



Enhanced Audio Solutions for the Digital World

Authoring Guidelines for Scalable Mobile Content:

Mobile XMF, SP-MIDI, and Mobile DLS



Contents

- 1. Detailed Overview ("White Papers").....3**
 - 1.1 What is Mobile XMF? 3**
 - 1.2 What is Mobile DLS? 5**
 - 1.3 What is SP-MIDI? 8**

- 2. Authoring Guidelines.....11**
 - 2.1 Mobile XMF Authoring Guidelines 11**
 - 2.2 Mobile DLS Authoring Guidelines 13**
 - 2.3 SP-MIDI Authoring Guidelines 16**

- Appendix: Built-In Instrument Bank & Program Numbers.....17**
 - GM Percussion Kit Keymap 17**
 - GM Melodic Instruments 18**

1. Detailed Overview ("White Papers")

This section introduces the essential technical concepts behind the major music content formats played by Beatnik’s mobileBAE audio engine: Mobile XMF, including Mobile DLS instruments and SP-MIDI musical score/performance files. These are all open standards adopted by 3GPP in order to allow MIDI-based content to be interoperable with a wide range of phone handsets.

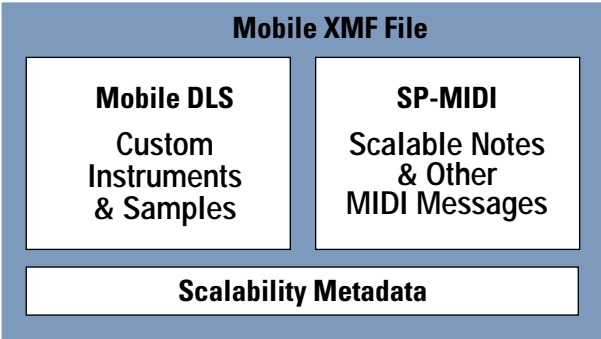
The benefit for the content developer is that they only need to author the content once for it to be played optimally on a wide range of handsets, as opposed to optimizing the content for individual handsets and therefore creating multiple versions of every composition.

Topics:

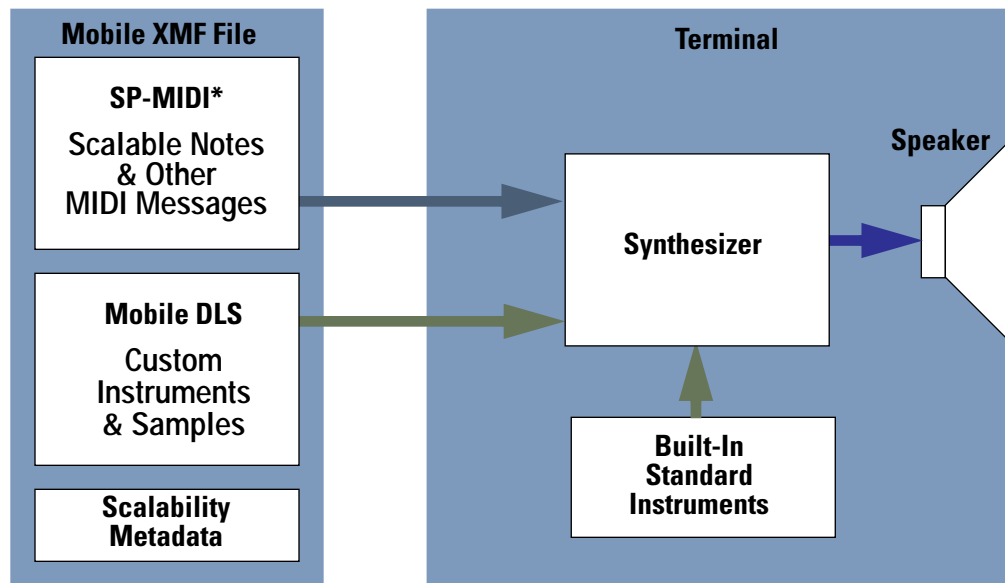
- 1.1 What is Mobile XMF? 3**
- 1.2 What is Mobile DLS? 5**
- 1.3 What is SP-MIDI? 8**

1.1 What is Mobile XMF?

Mobile XMF is an advanced, MIDI-based music and sound content format for mobile applications. Compared to earlier MIDI-based formats, Mobile XMF enables richer-sounding music by allowing the composer/arranger to select digital musical instruments that are especially chosen, or even custom-designed for each song. The same technology also allows a Mobile XMF file to contain a digital sample, for example a vocal sample or a game sound effect such as explosions, tire squeals, gunshots, and so forth. The custom instruments or samples use the new Mobile DLS format, the notes use the existing Standard MIDI File format, and they all travel together in the same Mobile XMF file to simplify transmission and server issues.



In the player, the music synthesizer has access to both the custom instruments and samples from the Mobile XMF file, and a built-in set of predefined standard instruments.



* Notes may be played on any combination of Built-In and Custom Instruments

1.1.1 Standardized Technology

Mobile XMF is based on the XMF Meta File Format, a bundling container technology developed and standardized by the MIDI Manufacturers Association (MMA), the worldwide authority¹ for interoperable MIDI technology. This container technology is also used in other MMA/AMEI content formats.

<http://www.midi.org/xmf>

Mobile XMF was also developed and approved by MMA/AMEI with Beatnik's participation, and has been adopted by the international mobile telephony standards body 3GPP as part of the Release 6 multimedia specification for mobile terminals and services.

<http://www.3gpp.org>

1.1.2 Expanded Musical Possibilities

Before Mobile XMF, MIDI-based content on mobile devices had a strictly limited palette of instruments that could be used. The legacy General MIDI and General MIDI 2 formats defined a pre-set palette of all possible instruments, which were selected by number.

There are two problems with this approach. First, if the particular instrument a composer really wanted to use for a musical part – say, a band's signature electric guitar sound, or a particular sampled drum sound – was not included in the pre-defined instrument list, there was simply no way to use that instrument in the piece. This limitation will in most cases seriously impact a ringtone's ability to evoke the original recording for the customer, and therefore its commercial value. Second, due to differences from terminal to terminal, the same instrument number could produce radically different sounding playback depending on what device was playing it. So consistent instrument timbres were a problem.

