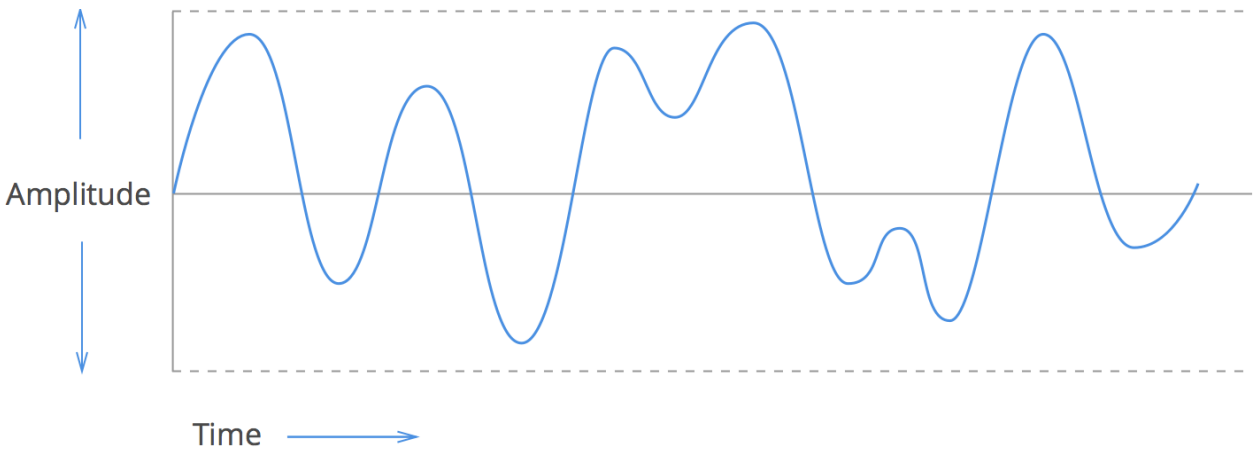


Figure 2.1: Analog Wave Form

The X-Axis represents time, the Y-Axis represents the position of the speaker cone, or the voltage on the wire, or more generally the "amplitude" of the signal at that point in time. An analog signal has an infinite number of possible X and Y values.

Microphones works the opposite way to a speaker, instead of converting voltages to sounds they convert sounds to voltages.

Digital Audio

In digital audio the analog signals are converted into a series of numbers that *approximate* the shape of the original analog signal. This series of numbers can then be stored, copied and otherwise manipulated by the computer.

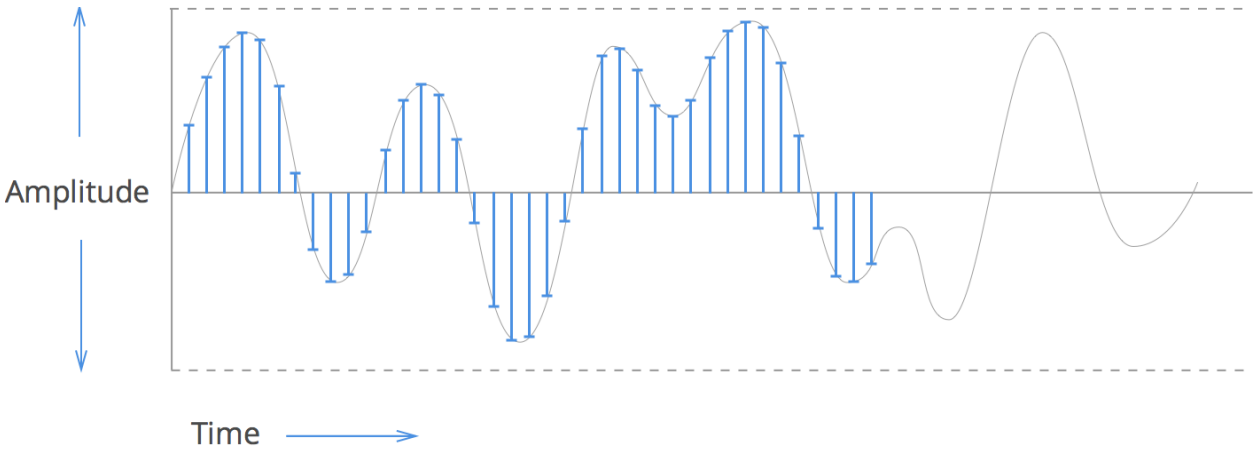


Figure 2.2: Sampling Analog to Digital

Each measurement of the analog signal (the blue bars in the above graph) is a called a sample – a measurement of the signal's amplitude at that point in time.

The speed at which these samples are recorded is called the sample rate – the number of samples captured for each 1 second of time. e.g. 48,000 samples per second.

The accuracy of each sample is determined by the bit depth – the precision of the digital number format uses to store the samples. e.g. a 16-bit sample can represent 65,536 different amplitudes.

Unlike an analog signal which has infinite positions on both axes, a digital signal has limited positions. The time axis precision is determined by the sample rate and the precision of the amplitude axis is determined by the sample bit depth.

Analog vs Digital

At this point you might be getting the impression that analog audio is superior to digital – it does after all seem to have more precision any way you look at it.

The problem with analog is that it's very difficult to capture and record in any format that maintains that precision and most analog formats lose precision each time the signal is copied.

The advantage of digital is that once the signal has been captured it can be copied without loss of quality and can be manipulated fairly easy (after all it's just a series of numbers which computers a great at working with).

The question is what sample rate and bit depth do we need to achieve a good quality signal. Higher sample rates and more precise bit-depths are better but also require more storage space and also place additional processing load on the computer.

The Ideal Sample Rate and Bit Depth

To work out the required sample rate and bit depth we can look at what humans are capable of hearing.

Humans can typically hear frequencies in the range of 20Hz up to about 20,000Hz. Without going into the details, the maths says that in order to accurately reproduce an analog signal we need a sample rate twice that of highest frequency that we want to reproduce.

So to reproduce the human hearing limit of 20KHz, we need a sample rate of least 40KHz. As for bit depth most humans can't tell any difference beyond 16-bit.

CD quality audio has a sample rate 44,100Hz and a bit depth of 16-bit – essentially just enough to accurately reproduce anything a human can hear.

ADC and DAC

In reading about sound cards you may have heard the terms ADC and DAC. These are acronyms for Analog to Digital Converter and Digital to Analog Converter and simply refer to the piece of circuitry in your sound card that performs the actual conversion from analog voltages to digital samples and vice-versa.

In no small part the audio quality of a sound card comes down to the accuracy of these two key components.

Buffering and Latency

Now that we have a basic understanding of digital audio you might be tempted to think that the computer can simply read one input sample, process it and generate an output sample. Do that 44,100 times per second and we should be good to go right? Unfortunately not.

As fast as modern PC's are, to interrupt it every time that one sample needs to be processed would be extremely inefficient. The computer would need to save the state of whatever it's currently working on, process that one sample, restore its state and then go back to what it was originally doing. This saving and restoring state is called task switching and is too slow to do 40,000 times per second.

To streamline this the sound card sends samples in batches resulting in far fewer task switches - a batch of 100 samples means 100 times fewer task switches.

We need to be careful however to not make these batches too large or there will be a noticeable delay between the input signal and the output signal. Any musician will tell you how difficult it is to play an instrument with these long latencies.

Let's define some terms:

- Buffering – the process of grouping samples into batches for processing.
- Buffer Size – the number of samples in one batch.
- Audio Cycle – the processing of one audio buffer.
- Latency – the time duration of the buffer.

Once again human anatomy helps us decide how much latency is tolerable. It's generally well accepted that most humans can't discern audio intervals less than about 10 milliseconds, or 0.01 seconds. In other words two sounds played 10 milliseconds apart sound instantaneous.

Latency can be calculated by dividing the buffer size by the sample rate and using some simple algebra we can work out the required buffer size:

Latency = BufferSize / SampleRate

BufferSize = Latency * SampleRate

BufferSize = 0.01 * 44100 = 441

At a sample rate of 44.1KHz, 10ms is 441 samples. Since some sound cards only support buffer sizes that are powers of 2 this is often rounded up to 512 samples (about 12ms) or down to 256 samples (6ms) – depending on what your computer is capable of.

I'll cover choosing a buffer size in more detail later.

But The Glitches! What's Causing the Glitches?

Finally we're at the point where we can think about what causes the dreaded audio glitches.

In order to produce a glitch free audio stream the computer needs to provide a fresh buffer of samples to the sound card roughly every 10 milliseconds. If that deadline is missed by even fraction of a millisecond then there's nothing the sound card can do to fill in the blanks and... glitch!

The type of glitch depends on a couple of things. Short delays typically sound like a "tick". Longer delays either produce no sound – "drop outs", or sometimes the buffered samples from the previous cycle are replayed causing "stuttering".

The key to performance tuning a PC for real-time audio processing is eliminating anything that might cause it to be held up on other tasks for longer than the duration of the buffer.

And to best understand that, we need to understand some basics of how the hardware and the operating system works...

Chapter 3

Operating System Basics

In this chapter I'll be providing a simplified explanation of how your computer's hardware, the operating system and your audio software work together to make real-time audio processing possible.

In particular I'll be explaining three key concepts of any modern operating system – multi-tasking, interrupts and virtual memory.

Multi-tasking

Multi-tasking is how it appears that multiple programs can be running at the same time even though your computer might only have a single processor.

The technique used by most modern operating systems (and for our purposes this includes Windows) is called "pre-emptive multitasking" in which the operating system runs each program for a short period time before switching to another program for a little while. If it does this quickly enough it appears as if all programs are running simultaneously.

It's called pre-emptive multitasking because the operating system forcefully re-takes control when a program's time is up – it pre-empts the running program no matter what it's doing.

These short periods of time in which programs run are called "time slices" and sometimes this technique is referred to as "time slicing".

The actual duration of a timeslice can vary depending on many factors but on Windows it's about 10 milliseconds (1/100 of a second).

The other type of multi-tasking is "co-operative multitasking" where each program must deliberately give control back to the operating system to allow other programs to run. The problem with co-operative multi-tasking is it makes software development more difficult and programs can easily hog the processor preventing other programs from running.

Threads and Thread Scheduling

In describing multi-tasking I said the operating system switches between "programs". In fact the correct term is a "thread".

You can think of threads as sub-programs within a program – each running independently of each other and each running in their own little time-slices.

A program with more than one thread is said to be multi-threaded. The advantage is that the different parts of the program can appear to run at the same time. For example the user interface of a word processor running on one thread can continue to run letting you type while another thread checks your spelling in the background.

The part of the operating system that controls all these threads is called the "thread scheduler". Its job is to decide which thread gets to run next and to handle the switching between different threads.

Conceptually Windows' thread scheduler is very simple. Every thread has a priority. When the thread scheduler needs to decide which thread to run next it chooses the highest priority thread that isn't waiting for something else to complete. That's it.

(Of course in practice it's a lot more complicated, but this is enough for our discussion).

The important point in the above statement is the bit about waiting.

Often a thread will need to wait for something else to complete. It might need to wait for another thread or a piece of hardware to complete something (reading from disk for example). It might be waiting on a timer to finish or perhaps it's deliberately put itself to sleep until some other thread wakes it up.

Threads that are in this waiting state are said to be "blocked" or "sleeping" and won't get to run again until the thread scheduler "unblocks" or "wakes" them.

The key is that: *most threads, most of the time are blocked*.

Even though high priority threads always run first, the lower priority threads still get to run when all the higher priority threads are blocked.

Interrupts, ISRs and DPCS

Interrupts are the mechanism by which the hardware in a computer notifies the operating system that something of significance has happened.

Let's take the keyboard as an example. When you press a key on your keyboard it raises an interrupt signal to the CPU indicating that something has happened.

On seeing the interrupt signal, the processor stops whatever its doing and calls a special piece of code called an Interrupt Service Routine (ISR). In this case the keyboard driver's ISR

would be called and it would read the key from the keyboard and put it in a queue where it can be later processed.

One of the primary goals of the ISR is to be fast so the processor can return to whatever it was doing before the interrupt occurred. Sometimes the ISR will capture just enough information about the event and schedule another special routine called a Deferred Procedure Call (DPC) which will finish the processing of the event a little later.

Nearly every device in your computer will be generating interrupts and invoking ISR and DPC routines – the keyboard, mouse, network devices, graphics card, hard disk, USB ports, timers, cameras and many others.

ISRs and DPCs are both special routines called in response to hardware interrupts. They must be executed quickly and in a timely manner so as to not miss information from the hardware and to not interrupt normal processing for too long.

Because of the special importance of these routines, Windows always runs them at a higher priority than everything else in the system – even higher than the real-time audio thread in your audio software.

Virtual Memory and Page Faults

Windows (like most other modern operating systems) uses a virtual memory system. A virtual memory system provides access to more memory storage than physically available in the computer by swapping sections of memory to disk when not in use.

These "sections of memory" are called pages. When Windows runs out of memory it will locate pages of memory that aren't in use, write them to disk and re-use the freed up space for other purposes. When the original page is accessed again the old page contents are read back from disk.

When a required page is not in memory and needs to be read back from disk this is called a page fault. Page faults are problematic because reading from disk is slow.

On Windows, there are two kinds of page faults – hard and soft. A hard page fault is one where the disk needs to be accessed to restore the page. A soft page fault is one where the page can be restored without accessing the disk (perhaps the page is mapped into another process or has been pre-fetched for other reasons).

It's the hard page faults we're most concerned about.

The Kitchen Analogy

If you're struggling to understand threads, interrupts and virtual memory the following analogy might help...

Imagine a typical restaurant kitchen.

In this kitchen the thing that gets the work done is the chef. The chef is like the the processor in a computer - it's the thing that does the work.

The chef's work is described by one or more recipes. Recipes are like programs - a list of instructions that the processor executes.

The kitchen provides a set of resources - stoves, ovens, pots, bowls, mixers, knives etc... A computer also has a set of resouces - memory, disk storage, keyboard, mouse, display etc...

If we now think about what the chef does in cooking a meal he basically works through a recipe from top to bottom performing each step.

A thread is everything related to the current progress of cooking a meal: which recipe, the current position in the recipe and which resources are in use - which stove top, bowl, oven etc...

A thread is not a physical thing in this kitchen but if the chef was to write down everything about the preparation of that meal - then that would constitute the "thread state".

Now consider what happens when the chef needs to cook two meals at once. There are now two threads. There's the current position in each recipe (or perhaps two positions in the one recipe if both orders are for the same meal) and a second set of resources in use. This is another "thread of execution".

The chef isn't necessarily busy the entire time a meal is being prepared. He might need to wait for something to cook before moving to the next step in the recipe. Similarly the CPU might start a disk read operation and need to wait for it to complete before continuing. What do the chef and CPU do while waiting? They leave that thread and go work on another.

A multi-core computer is like putting extra chefs in the kitchen. So long as all the chefs know the current "thread state" of each meal they can all contribute to getting the work done. There is an overhead to this however. Even though they can complete more work in less time there's extra work in that they need to regularly communicate, keep out of each other's way and someone needs to co-ordinate who's working on what.

Sometimes a chef will put something in the oven and set a timer. When the timer goes off it "interrupts" the chef who can return to that recipe. Timers, new meal orders, deliveries and returned meals are all analogous to processor interrupts.

Now lets really stretch the analogy to explain virtual memory...

What if the chefs in this kitchen have 50 ovens available to them but only 6 can fit in the kitchen at any one time? When they want to use a particular oven if it's not already in the kitchen they need to place an order to a service man to swap out that oven with another one.

That's just like a page fault. While the ovens that the chefs are using are already in the kitchen things run smoothly but when a chef needs an oven that's been moved out there's a lot of waiting involved (but at least he can go work on another thread while he waits and the service man will interrupt him when the oven is in place).

With 50 ovens the chefs can cook a lot more meals at the same time, but the process is less efficient. Expanding the kitchen to fit additional ovens is just like increasing the memory in your computer - it makes things run more smoothly.

Implications for Digital Audio Software

By now, you're probably wondering what all this has to do with your audio software. Well quite a lot. Let's think about what's happening under the covers when your audio software is running.

It all starts with your sound card and the audio driver software. When the audio driver is started, it creates a background thread to process audio to and from the sound card. This thread – typically called the "audio thread" is set to run at a high priority - higher than most other threads in the system.

When the sound card needs the next buffer of audio samples (roughly every 10ms or so), the audio driver wakes the audio thread which then calls into the audio software. The audio software is responsible for receiving the input audio buffers and filling the output buffers.

We know that if this process takes too long the sound will glitch so a well-designed program will be not only highly optimized to run quickly but more importantly it will carefully designed so that it never blocks.

The reason for this is that if it blocks, it gives up the rest of its time cycle and is then at the whim of the thread scheduler as to when it will get to run again. Meanwhile the sound card is desperately waiting for those audio buffers.

Windows' thread scheduler is extremely good at what it does. Even a poorly designed program will run perfectly fine for long periods of time. It's when circumstances arise where the audio thread isn't returned to quickly enough that you get one of those random unexplained glitches.

If we assume the audio thread is running at a high priority and that all our software is properly designed to not block, what could possibly go wrong?

Here are some scenarios:

- There is simply have too much work to do within one time slice. If it's too computationally expensive to complete the audio processing within one time slice there's a good chance the thread scheduler will pre-empt the audio thread and pass control to another.
- Poorly designed system drivers with slow ISR or DPC routines. Because these routines run at such a high-priority they can interrupt the real-time audio thread.
- Page faults – if the audio thread needs to access a page of memory that has been paged out everything will stall until that page has been re-read from disk.
- Power saving modes – most modern computers have various power saving modes. This might include shutting down parts of the system when not in use, or simply slowing

down the processors when load is light. The delay in resuming from these power saving modes can be enough to cause a glitch.

The rest of this book explains the things you can to do reduce the chances of these problems occurring.

Chapter 4

Power Management

Nearly every PC today ships with Windows configured to save power when not in use. In general this is a good thing and we should all be doing our bit for the environment. However, when it comes to real-time audio nearly all of these settings are detrimental to reliable audio.

