# Creating MIDI for the Web

by Jeff Essex

Sponsored by Apple Computer, Inc.
Worldwide Developer Relations Group

# Creating MIDI for the Web

by Jeff Essex

## Introduction

Over the past few years, the Web has become a sophisticated system for delivering media, starting mostly with text, then adding graphics, animation, video and audio. But bandwidth to most web clients (28.8 kbps modems) hasn't increased at the same rate, and isn't nearly fast enough to create a seamless multimedia experience. As a result, web developers must make difficult choices to reduce file sizes and download times.

In many cases, audio is one of the lowest priorities because audio files are cumbersome. Compression schemes and streaming technologies have made it more feasible to add audio to a web page, but compressed audio files are still big compared to text or graphics. Streaming audio reduces the download time, but it uses up a lot of valuable bandwidth. Depending on your needs, it may be that the optimal solution is one that predates the Web by about 10 years: MIDI, the Musical Instrument Digital Interface.

This article discusses strategies and tools for delivering MIDI via the Web. Topics include:


- What's MIDI?
- General MIDI - The Pleasure and the Pain
- How MIDI Plays on Computers
- Creating Consistent Web MIDI
- DLS and the Rebirth of MIDI
- Major MIDI Players on the Web
- MIDI and Real-Time Interactive Control
- Looking Ahead


At the end of this article is a resource guide that will help you find more information.

## What's MIDI?

First, this article assumes that most readers have some understanding of MIDI technology. Many, many books have been written about MIDI and it would be impossible to include enough material here. If you're just getting started with MIDI and are looking for detailed information about setting up your own MIDI studio, you should check the Resources section at the end of this article for more information.

But enough with the disclaimers. Let's review some general basics. MIDI (the Musical Instrument Digital Interface) is a serial protocol that provides realtime control and communication between musical instruments, computers, and audio peripherals. The best analogy for MIDI is that of the player piano. Like a player piano, a MIDI sequence contains all the control information a synthesizer needs to duplicate the playback of a musical piece. This is one of MIDI's main advantages over digitized audio for playing music over the Web. While CD quality audio requires over 10 megabytes of storage per minute, a 2 minute MIDI files is usually less than 30k. With MIDI, unlike digital audio, you can't really equate the length of the music with the size of the file. MIDI file size is determined more by the complexity and density of the piece (number of notes, instruments, and amount of controller data) than by time.

A basic MIDI setup includes:

- a device for generating MIDI data (often called a *controller)* like a synthesizer keyboard with a MIDI output. There are also MIDI-enabled guitars, drum pads, wind instruments and other alternative controllers for those who prefer a different method for creating MIDI performances.
- an interface between the computer and the MIDI device which translates the serial MIDI data into a format that can be read by the Mac serial (printer or modem) ports. Some of these interfaces are quite sophisticated, and can route MIDI data to multiple instruments or provide synchronization to video or audio tape decks. Some MIDI tone modules (a synthesizer in a box with no keyboard) have serial interfaces so that they can be connected directly to the serial ports of a Mac or PC.
- MIDI files are created using a *sequencer*, a software application that can record, edit and play back sequences of MIDI data. The Mac has long been the leading platform for creating MIDI files, and there are a number of excellent sequencers on the market (check the Resources section at the end of this article for more information).

MIDI files on the Web are usually played back on whatever General MIDI (GM) synthesizer happens to be available in the viewer's computer. The GM standard was created to enable reasonably consistent MIDI playback on different synthesizers. GM specifies a bank of 127 instrument types in which the Piano is always Program #1, Acoustic Bass is #33, and so on. This assures that a Standard MIDI File played on any GM device will give a relatively decent rendition of a piece of music: what was originally intended to sound like a tuba is mapped to the synthesizer's tuba sound instead of flute or violin.

## General MIDI - The Pleasure and the Pain

Just because a synthesizer supports General MIDI doesn't mean that the music will always sound the same. There are radical differences between synthesizers, and sound quality depends on what type of synthesis is used. Most PCs play General MIDI from a sound card with either FM or wavetable synthesis. Most Macs play MIDI through a synthesizer in software: the QuickTime Music Architecture (QTMA). Other software synthesizers can play on both platforms. There's a broad range of tonality in both hardware and software MIDI players. Getting the same piece of music to sound pleasing on many different playback devices is quite a balancing act filled with compromises.