Mobile XMF addresses both of these problems: the Mobile DLS custom instruments use wavetable synthesizer technology, in which the same digital recording is used as the basis of the instrument sound on all terminals. This minimizes differences between terminals. And because the recording can be of literally any

1. Together with the Association of Musical Electronic Industry (AMEI), who have responsibility for MIDI in Japan.

audible sound – and can be further manipulated with processors such as envelope generators, modulation oscillators, pitch shifters, and optionally lowpass filters – the instrument palette for mobile MIDI content is now wide open. With Mobile XMF, individual melodic or percussive instruments, recorded music, singing, even sound effects and speech can now be used in MIDI-based mobile content.

1.1.3 Scalability

Perhaps the greatest challenge in creating content for mobile devices is the extremely wide range of capabilities that different devices have. Scalability, therefore, can be a critical factor in a content format's success. Mobile XMF content is scalable in two different dimensions: number of simultaneous musical notes that the device can generate (musically this is called 'polyphony': literally, 'many sounds'), and amount of processing used for each custom Mobile DLS instrument.

Scalable Polyphony – Every MIDI device is capable of generating some maximum number of musical notes at once. This cap ranges from 4 or 5 on extremely limited devices, to 64 or more on high-end terminals. The challenge of making MIDI content adaptable, so that the arrangement can sound complete on any terminal, and so that bad-sounding artifacts from uncompleted notes can be avoided on low-polyphony, was met with Scalable Polyphony MIDI (SP-MIDI). SP-MIDI is also a MMA/AMEI content format. In SP-MIDI, a musician gets to arrange the MIDI channels in order of priority, and then tell the player how to turn off only the MIDI channels that can't be properly played given the player's polyphony level. Mobile XMF retains this functionality from the SP-MIDI content format.

See also: **1.3 What is SP-MIDI?**

Scalable Instruments – Mobile DLS synthesizers can have two different kinds of voice architecture: one that's more musically expressive but more hungry for processing power, and one that's more limited but also more efficient. Because Mobile XMF content can react to the player's voice type, it can be made to play on any player, taking advantage of advanced features when they're present.

See also: **1.2.3 Mobile DLS Synthesizer Voice Architecture**

1.2 What is Mobile DLS?

Mobile DLS is a MIDI-oriented content format originally intended for musical instrument definitions, but which can also be used for other sampled sound recordings in wavetable form. This is distinct from the Standard MIDI File or SP-MIDI formats, which contain the musical score in the form of notes and other timed MIDI messages and events. Mobile DLS is used as the instrument and sample format in Mobile XMF. Compared to earlier DLS-based formats, Mobile DLS is specifically tailored for resource-constrained mobile applications.

1.2.1 Mobile DLS Standardization

Mobile DLS is based on MMA/AMEI's earlier DLS-1 and DLS-2 wavetable instrument formats. DLS-1 described a simple synthesizer voice with no filter. DLS-2 extended this by adding a filter, an additional LFO, additional modulation options, and a backwards-compatible file format with conditional loading mechanisms. DLS-1 and DLS-2 are widely supported in computer operating systems and sound cards, and some professional musical instruments.

Like Mobile XMF, Mobile DLS was developed and approved by MMA/AMEI with Beatnik's participation, and has been adopted by the international mobile telephony standards body 3GPP as part of the Release 6 multimedia specification for mobile terminals and services.

1.2.2 Wavetable Synthesizer Technology

Mobile DLS uses wavetable synthesizer technology. In all wavetable synthesizers, a digital recording of a sound is used as basis for the instrument sound. By applying various playback and processing techniques, the synthesizer is able produce different note sounds from the recording, with whatever pitch, duration, and timbre changes are requested by the notes and other MIDI messages. Taken together, all these parameters allow for significant creative control of the sampled sound. It is typical for an instrument file to contain just a few samples, but many instruments based on them, each very different sounding from all the others, due solely to differences in the playback parameter settings.

Pitch – If the sound is played back at the same speed it was recorded at, the note pitch is the same as the original instrument – let’s say, A above middle C. By playing the recording at a slower rate, lower pitches are produced (for example, G); higher pitches can be produced by playing the recording back at a faster rate (for example, B).

Multi-Sampling (‘Key Splits’) – Because most kinds of instruments or voices can sound unnatural when played too fast or too slow, the DLS formats allow a single instrument definition to contain multiple samples, for example one recorded at a low pitch and another recorded at a high pitch, to reduce the amount of pitch shift the synthesizer needs to apply. Multiple samples can also be used to record the original instrument as played at different velocities – for example, a recording of a piano key played soft, and another recording of the same key played hard. Within an instrument definition, the synthesizer can be made to select the sample with the pitch and velocity closest to the requested MIDI note, for a more believable sound.

Duration – Sounds that sustain for an arbitrarily long duration can be produced by looping a well-chosen section of the selected sample until the note is released. Sounds that would not naturally sustain, for example drum hits, are typically not looped.