In order to tweak any of these power settings, you first need to open the Power Options part of the Windows Control Panel. The quickest way to locate this is to simply click the Windows Start button and type "power options":

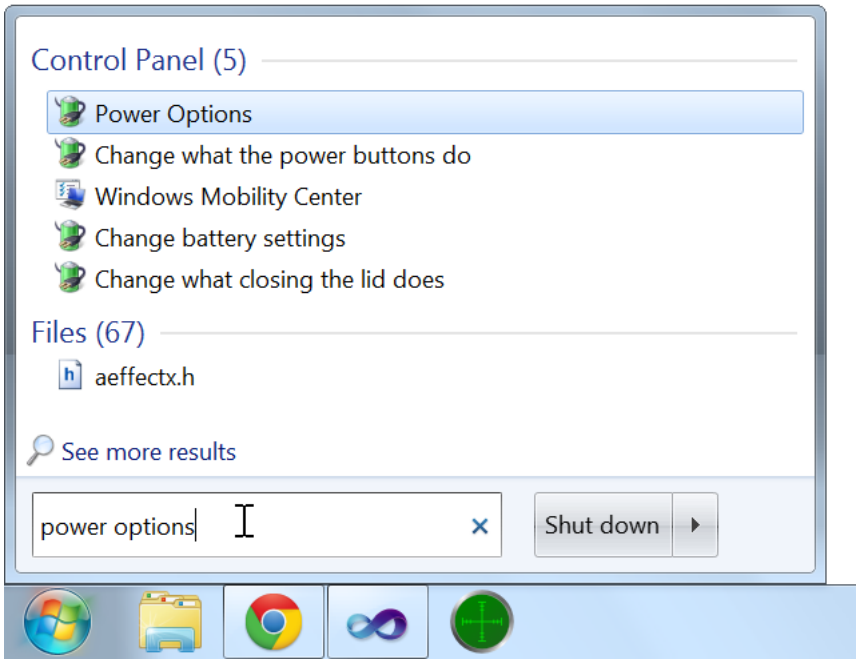


Figure 4.1: Opening Windows' Power Options

Once opened, switch to the "High performance" power plan:

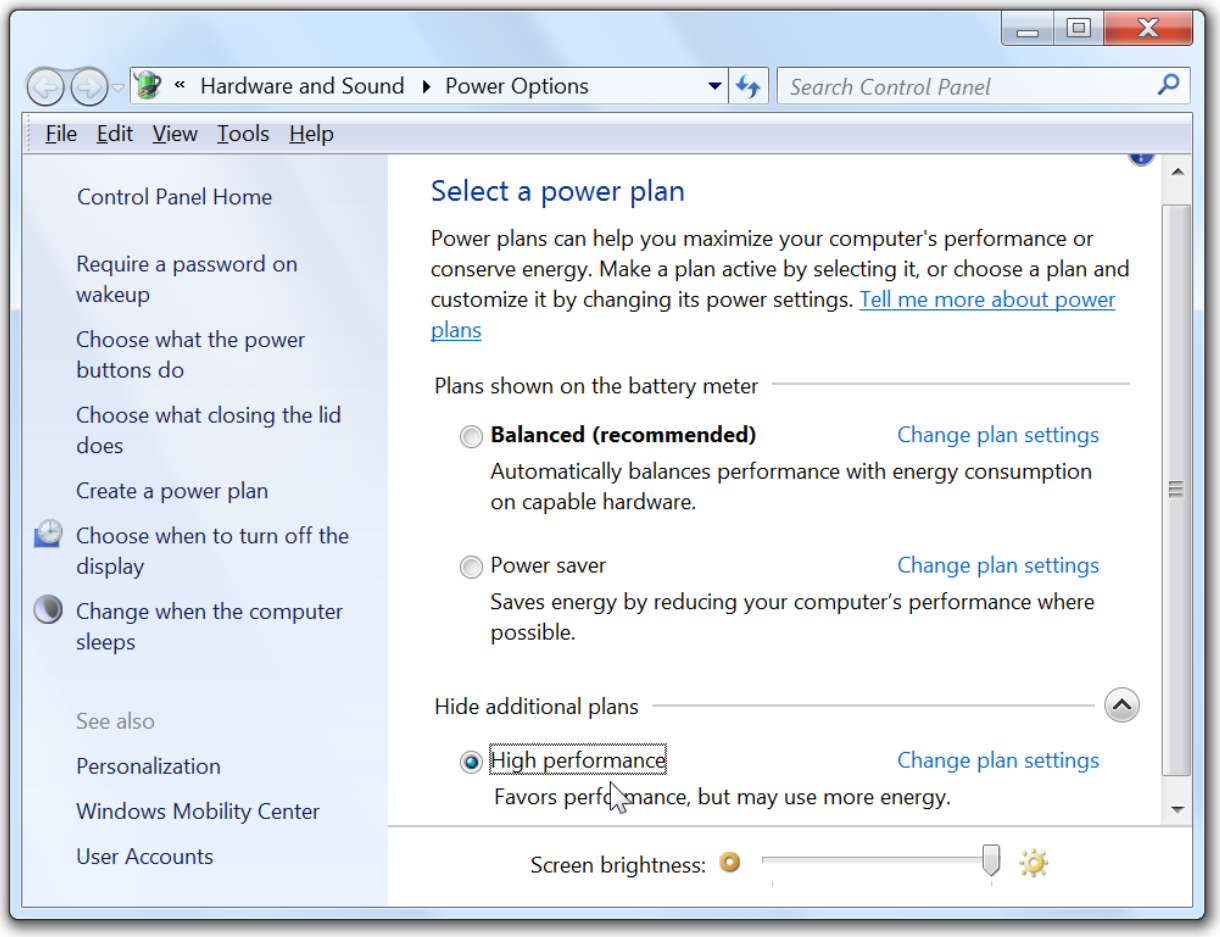


Figure 4.2: Selecting the "High Performance" Plan

Click the "Change plan settings" and set everything to "Never". If you're on a battery powered device you only need to adjust the fields on the column titled "Plugged in".

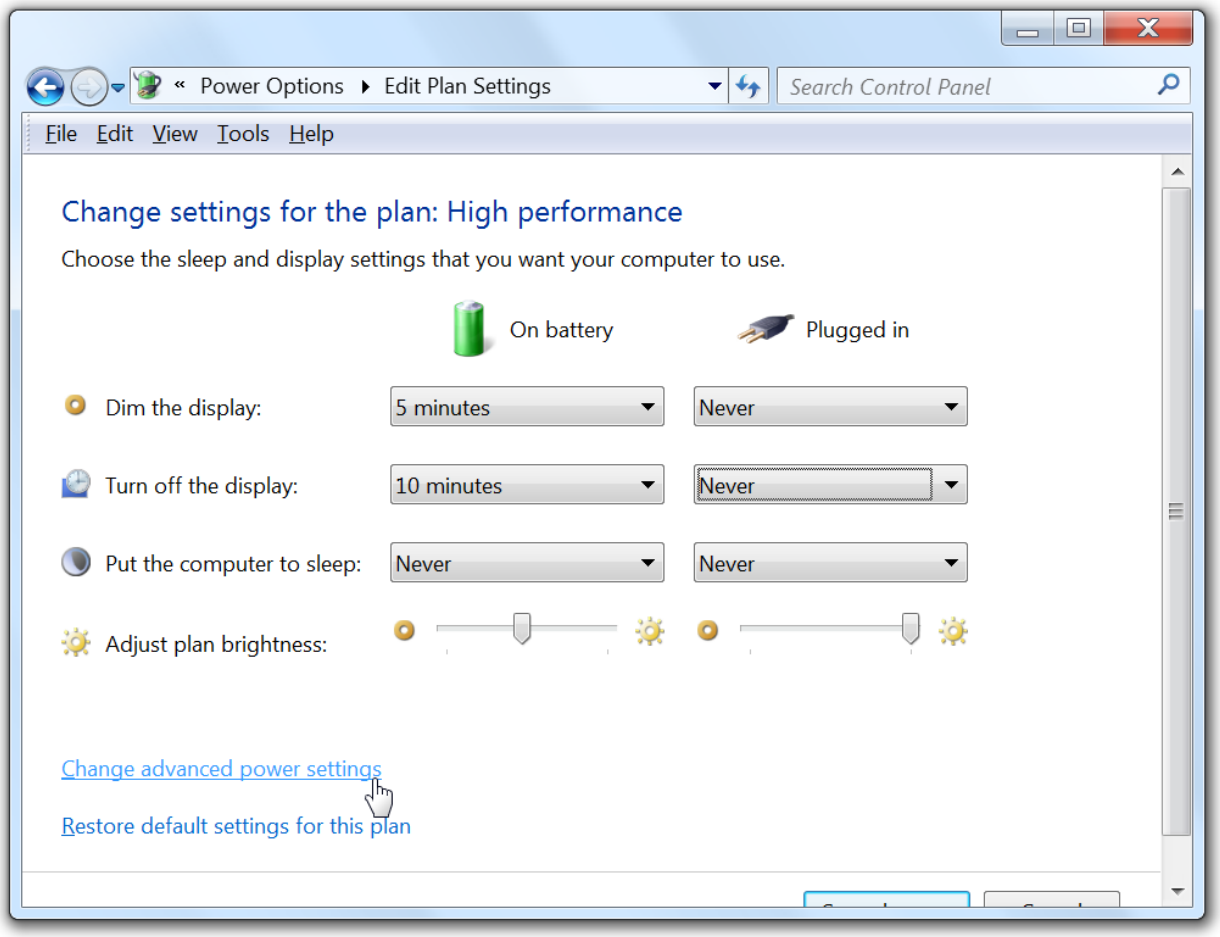


Figure 4.3: Accessing advanced power settings

Note: If you're using a portable device you'll have separate options for "On battery" and "Plugged in". In this guide I'll be assuming you'll only ever be using your computer for real-time audio while plugged in. You can make similar settings for when running on battery, but in general I don't recommend it.

Now click the "Change advanced power settings" link at the bottom left and work through the following sections to setup the power options as described.

Depending on your machine's hardware, you may not have some of the settings described. In this case you can generally ignore that section.

Hard Disk

There are a number of reasons why you don't want Windows to turn off the hard disk.

1. Mechanical hard drives can take a few seconds to spin up.
2. If you're recording audio to, or streaming audio from (including using large disk based sample libraries) you need the hard drive running continuously.
3. A hard page fault (discussed later) on a powered off hard drive can completely stall audio processing until it's fully powered up again.
4. Even for hard drives that you're not actually using during performance, just the act of shutting down or powering up a drive can stall other processes.

Set "Turn off hard disk after" to 0 minutes to prevent Windows from doing this:

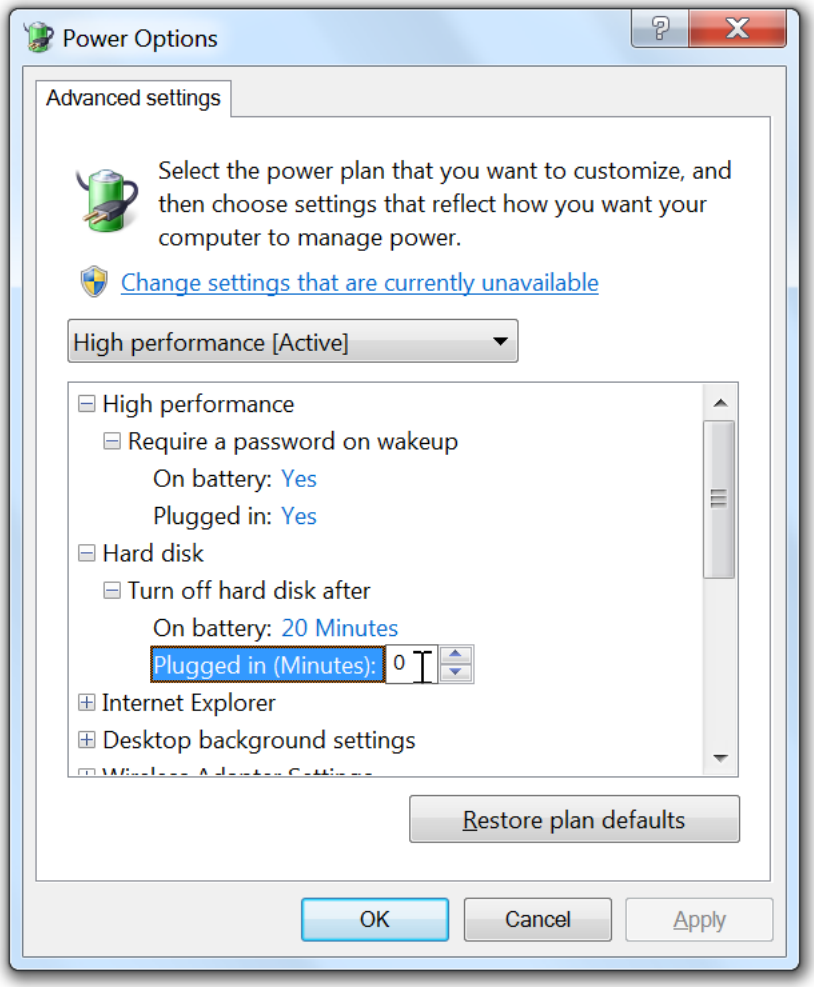


Figure 4.4: Hard disk power settings

Processor State

If you do nothing else from this entire guide, do not skip this one!

All modern CPU's have the ability to throttle down when not under load. There are various mechanisms that processors use for this including putting secondary processor cores to sleep, reducing the clock speed, slowing down the front side bus and more. All these strategies are designed to save power and modern laptops are getting remarkably good at conserving battery power.

The idea is that when load increases the CPU is throttled back up, contributes to getting the work done and then winds down again.

Power on tap, battery saving in the bank.

The problem is when you sit down at your keyboard and play that grand opening chord. All of a sudden the CPU has an awful lot to do – and it needs to do it right now - not when Windows gets around waking up the CPU.

Scroll through the Advanced Power Settings and locate the settings for "Minimum Processor State" and "Maximum Processor State" and make sure both are set to 100% - which basically means not to use any of these power saving mechanisms and to keep the CPU up and running ready to go.

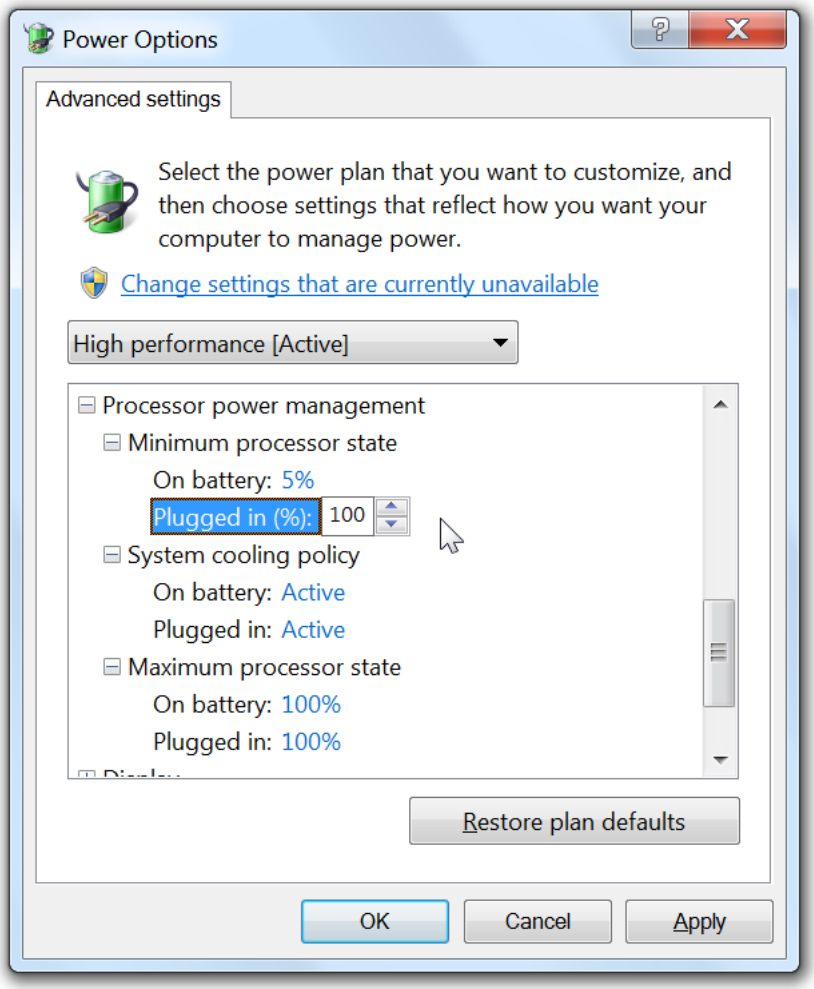


Figure 4.5: Processor Power Management

If you're running any modern PC and these options are missing, you might want to check with whoever setup or installed Windows on your computer and find out why. This setting really should be available and reviewed.

Desktop Slide Show

This one falls into the category of simply disabling background things that aren't needed. Desktop slide show periodically changes the background of the Windows desktop – typically cycling between various images. Unless you have some specific need for this, turn it off by changing this setting to "Disabled".

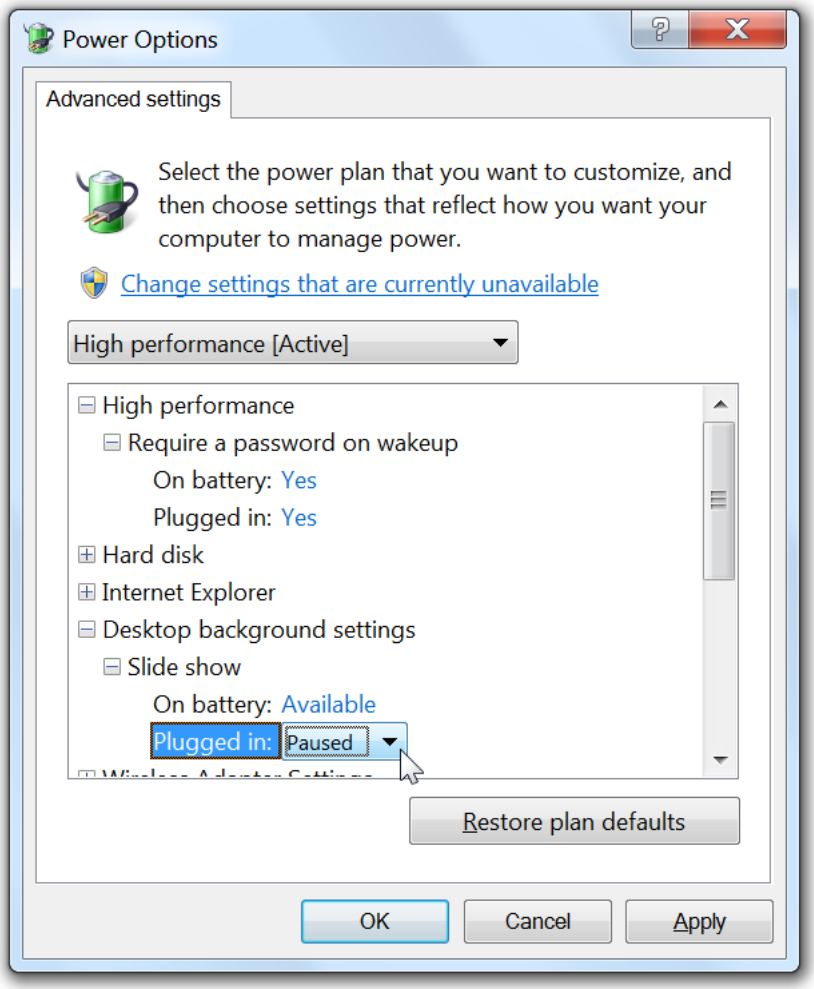


Figure 4.6: Slide Show Settings

USB Suspend

Almost every computer used for music performance is going to be using USB devices. Whether that's music related devices like audio devices, USB/MIDI keyboards or typical PC devices like a keyboard or mouse you really don't want these devices to be suspended while you're playing.

Even if you're not using USB devices, you still should make this change. In fact it's probably even more important– if you're not using USB it's more likely to be suspended and giving the device driver the opportunity to shut down could cause an undesirable stall.

In "USB Settings" -> "USB select suspend setting", change the setting to "Disabled".

