



Mobile Media API (MMAPI) Reference

This appendix is a reference to the MMAPI. All classes, interfaces, and exceptions are listed here so that you can find them in one easy-to-use place. The Javadocs, however, remain the best place for getting detailed explanation about each method and class and is used in this appendix to take the one-line explanation of each class, interface, and exception.

This API is listed alphabetically, grouped by package.

Package `javax.microedition.media`

Interface `Control`

A `Control` object is used to control some media processing functions.

```
public interface Control {  
    // no methods  
}
```

Interface `Controllable`

`Controllable` provides an interface for obtaining the controls from an object such as a `Player`.

```
public interface Controllable {  
    // methods  
    public Control getControl(java.lang.String controlType);  
    public Control[] getControls();  
}
```

Class Manager

Manager is the access point for obtaining system-dependent resources such as Player instances for multimedia processing.

```
public final class Manager {
    // Constructors
    public static final java.lang.String MIDI_DEVICE_LOCATOR;
    public static final java.lang.String TONE_DEVICE_LOCATOR;

    // Methods
    public static Player createPlayer(java.lang.String locator)
        throws java.io.IOException, MediaException;
    public static Player createPlayer(
        java.io.InputStream stream, java.lang.String type)
        throws java.io.IOException, MediaException;
    public static Player createPlayer(DataSource source)
        throws java.io.IOException, MediaException;
    public static java.lang.String[] getSupportedContentTypes(
        java.lang.String protocol);
    public static java.lang.String[] getSupportedProtocols(
        java.lang.String content_type);
    public static TimeBase getSystemTimeBase();
    public static void playTone(int note, int duration, int volume)
        throws MediaException;
}
```

Class MediaException

A MediaException indicates an unexpected error condition in a method.

```
public class MediaException extends java.lang.Exception {
    // Constructors
    public MediaException();
    public MediaException(java.lang.String reason);
}
```

Interface Player

Player controls the rendering of time-based media data.

```
public interface Player extends Controllable {
    // Constants
    public static final int CLOSED;
    public static final int PREFETCHED;
    public static final int REALIZED;
    public static final int STARTED;
    public static final long TIME_UNKNOWN;
    public static final int UNREALIZED;
```


Interface TimeBase

A TimeBase is a constantly ticking source of time.

```
public interface TimeBase {
    // Methods
    public long getTime();
}
```

Package javax.microedition.media.control

Interface FramePositioningControl

The FramePositioningControl is the interface to control precise positioning of a video frame for Player instances.

```
public interface FramePositioningControl extends Control {
    // Methods
    public long mapFrameToTime(int frameNumber);
    public int mapTimeToFrame(long mediaTime);
    public int seek(int frameNumber);
    public int skip(int framesToSkip);
}
```

Interface GUIControl

GUIControl extends Control and is defined for controls that provide GUI functionalities.

```
public interface GUIControl extends Control {
    // Constants
    public static final int USE_GUI_PRIMITIVE;

    // Methods
    public java.lang.Object initDisplayMode(int mode, java.lang.Object arg);
}
```

Interface MetadataControl

MetadataControl is used to retrieve metadata information included within the media streams.

```
public interface MetadataControl extends Control {
    // Constants
    public static final java.lang.String AUTHOR_KEY;
    public static final java.lang.String COPYRIGHT_KEY;
    public static final java.lang.String DATE_KEY;
    public static final java.lang.String TITLE_KEY;
}
```

```
// Methods
public java.lang.String[] getKeys();
public java.lang.String getKeyValue(java.lang.String key);
}
```

Interface MIDIControl

MIDIControl provides access to MIDI rendering and transmitting devices.

```
public interface MIDIControl extends Control {
    // Constants
    public static final int CONTROL_CHANGE;
    public static final int NOTE_ON;

    // Methods
    public int[] getBankList(boolean custom) throws MediaException;
    public int getChannelVolume(int channel);
    public java.lang.String getKeyName(int bank, int prog, int key)
        throws MediaException;
    public int[] getProgram(int channel) throws MediaException;
    public int[] getProgramList(int bank) throws MediaException;
    public java.lang.String getProgramName(int bank, int prog) throws MediaException;
    public boolean isBankQuerySupported();
    public int longMidiEvent(byte[] data, int offset, int length);
    public void setChannelVolume(int channel, int volume);
    public void setProgram(int channel, int bank, int program);
    public void shortMidiEvent(int type, int data1, int data2);
}
```