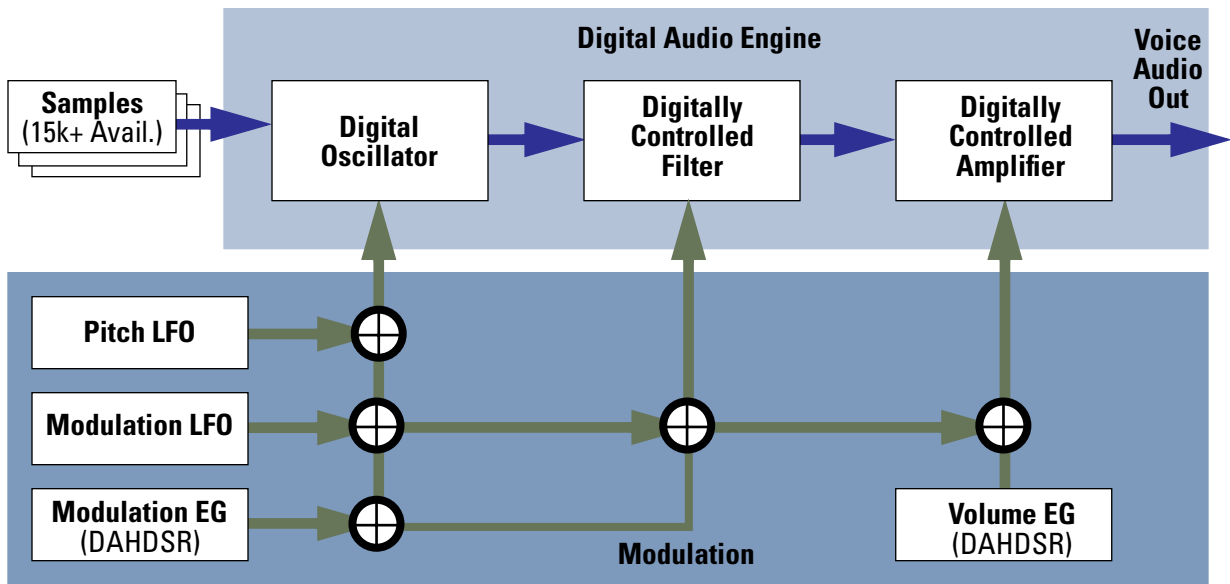
Processing and Modulation – Once the basic sampled tone is generated, it can be processed in various ways for musical effect. The spectrum of the signal can be varied by a Digitally Controlled Filter (DCF), both statically through a setting, and dynamically during the lifetime of a note through the use of modulators. Gain of the signal can be varied by a Digitally Controlled Amplifier (DCA), both statically through a setting, and dynamically during the lifetime of a note through the use of modulators. Modulators include low frequency oscillators (LFO) which produce a repeating pattern with a controllable period, and envelope generators (EG) which produce one-time curves with settings such as attack, decay, sustain, and release.

1.2.3 Mobile DLS Synthesizer Voice Architecture

There are two versions of the Mobile DLS synthesizer voice architecture: the ‘Base’ voice architecture and the ‘Plus’ voice architecture¹. Any Mobile DLS player may implement either version. All implementations of Beatnik’s mobileBAE engine shipped to date have used only the full voice architecture, not the reduced voice architecture.

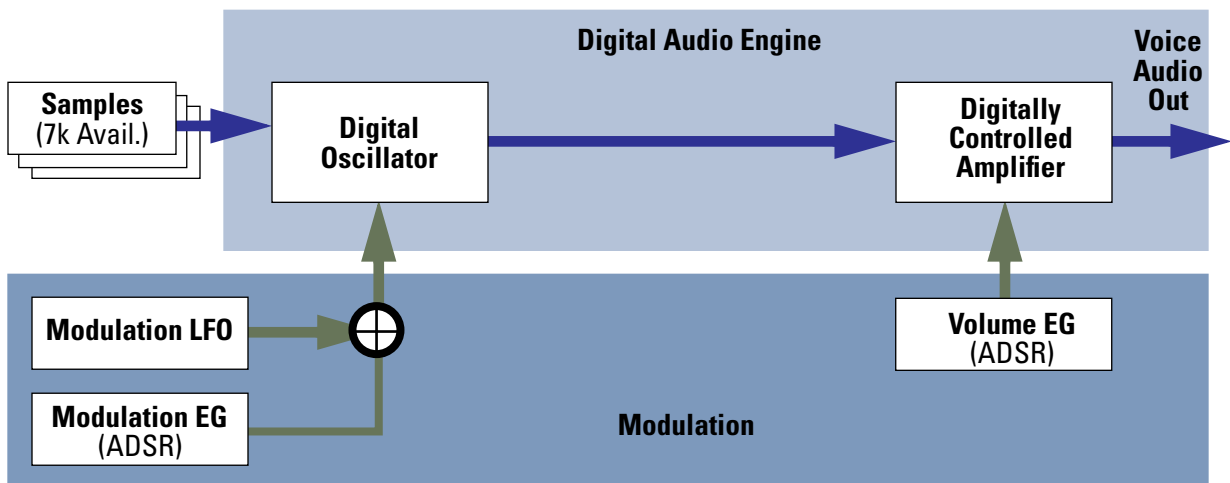
‘Plus’ Voice Architecture – The ‘Plus’ Mobile DLS voice architecture shown below incorporates a digitally controlled lowpass filter and other musically expressive synthesizer features. The filter allows the frequency spectrum of the sampled sound to be changed, either to darken the timbre by reducing higher frequency components, or to modify the spectrum as the note plays. The pitch of the sample, the cutoff frequency of the filter, and the gain of the amplifier can all be modulated from a total of four modulation sources (two Low Frequency Oscillators and two six-stage Envelope generators²). At least 15k bytes of uncompressed sample memory is available in players; some players may allow much more than that.

-
1. The terms ‘Base voice architecture’ and ‘Plus voice architecture’ do not appear in the Mobile DLS standard specifications as such. Instead, the more obscure term ‘optional voice processing blocks’ is used.
 2. These envelope generators offer Delay, Attack, Hold, Decay, Sustain, and Release controls.



'Plus' Mobile DLS Voice

'Base' Voice Architecture – Mobile DLS synthesizers can also have a 'Base' voice architecture that's more limited in its capabilities, but may also be more efficient and therefore able to produce higher polyphony counts for a given processor consumption cap. There is no filter, the Pitch LFO modulation source is removed, and the envelope generators lack the Delay and Hold controls (ADSR only). Only the sample pitch and the amplifier gain can be modulated. Also, sample memory is limited to 7k bytes.



'Base' Mobile DLS Voice: No Filter, Less Modulation, Less Memory

1.2.4 Sample Memory

It is important to understand that different Mobile DLS players may have different amounts of sample memory available. Players with the 'Base' voice architecture are not required to furnish more than 7k bytes of uncompressed sample memory. This is severely limited, and very challenging for instrument sound designers. Players with the 'Plus' voice architecture are required to furnish at least 15k bytes of uncom-

pressed sample memory, however there is no maximum boundary on the amount of sample memory that a player can provide. It is believed that market forces will naturally tend to drive these ‘Plus’ voice maximum sample sizes upwards for several years running.