Luckily, the situation is improving thanks to the development of higher quality synthesis in software. But this trend probably won't come to full fruition until 1999. Right now this technology is still a bit on the

leading edge. By next year it should be commonplace, and developers will be more comfortable integrating it into their sites. Until then, you should be prepared to deal with the installed base of MIDI playback devices. Here comes a quick overview!

## How MIDI Plays on Computers

There are four main playback devices one can expect to encounter when composing GM music for the Web: Wavetable, FM, QuickTime and software-based. Since you can't predict which one will be present on viewer's computer, you need to be familiar with each type.

*Wavetable* synths are so named because they create sounds by playing back actual recorded waveforms of instruments (called samples) stored in Read-Only Memory (ROM) on the sound card. As a general rule, wavetable delivers the best sound quality because the sample contains much of the harmonic complexity of the sound produced by an actual instrument. But this depends on how much memory is used to store samples on the card (more memory = better samples), and how well the instrument samples are used. The most popular models are from Creative Labs (www.creaf.com). Unfortunately, some of the sounds on this card differ quite a bit from the sounds on other cards or standard MIDI modules like the Roland Sound Canvas.

*FM synthesis* is the least-natural sounding type of synthesis. Sounds are created using Frequency Modulation (hence the name FM). One waveform is used to modify the frequency characteristics of another waveform, resulting in a variety of tones. The main benefit of FM is that it's very cheap to manufacture. The sound card stores tables of numbers (algorithms) used to create the waveforms, rather than actually using memory chips to store sample data. FM synths are good at creating sounds that have very regular waveforms (organ, clarinet, bassoon), percussive sounds (xylophone) and plucked strings (guitar, harp). But since they don't use sampled sounds, the tones sound very much like a cheap synthesizer (surprise!). On the whole, FM sound cards reproduce music that sounds like a cheap arcade game or one of the early Nintendo systems. As one composer noted, "It's great if all you're doing is circus music!"

*QuickTime synthesis* is is the main playback device on most Mac systems. Starting with QuickTime 2.0, Apple created the QuickTime Music Architecture (QTMA). With QTMA, every QuickTime-equipped Mac (which at this point means practically every Mac) can play MIDI files. A set of GM instruments (the QuickTime Musical Instruments extension) is loaded at startup. When a web-surfer hits a page that contains MIDI, the MIDI data is sent to QuickTime, and played by the QuickTime Musical Instruments. This is why the viewer usually see a progress bar with the words "Importing Movie" after hitting a page with a MIDI file.

With the advent of QuickTime 3.0 (www.apple.com/quicktime), there are major improvements in QuickTime's implementation of MIDI. In fact, it fixes just about everything I had hoped for. For a fuller discussion of MIDI in QuickTime, you may want to refer to my earlier article, "New Audio Features in QuickTime 3.0".

*Software-based synthesis* has been available for several years, but it's just beginning to become more feasible, thanks to the CPU horsepower available in Pentium and PowerPC processors (technically, the QuickTime synth falls in this category, too, but I felt it warranted its own discussion). Software synths, as their name implies, handle all the chores of synthesis in software, mix the result to the equivalent of a WAV or AIFF file, and send the result to the sound output device. Because all the work is handled in software, these synths are extremely flexible and can produce very good sound. On the flip side, since a lot of CPU power is required, the composer has to create a good musical score without bringing the computer to its knees. The most interesting potential for software synthesis is in conjunction with a new specification for downloadable sounds (DLS). One example of DLS is Beatnik, created by Headspace (www.headspace.com, or refer to the Beatnik article listed in the Resource section at the end of this article). In the DLS approach, synthesis data is downloaded along with the MIDI file, so that the user downloads not just the song, but all the sounds needed to play the song. This has tremendous advantages to the composer: we will finally be able to assure that users will hear our work exactly as we hear it. The main limiting factors in DLS become download time and CPU power. It will take a bit longer to send a DLS file than a MIDI file, because the instrument data is combined into the file. It also uses more of the CPU: 3% of the CPU per voice is not uncommon, and one 5-note chord might tie up 15% of the CPU resources. As CPUs get faster, this becomes less of a problem. DLS is an important new technology for composers and web designers, and we'll discuss it in detail in a few moments. First, let's look at some issues to consider when creating MIDI files for the installed base of synthesizers currently on the market.