Interface PitchControl

PitchControl raises or lowers the playback pitch of audio without changing the playback speed.

```
public interface PitchControl extends Control {
    // Methods
    public int getMaxPitch();
    public int getMinPitch();
    public int getPitch();
    public int setPitch(int millisemitones);
}
```

Interface RateControl

RateControl controls the playback rate of a Player instance.

```
public interface RateControl extends Control {
    // Methods
    public int getMaxRate();
    public int getMinRate();
}
```

```
    public int getRate();
    public int setRate(int millirate);
}
```

Interface RecordControl

RecordControl controls the recording of media from a Player instance.

```
public interface RecordControl extends Control {
    // Methods
    public void commit() throws java.io.IOException;
    public java.lang.String getContentType();
    public void reset() throws java.io.IOException;
    public void setRecordLocation(java.lang.String locator)
        throws java.io.IOException, MediaException;
    public int setRecordSizeLimit(int size) throws MediaException;
    public void setRecordStream(java.io.OutputStream stream);
    public void startRecord();
    public void stopRecord();
}
```

Interface StopTimeControl

StopTimeControl allows one to specify a preset stop time for a Player instance.

```
public interface StopTimeControl extends Control {
    // Constants
    public static final long RESET;

    // Methods
    public long getStopTime();
    public void setStopTime(long stopTime);
}
```

Interface TempoControl

TempoControl controls the tempo, in musical terms, of a song.

```
public interface TempoControl extends RateControl {
    // Methods
    public int getTempo();
    public int setTempo(int millitempo);
}
```

Interface ToneControl

ToneControl is the interface to enable playback of a user-defined monotonic tone sequence.

```
public interface ToneControl extends Control {
    // Constants
    public static final byte BLOCK_END;
    public static final byte BLOCK_START;
    public static final byte C4;
    public static final byte PLAY_BLOCK;
    public static final byte REPEAT;
    public static final byte RESOLUTION;
    public static final byte SET_VOLUME;
    public static final byte SILENCE;
    public static final byte TEMPO;
    public static final byte VERSION;

    // Methods
    public void setSequence(byte[] sequence);
}
```

Interface VideoControl

VideoControl controls the display of video.

```
public interface VideoControl extends GUIControl {
    // Constants
    public static final int USE_DIRECT_VIDEO;

    // Methods
    public int getDisplayHeight();
    public int getDisplayWidth();
    public int getDisplayX();
    public int getDisplayY();
    public byte[] getSnapshot(java.lang.String imageType) throws MediaException;
    public int getSourceHeight();
    public int getSourceWidth();
    public java.lang.Object initDisplayMode(int mode, java.lang.Object arg);
    public void setDisplayFullScreen(boolean fullScreenMode) throws MediaException;
    public void setDisplayLocation(int x, int y);
    public void setDisplaySize(int width, int height) throws MediaException;
    public void setVisible(boolean visible);
}
```

Interface VolumeControl

VolumeControl is an interface for manipulating the audio volume of a Player instance.

```
public interface VolumeControl extends Control {
    // Methods
    public int getLevel();
    public boolean isMuted();
    public int setLevel(int level);
    public void setMute(boolean mute);
}
```

Package javax.microedition.media.protocol

Class ContentDescriptor

A ContentDescriptor identifies media data containers.

```
public class ContentDescriptor {
    // Constructors
    public ContentDescriptor(java.lang.String contentType);

    // Methods
    public java.lang.String getContentType();
}
```