While the absence of any fixed maximum sample memory size represents a kind of future-proofing for Mobile DLS and Mobile XMF, it also complicates the content serving situation somewhat. So, several mechanisms have been introduced to help make it easier to match a given terminal with appropriate content, and to allow a given piece of content to play on as many terminals as possible. To help content servers select appropriate content, in 3GPP Release 6 a terminal is able to signal the size of its sample memory. Also, metadata in a Mobile XMF file can be used to summarize the varying sample memory requirements of a given piece of content when played back at different SP-MIDI polyphony levels.

1.2.5 Wavetable Compression Codecs

Mobile DLS is the first DLS format to allow the instrument samples to be data compressed using common audio file encodings, such as G.711 a-law. Data compression allows for optimization of the size and transmission time of Mobile DLS content, because a given amount of sample duration can be made to fit into fewer bytes than the uncompressed version takes. An extensibility mechanism allows additional codecs to be registered and used¹ in future. (Once decompressed, samples must still fit into the player’s 7k or 15k memory space.)

1.3 What is SP-MIDI?

SP-MIDI was the first scalable MIDI music content format, meaning it sounds good on all players no matter how many musical voices are available. ‘SP’ stands for ‘Scalable Polyphony,’ polyphony being the musical term for ‘number of voices’. SP-MIDI technology is used both in the popular SP-MIDI content format, and in the new, more powerful Mobile XMF content format².

SP-MIDI was developed and approved by MMA/AMEI with Beatnik’s participation, and has been adopted by the international mobile telephony standards body 3GPP as part of the Release 5 and later multimedia specification for mobile terminals and services.

<http://www.3gpp.org>

1.3.1 Background: MIDI Music and the Scalability Problem

To see why it was necessary to invent SP-MIDI, which is to say why MIDI is not inherently scalable, it helps to understand some basics about how MIDI music work.

1.3.1.1 How MIDI Music Works

All MIDI content is in the form of note events recorded in a file, which at playback time are sent to a music synthesizer. The synthesizer can have up to 16 channels, each playing a different instrument – for example, one channel can be an organ, another channel can be a bass guitar, and so on. For an organ note, the note event is sent to the channel with the organ instrument. Inside the synthesizer, each note is played by a unit called a voice, which generates a sound with the desired instrument tone and the right pitch for the desired musical note. All of the voices work the same, so any voice can play any instrument. In fact, voices are dynamically assigned to channels in real time based on the notes the synthesizer actually receives, taking on whatever instrument has been assigned to that channel. Each synthesizer generally has a fixed upper limit on the number of available voices; this number is called the synthesizer’s polyphony.

1. For 3GPP Mobile DLS and Mobile XMF content, only 3GPP-approved codecs will be interoperable (currently only G.711 a-law).
2. SP-MIDI’s instrument palette is fixed, not extensible, whereas Mobile XMF adds the capability to use custom-designed Mobile DLS instruments.

1.3.1.2 Voice Stealing

So, what happens when the MIDI content asks to play so many simultaneous notes that the synthesizer's polyphony is exceeded? There just aren't enough voices, and something has to give. What happens is that the synthesizer starts making its own decisions about which notes to play, which ones not to play, and which ones to cut short so their voices can be re-used for new incoming notes. Synthesizer designers call this unavoidable problem 'voice stealing,' and it's the root of the scalability problem – problematic because no synthesizer is smart enough to consistently make the best, most musically appropriate decisions about what notes to steal. As a result, voice stealing always degrades MIDI music playback to some degree, by introducing some amount of a choppy character, and by preventing the full duration of some of the notes from being heard.

It gets worse. In a scalable environment, the degradation increases with the ratio of requested notes to synthesizer voices. For example, consider a song that needs 16 voices. Play it on a 16-voice synthesizer and it should sound fine, because there are generally enough free voices to handle all the notes. But play the same content on a synthesizer that only has 4 voices, and on average only about 1/4 of the notes will be heard, and the music will sound odd, even unpleasant or wrong, because many or all of the notes will be clipped short.

What does this situation mean for the composer who wants to make rich arrangements that take advantage of feature-oriented terminals with high voice counts? It means it can't be done. There would have to be different versions of the content, each tailored for a given polyphony range: say, a 4-voice version, an 8-voice version, a 16-voice version, etc. etc. etc.

This sort of approach is simply not viable for mobile content, not only because of the obvious inefficiency and cumbersomeness, but also because the polyphony count of the playback synthesizer can't necessarily be known in advance, for example when the MIDI content is sent as MMS from one terminal to another.

1.3.2 How SP-MIDI Addresses Polyphony Differences

Rather than simply relying on the synthesizer to make the right calls when there aren't enough voices to play all the notes, SP-MIDI introduces a new MIDI message that allows the music arranger to provide the synthesizer with specific, tailored guidance about what to do. This message, called the MIP¹ Message, regards the piece of MIDI content in terms of its channels, in two important ways. First, the musician can set up a custom priority order for the channels (instruments) – for example, highest priority might be a saxophone melody, second priority might be a drum line, third priority might be piano chords, etc. Second, the musician can tell the synthesizer how many voices are needed for each channel – for example, one voice for the sax, two voices for drums, three voices for piano, etc. This gives the synthesizer all the information it needs to shut off whatever lower-priority channels can't be played correctly. For example, on a one voice synthesizer only the sax would be heard; with a 3-voice synth, both the sax and drums would play; with a 6-voice synth, you'd hear the sax, the drums, and the piano – all without any irritating voice stealing degradation.

While this example is simple, the principle applies all the way up to 16 channels, and synthesizers up to 128 voices. This means the same piece of SP-MIDI content can sound good on very limited players, yet also produce truly rich sound when played on high-quality, high-polyphony players.