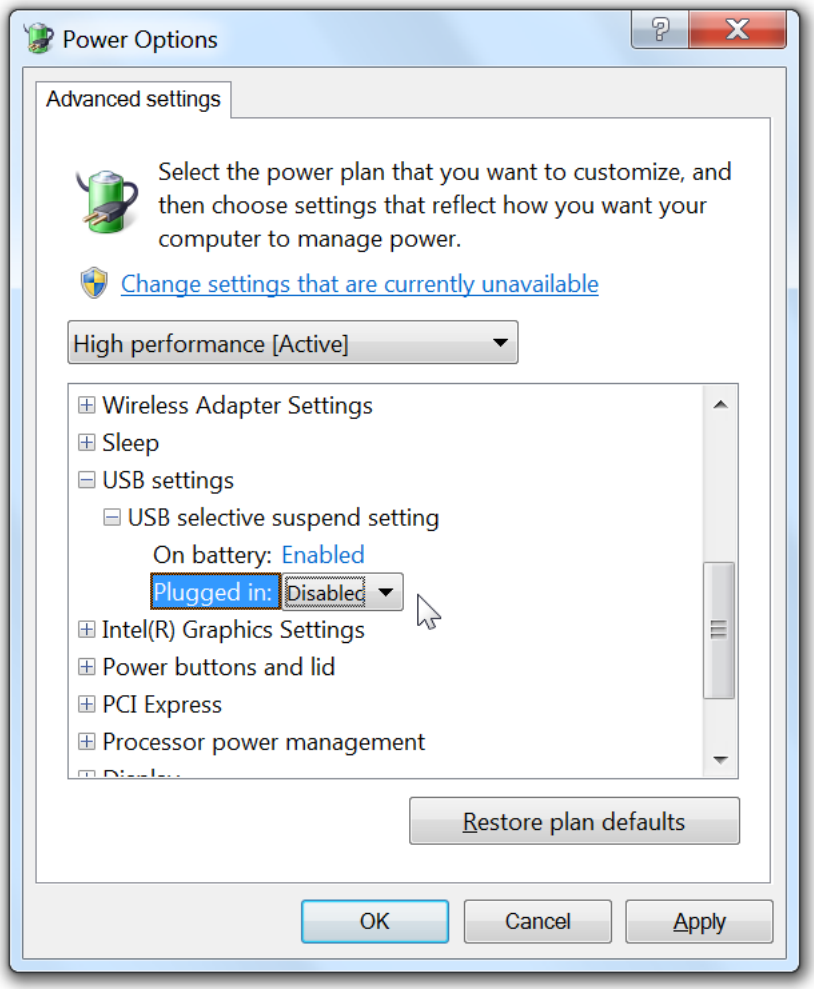


Figure 4.7: USB Suspend Settings

PCI Express Link State Management

PCI Express is the main high-speed system bus that connects the CPU to various other components in your computer. Notably the PCI Express bus is the main connection between the CPU and the graphics card.

The PCI Express standard defines various operating modes that control the balance between power saving and time to resume from a sleep state.

We're not interested in power savings, so set this to "Off".

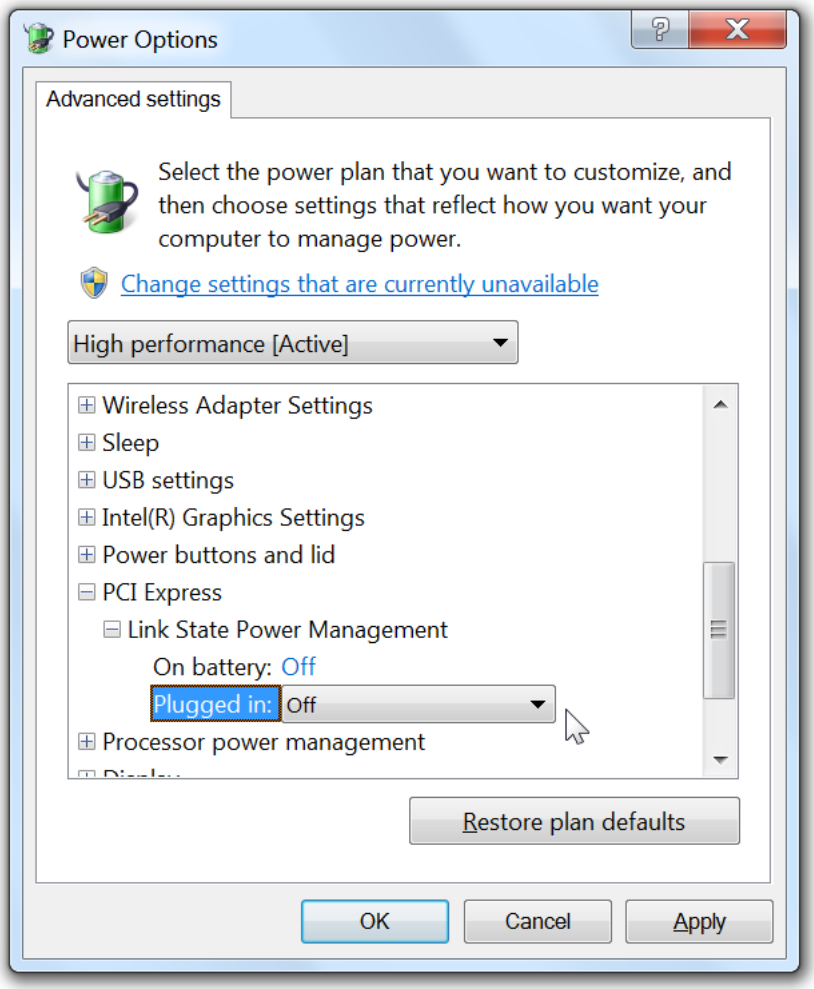


Figure 4.8: PCI Express Settings

Turn Off Display

The purpose of this setting is rather obvious – it controls whether Windows turns your computer's monitor off when not in use. This might seem a simple decision but it's actually a little more complicated than you might first think.

Often a music performance PC will be setup in such a way that it can be completely controlled via MIDI and the machine put away in the corner and the display simply isn't needed – so many users set the display to turn off automatically.

Unfortunately some display drivers cause some serious stalls both when turning the monitor on and off. I've seen at least one case where the entire system stalled for several seconds every time the display was turned on and off. I've not seen this in newer machines, but it's still something to be aware of.

There are a couple of possibilities here:

- If you use the monitor during performance, you don't want the monitor turning off – set this setting to 0 to disable shutting down the monitor.
- If you don't use the monitor and want it to turn off automatically, try setting this to a couple of minutes and see if causes any issue. If not, then leave it.
- If you don't use the monitor and automatically shutting it off does cause audio glitches, set this setting to zero and manually turn off the monitor yourself.

For machines where it's available the Dim display setting should typically be set to 0. If you want the monitor to dim then try enabling this option, but check it doesn't causes issues.

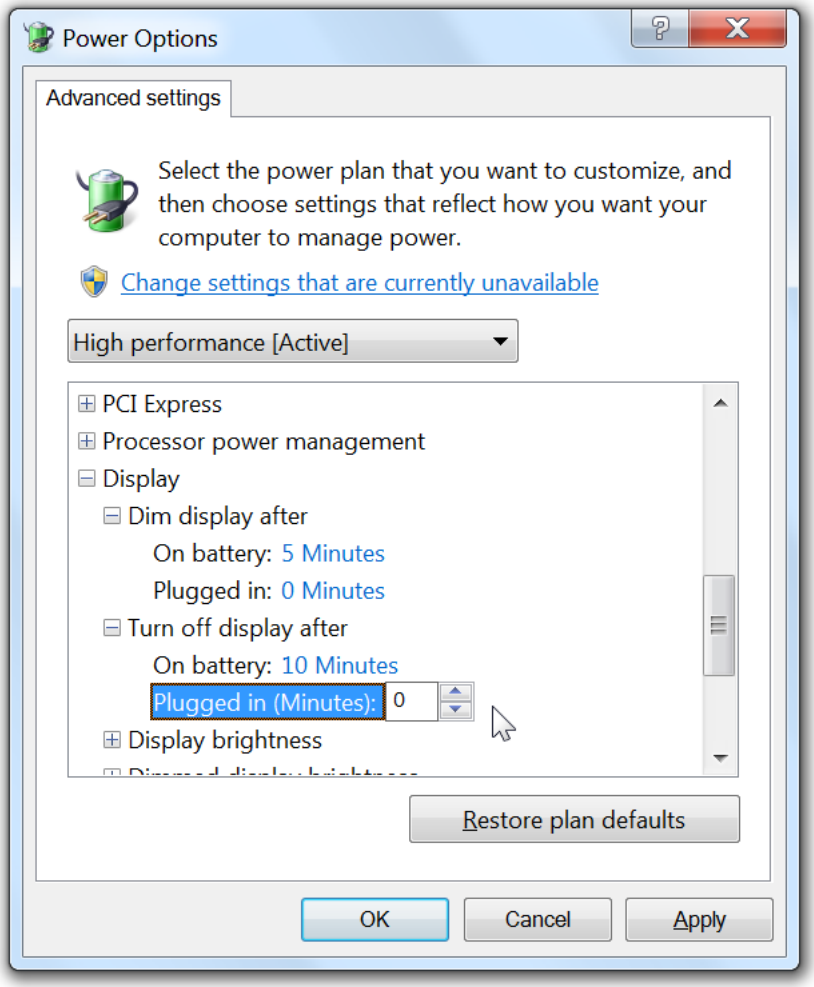


Figure 4.9: Display Settings

Sleep and Hibernate

The final set of power settings you want to check are the sleep and hibernate settings. You should really just set all of these so the machine never hibernates or sleeps.

If you're using the machine for live performance note that MIDI/Audio activity isn't detected by Windows and won't keep the machine awake. Unless you have some specific reason set all the sleep and hibernate settings to never.

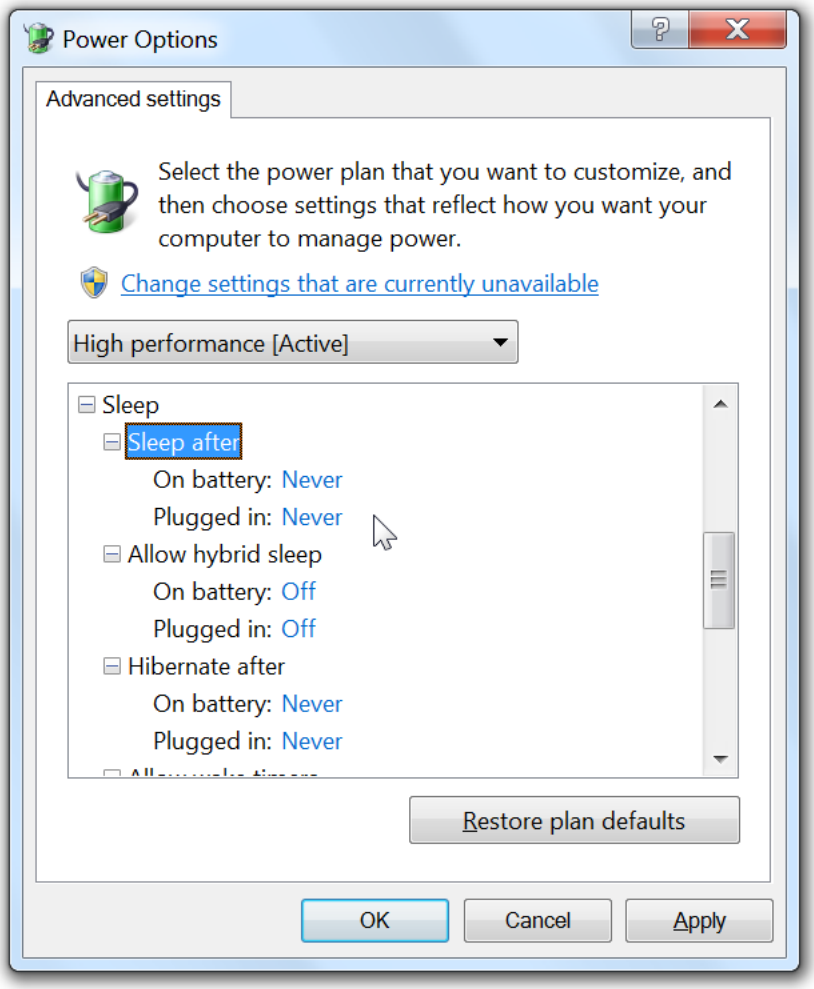


Figure 4.10: Sleep and Hibernate Settings

If you want to be able to manually put the machine to sleep, check the other power option setting titled "Power buttons and lid" and set one of the button actions to "Sleep".

One thing you'll need to check is that everything powers up again correctly and continues working – this is mostly dependant on the audio software you're using and whether it correctly stops and restarts itself, the audio driver and any MIDI devices on power sleep/hibernate.

Some Windows performance tuning guides recommend disabling the system hibernation file (Hiberfil.sys) but I've never came across a case where this was actually necessary for real-time audio performance.

USB Hub Power

The USB Hub and USB Root Hub devices have a setting that determines if Windows will power them off to save power.

The reasons for disabling this are essentially the same as for the "USB Suspend" setting in Power Options.

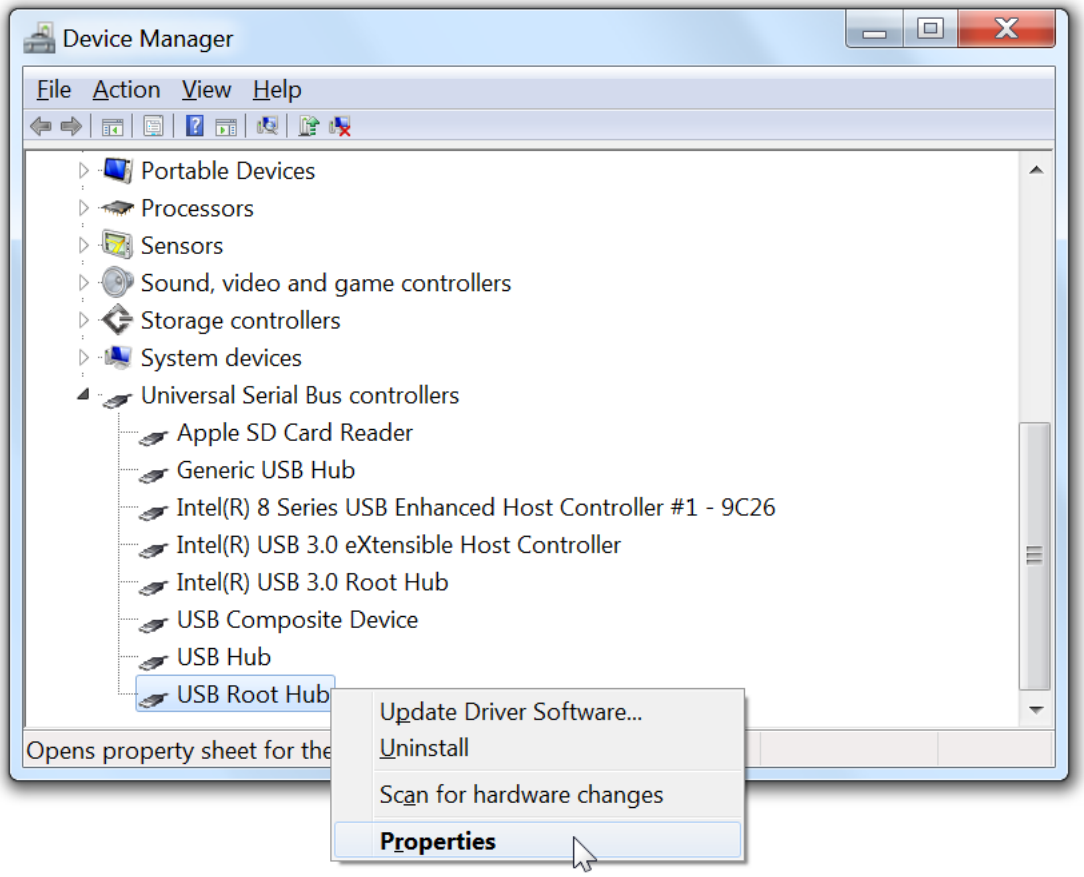


Figure 4.11: USB Hub Properties

1. Open the Device Manager, by clicking the Windows Start button, typing "device manager" and pressing Enter.
2. Scroll to the bottom of the list and expand the entry "Universal Serial Bus controllers".

3. Go through the list and right click on all devices including the words “USB Hub” or “USB Root Hub”.
4. Choose “Properties” from the popup menu.
5. Switch to the “Power Management” tab
6. Uncheck the option “Allow the computer to turn off this device to save power”
7. Repeat from step 3 for all hub devices.

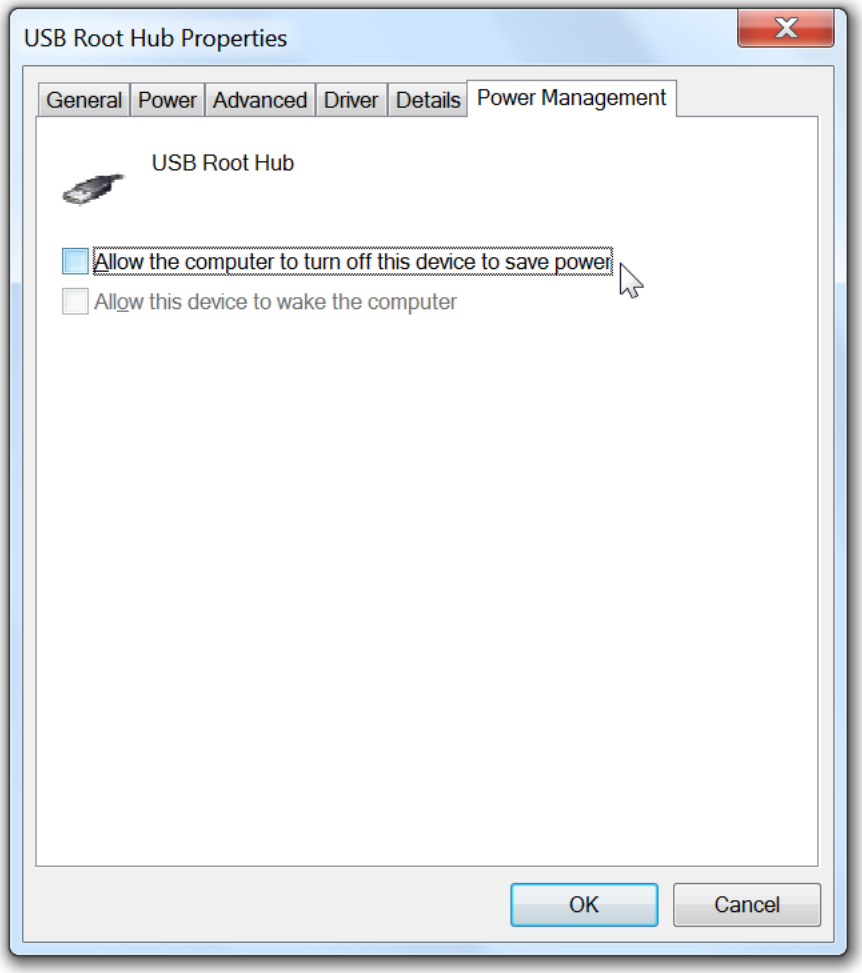


Figure 4.12: USB Hub Properties

Chapter 5

Other Settings & Services

This chapter looks at a various programs and background services that you might want to make sure aren't running while you're performing.

Hard Drive Compression and Indexing

If you're using any large sample libraries, or if you're recording or playing audio files you can gain some performance by disabling compression on the hard drives storing that media. Compressed drives take longer to read and write and consume CPU time to do the compression and decompression.

This is particularly important for sample libraries because not only are compressed drives slower to read, they're also slower to seek to particular location within a file – so sample players are going to be slower to locate the correct samples in those libraries.

Hard drives can also be indexed where Windows periodically scans the contents of files and builds an index to makes searching for files faster.

While maintaining an index in itself doesn't really affect performance, trying to build that index while also trying to stream audio to or from the same drive can. Unfortunately Windows tends to rebuild these indexes when it thinks the computer isn't in use such as when the keyboard and mouse isn't being used (regardless of whether you're playing audio or not).