## Creating Consistent Web MIDI

Now that you know more about where MIDI has been, you may have a better understanding of the potential pitfalls of MIDI playback. It's unfortunate that the prevalence of bad sounding FM synthesizer chips has led to a perception that "MIDI sounds bad." But it's important to keep in mind that MIDI in and of itself doesn't sound bad; it's just that there are lots of bad sounding synthesizers. To use an analogy from the world of graphics, this is akin to thinking that gradients in a vector-based graphics program "look bad" because they're displayed in 4-bit color rather than 24-bit. The graphics program has no control over the display device, in the same way that a MIDI file can't make a cheap synthesizer sound fabulous.

There are quite a few hoops one must jump through to get a decent MIDI file to play back from a Web page. What follows is a discussion of the issues and the solutions I've devised to meet these challenges.

1) *Always use Standard MIDI Files*
This is an obvious point for experienced composers and developers, but folks who are just starting out may need the reminder. You can't just save a MIDI file from your favorite MIDI sequencing application and expect it to play on the web. Instead, be sure to export the data as a Standard MIDI File (SMF). SMFs are a standard format that allows the exchange of MIDI file data between different sequencers. There are two types of SMF. Type 0 files combine all the data (including MIDI data on different channels) into one track. Type 1 files are multitrack, and export each track from the sequencer as an independent track. This makes it easier to import the data into another sequencing program. For web delivery, use Type 0 files.

2) *Every file must be initialized with the proper settings.*
A MIDI file can play multiple instruments at the same time. Each is on a discrete channel, somewhat like a multitrack tape recorder. Each of these channels must be initialized with the proper settings for instrument assignment (program changes), volume and placement in the stereo field (pan). These instructions must be placed before any music data, so that setup is complete before the music starts. At this point, the line between composing and programming begins to blur.

3) *The file must loop smoothly.*
One never knows how long the viewer will remain on a particular page, so the music should loop seamlessly. But achieving a seamless loop while also initializing the data (as in point #1 above) is quite a trick. If I tell the synth to play the first note while also telling it "change from a piano sound to a guitar sound", there will be a glitch in the rhythm of the piece while the command is executed. I've hit on three solutions to this problem:

a) start with just drums or piano.
The drum sound is always the same, and piano is the default sound that's played if no program change is sent. I sneak the initialization info for the other instruments in while the piano or drums are playing.

b) start with any one instrument by itself.
For example, one piece I composed starts with harp. The program data is placed just after the first note, so the first time a user hears it, they'll hear one note of piano before the rest of the harp notes. But on successive passes through the loop, the harp will play the first note, since its instrument assignment was set on the first run through the loop.

c) start with one tiny section of silence.
This is a bit more complex, and requires taking advantage of some musical math. Say a piece is in 4/4 time. It's often possible to go to the last measure of the piece and see if there's a tiny bit of silence at the end (for example, a 16th note rest). We can define the last measure as 16/16 instead of 4/4, then take the silence during the last 16th note rest and stick it at the beginning of the piece. We end up with a piece that starts with one measure in 1/16, then lots of 4/4, and a last measure of 15/16. Now there's a 1/16 measure of silence where we can initialize the file.

4) *The file has to sound good on whatever synthesizer is present in the viewer's machine.*

As mentioned earlier, the first step is to know the enemy. After a piece is finished, I play it on a PC through a Creative Labs SoundBlaster AWE32, a basic SoundBlaster 16 with FM synthesis, and on a Mac through the QuickTime Musical Instruments. Most of my pieces go through several revisions to balance the instrument levels between the different players.

5) *The file will never sound as good as you would like.*

This is perhaps the toughest trick to master: knowing when to stop tweaking. It's daunting enough that our fine creations may be reduced to a collection of chirps and bleeps. We can only continue to hang on until improved technology rides to the rescue. Help is on the way thanks to DLS.

## DLS and the Rebirth of MIDI

DLS (**D**own**L**oadable **S**ounds) represents the future of MIDI playback in Web, CD-ROM and game applications. The combination of software synthesis and DLS will give composers the power to create richer musical experiences. As mentioned above, a DLS file combines MIDI data along with the instrument samples required to play the MIDI sequence. The samples are loaded into a software synthesizer for playback at runtime. The major advantage of DLS is that the performance heard by the listener is exactly what the composer heard when creating the piece. This is a tremendous improvement over the current hodgepodge of synthesizers currently installed in PCs. Luckily, Macintosh users have largely been spared this experience, and the improvements in QuickTime 3.0 have further reduced the difficulty.