1. Acronym for 'Maximum Instantaneous Polyphony'

1.3.3 Details of the SP-MIDI Mechanism

To illustrate how SP-MIDI works at the detail level, let's continue with the same example. To encode the information about channel priorities and needed voices, the musician inserts a MIP Message into the MIDI data, using either the same MIDI tool they usually compose with, or a specialized SP-MIDI tool. Collecting our example data into a table, and assigning the instruments to some commonly used MIDI channels, it looks like this:

Instrument	MIDI Channel	Channel Polyphony Needed	Total Polyphony Needed
Sax	1	1	1
Drums	10	2	3 (1 + 2)
Piano	2	3	6 (3 + 3)

The last column, Total Polyphony Needed, is calculated cumulatively from the Channel Polyphony Needed plus the previous Total Polyphony Needed.

In the MIDI file, the MIP Message requires a list of number pairs, one pair for each MIDI channel in use. The first number in each pair is the MIDI channel number, and the second number is the total player polyphony level needed to let that channel play. So to add the MIP Message, the musician simply enters the required header and termination bytes, and types the three pairs from the MIDI Channel and Total Polyphony Needed columns above (all shown in hex notation):

```
F0 7F 7F 0B 01      MIP Message Header
01 01              Channel 1 needs player polyphony >= 1 to play
0A 03              Channel 101 needs player polyphony >= 3 to play
02 06              Channel 2 needs player polyphony >=6 to play
F7                  End of MIP Message Header
```

Once this MIP message is present in the MIDI file, SP-MIDI compliant players are able to prevent playback degradations due to note stealing by masking off lower-priority MIDI channels in a musically appropriate manner.

1. The hex representation of decimal number 10 is 0A.

2. Authoring Guidelines

The following Authoring Guidelines are intended for developers of Mobile XMF, Mobile DLS, and SP-MIDI content. Readers are assumed to be familiar with the basics of these formats, and with MIDI systems and music generally.

Topics:

- 2.1 Mobile XMF Authoring Guidelines 11**
- 2.2 Mobile DLS Authoring Guidelines 13**
- 2.3 SP-MIDI Authoring Guidelines 16**

2.1 Mobile XMF Authoring Guidelines

This section describes the tools and authoring practices that content creators need to use when producing Mobile XMF content.

See also: **1.1 What is Mobile XMF?**

2.1.1 Mobile XMF Tools

For best musical results, the SMF file should be created and edited in an environment that allows the actual Mobile DLS instruments to be used directly, not simulated. This is especially true in cases where highly specific responses to controllers and other real time MIDI messages is required. To achieve this, you'll need a desktop or laptop computer and three or four commercial MIDI music software applications:

1. A DLS or Mobile DLS instrument editor, such as Awave, to create and edit the custom Mobile DLS instruments. At this time, Awave is, to the best of our knowledge, the only available editor that specifically supports Mobile DLS.

See also: **2.2 Mobile DLS Authoring Guidelines**

2. A professional MIDI sequencer, such as Cubase, Digital Performer, Logic, Sonar, or Nuendo¹, to create and edit the SP-MIDI SMF file. Any sequencer that offers good control of exported SMF can be used for this.
3. A Mobile DLS synthesizer with SP-MIDI capabilities, to audition the Mobile DLS instruments with the SMF content. At this time, Beatnik Mobile Sound Builder (MSB) is, to the best of our knowledge, the only available Mobile DLS synthesizer for computers.
4. A Mobile XMF tool to assemble the parts into a single file. At this time, Beatnik MSB is, to the best of our knowledge, the only available Mobile XMF tool.

1. The sequencer product names Cubase, Digital Performer, Logic, Sonar, and Nuendo are trademarks of their respective owners, and are used for illustration only. No endorsement of any particular product should be inferred.

2.1.2 Preparing Mobile XMF Content

While it is beyond the scope of this document to teach how to create wavetable instruments and MIDI arrangements, content creators should be aware of several aspects of how Mobile XMF content works.

- In creating the Mobile DLS file and the SP-MIDI SMF file, it is critical that the instrument selection MIDI messages in the SP-MIDI SMF file directly agree either with the GM (or GM2) instrument set, or for custom instruments, with the instrument definitions present in the Mobile DLS file. See section 2.1.3.1. Instrument Bank Numbering.
- Mobile XMF players support SP-MIDI polyphony management, and your content may be played on synthesizer with different polyphony levels, perhaps radically different. During editing, content should be auditioned at different SP-MIDI polyphony levels. Note that at lower SP-MIDI polyphony levels, fewer of the Mobile DLS custom instruments, or even none of them, may be used.

Mobile DLS defines two different, though related, synthesizer voice architectures, and your content may be played on either one of them. If custom Mobile DLS instruments are used, then the content should be auditioned at both levels of synthesizer voice complexity ('Plus' voice vs. 'Base' voice).

In musical terms, the difference between the Base and Plus voice is that the Base voice lacks the following elements:

- Digitally Controlled Filter (DCF)
- Pitch ('vibrato') LFO
- Delay and Hold envelope generator stages.

Mobile DLS synths with the 'Base' voice are also limited to a total of 7k bytes of uncompressed wavetable memory.

2.1.2.1 Instrument Bank Numbering

Mobile XMF content can use two kinds of instruments: General MIDI (GM) instruments which the player provides automatically, and custom Mobile DLS instruments that you include in the Mobile XMF file. The GM instruments are located at known, fixed bank numbers, whereas any bank numbers can be used for your custom instruments – so long as the MIDI file and the custom instruments agree.

Note: The GM instruments' bank and program numbers are listed in the Appendix.

2.1.2.2 In the Standard MIDI File

Accessing GM Instruments – GM bank numbering is based on the General MIDI 2 spec. The player's GM percussion kits are located at bank MSB 78h LSB 00h (in decimal numbers: MSB 120, LSB 0). The GM melodic instrument set is located at bank MSB 79h and LSB 00h through 09h (in decimal numbers: MSB 121, LSB 0 through 9).

To access these GM instruments, your SMF content should observe the SP-MIDI and GM2 Bank Select and Program Change message rules: First set the bank by sending the Bank Select MSB and then the Bank Select LSB messages, in that order, and then select the instrument by sending the Program Change message. Bank Select MSB is MIDI controller number 0 and the value will be either 0x78 for percussion kits or 0x79 for melodic instruments. Bank Select LSB is controller 32 and the value will be either 0x00 for percussion kits, or 0x00 through 0x09 for melodic instruments.