To make sure these features are disabled:

1. Start Windows Explorer (Windows Key + E)
2. Select "Computer" (or "This PC" on Windows 10) in the left-hand panel:

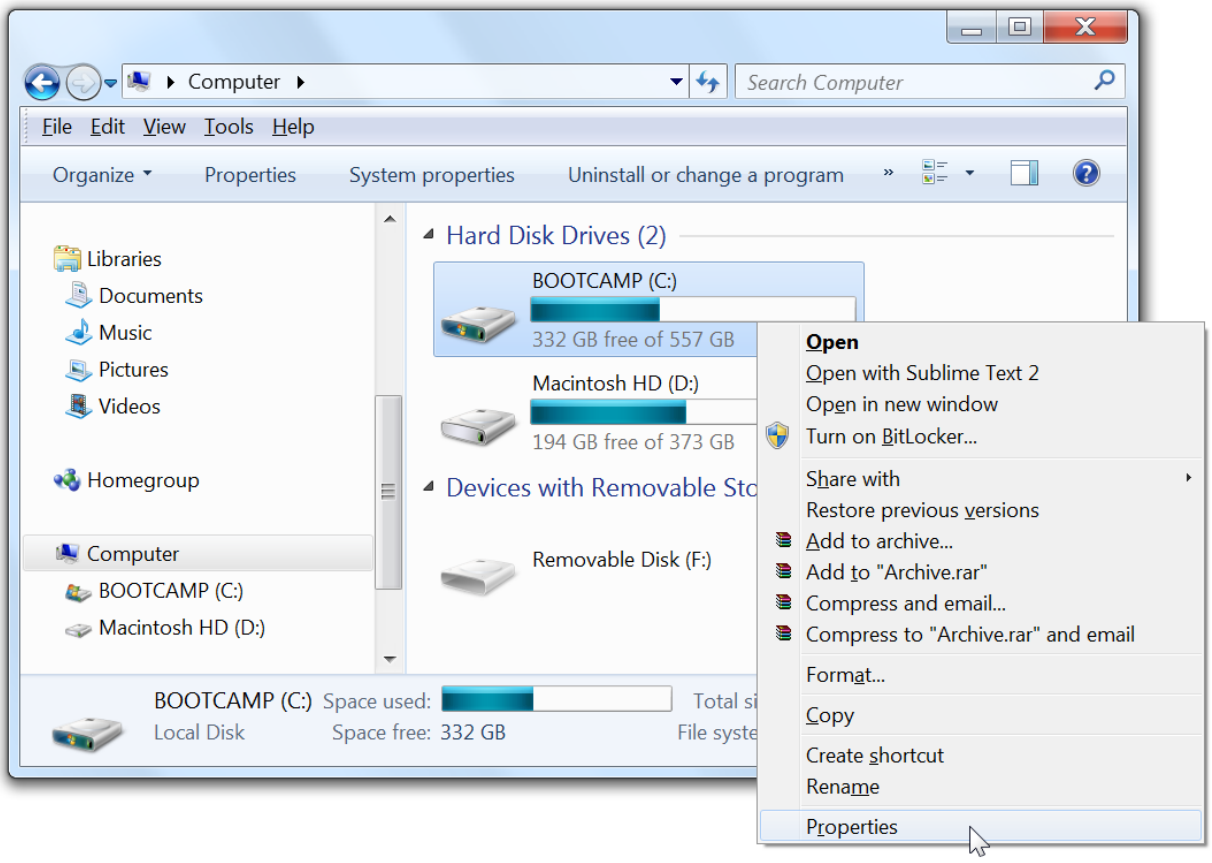


Figure 5.1: Accessing drive properties

1. Right click on each hard drive and select “Properties”
2. Turn off the two options at the bottom of the window:

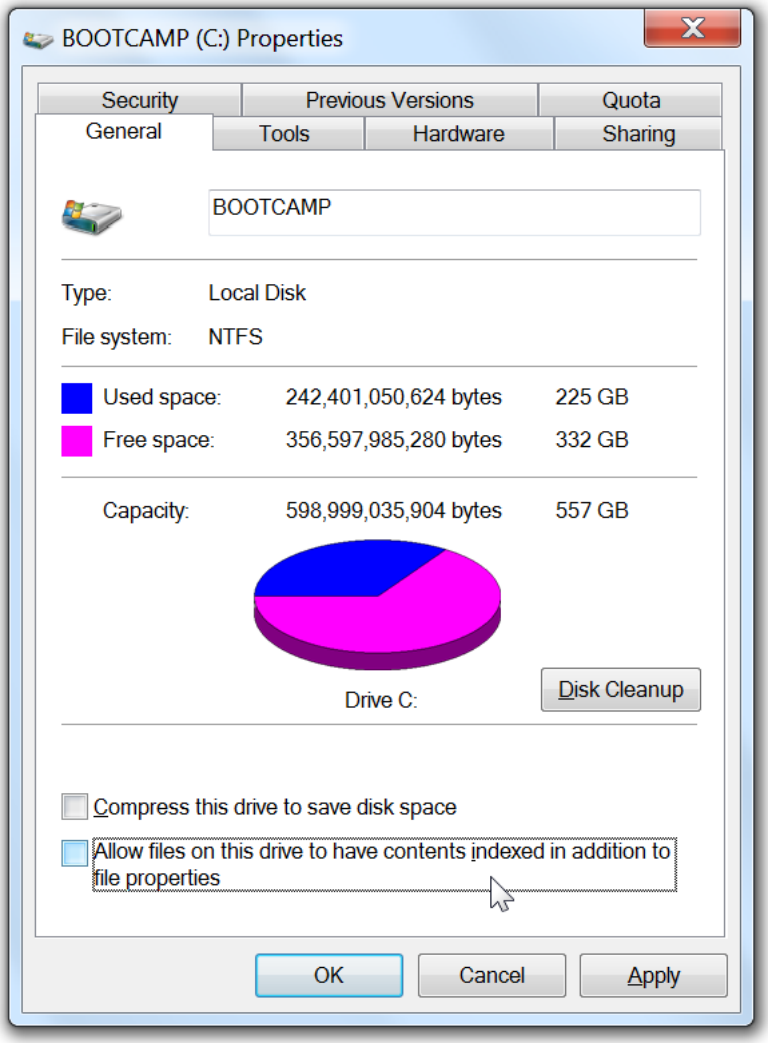


Figure 5.2: Drive Properties

Screen Saver

Screen savers are a bit old school these days, but if you do use one it's going to be creating unnecessary processing load CPU while you're trying to perform. Some screen savers can be very demanding on CPU.

The other reason for disabling the screen saver is simply because you might be using the screen during performance in which case you don't want it starting mid-performance.

To disable the screen saver:

1. Click the Windows Start button and type "screen saver" and press Enter.
2. From the drop down choose either "None" to keep the screen display turned on, or if

you're not using the screen during performance you can set it to "blank" – this will clear the screen without using any CPU.

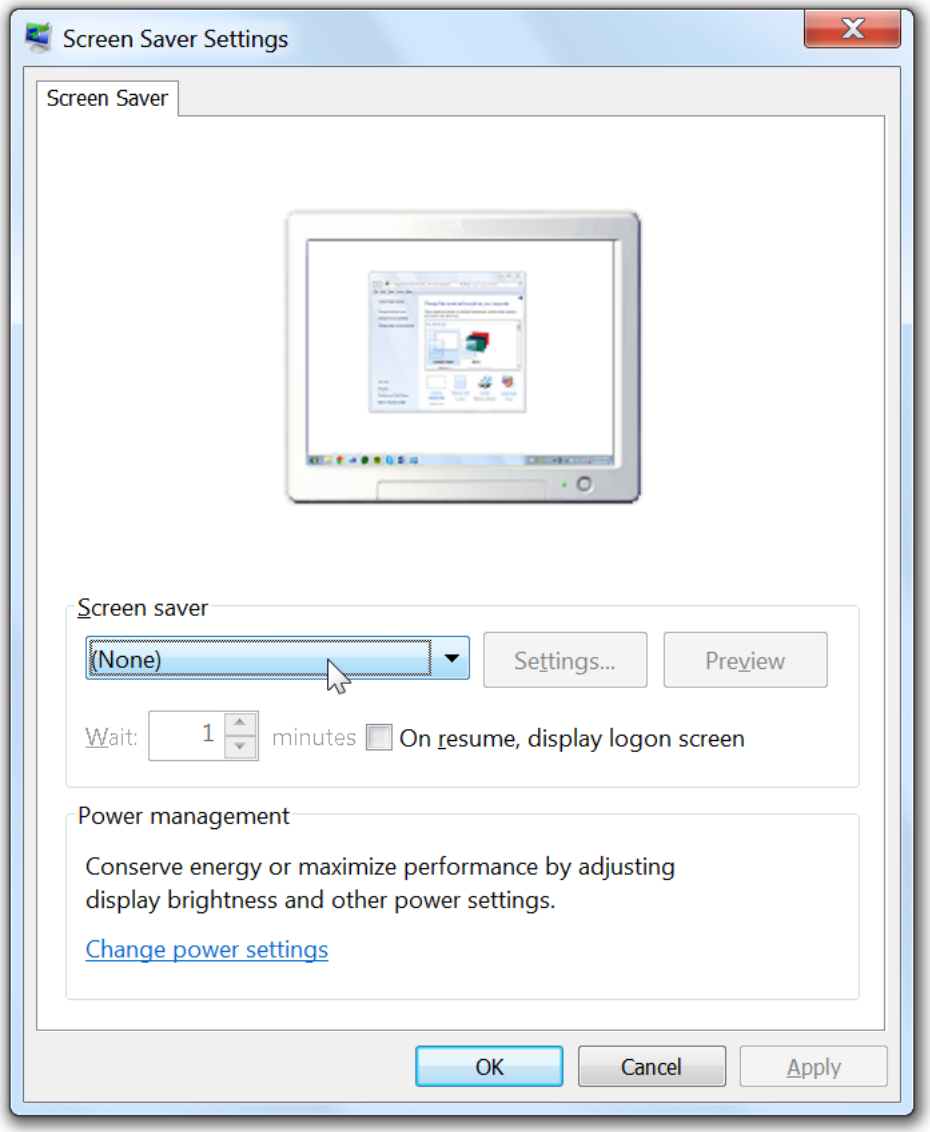


Figure 5.3: Screen Saver Settings

Visual Effects

By default Windows enables various "visual effects" to improve the appearance of Windows. This includes animations when hiding and show windows, drop shadows, better looking buttons and other controls and Windows 7's Aero Glass theme (the semi-transparent window captions and borders).

Although these effects do introduce some processing load, for any modern PC's most of this will be handled by the graphics card and the overhead is quite minimal – especially in terms of audio processing.

If you like these effects and they don't seem to be causing problems I don't think there's too much harm leaving them on. If you're trying to squeeze the best performance from your PC or if you suspect they're causing problems you can turn these features off.

To disable Windows 7 Aero Glass theme:

1. Right click on the desktop and choose "Personalize"
2. Scroll down and locate the theme "Windows 7 Classic" or "Windows 7 Basic".

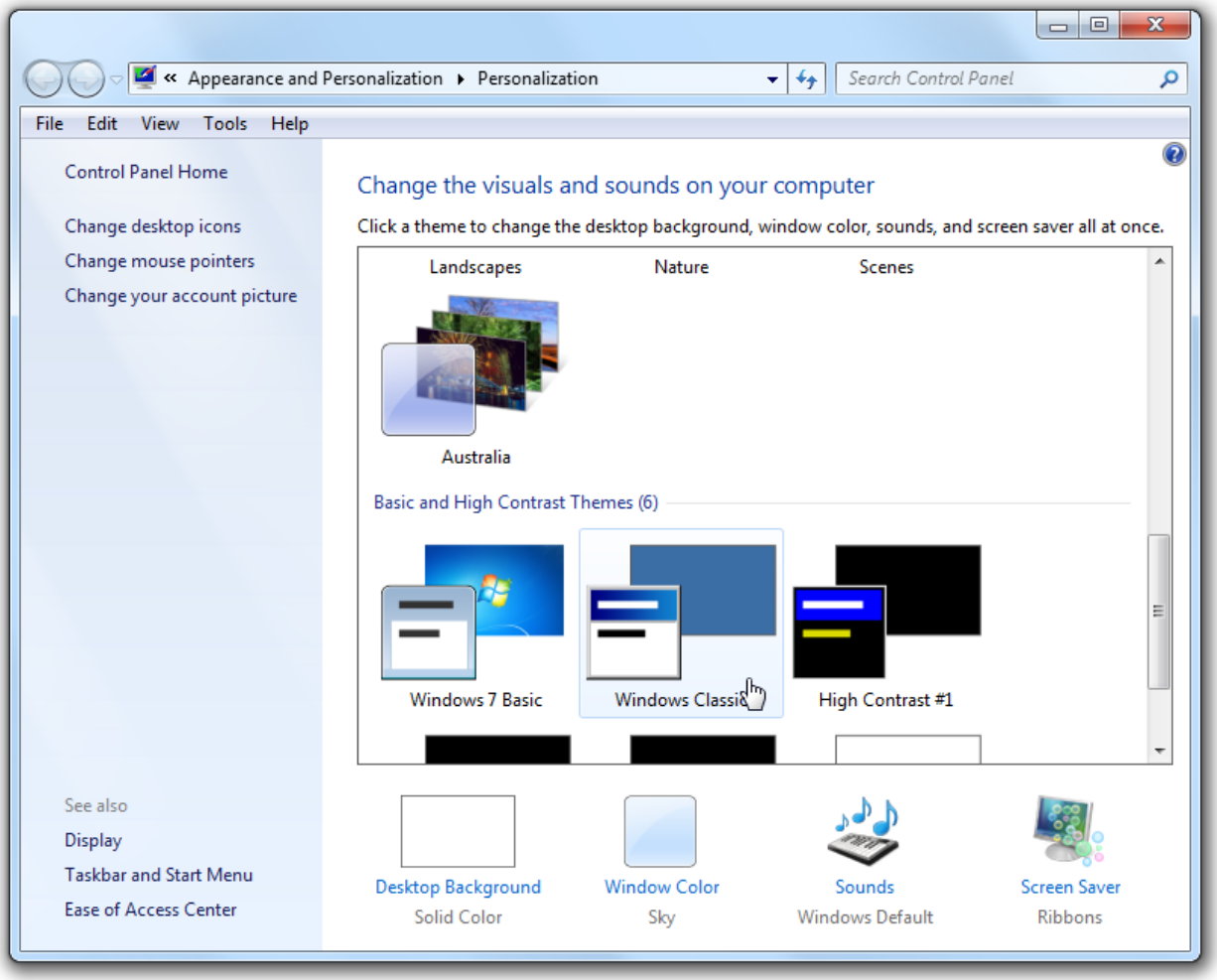


Figure 5.4: Disabling Aero Glass

To disable other visual effects:

1. Click the Windows Start button

2. Type “appearance” and choose “Adjust the appearance and performance of Windows”
3. Choose “Adjust for best performance”

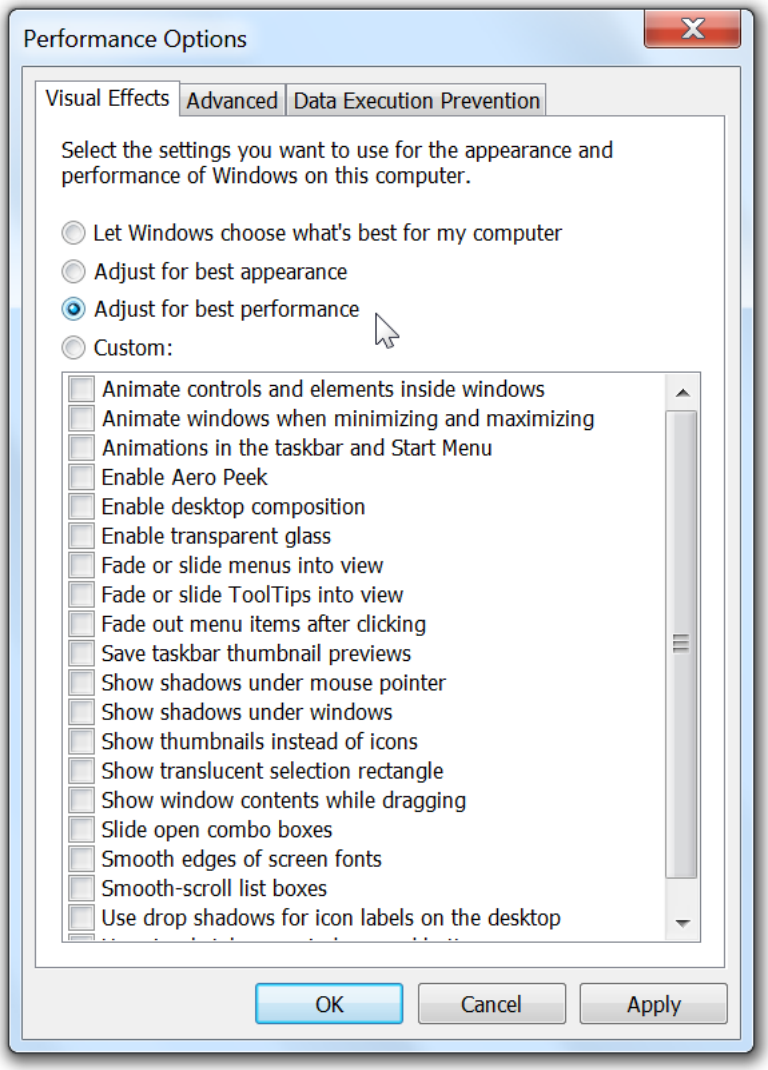


Figure 5.5: Disabling Visual Effects

Processor Scheduling

This is a bit of tricky one. As mentioned, Windows implements multi-tasking using time-slicing – it gives each program a tiny slice of time to run. By continuously cycling through all the running programs it appears as if they’re all running at once.

This Processor Scheduling setting controls the duration of each of these time slices. Longer time slices are better for background processing, whereas shorter timer slices can make the user-interface seem more responsive.

The question is which is best for audio performance? There's no correct answer – depending on what you're doing one might perform better than the other. It's like a double edged sword – longer time slices might give your audio software more time to process audio but they can also keep the processor tied up on other tasks and prevent audio processing.

It's probably best to start with "Background services" mode:

1. Click the Window Start button and type "advanced system settings" and press Enter.
2. In the "Processor scheduling" section select either "Programs" for shorter time slices or "Background services" for longer.

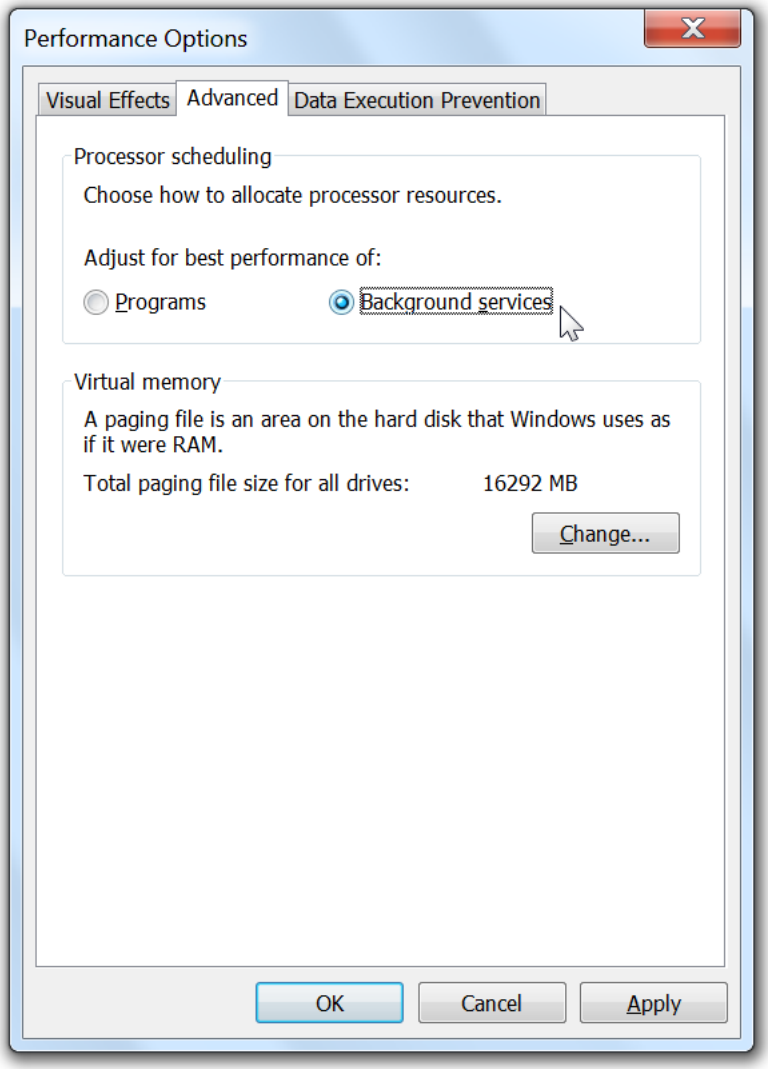


Figure 5.6: Processor Scheduling Settings

Core Parking

Core parking is feature available in some modern processors (eg: Intel i7 processors) in which entire CPU cores can be shutdown to save power.