The DLS standard was developed by the Interactive Audio Special Interest Group (IA-SIG) of the MIDI Manufacturer's Association (www.midi.org). The IA-SIG membership includes composers, sound designers, software and hardware developers, and content publishers. The MIDI Manufacturer's Association is the group responsible for creating, maintaining and modifying the MIDI specification. Together, these groups developed the DLS Level 1 specification which defines the requirements and capabilities of DLS-compatible software synthesizers, and the DLS file format.

At this point, there are two main DLS-compatible synthesizers available for Web playback: QuickTime 3.0 and Beatnik from Headspace. There are also several MIDI synthesizers and players that have gained a substantial installed base for Web delivery, including LiveUpdate's Crescendo and Yamaha's MIDPlug. Let's take a look at these tools and their capabilities.

## Major MIDI Players on the Web

*Beatnik from Headspace*

Beatnik (www.headspace.com) is probably the most powerful MIDI playback device currently available for the web. It combines a General MIDI-compatible software synthesizer, the Beatnik editor for adding custom samples, and extensive hooks into JavaScript for realtime control during playback. The Beatnik plug-in includes a full complement of General MIDI instruments. As a bonus, a second custom bank is available, drawn from the same samples as the standard bank, but with different synthesis parameters applied. Right off the bat, you have access to 256 different instrument sounds on the client's machine.

To create a file with Beatnik, you can either import a Standard MIDI File, or access Beatnik's instruments directly from Opcode's Vision, one of the most popular MIDI sequencing applications. It's difficult to play the Beatnik synthesizer directly from within Vision, however, because there's quite a bit of latency between the time when a note is struck on the MIDI controller's keyboard, and the actual sounding of the note. When composing, it's best to create with a General MIDI module like the Roland Sound Canvas, then switch the playback over to Beatnik to audition the performance and tweak it to

your taste. You can also import AIFF files into the Beatnik editor to create your own custom samples, or work from the existing samples included with the plug-in to create your own new sounds.

Beatnik's proprietary file format, RMF, combines the MIDI and audio file data. When you save the RMF file, Beatnik applies data compression to both the MIDI and audio data to achieve the smallest possible file size for Web delivery.

Beatnik's greatest strength may be its real-time control. Using JavaScript, you can send MIDI commands to the Beatnik plug-in, triggering note events, adjusting volume, swapping out instrument samples and even muting or unmuting tracks. This makes Beatnik great for button rollover sounds as well as background music. And while Type 0 Standard MIDI files are the general norm for web MIDI, Beatnik can access up to 64 tracks in a Type 1 Standard MIDI File. Armed with this ability, a composer could create a short piece with multiple layers of instruments, then control the muting and unmuting of the tracks so that a 15-second music loop could become a five minute piece!

*QuickTime 3.0*

As I mentioned earlier, the MIDI playback in QuickTime 3.0 is a tremendous improvement over earlier versions. QuickTime 3.0's new instrument set was licensed from Roland Corporation, the makers of the Sound Canvas MIDI sound module (i.e. a traditional hardware synthesizer). Over the years, the Sound Canvas has emerged as a standard for composers, since it was one of the first General MIDI devices on the market. I think of it as the Rosetta Stone of General MIDI playback devices; if something sounds good on the Sound Canvas, it's likely to sound okay on most other General MIDI players. Happily, QuickTime 3.0's implementation of the Sound Canvas instruments is remarkably close to the hardware synthesizer version (the only major difference is that QuickTime's synthesizer doesn't provide effects like reverb or chorus).

QuickTime 3.0 is also DLS-compliant, because instrument samples can be built right into QuickTime music tracks. There are two ways to do this:

- using Drag and Drop to place .snd resources into the Instruments window of MoviePlayer 3.0 (only supported in QuickTime Pro)
- using Drag and Drop from the Atomic Editor utility for creating custom instrument extensions

Both of these techniques are described in my article "New Audio Features in QuickTime 3.0"; I strongly encourage you to review that information if you're interested in putting your own samples into a QuickTime movie.

Best of all, QuickTime 3.0 is fully supported on both Mac and PC platforms, so anyone with the QuickTime extension and QuickTime plug-in will hear the music performed exactly as it was created, even if they only have an FM synthesis chip on their PC sound card.

