

# The Downloadable Sounds Level 1 Specification

James Wright<sup>(1)</sup>, Thomas White<sup>(2)</sup>, Todor Fay<sup>(3)</sup>, Daniel Petkevich<sup>(4)</sup>

(1) Computer Music Center, T.J. Watson Research Center, IBM  
jwright@watson.ibm.com, <http://www.research.ibm.com/music>

(2) MIDI Manufacturers Organization, MMA@midi.org;

(3) Microsoft, todorfay@microsoft.com;

(4) S3, dannyp@s3.com

## Abstract

The Downloadable Sounds specification (DLS) defines a base-line architecture and file format which supports the playback of runtime-configurable sounds. Primary goals include consistency of audible results, flexible range of sounds, interactive control and reduced data size. The specification is primarily aimed at multimedia computing applications and platforms, but should also be useful for a range of other purposes. DLS was developed by a group of hardware and software companies under the auspices of the Interactive Audio Special Interest Group and the MIDI Manufacturers Association.

This paper presents an overview of DLS Level 1, with a focus on the rationale, device architecture, file format and recommended protocols, and looks forwards to planned enhancements for DLS Level 2.

## 1 Introduction

Composers of interactive audio and music for multimedia or other computer-based applications face a basic dilemma. It is impossible to predict what their work will sound like, if they rely on common MIDI-based sound cards! Yet the only other alternative – recorded digital audio – is inflexible and requires massive amounts of storage space.

The Downloadable Sounds specification (DLS) offers a third choice, using a standardized sample playback architecture and file format, which combines the best features of MIDI-based synthesis engines and digital audio playback. To understand the intent and motivation behind DLS, it is useful to examine the first two approaches from the perspectives of consistency, interactivity, fidelity, range of sounds and data size. In this context, consistency means the ability to recreate an audio experience more-or-less identically on different systems, while interactivity is the ability to modify the audio performance in near-real-time in response to external events.

Digital audio is extremely consistent, offers selectable fidelity, and can reproduce any sound that can be recorded or created (although in practice fidelity and range of sounds is limited by data size). However, digital audio does not support interactivity very well: signal creation or significant processing is largely impractical with multimedia hardware. There is an inverse correlation between interactivity and fidelity: a broader range of sounds and more flexible runtime processing can be provided if fidelity is reduced.

MIDI-based synthesis engines offer a completely different set of characteristics. Unlike digital audio, MIDI data is extremely compact (three to four orders of magnitude more compact than CD-quality audio). And MIDI data supports a high degree of interactivity, since the actual sound is rendered in real time. Fidelity can be very good, depending on the synthesizer. Which brings us to a major aspect of MIDI: everything depends on the synthesizer. General MIDI (GM) offers a specific sound set intended for multimedia applications, but GM is not a complete solution in at least two respects. First, GM engines from different manufacturers will render the same score differently – sometimes to a large degree. A given score will only sound as intended on a small fraction of the installed base of GM synthesizers. Second, GM has a fairly limited palette of sounds (128 instruments, 47 drums).

Let's summarize these key factors:

	Digital Audio	GM
Consistency	Very good	Poor
Range of sounds	Excellent	Limited
Interactivity	Poor	Excellent
Fidelity	Variable	Variable
Data size	Huge	Compact

Consistency and range of sounds are the two most important considerations for most multimedia applications, with interactivity running a poor third. Sound designers want their work to sound the same, regardless of what hardware is used to render it. And sound designers also want to the freedom to use any

sounds they want. Digital audio has become the medium of choice for most multimedia titles, despite its inherent and quite serious limitations.

Imagine what would happen if the key benefits of MIDI – interactivity and compact data size – could be combined with the consistent playback and unlimited palette offered by digital audio. That, in essence, is what DLS provides.