This is good for power management but can affect real-time audio performance because there is a small delay when unparking which can lead to drop outs if that processing power is needed.

By default Windows hides the setting for core parking but it can be shown with a few tweaks to the system registry.

1. Click the Windows Start button and type "regedit" and press Enter to launch Window's Registry Editor program
2. Press the Home key to move the selection in the left hand pane to the very top
3. Press Ctrl+F to bring up the Find dialog and search for "dec35c318583" (without the quotes)
4. Once found, make sure the found key is related to power settings by checking the status bar – it should include "Control\Power\PowerSettings". If not, ignore it and repeat from step 3.
5. Double click the "Attribute" setting in the right hand panel and change the value to 0 (zero) as shown below:

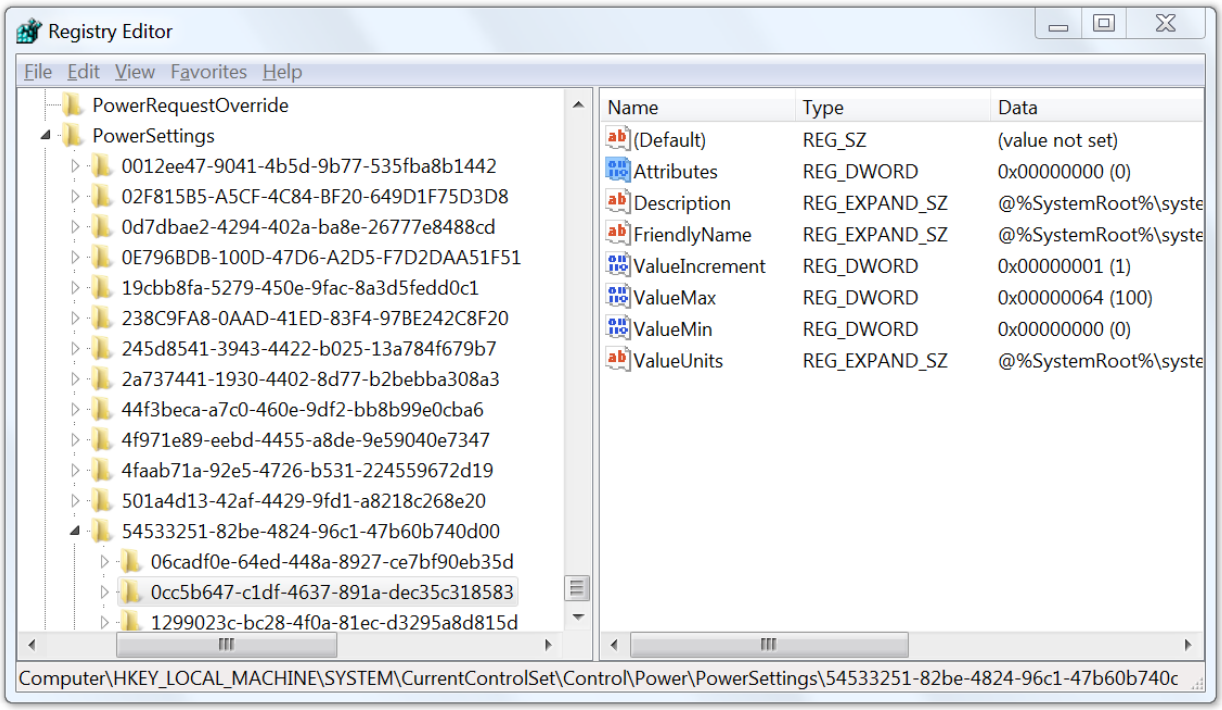


Figure 5.7: Registry settings to show core parking settings

1. Repeat steps 3-5 until all such entries have been changed (there may be several).

To be clear: it's only the "Attribute" values under the "0cc5b647-c1df-4637-891a-dec35c318583" keys that should be changed.

Once you've made these changes a new settings will appear in Power Options that will control core parking:

1. Go to Control Panel -> Power Options -> Change Plan Settings -> Change Advanced Power Settings
2. In the Advanced Settings window navigate to Processor Power Management -> Processor performance core parking min cores

The value you enter for this setting is the minimum percentage of processor cores that must kept running (not parked). Set this to 100% to prevent any cores from being parked.

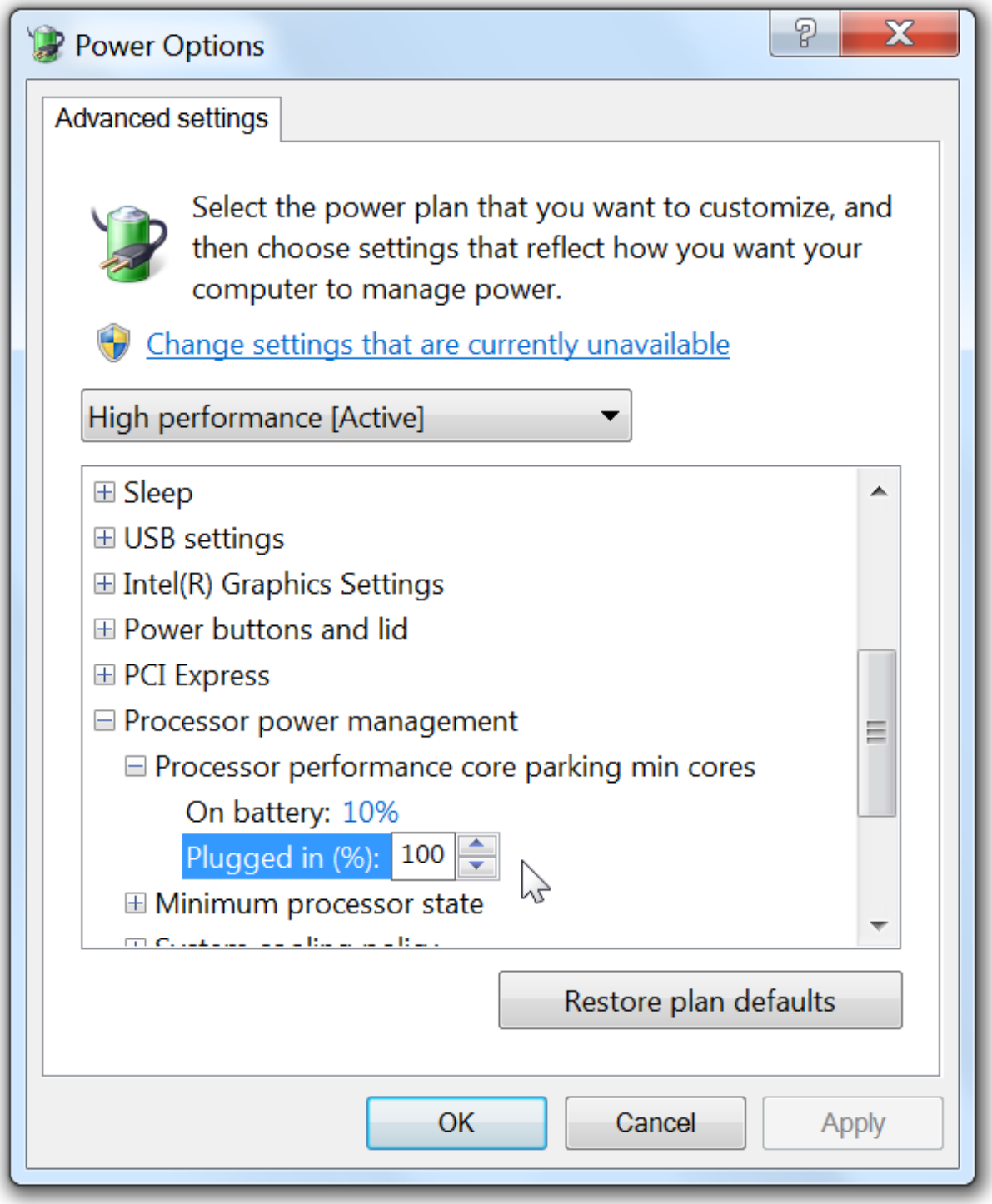


Figure 5.8: Core parking settings

If you don't feel comfortable editing the system registry, there is an alternative method for

disabling core parking:

1. Make sure the "High Performance" scheme is selected in Windows' Power Options
2. Click the Window's Start button, type "cmd", right click on the found "cmd.exe" and choose "Run as Administrator"
3. In the command prompt window that appears enter: (all on one line)

   powercfg -setacvalueindex scheme_current sub_processor bc5038f7-23e0-4960-96da-33abaf5935ec 100

and then:

    powercfg -setactive scheme_current

Note that this command actually changes the core parking and doesn't require showing the normally hidden option in Power Options.

Scheduled Tasks

Scheduled tasks are programs that are configured to be run at certain pre-defined times. For example your backup program might schedule itself to run at 9pm every night.

The idea here is to check these scheduled tasks to make sure there's nothing too intensive running while you're trying to use the computer for performance. You really don't want your backup software to start while you're in the middle of a gig.

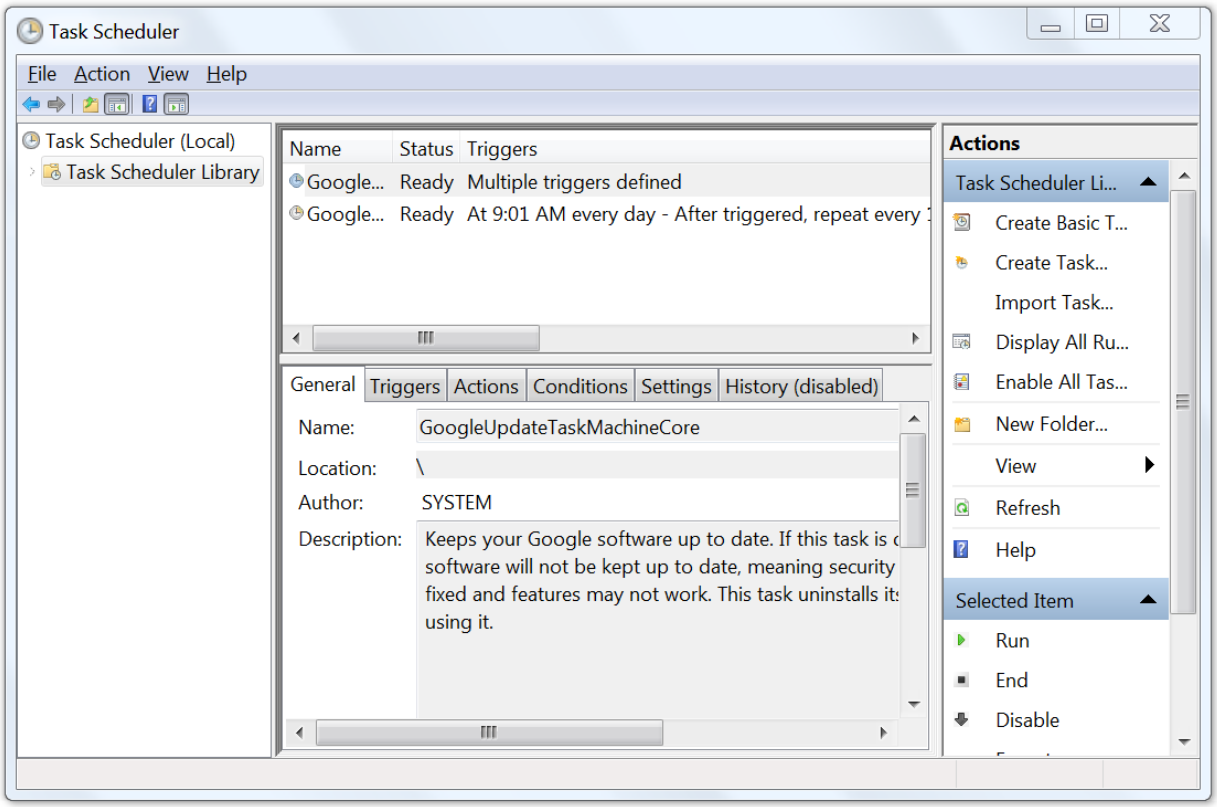


Figure 5.9: Task Scheduler

1. Click the Windows Start button and type "task scheduler" and press Enter to open the task scheduler.
2. In the left-hand panel, select the "Task Scheduler Library" entry. The middle panel will now show a list of scheduled tasks
3. Double click each entry and switch to the "Triggers" tab to check when the task is scheduled to be run.
4. Check each entry and make sure they won't cause the task to start when you're likely to be performing.
5. Expand the "Task Scheduler Library" node on the left hand side and review any other entries that you feel might affect performance. In particular I recommend checking Microsoft/Windows/Defrag and Microsoft/Windows/SystemRestore

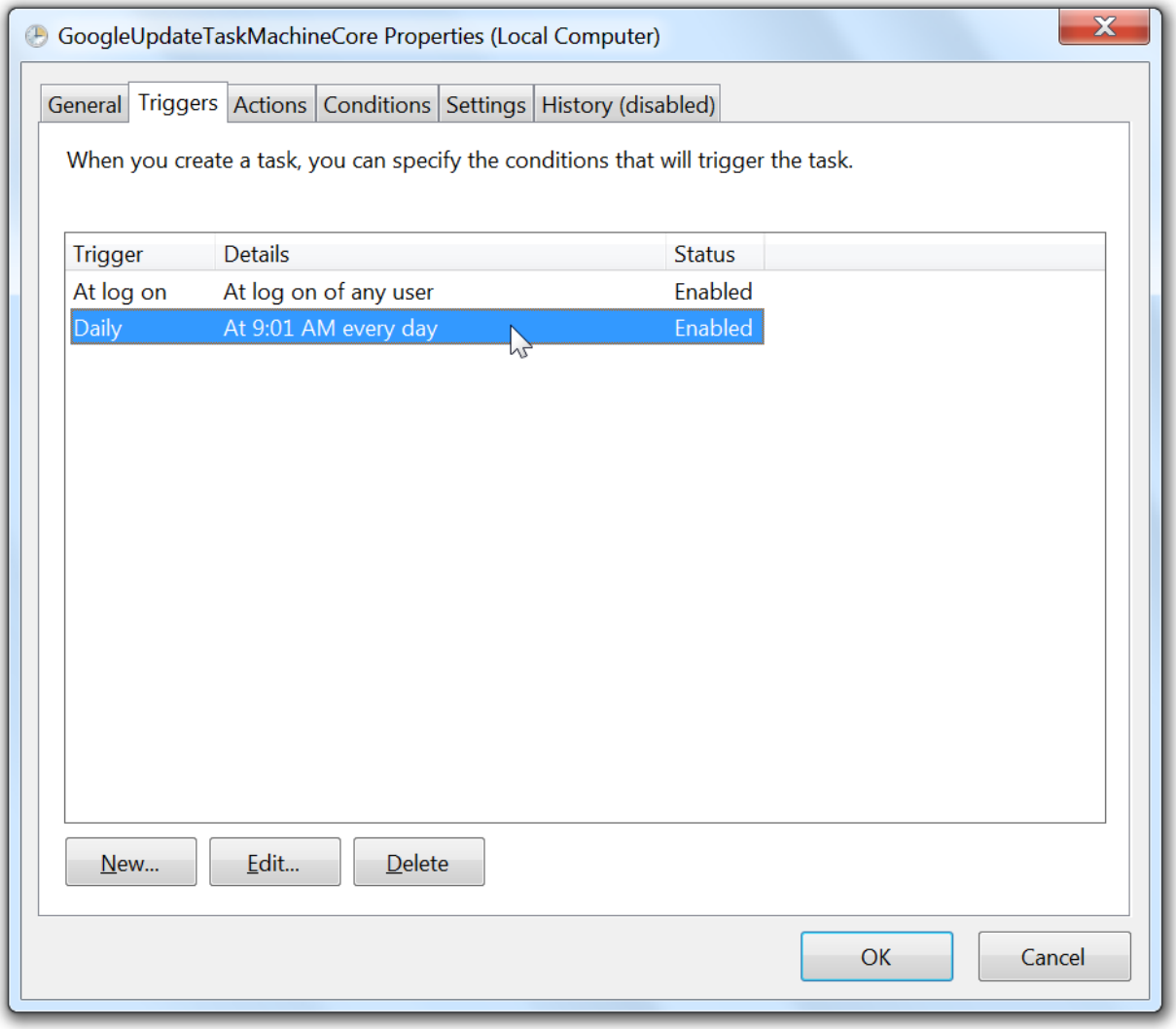


Figure 5.10: Scheduled Task Triggers

In general I don't recommend disabling or deleting scheduled tasks – the idea is simply to make sure nothing too intensive will run when you least want it.

The other thing to watch out for here is tasks that are configured to run when idle – if you're performing and not interacting with the screen, keyboard or mouse, Windows might consider the machine idle and start those tasks.

The other setting that might be handy here is the setting to "Wake the computer to run this task" - found on the Conditions tab. If you want to schedule your tasks to run at a time when you know you won't be performing but it's likely your computer won't be powered on, you can use this setting to wake the computer to run the selected task.

Finally, you should consider the impact of the "Run the task as soon as possible after a

schedule start is missed" setting on the Settings tab. If Windows can't start a scheduled task for some reason (perhaps your computer was turned off at the scheduled time), this option will cause it to run when it's next possible and that might be right when you turn the machine on and want to use if for performance.

Page File Settings

The Paging File is a special file that Window's uses for additional storage when physical memory starts to run low. The paging file is also referred to as the "swap file" because pages of memory are swapped between it and physical memory.

By default Windows is typically configured automatically manage the size of the paging file however you should consider setting it to a fixed size so it doesn't need to be resized while performing.

To configured the paging file:

1. Click the Window Start button
2. Type "advanced system settings" and press Enter
3. In the Performance group, click the "Settings" button
4. Switch to the "Advanced" tab
5. In the Virtual Memory group, click the "Change" button
6. Turn off "Automatically manage paging file size for all drives"
7. Click on the first drive in the list
8. Select the "Custom Size" radio button
9. Enter the desired initial and maximum size values (see below)
10. Repeat Steps 7-9 for each of the other drives.

You now need to consider how big the paging file on each drive should be:

- Always put the page file on the fastest hard drive. If you have an SSD drive you definitely want to put the swap file on that drive.
- You generally only need a paging file on one drive however you can get a very small performance increase by using multiple drives.
- A guideline for the total size of all paging files should be roughly 1.5 times the amount of physical RAM in your computer. Eg: 4GB Physical RAM = 6GB Paging File.
- If you have lots of physical RAM (eg: $>$ 8GB) you can usually get away with a smaller paging file. Eg: if you have 32GB RAM it doesn't much sense to allocate 48GB to the paging file (this might be a considerable fraction of an SSD drive).