Class DataSource

A DataSource is an abstraction for media protocol handlers.

```
public abstract class DataSource implements Controllable {
    // Constructors
    public DataSource(java.lang.String locator);

    // Methods
    public abstract void connect() throws java.io.IOException;
    public abstract void disconnect();
    public abstract java.lang.String getContentType();
    public abstract java.lang.String getLocator();
    public abstract SourceStream[] getStreams();
    public abstract void start() throws java.io.IOException;
    public abstract void stop() throws java.io.IOException;
}
```

Interface SourceStream

Abstracts a single stream of media data.

```
public interface SourceStream extends Controllable {
    // Constants
    public static final int NOT_SEEKABLE;
    public static final int RANDOM_ACCESSIBLE;
    public static final int SEEKABLE_TO_START;

    // Methods
    public ContentDescriptor getContentDescriptor();
    public long getContentLength();
    public int getSeekType();
    public int getTransferSize();
    public int read(byte[] b, int off, int len) throws java.io.IOException;
    public long seek(long where) throws java.io.IOException;
    public long tell();
}
```



URI Syntax for Media Locators

The Uniform Resource Identifier (URI) syntax is described using a Request For Comment (RFC) paper as a universal means of describing the location of different types and kinds of resources. The RFC in question is 2396 and its full text can be accessed at <http://www.ietf.org/rfc/rfc2396.txt>.

Essentially, this RFC defines a way for providing a uniform and consistent way of describing the location of different types of resources, wherever they may be located. This is important to us in MMAPi, because MMAPi, being protocol and format agnostic, requires consistent means to access resources.

URI defines two parts to each identifier string: the scheme part and the scheme specific part separated by a ':' as shown here:

```
<scheme>:<scheme-specific-part>
```

As it applies to MMAPi, the scheme part is the protocol over which the resource is being referenced, whereas the scheme-specific part is a hierarchical, location-specific string, which tells the system the exact physical location of the resource.

Not surprisingly, the most common example of this is when resources are accessed over the Internet using HTTP. As an example, <http://www.ietf.org/rfc/rfc2396.txt> consists of the scheme "http" and the scheme-specific part of ["/www.ietf.org/rfc/rfc2396.txt"](http://www.ietf.org/rfc/rfc2396.txt). The hierarchies in the scheme-specific parts are separated by the "/" character. Thus, [rfc2396.txt](http://www.ietf.org/rfc/rfc2396.txt) is a physical document located in the `rfc` folder on the `www.ietf.org` server.

Of course, with MMAPi, the scheme part can be an entire range of options. As you have seen in this book, the scheme part can be any one of `capture`, `rtp`, `file`, `resource`, and, of course, `http` strings.

With each scheme part, different options for the scheme-specific part can be set. For `capture` scheme, you have used the `audio` and `video` scheme-specific parts. These can be further qualified to specify attributes, for example: `capture://audio?encoding=pcm` would give you a specific URI for capturing WAV audio files.



Advanced Multimedia Supplements—JSR 234

The Java Community Process (JCP) at <http://www.jcp.org> defines new technology for the Java technology using a community process, whereby new specifications are released for public overview and debated on. Once approved, the specs are introduced in the Java language. One of the most prolific areas of technology development is Java ME, and new APIs are being constantly created and released to target this area of Java development.

Specifications are released with a Java Specification Request (JSR) number. MMAPI was released as JSR 135 (<http://www.jcp.org/en/jsr/detail?id=135>). However, this specification is now more than two years old, and newer advances in technology have necessitated an update. JSR 234 (<http://www.jcp.org/en/jsr/detail?id=234>), titled Advanced Multimedia Supplements, is a step in updating the MMAPI for new technology devices. The final release of this specification was in June 2005.

However, note that JSR 234 is a supplement to JSR 135. Thus, it doesn't replace MMAPI nor is it used on its own, but provides supplementary implementations for newer technology devices. This appendix gives you an overview of the new JSR.

Introduction to JSR 234

JSR 234 was initiated by a Nokia request to create an API that could provide richer experiences to users of mobile devices. Mobile devices are increasing in their processing power capacity, so this enables sophisticated media processing functionalities available to MIDlet developers and distributors. Specifically, better controls for camera, radio and audio, and sophisticated MIDlets created based on them is the target aim of this specification.