For example, to select the GM Honky Tonk Piano instrument, which is instrument 3 in bank MSB 0x79 / LSB 0x00, your SMF should include the following sequence of MIDI messages:

```
Control Change: Controller 0, Value 121      : Bank Select MSB 0x79
Control Change: Controller 32, Value 0       : Bank Select LSB 0x00
Program Change: Program 0x03                 : Program Change 0x03
```

Accessing Custom Mobile DLS Instruments – Custom instrument bank numbering is up to whoever is creating your Mobile DLS instruments. Any custom instrument can use any Bank Select MSB + LSB address and any Program Change number. There is no distinction between ‘percussion instruments’ and ‘melodic instruments,’ so instrument selection produces the same results on all MIDI channels. In other words, channel 10 selects from the same instrument set as all other channels.

To access a custom Mobile DLS instrument, just use the same Bank Select MSB+LSB and Program Change numbers that the instrument uses. For example, to access program 0x56 in bank MSB 0x12 / LSB 0x34, your SMF should include the following sequence of MIDI messages:

```
Control Change: Controller 0, Value 0x12      : Bank Select MSB 0x12
Control Change: Controller 32, Value 0x034    : Bank Select LSB 0x34
Program Change: Program 0x56                  : Program Change 0x56
```

Custom Instruments Override GM Instruments – If a given instrument number (Bank Select MSB + LSB plus Program Change number) is present in both the custom instruments and the GM instruments, the player will use the custom version.

2.1.2.3 In the Mobile DLS File

Mobile DLS instrument sound designers can use any Bank Select MSB + LSB and Program Change number for any instrument, however every instrument in a given Mobile DLS should have a unique Bank Select MSB + LSB plus Program Change number¹.

Providing a Custom Override for a GM Instrument – If you want to, you can temporarily ‘override’ any of the built-in GM instruments by creating a custom Mobile DLS instrument that uses exactly the same Bank Select MSB + LSB and the same Program Change number as the desired GM instrument. Bank Select MSB 78h / LSB 00h overrides the GM percussion instruments, and MSB 79h / LSB 00h through 09h override the GM melodic instruments.

2.1.3 Checking XMF Content

Before shipping final Mobile XMF content, each file should be checked for the following common errors:

- Before creating the final Mobile XMF file, content creators should generally take care to remove any and all Mobile DLS instruments not used by the SP-MIDI SMF file. An exception would be if the Mobile XMF file was intended for use in an interactive application where ‘live’ MIDI note events were going to be sent to the Mobile DLS synthesizer from a source other than the SMF file.
- For best playback in mobile devices, Content Description Metadata should be included in the final Mobile XMF files. Beatnik MSB will automatically generate the necessary metadata and add it to your Mobile XMF file.

2.2 Mobile DLS Authoring Guidelines

This section describes the tools that content creators need to use, and authoring practices that should be observed, when producing Mobile DLS content.

See also: **1.2 What is Mobile DLS?**

1. Note that Mobile DLS ignores the DLS-1 and DLS-2 ‘ulBank bit 31’ part of the Bank address, which is used to distinguish percussion vs. melodic instruments. As a result, there is only a single instrument space in Mobile DLS, rather than one space for melodic instruments and a separate space for percussion instruments.

2.2.1 Mobile DLS Tools

For best musical results, Mobile DLS instruments should be created and edited in an environment where they can be heard and adjusted directly, not simulated. To achieve this, you'll need a desktop or laptop computer and a DLS or Mobile DLS instrument editor. At this time, Awave is, to the best of our knowledge, the only available editor that specifically supports Mobile DLS.

2.2.2 Preparing Mobile DLS Instrument Content

Making Mobile DLS instrument content is somewhat more involved than creating instruments or patches for professional instruments in the music studio. When creating instruments for the mobile environment, it is necessary to be careful about the amount of sample memory and what synthesizer voice features you use, because different devices will have different amounts of available memory and different synthesizer voices ('Plus' vs. 'Base'). It's also advisable to be aware of how much of a processing load your instruments will put on the playback device, since mobile terminals tend to have limited resources and it may be possible to overburden some less-powerful units.

Depending on your situation, you may be more concerned with portability, efficiency, or best sound. Below is one section with tips on optimizing your Mobile DLS content for each of those concerns.

2.2.2.1 Creating Portable Instruments

The following tips will help you create Mobile DLS instruments that are able to play on the widest possible range of Mobile DLS players.

- Use the 'Base' voice architecture, not the 'Plus' voice architecture. 'Base'-voice instruments will play on all Mobile DLS players, whereas 'Plus'-voice instruments will play only on 'Plus'-voice players. The 'Base' voice excludes the DCF, the Vibrato LFO, and the Delay and Hold stages in both EGs.
- Limit your samples to 7 kBytes. This is the most that all players are guaranteed to accept.

2.2.2.2 Creating Efficient Instruments

The following tips will help you minimize the memory and processor consumption of your Mobile DLS instruments.

Note: Although the same tools are used to create Mobile DLS instruments for use in Mobile XMF content and for use as the built-in instruments for a device or terminal, there may be different memory and/or efficiency constraints in each situation. For example, typically built-in instruments are optimized for minimum processor usage, whereas instruments for Mobile XMF use need to meet stricter memory limits.

- Use mono samples only, not stereo samples. Some Mobile DLS players only use the left channel, and stereo consumes twice the space of mono.
- When creating a bank such as a General MIDI instrument set with the Beatnik MSB tool, try to re-use existing instruments via aliases where possible, instead of creating a new one. An alias is simply a pointer that allows more than one bank/program number to use the same instrument definition. For example, if the sample size is small, it might be acceptable to use a flute instrument as the piccolo. This may in some cases require some minor work on the existing instrument, for example to extend the pitch range, but even so this is usually much more efficient than creating a second instrument.
- Use the filter only when necessary. Instruments that use the filter consume more processor bandwidth than unfiltered instruments.
- Use lower sampling rates in your audio samples. Lower rates mean smaller samples.
- Use 8-bit samples instead of 16-bit samples. They're half the size.
- Re-use samples in multiple instruments where possible. Any number of instruments can share the same sample.
- Don't have your loop end at the very end of the sample, leave a little space after the loop. When the

loop is at the very end, the audio engine's interpolator function has to work harder to smooth the signal.