- I don't recommend completely disabling the paging file unless you have a lot of memory and you're not using most of it.

By monitoring your total memory usage during a typical performance you can get a much better idea for how much memory you need. This estimate should take into account everything you might have running on the machine at any one time.

Once you've determined roughly how much memory you need you should ensure that the amount of physical memory plus the size of your paging file is greater than the required memory plus a generous margin.

Windows Update

Windows Update is a service that regularly checks with Microsoft for updates to the Windows operating system and associated software.

For a performance critical machine you want to make sure these changes are only done in a way that you have time to test the updated machine before using it live. In other words, at a time that suits you and not when Windows decides is a good time.

For this reason I recommend switching off automatic Windows Update checks:

1. Click the Windows Start button
2. Type "windows update" and press Enter
3. Click the "Change Settings" link on the left
4. Choose "Never check for updates (not recommended)"

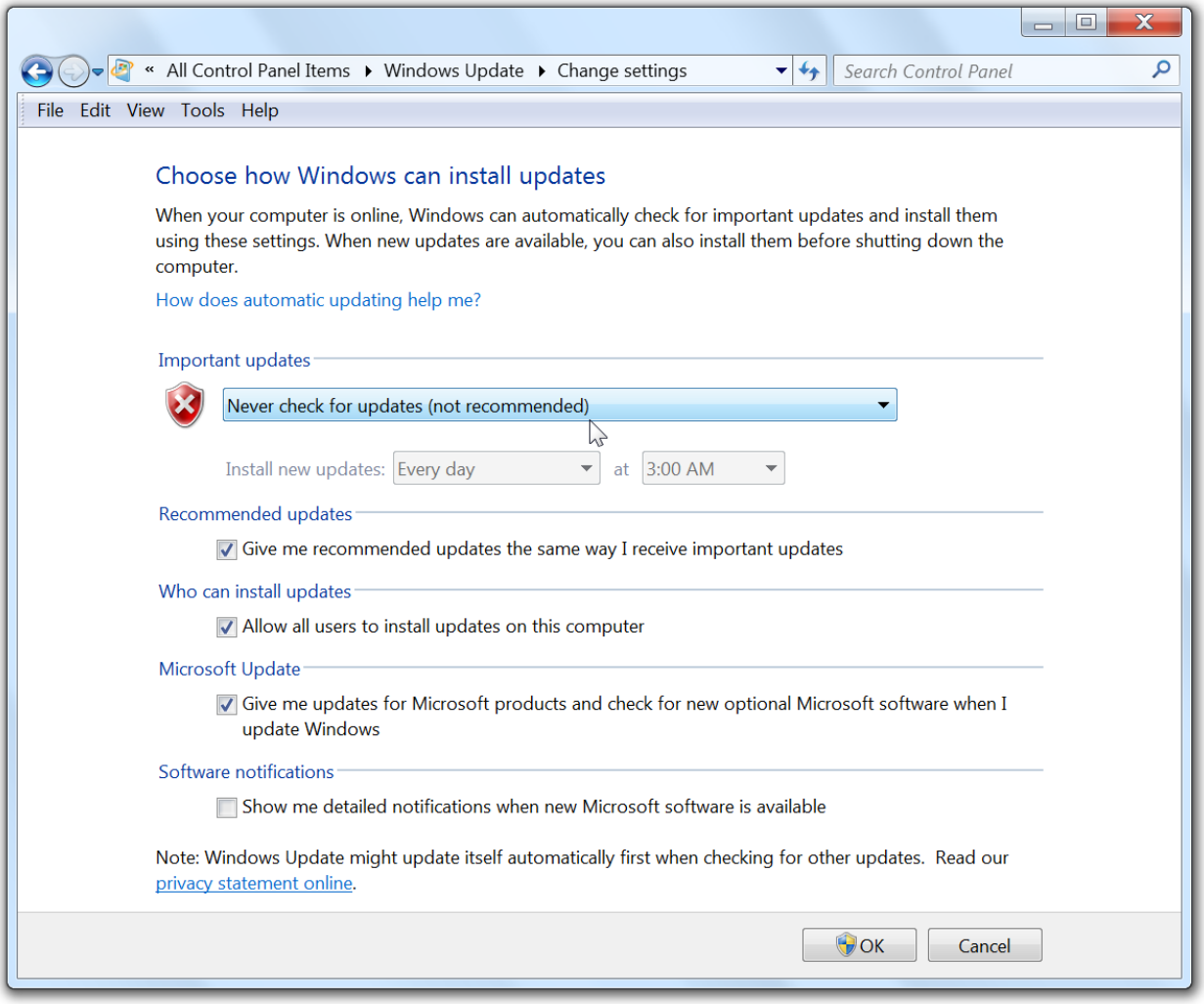


Figure 5.11: Windows Update Settings

If you're less concerned about automatic updates you might also consider choosing *"Check for updates but let me choose whether to download and install them"* but note that sometimes this will cause Windows Update to automatically update itself first – which you don't want to happen during performance.

If you disable the Windows Update Service don't forget to regularly check for and install important updates yourself using the Check for Updates and Install Updates buttons:

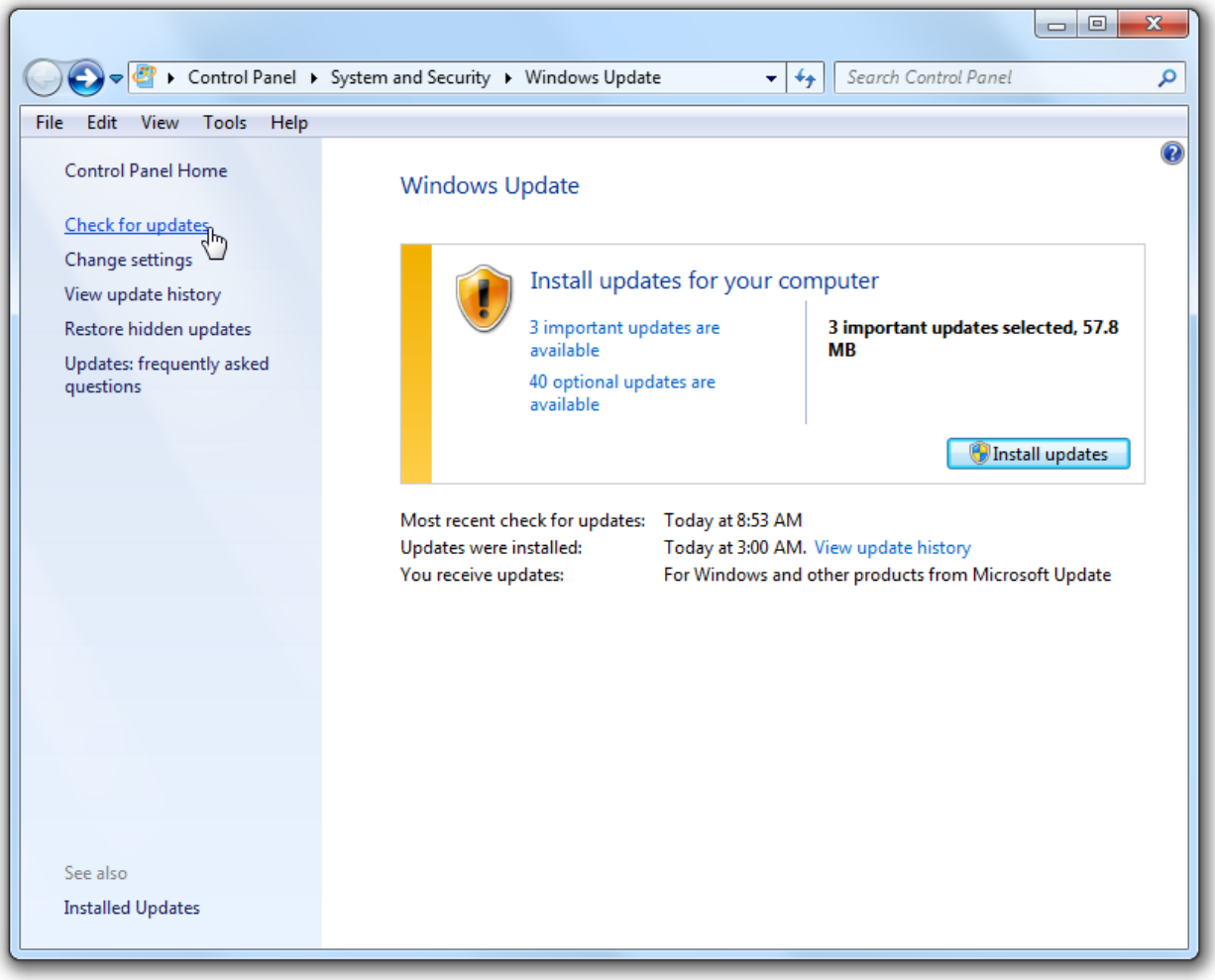


Figure 5.12: Accessing "Check for Updates" and "Install Updates"

Disk Defragmentation and System Restore

Windows is usually configured to automatically run Disk Defragmentation tasks and to create System Restore points.

Since these are both disk intensive activities you should make sure these are not scheduled to run while you're performing and is covered in the section Scheduled Tasks above.

4GB Tuning on 32-Bit Windows

This setting only applies if you're running the 32-bit (x86) version of Windows and only needs to be changed if you have more than 2GB RAM in your computer.

Normally 32-bit Windows only makes 2GB of memory available to programs, reserving the other 2GB of address space for Windows itself.

There is a boot option that can change this to give programs access to 3GB. When using many plugins, or large sample based plugins this extra 1GB of memory can significantly improve performance and reduce page faults.

To make this change:

1. Click the Windows Start button
2. Type "cmd", right click on the found "cmd.exe" entry and choose "Run as Adminstrator"

Enter the following into the displayed command prompt window:

```
bcdedit /set IncreaseUserVA 3072
```

Reboot your computer and you should now find that compatible programs have access to the additional 1GB of RAM.

Note that this also requires your audio software supports it. Most audio programs do but if you're not sure check with the software's developer and ask if it is "Large Address Aware".

Anti-Virus and Anti-Malware Software

Nearly all anti-virus software periodically performs a full scan of your system – that is it goes through every file on your computer and checks for viruses. Although you don't want viruses on a machine you're relying on for reliable performance, you also don't want to run a full system scan while performing.

There are too many virus scanners to cover in this guide, and you should refer to the software's documentation for details on how to configure it.

Here are some notes and tips though:

- Most virus scanners include an icon in the system tray (the icons in the bottom right corner of the screen). You can typically double click the icon to get to the scanner's settings.
- Check the settings for when a full system scan is scheduled to run. Similarly to Scheduled Tasks above – you want to configure this to run when you're unlikely to be using the computer for performance.
- Some virus scanners have the ability to scan files when they're accessed. This is generally not a good idea for our purposes (do you really want to virus scan your VST plugins and media files when loading a session into your audio software? Probably not if you're on stage). If you disable this feature, make sure you run a regular full scan.

Virus scanners are a must for any machine being used for real-time performance – the impact of a virus can be far worse than overhead of the virus scanner. We just want to make sure that overhead is not at an inconvenient time.

Other Software

If the machine you use for real-time audio work is also a general purpose machine that you use for regular daily work there's a good chance you have lots of other software installed. It's a good idea to run a quick check through everything that runs in the background and consider the implications that might have on real-time audio performance.

Here are some tips for finding software running in the background:

1. Close all open windows and programs.
2. Go through all the icons in the system tray – some background programs add an icon here for convenient access. E.g. the Dropbox software does this.
3. Bring up the Task Manager (press Ctrl+Shift+Escape), switch to the Processes tab and click the button "Show processes from all users". Run through the list and see if anything stands out. If you're not sure what something is, right click on it and choose "Open File Location". The name of the folder where the program resides might give a clue as to what it is.

Some common things to look out for include:

- Instant messaging and conferencing software (e.g. Skype) can both interfere with your audio configuration (if they try to use the same audio devices) and open you to the risk of an incoming call or message while you're performing.
- File synchronisation tools (e.g. Dropbox) – particularly if you're using them to sync audio files. You don't want these tools running while you're performing. Often these tools have a pause feature available from the system tray that I recommend using while performing.
- Other media players (e.g. Windows Media Player, Spotify, Pandora, iTunes) might contend for the audio gear on your computer and/or might download content in the background.

If the machine in question is dedicated to on-stage performance, it's generally a good idea to simply uninstall anything you don't absolutely need.

System Sounds

System sounds are the sounds that Windows plays when various events occur - the sound when a message box prompts you to save a file, critical error message sounds, notification

popup sounds, device plugged in sounds etc. . .

If you're doing live performance you'll almost certainly want to disable these sounds. Another thing to consider is the ways these sounds might interfere with the audio driver support in your audio software – I've seen a couple of times where the audio software failed to start correctly because a message box was displayed during start up, which played a sound and made the sound device unavailable to the audio software.

The easiest way to disable these sounds is to select the No Sounds scheme:

1. Click the Window Start button, type "change system sounds" and press Enter
2. From the Sound Scheme drop down, choose "No Sounds"

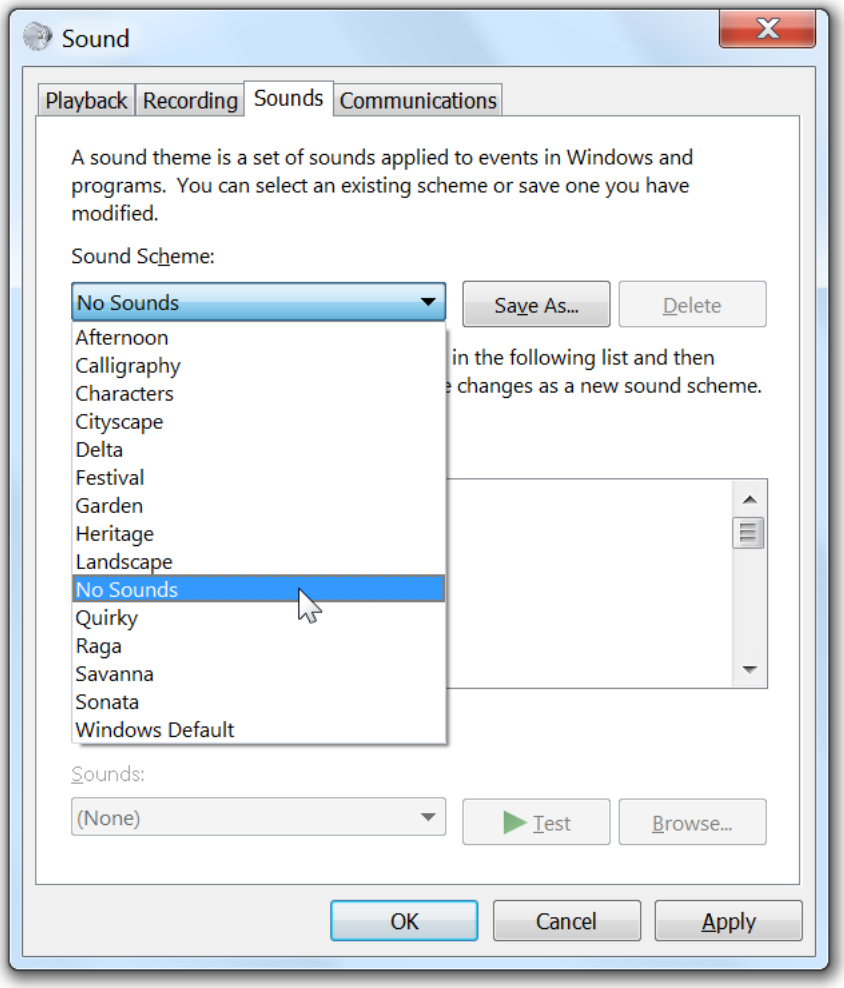


Figure 5.13: Selecting the "No Sounds" scheme

Multiple Audio Devices

If you have a dedicated sound card for audio performance its worth disabling the built in sound device, or other sound cards if you're not using them.

Sound card drivers are notorious for causing DPC latency issues. Also disabling them can also simplify the setup of your audio software since the ability to select these devices will be removed.

1. Click on the Windows Start button, type "device manager" and press Enter
2. Expand the group named "Sound, video and game controllers"
3. Right click on any sound cards that you don't need and select "Disable" from the popup menu

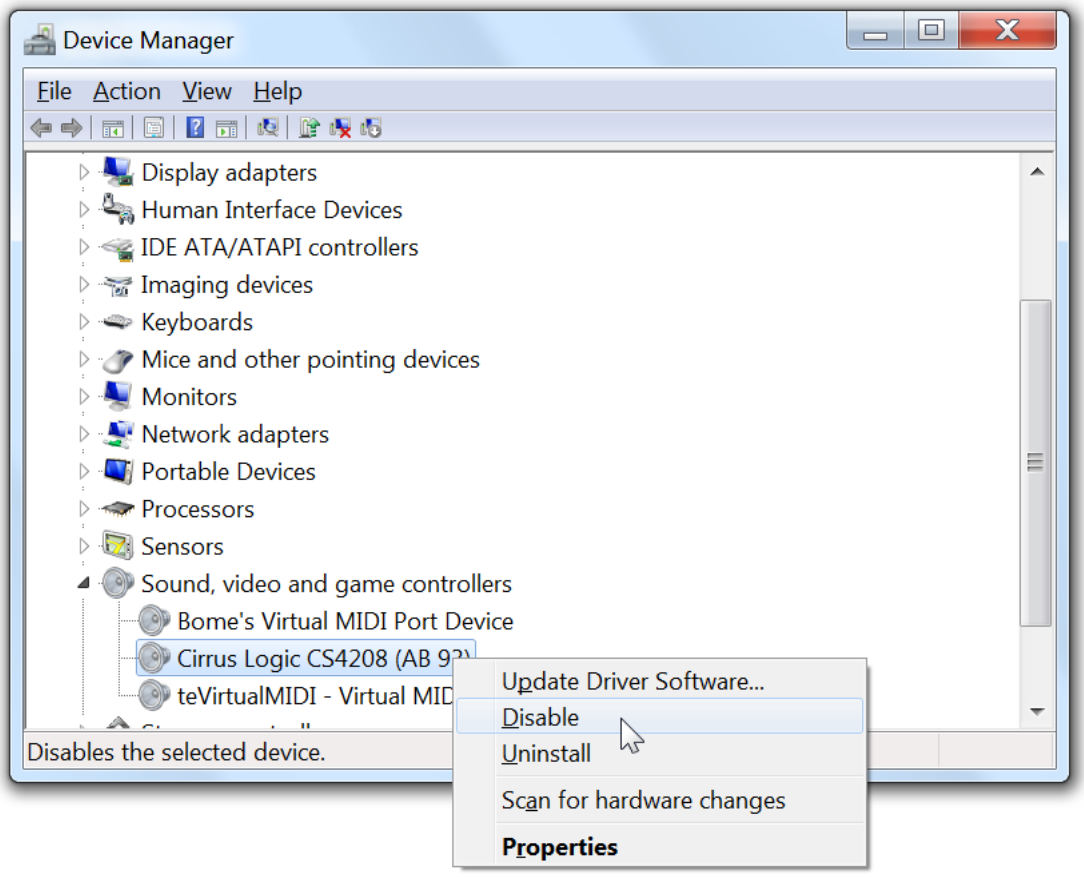


Figure 5.14: Disabling unused sound devices

Firewall

Some performance tuning guides recommend disabling the system firewall. In general I don't recommend doing this for two reasons:

1. Any well designed audio application won't be accessing the network from the audio thread and any latency introduced by a firewall is not going to have any effect on real-time audio processing.
2. Disabling the system firewall is generally just a bad idea and leaves you open to all sorts of attacks, some of which might have a bigger impact on performance than the firewall itself.