The specification creates a new package, `javax.microedition.amms`, where `amms` stands for Advanced Multimedia supplements. The specification mostly defines new controls that are derived from the `javax.microedition.media.control.Control` interface, thus making these controls compatible with MMAPI (as you would expect, as JSR 234 is a supplement to JSR 135). It also defines four new concepts in the main `javax.microedition.amms` package, `GlobalManager`, `Spectator`, `Module`, and `MediaProcessor`.

GlobalManager

`GlobalManager` is like a `Manager` class for JSR 234. It provides methods to retrieve the specialized objects present in this specification. However, it doesn't replace or extend the `Manager` class, and you still use the `Manager` class for creating and managing `Player` instances.

Spectator

The `Spectator` class represents a virtual listener. It provides acoustic controls that can help create MIDlet applications that mimic sound coming from different locations and areas. You can create a `Spectator` class by using the `getSpectator()` static method in the `GlobalManager` class. This class implements the `Controllable` interface, which allows it to provide implementation of the `getControl()` and the `getControls()` methods.

Module

The `Module` interface is a logical grouping of multiple `Player` instances and/or MIDI channels. This functionality was missing from MMAPI, where you can treat a group of such instances as one entity. This interface is extended by two subinterfaces, `EffectsModule` and `SoundsSource3D`, which represent different types of groupings. `EffectsModule` represents `Player` instances and/or MIDI channels, to which a common effect can be applied, whereas `SoundsSource3D` represents this group in a virtual acoustical space.

Modules can be fetched from the `GlobalManager` class using the `createEffectModule()` or the `createSoundSource3D()` methods. Note that the `Module` interface also implements the `Controllable` interface.

MediaProcessor

`MediaProcessor` is an interface for postprocessing of media data. A `MediaProcessor` works very much like a `Player` interface and defines various states that mimic it as well. A `MediaProcessor` instance can be in one of `UNREALIZED`, `REALIZED`, `STARTED`, or `STOPPED` states.

You can retrieve an instance of `MediaProcessor` by using the `GlobalManager` class's static method `createMediaProcessor(String type)`, passing to it the MIME type of target processor.

This interface also extends the `Controllable` interface.

Controls

The rest of the interfaces in this API are various controls and are divided in various packages depending on the type of functionality that they provide.

Package `javax.microedition.amms.control`

There are two main control interfaces in this package that impact the rest of the packages as well.

`FormatControl` is used for specifying the format of the various media types. This interface is implemented by four different types of media formats using their own interfaces: `AudioFormatControl`, `ContainerFormatControl`, `ImageFormatControl`, and `VideoFormatControl`.

`EffectControl` is used to preset a filter on some media data to provide different types of effects. This interface provides a method, `setEnabled(boolean enable)`, to turn the effects on or off. This interface is supported with the `EffectOrderControl` interface, which controls the order in which multiple effects are applied.

Finally, this package contains the interface `PanControl`, which controls the panning of audio and video media data, and `PriorityControl`, which allows the manipulation of the priority given to various `Player` instances in a group situation.

Packages `javax.microedition.amms.control.imageeffect` and `javax.microedition.amms.control.audioeffect`

All of the interfaces in these two packages extend the `EffectControl` interface and provide various effects that can be applied to audio and image media data.

Package `javax.microedition.amms.control.audio3d`

All of the interfaces in this package provide controls for manipulation of audio in a 3D environment. The controls are retrieved from either the `SoundSource3D` or `Spectator` class, as the case may be.

Package `javax.microedition.amms.control.camera`

The controls in this package are retrieved from a `Player` instance that is created by using the `Manager.createPlayer("capture://camera")` method. These controls provide control over the exposure, the flash, focus, zoom, and burst shooting of snapshots (as opposed to single snapshots using the `getSnapshot()` method of the `VideoControl` interface provided in MMAPI). A `CameraControl` control provides further control over the camera by providing methods for shutter control, camera rotation, and so on.