*Yamaha's MIDPlug*

MIDPlug from Yamaha is a MIDI file player that features Yamaha's SoftSynth, a General MIDI-compatible software synthesizer with (www.yamaha-xg.com/english/xg/midplug/mplug.html). MIDplug goes a step beyond the regular General MIDI instrument set, and incorporates Yamaha's XG instruments. Briefly, XG is an extended set of General MIDI instruments which gives the user more instrument choices (multiple pianos, saxophones, trumpets, etc.) so that composers can create with a broader palette of sounds. Unlike Beatnik or QuickTime, however, there are no provisions for adding custom instrument samples.

*Crescendo*

Crescendo from Live Update (www.liveupdate.com) is a plug-in for controlling the playback of MIDI files, and only plays MIDI data through whatever synthesizer is available on the client machine. In essence, Crescendo provides a CD player-style interface for playing MIDI files, giving the user more control than they might otherwise have through JavaScript or HTML tags. It also streams MIDI data so that a file can start playing before it has finished downloading. Granted, a 30k file should only take 12-15 seconds to download on a 28.8 modem, but Crescendo's streaming reduces the wait time to about five seconds. Of all the plug-ins covered here, Crescendo is the most basic in functionality, but still provides helpful features for MIDI playback.

## MIDI and Real-Time Interactive Control

By now you can probably appreciate that MIDI is a superior format for delivering music over the Web. But it has other advantages than just small file sizes. MIDI is also ideal for interactive applications. Remember that MIDI is a stream of control data rather than just a representation of an audio waveform, so events can be added or modified on the fly during playback. Digitized audio (whether it's streaming from the web, or an AIFF file playing from CD-ROM) is much more limited in terms of realtime control. Beatnik, with its extensive JavaScript support, is an excellent example of the possibilities here. Another interesting new application is ResRocket DRGN (Distributed Real-time Groove Network) (www.resrocket.com), a product for jam sessions on the net. Musicians around the world can collaborate on music in real time (puts a whole new spin on "world music"!).

## Looking Ahead

Now that the DLS Level 1 specification has been established, it's likely that MIDI will become an increasingly important tool for delivering music content over the web. Thanks to continuing improvements in CPU performance, synthesis in software will become the most desireable method for MIDI playback. Also, work on the DLS Level 2 specification has recently been completed by the IASIG, and is likely to be adopted as part of the MPEG 4 spec. As industry-wide support of DLS becomes the norm, sound designers, composers and web developers will have more control over music and audio production, and web surfers will have a much better listening experience.

## Resources

*More About MIDI*

| | |
|---|---|
| **MIDI Manufacturer's Association** | **http://www.midi.org** |
| Keyboard Magazine | http://www.keyboardmag.com |
| Electronic Musician Magazine | http://www.e-musician.com |
| Harmony Central | http://www.harmony-central.com |
| MIDI Farm | http://www.midifarm.com |
| Creative Labs | http://www.creaf.com |

*MIDI Sequencing Software*

| | |
|---|---|
| Opcode Vision | http://www.opcode.com |
| Mark of the Unicorn Performer | http://www.motu.com |
| Emagic Logic | http://www.emagic.de |
| Steinberg Cubase | http://www.us.steinberg.net |

*Web MIDI Players*

| | |
|---|---|
| Beatnik | http://www.headspace.com |
| MIDPlug | http://www.yamaha-xg.com/english/xg/midplug/mplug.html |
| Crescendo | http://www.liveupdate.com |
| QuickTime 3.0 | http://www.apple.com/quicktime |
| ResRocket | http://www.resrocket.com |

*Other Articles of Interest* ftp://dev.apple.com/devworld/Interactive_Media_Resources

1) New Audio Features in QuickTime 3.0
2) Interactive Sound Design on the Web Featuring Beatnik Technology

## About the Author

Jeff Essex, Creative Director of audiosyncrasy, specializes in the use of digital audio and MIDI to create music, sound effects and voiceover for multimedia. In addition to his creative and audio engineering skills, he is intimately familiar with multimedia tools and technology. A veteran of MacroMind and Macromedia, he served two years as Technical Support Lead for Macromedia Director and SoundEdit. He is credited on over 40 CD-ROM titles, including products from Virgin Sound and Vision, Corbis Productions, Mindscape, 3DO, and Disney Interactive. For the past year he has worked as the primary composer and audio consultant for the leading children's interactive web site. His book, Multimedia Sound and Music Studio, (Random House/Apple New Media Library,1996) is the definitive guide to multimedia audio production. It won the Computer Press Award for Best Advanced How-To Book of 1996.