Of course there are always exceptions. If you're using network based MIDI protocols, you're finding the latency unacceptable and you're certain the machine is isolated from malicious attacks then go right ahead.

Disabling Nagle's Algorithm

If you're using any real-time network protocols while performing (e.g. network based audio or MIDI protocols) you can reduce the overall network latency by disabling support for Nagle's algorithm.

Nagle's Algorithm is a technique that combines network packets together to reduce the total network load. To do this however it needs to delay slightly before sending a packet to see if there are any others that it can be combined with and this introduces additional latency of most packets.

To disable Nagle's Algorithm requires modifying the system registry:

1. Click the Window Start button
2. Type "regedit" and press Enter to launch the system Registry Editor
3. In the left hand panel navigate to: HKEY_LOCAL_MACHINE -> SYSTEM -> CurrentControlSet -> Services -> Tcpip -> Parameters -> Interfaces
4. Under the "Interfaces" node you'll see several entries for each network your machine is configured for. You need to locate the one with the IP address of the network you want to update.
5. Once you've found the correct interface, right click on it and choose "New DWORD (32-bit) Value":

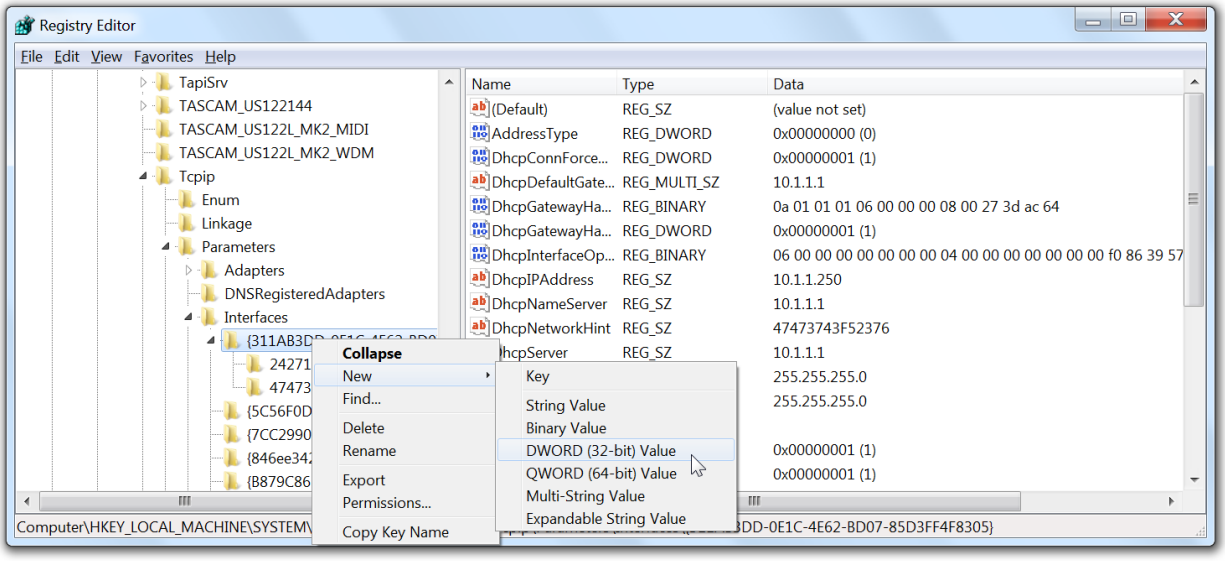


Figure 5.15: Creating keys to disable Nagles algorithm

1. Enter the name of the key as “TcpAckFrequency” (without the quotes)
2. Select the new value in the right hand pane and set its value to 1.
3. Repeat steps 5-7 to create a second DWORD value named “TcpNoDelay” and also set its value to 1.
4. Reboot for the changes to take effect.

When you’ve finished it should look similar to this:

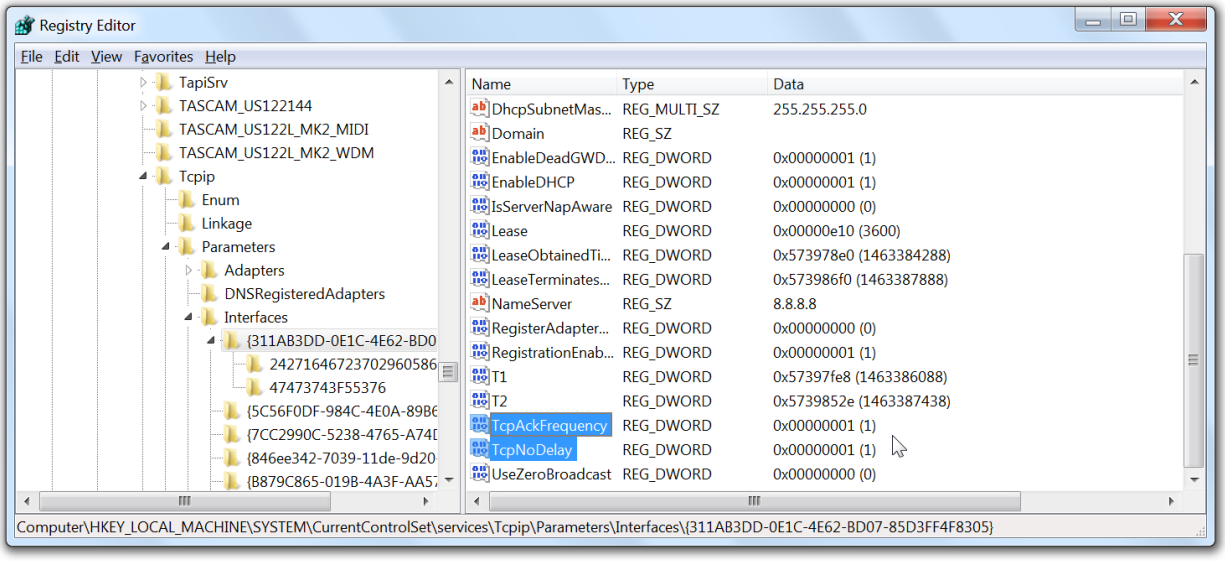


Figure 5.16: Both TcpAckFrequency and TcpNoDelay should be set to 1

Disabling Spread Spectrum

Spread spectrum is a technique motherboard developers use to reduce the amount of emitted electro-magnetic interference (EMI). The primary reason for this is to comply with FCC regulations.

To reduce the amount of EMI the frequency of various system clock generators is varied ever so slightly so that the interference is spread over a wider frequency spectrum.

Although uncommon, these slight variations in timing can cause timing issues and data transfer errors (which may require the operation to be retried).

Many motherboards have a BIOS setting to disable spread spectrum - refer to the documentation for your machine and/or motherboard for more information on this.

Wireless Networking and Bluetooth

Unless you absolutely need them I recommend disabling these devices while performing for the following reasons:

- If you're using your gear in an unfamiliar environment (say, at a gig) there's a good chance these devices will be performing at least a minimum level of background communications with surrounding equipment (wireless access points, mobile phones etc. . .)

- Wireless network drivers and Bluetooth drivers are both notorious for having poorly written system-level drivers and are often the cause of DPC latency issues.

Chapter 6

Setting Up Your Audio Software

In this chapter we'll be looking at various settings in your audio program that can affect performance. It's beyond the scope of this book to cover all software packages however most of these concepts will have related settings in most audio programs.

Typically you'll find these settings in the program's settings or options screen which can typically be accessed from a File|Options, Tools|Options or Edit|Preferences menu command.

Note that some of the suggestions I make in this section will be argued by some as less than optimal – particularly in regard to sample rate and sample size. This is a bit of an ongoing debate amongst audiophiles – some who swear they can hear differences. However, the point of this book is to optimize a machine for real-time audio performance where audiophile quality audio generally isn't necessary.

Also note that most audio programs have a separate set of settings for exporting rendered (bounced) compositions – for these use whatever settings you like – those settings have no impact on performance while you're using the software for auditioning/performance and typically just cause the export process to take longer. This is where it makes sense to use higher sample rates, wider sample formats etc...

Audio Drivers

Besides the buffer size and sample rate (which are discussed in the next section) the most important setting that will affect the performance of your software is correct audio driver selection. Many programs support different audio driver technologies such as ASIO, MME, WME, DirectX (or DirectSound), WASAPI, and others.

Where ever possible I recommend using ASIO drivers. Most ASIO drivers provide a very direct connection to the audio hardware, they're usually designed with real-time audio performance in mind, are widely supported by most software packages and practically all decent quality sound cards come with ASIO drivers.

When selecting your audio driver try to select an ASIO driver with the same name of the audio card you'll be using for audio I/O. If no such driver is listed check with the manufacturer of your sound card for a driver installation program and make sure it's installed correctly.

Another good option is to use the generic ASIO4ALL driver (available here http://www.asio4all.com). ASIO4ALL uses Windows kernel mode drivers to get low level access to the sound card and provides an ASIO interface to it.

I highly recommend ASIO4ALL if your sound card manufacturer doesn't provide an ASIO driver, or if you're having problems with it.

(Note: some cheap sound cards do include ASIO drivers, but are so poorly written they're not worth using and ASIO4ALL will often outperform them. Even if your standard sound card drivers are working fine, it's handy to have ASIO4ALL installed for testing and comparison.)

Note that ASIO4ALL may require you to configure which audio cards you want to use with it. Refer to the ASIO4ALL documentation for more details.

There are a couple of other generic ASIO drivers such as the "Generic Low Latency ASIO Driver" and "ASIO DirectX Full Duplex". In general I don't recommend these as their performance can vary greatly and often cause drop outs due to synchronisation issues and other instability problems. If you have no alternative you can try your luck, but don't expect too much.

Most ASIO drivers include a settings panel that can be typically accessed from within your audio software - typically via a button labelled "Control Panel" or "Audio Driver Settings" or similar. These settings panels are typically used to enable/disable channels, audio loop back (monitoring) modes, sample rates, buffer sizes etc... Again, refer to the sound card or driver's documentation for more information.

Some ASIO drivers don't always apply their settings correctly while the audio program is running. If this seems to be happening, try restarting your audio software after making changes.

Finally, some sound cards and ASIO drivers have a setting panel in the Windows Control Panel where additional or default settings can be tweaked.

Sample Rate and Buffer Size

In the *Introduction to Digital Audio* chapter at the start of this book I described how we can use the limits of human anatomy to determine a reasonable buffer size and sample rate for good quality audio.

We determined that anything humans can hear can be accurately represented with a sample rate of 44,100Hz and a bit depth of 16-bit. We also determined that humans can't discern latencies less than about 10ms and that a buffer size of 256 samples will satisfy that limit providing latency of about 6ms.

And that is where I recommend starting with your audio software configuration – a sample rate 44,100 KHz and a buffer size of 256 samples.

If you're not processing input audio (i.e. all your sounds are coming from within your audio software) set the buffer size of 256. If you are processing input audio the total latency will be affected by the length of the input buffer and the output buffer. Setting the buffer size to 128 samples gives a total end-to-end latency of 256 samples.

You certainly don't want to go below 44,100 KHz or audio quality will noticeable degrade and in general you don't need to go higher than this.

Remember the higher you go with the sample rate the more data needs to be processed and the less time it has to do it. Say for example you doubled the sample rate to 88,100 KHz – the number of samples per second has doubled and if you leave the buffer size at 256 samples the latency has dropped to about 3 milliseconds. If you do increase the sample rate, make sure you also increase the buffer size to compensate.

So when would you increase the sample rate? The main reason for doing this if you have a long chain of plugins or effects that result in the audio signal being processed many times. Increasing the sample rate can reduce the chances of audio artefacts creeping in.

The reason I recommend 256 samples is that it's well below the 10ms human hearing limit and provides enough head room for any additional latency that the sound card might introduce - there will be some. Also, it's a big enough buffer that it can be processed efficiently.

For nearly all applications these default settings will work just fine - and more importantly they'll work reliably. There are however exceptions:

- If you're running more audio effects and/or plugins than your computer can handle you may need to increase the buffer size to compensate.
- If you're routing audio between multiple programs or devices where each introduces its own latency you might need to optimize the latency down as far as possible so that the total latency of the entire chain is acceptable. In general keep the buffer as big as you can while maintaining acceptable latency.

See the section below on CPU Load vs Audio Load for determining how close to the limits of the buffer your software is running.

MIDI Drivers

MIDI is not something I've really covered in this book because in general it doesn't have a great impact on performance.

When it comes to configuring your audio software however you should definitely not enable any MIDI devices you're not using. Most audio software will still need to perform a minimal amount of processing for enabled MIDI ports even if they're not being used.

One especially notable case here is the Microsoft Wavetable GS Synth device. This device opens the default sound driver, almost certainly consumes CPU and is well known to conflict with the ASIO4ALL audio driver (and probably other audio drivers as well). Besides, it's a terrible sounding device - just don't enable it unless you really need it.

For software that supports acting as a master or slave for MIDI clock synchronisation the other thing to check is that you only enable MIDI clock on devices where you're actually going to use it. When MIDI clock is enabled quite a few MIDI events are transmitted just for the purposes of maintaining time. You don't want your software spending time processing the receipt of these events, nor the time required to generate them if they're not needed.

Multi-Core Processing

Nearly all modern PC's have multiple processors or multiple processor cores. In essence this means the computer is able to execute two pieces of code at the same time.

(In this book I use the term "processor", "processor core" and "core" interchangeably)

Say for example you have two virtual instruments – a piano and an electric piano in a keyboard split type setup.

With a single-core machine each instrument would be processed sequentially, i.e. one after the other.

On multi-core machine both instruments can be processed in parallel – one on each processor core effectively reducing the total time to process both of them.

However, there's also additional overhead to manage these multiple cores. The audio software needs to schedule each instrument to run separately, needs to ensure the other processor core is up and running and might need to wait for the other processor to finish.

The thing to remember here is that multi-core machines can usually do more work in less time however a single-core setup can almost always do it more efficiently.

Because of the obvious performance advantages that multi-core processing can provide most audio programs do include support for them. Similarly some instrument and effect plugins (particularly those that are CPU intensive) also leverage multi-core processors to improve performance.

This presents a complicated situation. Imagine for example you have two plugins that both support multi-core processing and your host program also supports multi-core processing and you've got a dual-core processor. You've now effectively created the situation where the computer is being tasked with completing four tasks at once on two processor cores. This is a not an ideal situation and can result in the plugins fighting each other for CPU resources and actually slowing things down.

Unfortunately there's no hard and fast rules for the best way to set this up and it can depend greatly on the plugins and effects you're using, how many of them, how they're chained

together and more.

In general I recommend this:

- If you're running many plugins enable the multi-core support in your host program and disable multi-core processing in any plugins that support it.
- If you're running just a few very CPU intensive plugins that support multi-core processing, disable the multi-core support in your host program and enable it the plugins.
- If you're running just a few lightweight plugins, disable the multi-core support in your host program – it's probably not necessary and might actually make things slower.

Having said all that, Windows is pretty good at scheduling CPU resources and in general most combinations of enabled/disabled support will work. By playing with the different combinations however you can hit on a more efficient setup and that can make for a more stable machine.

Hyper Threading

Some modern processors such as Intel's i7 processors support a technology called "hyper-threading". With hyper-threading the CPU has the ability to *sometimes* run multiple tasks at the same time on one CPU core. So a dual-core hyper-threaded processor can occasionally process 4 tasks at once.

In practice however for most audio applications this often doesn't work well. This is mainly because all the tasks that an audio program might try to run on separate cores are all trying to do the same thing (typically math operations) and those things can't be done simultaneously by one processor core.

Using hyper-threading for real-time audio processing can result in a significant *loss* of performance.

My advice here is if your audio software allows selecting the number of threads to be used for audio processing do not exceed the number physical processor cores.

64 vs 32-Bit Audio

(Note that this discussion refers to 32 vs 64-bit audio sample size and is not related to the question of 32 vs 64-bit operating system – commonly called x86/x64 – which is discussed in the next section. These are completely separate topics. x86 and x64 computers both support 32 and 64-bit audio).

In the Introduction to Digital Audio chapter I stated that CD quality audio uses a 16-bit integer sample. This works well for a final audio file but for audio processing a more precise floating point format is used.

Floating point numbers provide a better range of values and precision for audio processing than integer numbers. Also, the smallest floating point data type supported by Intel processors is 32-bit so nearly all audio software processes audio as a 32-bit floating point sample.

Many software packages however also provide the option to process audio as 64-bit audio samples, providing an even higher degree of accuracy.

The reason for using 64-bit audio over 32-bit audio is the same as for using a higher sample rate – to maintain accuracy and to prevent loss of quality when processing audio through a long chain of effects.

As far as the processor is concerned there's practically no difference in performance when comparing 32 and 64-bit audio processing.

However, if you're using VST plugins, not all plugins support 64-bit samples. If you've configured your audio program to use 64-bit audio and you use a plugin that only supports 32-bit audio the host program will need to down convert the samples to 32-bit before sending them to the plugin and then up convert the result back to 64-bit. There are two problems here – you've lost the audio quality you were trying to maintain by using 64-bit audio but more importantly you've introduced two extra steps in the audio pipeline (i.e: more work to do).

Normally the cost of these conversions are fairly transparent and you'd hardly notice any performance loss, but it's something to be aware of.

- Since all plugins support 32-bit samples, only enable 64-bit audio processing if you really need it.
- For live performance you almost certainly don't need 64-bit audio quality.
- If you do use 64-bit audio check that the majority of the plugins you'll be using support 64-bit audio.

64 vs 32-Bit Processor Architecture

Windows is available in two platform architectures - a 32-bit version (aka x86) and a 64-bit version (aka x64).

The bit depth of the operating system basically defines how much memory the CPU can address. An x86 computer can address a maximum of 4GB of memory (although Windows restricts this to just 3GB). An x64 computer can address a lot more than that – more than you'll ever need.

The 32-bit version of Windows can only run 32-bit software. The 64-bit version of Windows requires a 64-bit processor but can run 32 or 64-bit software. A 32-bit program running

under 64-bit Windows is still restricted to 3GB of RAM even though the machine may have a lot more.

Most modern Windows PCs are capable of running the 64-bit version of Windows and it's the typical version installed these days. You can check which version you're running by right clicking on "Computer" in Windows Explorer and choosing "Properties". Look for the entry titled "System type".

If you're running the 32-bit version of Windows this section is not applicable.

When you purchase or download or install your audio software you may be prompted to select the x86 or x64 version. Most plugins have similar options.

Whether you decide to use the 32 or 64-bit version of the software will mostly depend if the software itself and all the plugins you need to use are available as 64-bit versions. Often a 64-bit host program can't load a 32-bit plugin and vice versa.

Here's how to decide:

- If all the software you need to run is available as 64-bit versions then go with 64-bit.
- If many of your plugins (or the host program itself) are not available in 64-bit and you don't have the need for more than 3GB of RAM then choose the 32-bit versions.
- If you need the large memory space of 64-bit but you also need one or more plugins that are only available in 32-bit versions use the 64-bit version and look for a bridging option.

Bridging is a technology that allows a 64-bit program to load 32-bit plugin, or a 32-bit program to load a 64-bit plugin. It works by running the plugin as a separate program and using various communication methods to pass audio and MIDI data between the two programs.

The problem is that communications between separate programs is considerably slower than communication within the one program. Not only that, the way this communication works often requires the host program to wait in a way that opens it up to a greater possibility of being stalled by the operating system or some other task.

Bridging can be a good option when needed. Some audio programs include built-in support for bridging or the third party tool jBridge can be used for bridging VST plugins.

CPU Load vs Audio Load

Most audio programs have a load meter – an indicator that shows how much load the system is under with respect to audio processing. Windows Task Manager reports a similar load measurement as a percentage of CPU resource usage.