Package `javax.microedition.amms.control.tuner`

This package contains two control interfaces for radio tuning. `RDSControl` is used for Radio Data System (RDS) tuning and provides methods to manage that. `TunerControl` isn't specifically targeted for radio tuning, but it can be used to tune into any `Player` instance that supports tuning, however unlikely.

JSR 234 Implementations

At this point, being a relatively new API, there are no known implementations on any devices, and neither is there any known emulator support. However, Nokia provides a reference implementation with some examples, which can be downloaded at <http://forum.nokia.com/java/jsr234>.

Index

■ Numbers and symbols

- # sign, used to denote a sharp, 76
- 3rd Generation Participation Project (3GPP), Adaptive Multi-Rate (AMR) format standard adapted by, 134

■ A

- Adaptive Multi-Rate (AMR) format, standard adapted by the 3rd Generation Participation Project (3GPP), 134
- addPlayerListener(PlayerListener listener) method, using to register an implementation with a Player instance, 47
- Advanced Multimedia supplements (JSR 234), 251–253
- After-Touch message, function of, 100
- AMR format. *See* Adaptive Multi-Rate (AMR) format
- AMR-Narrow Band (AMR-NB) format. *See* Adaptive Multi-Rate (AMR) format
- AMR-Wide Band (AMR-WB) format. *See also* Adaptive Multi-Rate (AMR) format for higher speech quality of sampled audio, 134
- AMS (Application Management Software). *See also* Application Management Software (AMS)
 - halts installation if unable to put MIDlet in a trusted domain, 65–66
- applause.wav, playing in Motorola C975 emulator, 23
- Apple's Darwin streaming server, Web site address for, 182
- Application Management Software (AMS), 29
- Apress Web site, for source code downloads, 180, 202
- audio
 - capturing, 146–162
 - getting user permission before recording to the Device Blog, 190–191
 - saving captured, 158–162
 - timed capture and playback, 147–151
 - audio and tone generation, MMAPI support for, 2
 - audio and video, working with, 127–184
 - AudioBlogEntry, in Device Blog MIDlet application, 193

- AudioEditCanvas class, used as the editable canvas for audio recording, 216–220
- AudioPlayer MIDlet
 - code for creating AudioPlayerCanvas, 27–29
 - examples in operation showing list of audio files, 22–23
- AudioPlayerCanvas
 - code for creating, 27–29
 - improving by caching Player instances, 29–36
 - interface for playing audio files, 23–26
 - modifying to CachedAudioPlayerCanvas, 30–33

■ B

- b, used to denote a flat, 76
- baby.wav
 - playing in Sun's DefaultColorPhone emulator, 23
 - table showing time taken to play back with caching of Player instances, 36
 - table showing time taken to start playback of across three devices, 30
- bank (or soundbank). *See also* soundbanks used to support more instruments than General MIDI specification allows, 102
- bar (segment of time), using a time signature to define beats in each, 82
- basic controls, MMAPI support for, 2
- BBC's live radio broadcast, Web site address for, 182
- Beginning J2ME: From Novice to Professional*, Third Edition, by Jonathan Knudsen and Sing Li (Apress, 2005), 3, 17
- BenQ (formerly Siemens) CX 75 emulator, Web site address for downloading, 3
- BenQ M75 emulator and device
 - capturing and previewing images and video on, 192
 - example of MMAPI capabilities in, 131
 - looking through the viewfinder in, 178
 - starting the Device Blog MIDlet on, 186
 - storing captured audio on, 162
 - table of properties supported by, 131
- BLOCK_END constant, for marking the end of a block, 87

BLOCK_START constant, for marking the start of a block, 87

blog editing classes, defining in the Device Blog MIDlet application, 196

blog entries, creating view for editing and previewing, 195

BlogEntry interface

- for all types of blog entries, 199
- code for handling user selection of type of, 231–232
- in Device Blog MIDlet application, 193

blogging. *See also* Case Study: Device Blogging

- building a mobile blogging MIDlet called Device Blog, 185–237

BlogServer class

- as a container for the blog server, 204
- BootstrapMIDlet or Controller class responsible for creating, 205
- in Device Blog MIDlet application, 194