- Avoid use of Loop-and-Release where possible. Few instruments benefit much from this technique, so in most cases the extra memory used by the release portion of the sample is probably better used for other things.
- Use of key splits is OK, but avoid layers (key splits that overlap) wherever possible. Notes in the split range will consume two synthesizer voices, not just one. This takes twice the processor bandwidth, and also reduces the player's effective polyphony.
- Limit the number of key splits per instrument. We recommend a maximum of no more than five. Every split consumes more memory, especially if a new sample is used.
- Plan your key splits and samples to avoid pitching samples up more than two octaves. It not only sounds better, it also places less processing load on the synthesizer.
- Try to keep your envelope Shut-Down times at the default minimum value (15 mSec). Longer values can cause old notes to hang on and overlap newer notes, reducing polyphony.
- Use the Modulation LFO only when necessary. It increases processor load somewhat.

The following recommendations impact only the instrument file size, they have no effect on processor usage:

- Leave both LFOs' initial Delay settings at zero, which is the default. A non-default value will have to be stored in the Mobile DLS file, increasing its size somewhat.
- Leave both EGs' Delay and Hold settings at zero, which are the defaults. A non-default value will have to be stored in the Mobile DLS file, increasing its size somewhat.

2.2.2.3 Creating Better-Sounding Instruments

The following tips should help you when creating instruments where the quality of the sound is your most important goal.

- Choose your samples and key splits so that the sample doesn't have to be pitched up by more than a major third, or down by more than 4 octaves. This may force you to use more key splits and/or more samples.
- While low sample rates will help you conserve wavetable memory, sound quality will suffer if you use too low a sample rate. Try to find the best balance between sample size and sample rate, rather than always using the lowest possible sample rate,

2.2.3 Checking Mobile DLS Instrument Content

Before shipping your Mobile DLS instruments, be sure to check the following points:

- Are there any parts (samples, key splits, articulations, etc.) not used in the instruments? Stray parts should always be removed to save memory.
- Are all your samples normalized? Synthesizers will provide better signal-to-noise performance with normalized samples.
- Is the average volume of your samples high? Due to background noise, mobile phone sounds work best when they have limited dynamic range.
- Do your samples have roughly the same perceived loudness? This makes balancing and editing instruments easier.
- Are your sample loops good? Bad loops are annoying, even on mobile phones.
- Do your samples exhibit DC offset? If extreme, this can cause distortion in the synthesizer.
- Do all of your instruments respond to all the notes (MIDI note numbers) they need to? Is the high-low

note range correct? Composers don't like instruments that don't play all the notes.

- In terms of volume, are all of your instruments well-balanced relative to one another? At all MIDI velocities? This shows polish that composers and arrangers will appreciate.

2.3 SP-MIDI Authoring Guidelines

Beatnik's authoring recommendations for SP-MIDI content are available on the web:

<http://www.beatnik.com/developers/practices.html>

See also: **1.3 What is SP-MIDI?**

Appendix: Built-In Instrument Bank & Program Numbers

This section details the built-in instruments in the General MIDI banks, with bank and program information for each instrument. There are two kinds of instruments: the GM Percussion Kit, and the GM Melodic Instruments.

GM Percussion Kit Keymap

The GM Percussion Kit is found at bank MSB 0x78 / 0x00, program 0x00. Once that program has been selected, the following instruments are available on each listed MIDI note number. Note numbers not listed here will be silent. Within an exclusivity group, every new Note On message will end all previous notes.

MIDI Note Number	Note Name	Instrument Name	Exclusivity Group
35	C_3	Acoustic Bass Drum	—
36	C#3	Bass Drum 1	—
37	D_3	Side Stick	—
38	D#3	Acoustic Snare	—
39	E_3	Hand Clap	—
40	F_3	Electric Snare	—
41	F#3	Low Floor Tom	—
42	G_3	Closed Hi-hat	1
43	G#3	High Floor Tom	—
44	A_3	Pedal Hi-hat	1
45	A#3	Low Tom	—
46	B_3	Open Hi-Hat	1
47	C_4	Low-Mid Tom	—
48	C#4	High Mid Tom	—
49	D_4	Crash Cymbal 1	—
50	D#4	High Tom	—
51	E_4	Ride Cymbal 1	—
52	F_4	Chinese Cymbal	—
53	F#4	Ride Bell	—
54	G_4	Tambourine	—
55	G#4	Splash Cymbal	—
56	A_4	Cowbell	—

57	A#4	Crash Cymbal 2	—
58	B_4	Vibra-Slap	—
59	C_5	Ride Cymbal 2	—
60	C#5	High Bongo	—
61	D_5	Low Bongo	—
62	D#5	Mute Hi Conga	—
63	E_5	Open Hi Conga	—
64	F_5	Low Conga	—
65	F#5	High Timbale	—
66	G_5	Low Timbale	—
67	G#5	High Agogo	—
68	A_5	Low Agogo	—
69	A#5	Cabasa	—
70	B_5	Maracas	—
71	C_6	Short Whistle	2
72	C#6	Long Whistle	2
73	D_6	Short Guiro	3
74	D#6	Long Guiro	3
75	E_6	Claves	—
76	F_6	Hi Wood Block	—
77	F#6	Lo Wood Block	—
78	G_6	Mute Cuica	4
79	G#6	Open Cuica	4
80	A_6	Mute Triangle	5
81	A#6	Open Triangle	5

GM Melodic Instruments

The basic GM Melodic instruments reside at bank MSB 0x79 / LSB 0x00, and depending on the player implementation, alternate versions of the instruments may reside at banks LSB 0x01 through 0x79. If the alternate version of a given program is not present, the basic version at LSB 0x00 will automatically be used instead.