Although these two metrics are similar and seem related they are in-fact measuring quite different things and can't be compared.

The load reported by most audio programs is the amount of time taken to process an audio cycle as a ratio of the buffer duration. For example, if it takes 3ms to process one audio cycle and the audio buffer duration is 6ms, the load will be displayed as 50%.

In contrast, the CPU load reported by Windows is the percentage of available CPU cycles that are being used by a particular program. Note that it's possible for a program to be waiting for something (which takes time) without consuming CPU cycles (because Windows has suspended the program's execution and possibly using the CPU for something else while waiting).

In other words there's not necessarily a relationship between time taken and CPU resources used – which is why these two metrics rarely align.

The reason that audio programs display load as a measure of time taken is that time is the critical factor in providing reliable audio – remember that if the program fails to deliver the next audio buffer in time, a glitch will result.

You should use the audio load meter to determine how close you are to exceeding the capabilities of your computer and adjust the buffer duration accordingly.

The best way to test this is to load up the most demanding configuration you have and if you're using virtual instruments, play a MIDI file that exercises all the loaded instruments and keep an eye on the load meter.

A rough rule of thumb is for the load meter stay below about 50% - however this also depends somewhat on the sample rate and buffer size. You'll notice that shorter buffers are far less stable than large buffer sizes. This is because a delay in a smaller audio buffer leads to a bigger percentage change than the same delay with a longer buffer.

For example, a 1ms stall with a 6ms audio buffer equates to about 16% difference in the load meter. That same 1ms stall on a 12ms audio buffer adds just 8%.

In other words the smaller the audio buffer – the higher the risk of a dropout and therefore the lower overall load you want to start with.

Virtual Audio and MIDI Cables

Virtual audio and MIDI cables let you route audio and MIDI signals between different programs. This can be handy when different programs provide different sets of features that you need use at the same time.

In general these virtual cables work fine however whenever possible you should favour a virtual MIDI cable over a virtual audio cable.

The reason for this is that MIDI events don't need to be processed as part of the audio processing pipeline of the audio program. A MIDI event delayed by a millisecond or two will rarely, if ever, be noticed by anyone. That same 2 millisecond delay in audio processing however could be enough of a delay to cause a glitch.

The reason for avoiding virtual audio cables are essentially the same as for avoiding plugin bridging tools. Basically if you need them, test them carefully, be aware of their pitfalls and try to include a little extra head-room in your buffer size selection.

Chapter 7

ISR, DPC and Page Faults

So far in this book we've covered how to setup your system and your audio software to give the best chance of reliable performance. In this chapter we'll be looking at how to diagnose and fix the most common problems – ISR and DPC Latency issues and Memory Page Faults.

LatencyMon

Windows doesn't include any built-in tools to check for ISR and DPC latency issues however there's a great free tool from Resplendence Software call LatencyMon.

LatencyMon is available here: http://www.resplendence.com/latencymon

Download and install it as you would any other software. Once installed:

1. If you're running on a battery powered device, make sure you have mains power attached
2. Make sure your power settings are configured as they would be during performance
3. Close all other running software
4. Start LatencyMon
5. Press the green "play" button to start the test
6. Let it run for a few minutes
7. Press the red "stop" button to stop the test

LatencyMon will show a set of graphs and latency figures that reflect the ISR and DPC performance of your machine.

Ideally, you want the longest of these times to be less that the headroom you have in your audio buffer size. For example suppose your running with a buffer size of 6ms and the peak load as reported by your audio software is 33% (2ms) then you have about 4ms of headroom.

If all the times reported by LatencyMon are less than the amount of headroom then DPCs and ISRs probably aren't going to be causing you issues.

If on the other hand the reported times are longer than the available headroom (or if they're more than about 500µs (0.5ms), then you should probably take closer look. LatencyMon will show which drivers had the slowest ISR and DPC times.

1. Check for an updated driver. If one is available update it and re-run the test.
2. If you have the latest driver and it's for a device you know you don't need during live performance you might be able to simply disable it in Windows Device Manager and re-run the test. (Never uninstall a device and be careful not to disable devices that the system needs for correct operation – see below)
3. Try searching for the name of the driver and the terms "DPC" or "ISR". You'll often find forum discussions on particular drivers that are problematic and sometimes a specific version (perhaps even an older version) are known to perform more reliably.
4. Depending on the problematic device you may need to replace the device with a similar device from a different manufacturer

If you still can't find a solution for an essential driver all may not be lost.

- If your machine has multiple processors and your audio software provides a setting to control the number of audio threads you could try reducing this by one.
- If your audio software doesn't have an option to control the number of audio threads, look for a setting to disable multi-core processing.

Once you've reduced or disabled multi-core processing test your system by running your audio program over a period of time - you might find that even with the slow ISR or DPC issue the free processor cores are enough to keep things running smoothly.

Another handy tool is DPC Latency Checker by TheSysCon. This provides similar information to LatencyMon, but doesn't include details on which driver is causing the stalls.

Disabling Devices

Once you've determined that a system driver is causing ISR or DPC issues you might consider disabling the driver.

You should be careful not to disable a driver that's required by the system as this could adversely affect the system as a whole – perhaps even rendering it unable to boot.

In general you should never disable devices in the following groups:

- Computer
- IDE ATA/ATAPI controllers

- Processors
- System Devices
- Universal Serial Bus controllers

There are occasional exceptions for the above, but unless it's for a very specific device that you know is not necessary try to avoid these groups.

Devices that are generally safe to disable include:

- Batteries
- Bluetooth Devices
- Imaging Devices
- Network Adapters
- Sensors
- Sound, video and game controllers

You should never uninstall drivers unless the associated hardware has been physically removed from the machine.

To disable a device:

1. Click the Window start button, type "Device Manager" and press Enter launch Window's Device Manager Utility
2. Locate the device in the list
3. Right click on the device and choose "Disable"

After disabling a device you should re-run the ISR and DPC latency check tools in-case there is another device causing latency issues.

Notorious and Unnecessary Devices

Some kinds of devices are notorious for causing problems:

- Wireless network drivers – perhaps due to the nature of the associated hardware, these drivers often cause DPC latency problems. Options include going without network access or switching to a wired network.
- Bluetooth drivers – fortunately for live performance Bluetooth is often not required and can be simply turned off. Switch to wired or built-in devices in preference to Bluetooth keyboards and mice.

- Sound drivers – ironically sound drivers often cause DPC issues for audio performance. This particularly applies for built-in sound devices not designed for real-time performance. Use a sound card or device from a reputable manufacturer and disable any unnecessary built-in devices.
- ACPI.sys will often show slow DPC latency times on battery powered machines. *Do not disable the ACPI driver nor the Microsoft ACPI-Compliant System device.* Rather, disable the "ACPI compliant battery" or "Microsoft ACPI-Compliant Control Method Battery" device. Unfortunately you'll lose you battery icon and some power management settings.

Other devices you might consider disabling because they're often not needed during performance include:

- DVD and CD Drives
- External storage devices (flash drives, card readers etc. . .)
- Cameras and scanners

Diagnosing Page Fault Issues

Diagnosing whether page faults are the cause of audio issues is particularly difficult:

- Page faults are a common and a normal part of the operating system's functionality.
- Page faults on threads other than the audio thread generally don't cause audio issues in properly designed software but still show up in page fault metrics.
- Generally only hard page faults cause issues however most diagnostics tools combine hard and soft page fault statistics making it difficult to tell what exactly is happening.

One of the best ways to monitor for hard page faults is with LatencyMon. The trick is to try to correlate hard page faults with audio drop outs. One technique to try is this:

1. Make sure any screen savers and power save options are turned off.
2. Start your audio software and load up the session you want to test against. Make sure any sample libraries are fully loaded.
3. Make sure your external MIDI keyboard is connected and routed to whatever plugins you're testing against.
4. Play a glissando across the full keyboard to help pre-load pages.
5. Start LatencyMon and press the green "Play" button.
6. Switch to the Processes tab, sort by name and locate the name of your audio program and make sure it's visible.

7. Step away from the computer – don't touch the mouse or keyboard and just let it sit there for a few minutes.
8. After a few minutes, play a chord of say 8 notes and watch the hard page fault count in LatencyMon – if it jumps rapidly then you may have an issue with page faults.

Unfortunately, while the above may give a clue to page fault issues it's by no means reliable since Windows memory management is a very dynamic system.

In short, you can't really test for page fault issues – the best you can do is try to correlate audio glitches to high page fault counts associated with your audio software.

Some audio software includes a page fault counter. If so, I recommend enabling it and monitoring it while playing – you should be looking for high page fault counts while doing nothing other than playing. Doing just about anything else in the program will often cause a normal page fault spike and typically these page faults won't be on the audio thread.

Fixing Page Fault Issues

If you've determined (or suspect) that the cause of audio glitches is related to page faults there are a number of steps you can take to minimize the issue:

1. Reduce total memory usage by stopping unnecessary programs and services. Use Windows task manager to find programs using excessive memory and stop them. Modern web browsers and Windows search indexing services are notorious memory hogs.
2. Reduce memory usage by using smaller sample libraries. Most audio samplers include options to use smaller sample sets which are often more than adequate for live performance.
3. Reduce memory usage by using smaller sample buffers. Similarly to the above most samplers have buffer settings for how much of each sample to pre-load. If you're definitely getting page fault issues but your hard disk is fast enough to not require the additional buffers, reducing the buffer size might help.
4. Add more memory to your system. Memory is reasonably cheap these days and can dramatically improve the overall performance of your machine and reduce the number and likelihood of a hard page fault.
5. Cantabile has an option to try and keep memory paged in. This works by running a low priority background thread that slowly goes through touching every page of memory so that Windows thinks it's in use and doesn't page it out so readily. See Options -> Audio Engine -> Prevent Memory Paging and set it to either Normal or Aggressive. (But don't use this option on Windows 10 – this option is incompatible with its new memory manager)

Although I generally don't recommend this the other thing you can try is disabling the page file altogether. Before you consider this however:

- Make sure you have more than enough physical memory to load everything your performance session requires.
- Note that you're much more likely to run out of memory and some programs will simply crash. Windows memory management with page file is so good that many programs are just not well tested for out of memory conditions.
- Windows can't generate crash reports when the page file is disabled.

To disable the page file:

1. Click the Windows start button and type "advanced system settings"
2. In the Performance group, click the "Settings" button
3. Switch to the "Advanced" tab
4. In the Virtual Memory group, click the "Change" button
5. Turn off "Automatically manage paging file size for all drives"
6. Click on each drive in turn and the select "No paging file" then press the "Set" button
7. Press OK

The other reason you might want to disable the page file is simply as a test. If you disable the page file and you're still getting audio glitches you can be pretty sure it's unrelated to page fault issues.

Chapter 8

Miscellaneous

Hard Drive Performance

If you're using large sampler plugins, poor hard drive performance can cause issues with real-time audio rendering.

Often these sample libraries are far larger than the amount of system RAM and need to be streamed from the hard drive as the sound plays.

There are three main classes of hard drive performance:

- Slower mechanical drives – typically 5400rpm
- Faster mechanical drives – typically 7200rpm
- Very fast solid state drives – often referred to as SSD drives

If you're using sampled plugins I generally recommend using a 7200rpm drive – they're cheap, large and generally fast enough. Depending on the plugins in use, 5400rpm drives often only provide borderline performance.

On the other hand, if you can afford the extra cost and can live with the typically smaller size, SSD drives provide a great performance boost. Not only are they more reliable during performance, loading anything from that drive will also be faster.

Drive manufacturers will often promote better performance of their drives based on large cache buffers, or on-drive flash memory. For general computer usage this often works really well. You should note however that these technologies provide little benefit when streaming large sample sets– for this you need a drive with fast underlying performance.

Multiple Drives

If you have room in your machine to install multiple drives, you might like to consider installing the sample libraries for your large plugins on a separate drive from your system drive.

By placing your sample libraries on a separate drive you can ensure the drive will only be accessed from the associated plugins. This can help to avoid contention for the drive from other running software and access to the system swap file.

Note that using separate hard drive partitions isn't enough to help with performance - they need to be separate physical drives.

Midi-Clock Sync Jitter

Many plugins offer the ability to synchronize their sounds to the musical timing of the host program.

Normally the host program will provide very stable timing information and the plugin behaves as expected.

If you have your host program synchronized to an external MIDI clock source, the software typically needs to calculate the current tempo by timing the incoming events. Often this will result in "jitter" – where the calculated tempo fluctuates slightly over time.

Some plugins don't cope well with this jitter resulting in audio glitches.

The easiest way to test for this is to simply switch from the MIDI clock source to the hosts built in metronome and see if the problem goes away.

If you do encounter this problem, some host programs provide a setting to control the responsiveness of the MIDI clock synchronization. Less responsive settings are typically more stable and can often eliminate this problem.

Non-Shared Audio Clocks

Some audio drivers (eg: ASIO4ALL) have the ability to combine multiple audio hardware devices and make them appear as a single device to the host program.

You should take care when using multiple underlying devices that don't share an underlying hardware clock. These devices will drift over time – i.e. one will always run ever so slightly faster than the other.

To compensate for this drift, the audio driver might take one of two approaches:

1. Dropping or inserting extra samples to re-synchronize the two devices. This approach doesn't affect performance too much, but can result in a glitch when the resync occurs.
2. Gradual resampling over time to keep the devices in sync. In this case the resampling will affect performance if it's happening often but in general shouldn't cause glitches.

Neither of these approaches are ideal and if possible you should avoid using multiple devices that don't share a common clock signal.

The only real workaround for this situation is to switch to a different audio interface that has enough inputs/outputs so that you don't need to use multiple audio drivers.

Avoiding Resampling and Time/Pitch Shifting

Audio resampling can be a very computationally intensive operation – particularly if done using a high-quality setting.

If you're performing live, you can help avoid this performance hit by ensuring any audio files you're playing during the performance have the same sample rate that the audio driver is running at.

Most audio editing software can resample audio files.

The same argument applies to audio time and pitch shifting. If you can, pre-process any such operations and use the processed files to avoid these expensive operations during performance.

BIOS and Chipset Updates

Although rare, it's possible that performance issues can stem from poor firmware and chipset drivers.

In general, if your machine is running without issues you shouldn't need to update these drivers. If on the other hand you're encountering unexplained audio problems it might pay to check for updates.

In both cases, refer to the manufacturer of your hardware to see if updated software is available.

Graphics Card Drivers

Many modern graphics cards include power saving modes. Eg: NVidia Powermizer and ATI Power Play.

Often these power modes will throttle down the graphics card in order to save power and this can result in long DPC latency stalls.

Most graphics cards include a utility to control the power settings of the card – check the settings and ensure that power saving modes are disabled.

If you don't have such a utility check the specifications of your card and/or the manufacturer for information on whether the card supports power saving modes and how to disable them.

BIOS Settings

Many system BIOSes have settings to disable features like multi-core support, hyper-threading, clock throttling technologies and other power saving technologies like "SpeedStep".

In general I recommend leaving all these options turned on ans use the operating system supplied settings for these. (This let you more readily switch between high-performance and power saving modes).

Chapter 9

Wrap Up

My goal with this book was to provide more than just a long check list of settings to adjust. I really hope that in reading this book you now have a deeper understanding of what's happening under the covers, why audio glitches occur and how to best setup your machine for your particular requirements.

This book is full of personal opinion – not everyone will agree with everything mentioned, but the recommendations I've given are based on the things I've found matter most and work best. Feel free however to try other approaches and if you find something that works please let me know – you can reach me at:

- brad@cantabilesoftware.com

I can also be found on Twitter and Facebook as @CantabileApp. If you enjoyed this book let me know or you can help spread the word with a like, retweet or follow:

- http://twitter.com/CantabileApp
- http://facebook.com/CantabileApp

If you'd like to discuss anything mentioned in this book, the Cantabile discussion forum is a great place for that.

- http://community.cantabilesoftware.com

This book will continue to be updated from time to time as new versions of Windows are released and other approaches to improving its real-time audio performance are found.

In the meantime I hope you've found this guide useful and all the best in your music making endeavours.

Chapter 10

About Cantabile

Cantabile is a music workstation for live performing musicians. It's been highly optimized for stable real-time audio performance and it's the best way we know to play virtual synths and effects live.

- Play virtual instruments and effects live
- Create re-usable instrument and effect racks
- Instantly switch between songs
- Per-song keyboard splits and transpose
- Trigger audio and MIDI clips on the fly
- Control everything via MIDI
- Remember where you're up to with show notes
- Automatically record your performances

You can get a free copy of Cantabile Lite or a free 30-day trial of Cantabile Solo and Performer from the website: www.cantabilesoftware.com

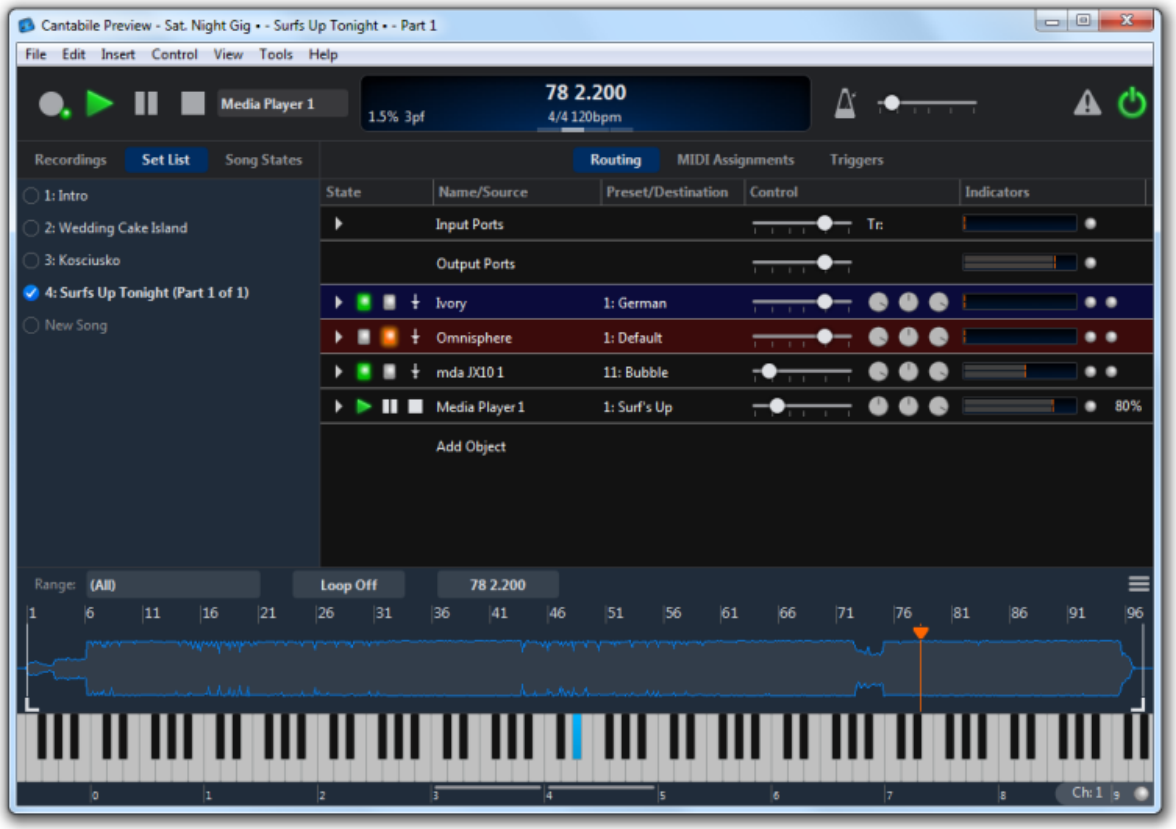


Figure 10.1: Cantabile 3 - Live Performance Workstation