## 2 Overview

The DLS specification was designed to meet five major objectives:

1. Work with most or all current hardware platforms that support some form of downloadable instruments. This is required for initial acceptance.
2. Guarantee a common playback experience. The combination of a given MIDI sequence and set of downloadable instruments should yield highly consistent results from platform to platform.
3. Support extensibility for new capabilities over time. While Level 1 must work with existing hardware, the Level 2 specification should anticipate and even help define new hardware.
4. Define an architecture that can be readily extended in future to support streaming audio and the downloading of new instruments during playback.
5. Provide an open, non-proprietary specification.

The actual specification contains two parts. The first part describes the playback device architecture, and how device parameters are defined. The second part describes a standard file format for distributing these samples. The membership of the MIDI Manufacturers Association (MMA) voted to approve the device architecture and file format in January 1997; formal adoption is pending. While not currently part of the approved specification, an additional document giving message and protocol recommendations for using DLS on various platforms and environments is currently under development and will be released later in 1997.

### 2.1 Instruments and Devices

Instruments and Devices are primary concepts in DLS. A *device* embodies a means of producing a variety of sounds: it may be hardware, software or both. An *instrument definition* is a specific device configuration for producing a particular set of related sounds. An *instrument* is a set of device resources configured to generate sound using a given instrument definition. One instrument might produce a traditional sound such as flute or piano, while another might produce sound effects, ambient textures or some other kind of sonic entity. Some non-DLS devices only support a fixed set

of instruments (e.g. the GM program set). DLS devices support *downloadable* instruments, which allow the current set of instruments to be changed by loading new instrument definitions.

DLS and General MIDI devices can typically render sixteen different instruments concurrently, with each instrument able to sound one or more notes at the same time. By contrast, digital audio devices generally render audio *streams* rather than instruments, and the total number of concurrent streams is usually less than sixteen (often between one and eight in multimedia applications). Indeed, the concept of an instrument that is *performed* is not supported in most digital audio devices: their fundamental model is a stream that is *reproduced*. Interactivity is inherently limited (or nonexistent) in such a model.

### 2.2 Device Architecture

DLS Level 1 defines a generic sample-playback device architecture with the following features:

- A digital oscillator that plays back sampled sounds (digital audio fragments) with optional looping and both stepped and continuous pitch control. Most instruments employ more than one sample.
- A digitally-controlled amplifier for controlling the amplitude of the sample.
- A single LFO (low frequency oscillator) used to modulate the pitch and/or amplitude of the sample.
- Two ADSR envelope generators used for pitch and amplitude variations.
- A defined set of articulation routings for configuring these elements.
- Standard behavior for a defined set of MIDI messages: Bend, Mod Wheel, Volume, Expression, Pan, Sustain, Fine and Coarse Tune (RPN 1 & 2).
- Extremely precise definition of all parameters, to ensure consistent results from compliant devices.

With a sample-playback architecture, the fundamental character of the sound is determined by the audio sample. Expressive variation is provided by modifying the pitch and amplitude characteristics of a given sample during playback. This can be accomplished either automatically (via envelopes or LFO) or in response to MIDI control events. Note that the Level 2 specification will encompass much greater capabilities.

DLS Level 1 organizes its sonic resources into two kinds of instruments: Melodic and DrumKit. There can be up to fifteen Melodic Instruments, but only one DrumKit. Each instrument definition organizes sample and articulation data using one or more regions, where each region spans a contiguous note range. Each region

specifies one wave sample and a set of related parameters:

- A Melodic Instrument is composed of up to 16 regions (each specifying a different sample) and a single global set of articulation data for all samples.
- The DrumKit instrument supports up to 128 distinct regions, each specifying one sample and independent articulation data. A DrumKit has no global articulation data.

A Level 1 device must provide at least 24 voices (independent notes) at a sampling rate of 22 kHz, using 16 bit data. At least 512 KB (256K 16-bit words) must be available for storing sample and articulation data. These are all minimum requirements, and suppliers of DLS devices are free to surpass any or all of them.