Program Number		Instrument Name	Recommended MIDI Note Range	
Hex	Decimal		Lowest	Highest
0x00	0	Acoustic Grand Piano	21	108
0x01	1	Bright Acoustic Piano	21	108
0x02	2	Electric Grand Piano	21	108
0x03	3	Honky-tonk Piano	21	108
0x04	4	Electric Piano 1	28	103
0x05	5	Electric Piano 2	28	103
0x06	6	Harpsichord	41	89
0x07	7	Clavi	36	96

0x08	8	Celesta	60	108
0x09	9	Glockenspiel	72	108
0x0A	10	Music Box	60	84
0x0B	11	Vibraphone	53	89
0x0C	12	Marimba	48	84
0x0D	13	Xylophone	65	96
0x0E	14	Tubular Bells	60	77
0x0F	15	Dulcimer	60	84
0x10	16	Drawbar Organ	36	96
0x11	17	Percussive Organ	36	96
0x12	18	Rock Organ	36	96
0x13	19	Church Organ	21	108
0x14	20	Reed Organ	36	96
0x15	21	Accordion	53	89
0x16	22	Harmonica	60	84
0x17	23	Tango Accordion	53	89
0x18	24	Acoustic Guitar (nylon)	40	84
0x19	25	Acoustic Guitar (steel)	40	84
0x1A	26	Electric Guitar (jazz)	40	86
0x1B	27	Electric Guitar (clean)	40	86
0x1C	28	Electric Guitar (muted)	40	86
0x1D	29	Overdriven Guitar	40	86
0x1E	30	Distortion Guitar	40	86
0x1F	31	Guitar Harmonics	40	86
0x20	32	Acoustic Bass	28	55
0x21	33	Electric Bass (finger)	28	55
0x22	34	Electric Bass (pick)	28	55
0x23	35	Fretless Bass	28	55
0x24	36	Slap Bass 1	28	55
0x25	37	Slap Bass 2	28	55
0x26	38	Synth Bass 1	28	55
0x27	39	Synth Bass 2	28	55
0x28	40	Violin	55	96
0x29	41	Viola	48	84
0x2A	42	Cello	36	72
0x2B	43	Contrabass	28	55
0x2C	44	Tremolo Strings	28	96
0x2D	45	Pizzicato Strings	28	96
0x2E	46	Orchestral Harp	23	103
0x2F	47	Timpani	36	57
0x30	48	String Ensembles 1	28	96
0x31	49	String Ensembles 2	28	96
0x32	50	Synth Strings 1	36	96

0x33	51	Synth Strings 2	36	96
0x34	52	Choir Aahs	48	79
0x35	53	Voice Oohs	48	79
0x36	54	Synth Voice	48	84
0x37	55	Orchestra Hit	48	72
0x38	56	Trumpet	58	94
0x39	57	Trombone	34	75
0x3A	58	Tuba	29	55
0x3B	59	Muted Trumpet	58	82
0x3C	60	French Horn	41	77
0x3D	61	Brass Section	36	96
0x3E	62	Synth Brass 1	36	96
0x3F	63	Synth Brass 2	36	96
0x40	64	Soprano Sax	54	87
0x41	65	Alto Sax	49	80
0x42	66	Tenor Sax	42	75
0x43	67	Baritone Sax	37	68
0x44	68	Oboe	58	91
0x45	69	English Horn	52	81
0x46	70	Bassoon	34	72
0x47	71	Clarinet	50	91
0x48	72	Piccolo	74	108
0x49	73	Flute	60	96
0x4A	74	Recorder	60	96
0x4B	75	Pan Flute	60	96
0x4C	76	Blown Bottle	60	96
0x4D	77	Shakuhachi	55	84
0x4E	78	Whistle	60	96
0x4F	79	Ocarina	60	84
0x50	80	Lead 1 (square)	21	108
0x51	81	Lead 2 (sawtooth)	21	108
0x52	82	Lead 3 (calliope)	36	96
0x53	83	Lead 4 (chiff)	36	96
0x54	84	Lead 5 (charang)	36	96
0x55	85	Lead 6 (voice)	36	96
0x56	86	Lead 7 (fifths)	36	96
0x57	87	Lead 8 (bass + lead)	21	108
0x58	88	Pad 1 (new age)	36	96
0x59	89	Pad 2 (warm)	36	96
0x5A	90	Pad 3 (polysynth)	36	96
0x5B	91	Pad 4 (choir)	36	96
0x5C	92	Pad 5 (bowed)	36	96
0x5D	93	Pad 6 (metallic)	36	96

0x5E	94	Pad 7 (halo)	36	96
0x5F	95	Pad 8 (sweep)	36	96
0x60	96	FX 1 (rain)	36	96
0x61	97	FX 2 (soundtrack)	36	96
0x62	98	FX 3 (crystal)	36	96
0x63	99	FX 4 (atmosphere)	36	96
0x64	100	FX 5 (brightness)	36	96
0x65	101	FX 6 (goblins)	36	96
0x66	102	FX 7 (echoes)	36	96
0x67	103	FX 8 (sci-fi)	36	96
0x68	104	Sitar	48	77
0x69	105	Banjo	48	84
0x6A	106	Shamisen	50	79
0x6B	107	Koto	55	84
0x6C	108	Kalimba	48	79
0x6D	109	Bag pipe	36	77
0x6E	110	Fiddle	55	96
0x6F	111	Shanai	48	72
0x70	112	Tinkle Bell	72	84
0x71	113	Agogo	60	72
0x72	114	Steel Drums	52	76
0x73	115	Woodblock	—	—
0x74	116	Taiko Drum	—	—
0x75	117	Melodic Tom	—	—
0x76	118	Synth Drum	—	—
0x77	119	Reverse Cymbal	—	—
0x78	120	Guitar Fret Noise	—	—
0x79	121	Breath Noise	—	—
0x7A	122	Seashore	—	—
0x7B	123	Bird Tweet	—	—
0x7C	124	Telephone Ring	—	—
0x7D	125	Helicopter	—	—
0x7E	126	Applause	—	—
0x7F	127	Gunshot	—	—