BootstrapMIDlet class

- code for creating, 227–230
- control class of the Device Blog MIDlet application, 197–198

bpm (beats per minute), using

- TempoControl to set, 107–108

byte array, specifying notes you want played in any sequence you want in, 81

C

CachingAudioPlayerCanvas class, enabling event handling in, 53–55

camera viewfinder, looking through, 174–178

Canvas, code for displaying video in, 164–168

CapabilitiesMIDlet, code for listing capabilities of a MMAPI implementation, 128–129

captured audio, saving, 158–162

Case Study: Device Blogging, 185–237

channel, changing program and volume of, 115

Channel messages

- how they are differentiated from System messages, 98
- Voice or Mode Channel subclasses of, 97

Channel Pressure message, function of, 101

CLDC. *See* Connected Limited Device Configuration (CLDC)

cleanUp() method, for closing each Player instance after it is no longer required, 27

close() method

- actions to be performed when calling during different states, 46
- for releasing any resources held by the player, 27
- in a simple linear transition path for a Player instance, 40

closeAll() method, to shut down all Player instances, 34

CLOSED state

- of Player instance, 39
- Player instance in, 46

code example

- for adding the keyPressed(int keyCode) method to the CaptureVideoCanvas class, 179
- AudioEditCanvas used as the editable canvas for audio recording, 216–220
- AudioPlayer MIDlet creates AudioPlayerCanvas, 27–29
- for AudioPlayerCanvas—an interface for playing audio files, 23–26
- for BlogEntry interface for all types of blog entries, 199
- for BlogServer class acting as a container for the blog server, 204
- for the BootstrapMIDlet class for starting the Device Blog MIDlet, 227–230
- of a byte array for playing the first two stanzas of “Happy Birthday to You”, 87
- CachedAudioPlayerCanvas caches Player instances, 30–33
- CachingAudioPlayer uses the new CachedAudioPlayerCanvas, 34–36
- for CapabilitiesMIDlet for listing capabilities of a MMAPI implementation, 128–129
- for changing volume of a channel, 117
- for a class to create a binary JTS file from a hex representation, 89–91
- for controlling audio capture, 151–157
- controlling MIDI with pitch, tempo, and volume controls, 104–107
- converting EventHandler to use reference checking instead of object equality, 56–57
- for creating a standalone MIDI Player instance, 113
- for creating EditableDisplay as base interface for all editable objects, 211
- for custom encodings that allow you to create your own encodings, 133
- for displaying video in a Form or a Canvas, 164–168
- EchoEventsMIDlet echoes player events onscreen, 47–48
- EditCanvas class used as the base class to create editable canvases, 215–216
- for EditForm class to implement EditableDisplay interface, 211–213
- for enabling event handling in the CachingAudioPlayerCanvas class, 53–55
- EventHandler class, 51–52