For consistent results, it is necessary to define what happens when all voices are sounding and a new MIDI Note On is received. DLS Level 1 uses a static voice allocation scheme. The drum channel (channel 10) has the highest priority, followed by the remaining channels in order (10, 1-9, 11-16). A channel may only “steal” voices from lower-priority channels. This guarantees that higher-priority notes will continue to sound even when device resources are overcommitted, and that other notes will be stolen in a predictable manner.

### 2.3 File Format

A DLS file holds the articulation and waveform sample data for a collection of instruments (see figure 1).

The DLS file format is based on the widely-used RIFF file format (introduced by Microsoft in 1992 and based on the pre-existing IFF format). WAVE and AVI files use RIFF format; AIFF audio files and Standard MIDI files use a similar chunk-based format.

RIFF has several benefits:

- The basic format is composed of a set of tagged data chunks which may be nested indefinitely.
- Custom-format chunks may be added at any level, and easily skipped over by applications which do not recognize them. This supports both proprietary extensions and future enhancements to DLS.
- Tools for working with RIFF format are widely available.

Each instrument is defined as a set of one or more regions. Waveform data is accessed indirectly through a pool table that contains a cross-reference table to the actual waveform data. This allows multiple instruments and/or regions to share sample data. Waveform data is stored as a set of nested WAVE files, to facilitate import from and export to standard audio editors. Optional version stamps and globally unique identifiers may be used for change and resource management.

### 2.4 Message & Protocol Recommendations

A set of software guidelines known as the Message and Protocol Recommendations (MPR) has been developed to augment the Device Architecture and File Format specifications. This document describes (but does not mandate) a set of software protocols and a reference runtime object architecture (figure 2) which implements these protocols.

The MPR includes a rich set of facilities and a C++ API definition. A basic MIDI SysEx protocol for DLS Device Control has also been proposed. Since DLS covers a broad range of potential uses and platforms, no single API is likely to satisfy all situations. Therefore, the MPR divides facilities into core and supplemental categories, and defines two different levels of service:

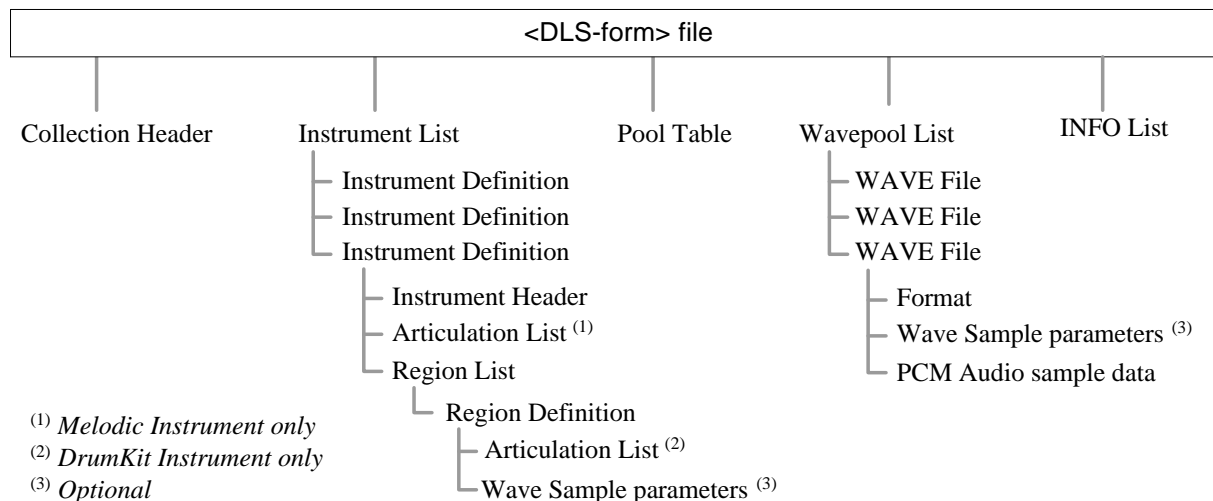


Figure 1. Partial schematic of DLS Level 1 File

- The basic level supports device-level resource management and convenient access to standard DLS objects.
- The extended level supports system-level resource management, extensibility, and fine-grained data access.

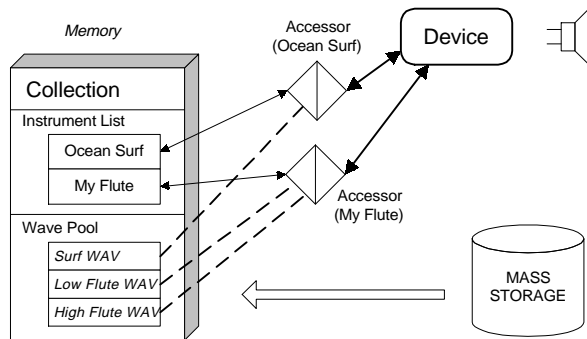


Figure 2: DLS Reference Runtime Architecture

While most DLS implementations will be relatively self-contained, it is also possible to use DLS protocols in systems connected to external DLS devices or collections. For example, one system might be connected to an external synthesizer using MIDI, Universal Serial Bus or IEEE-1394 (a.k.a. Firewire). Another system might use DLS protocols (together with TCP/IP or other network protocols) to access Collection data stored at a remote location. In these cases, DLS messages would be sent as actual message packets using the appropriate transport protocol. While bi-directional communications are highly desirable, the most basic DLS protocols can operate over a unidirectional link. The proposed MIDI SysEx DLS Device Control protocol can work with either one or two MIDI cables between an external synthesizer and the client system.

The Message and Protocol Recommendations have been submitted for review by the Interactive Audio Special Interest Group (IASIG) and MMA, but have not yet been adopted as part of DLS Level 1.

## 4 DLS Level 2

DLS Level 2 is currently under active development within the IASIG. Additions are expected to include filters, effects (e.g. reverb and chorus), improved articulation capabilities, more voices (polyphony), a higher minimum sample rate and increased amounts of base memory. The new specification should be completed in time for adoption by the MMA in January 1998.

## 5 Conclusions

DLS and MIDI offer compelling benefits for multimedia applications:

- DLS provides a standard wavetable sample playback architecture and file format. Since the standard is extremely precise, developers can be assured of a common playback experience across a diverse range of hardware. The MMA is planning a certification program to verify compliant devices in order to ensure this promise is realized.
- DLS provides a very broad potential sound palette – anything that can be captured in a digital audio sample and rendered with the L1 architecture.
- Using MIDI, DLS provides **much** better interactive control than streaming digital audio.
- Storage and bandwidth requirements are greatly reduced, compared to regular digital audio. This will enable an entirely new set of network-based music and multimedia applications.
- A large installed base of DLS devices is likely to be available quickly, through software driver upgrades and new product introductions (dedicated DLS chips are already in production).

## 6 Acknowledgments

DLS is the work of many hands. David Sparks (Sequoia) led a small group that focused on device architecture; their work was largely complete by the end of 1995 and forms the basis of DLS Level 1. Monty Schmidt (Sonic Foundry) and the Creative Labs/E-mu SoundFonts™ team were major influences on the file format. Jim Wright (IBM) is responsible for the MPR. Todor Fay (Microsoft) and Danny Petkevitch (S3) co-chaired the IASIG Downloadable Sounds Working Group during 1996, when DLS Level 1 was completed. Participating companies included AMD, Creative Labs/E-mu, Crystal Semiconductor, ESS, Euphonics, IBM, Invision Interactive, Microsoft, NVIDIA, Roland, S3, Sequoia Development, VLSI and Yamaha. We gratefully acknowledge all of the contributions involved in bringing DLS to fruition.

Portions of this paper are based on excerpts from the DLS Level 1 specification, adapted with permission from the MIDI Manufacturers Association.

## References

- [1] DLS Level 1 Specification (Device Architecture and File Format), January 1997, MMA
- [2] DLS Message & Protocol Recommendations, 1997
- [3] The Complete MIDI 1.0 Detailed Specification, version 95.1, MMA