- for first stanza of “Happy Birthday to You”, 87
- general format for each type of encoding string, 132
- for handling user selection of type of BlogEntry, 231–232
- of HTTP for accessing files, 8
- for ImagePreviewCanvas showing associated image to user as a preview, 223–224
- loading and playing JTS file from a JAR file, 91–92
- looking through a camera’s viewfinder and displaying onscreen, 174–177
- MediaBlogEntry class representing blog entries that have a media component, 200–201
- MIDICapabilitiesMIDlet for providing information about MIDIControl’s capabilities, 113–114
- MIDlet for timed capture and playback of audio, 148–151
- of MIDlet that plays all notes from 0 to 127 sequentially, 79–80
- for mixed encoding, 132
- mixing MIDI and sampled audio, 143–146
- modifying commandAction() method to find time taken to play an audio file, 30
- for modifying the JAD manifest file for the NetworkTest MIDlet, 65
- moving video in vertical space and muting volume, 169–173
- NetworkPlayerManager class implementing the Runnable interface, 69–70
- NetworkRunner is a single-threaded HTTPConnection class, 205–209
- of permission needed to record any media (audio or video), 147
- for playing a local MIDI file, 102–103
- for playing an audio file over the network, 63–64
- for playing C# for 5 seconds at max volume, 76–77
- for playing tone sequences, 88
- postEntry(BlogEntry entry) method of the Controller class, 230–231
- of postentry.jsp file, 232–235
- for PreviewCanvas abstract class, 221–223
- ProgramNamesMIDlet queries soundbanks/programs and plays notes, 121–124
- RateControllableMIDIMIDlet controls playback rate, 110–112
- for sending MIDI message commands, 116
- of the shortMidiEvent() method in action, 118–120
- a simple Java class to calculate note frequencies and int values, 77–78
- for a simple MMAPi MIDlet, 19–20
- specifying the encoding formats for audio.encodings property, 132
- StopTimeControlMIDlet allows you to play with the StopTimeControl, 135–138
- for storing and retrieving captured audio, 159–161
- streaming media over the network, 182–183
- TextBlogEntry class representing all entries that have a textual component, 199–200
- TextEditForm used to create editable form screens for text blog entries, 213–214
- that allows the user to start and stop recording video clips, 180
- ThreadedMIDlet creates a new thread to play back a media file, 67–69
- understanding threads in MIDlets, 60–61
- for the URLEncoder class, 209–211
- User class represents the user of the Device Blog MIDlet, 202–204
- using MetadataControl to display meta information, 139–141
- using RingToneConverter class to convert RTTTL to MMAPi format, 84–86
- using the BLOCK_END constant, 87
- using the BLOCK_START constant, 87
- using the PLAY_BLOCK constant, 87
- using the ToneControl interface for creating a tone sequence to play, 83–84
- video capture formats supported by Sun’s DefaultColorPhone emulator, 132
- VideoBlogEntry extends MediaBlogEntry and represents video blog entries, 202
- for the VideoPreviewCanvas that previews the video to the user, 225–227
- commandAction() method
 - function of for displaying video, 168
 - modifying to find time taken to play one of the audio files, 29–30
- common time, defined, 82
- Connected Limited Device Configuration (CLDC), 2
- ContentDescriptor class, in javax.microedition.media.protocol package, 246
- Control Change message, function of, 100
- Control Change messages, table of values, 117
- control classes, creating, 227–232

- Control object, in `javax.microedition.media` package, 239
- Control objects, that provide control over the functionality of a Player, 13–14
- Controllable interface, in `javax.microedition.media` package, 239
- ControllableMIDIMIDlet, running to control pitch, tempo, and volume, 104
- controlled capture and playback, of audio, 151–157
- Controller class, of the Device Blog MIDlet application, 197–198
- CreateJTSFileFromHexString class, for creating a binary JTS file, 89–91
- `createPlayer()` methods, provided by Manager class, 10
- `createPlayer(DataSource source)` method, 10
- `createPlayer(InputStream is, String type)` method, 10
- `createPlayer(String locator)` method, 10
- custom encodings, code example for, 133
- custom events, handling, 55–57

D

- DataSource class
 - basics, 8–9
 - function of, 7
 - in `javax.microedition.media.protocol` package, 246
- DataSource instance, creation of, 8
- `deallocate()` method, attempting to call on the Player instance, 41
- Device Blog MIDlet application
 - building, 185–237
 - bundling together common functionalities of a List in, 194
 - companion Web site, 185
 - control classes of, 198
 - creating a text-only entry for, 189
 - creating the code for, 198–235
 - creating the design for, 192–198
 - defining the blog editing set of classes in, 196
 - defining the blog preview set of classes, 197
 - the finished in action, 185–192
 - getting permission from the user for recording audio, 190–191
 - image recording on, 191–192
 - preview of the text-only blog entry, 189
 - starting, 186
 - Unified Modeling Language (UML) model for, 193
 - URL for posting entries to, 190
 - views in, 194–197
- Device Blog server, in action, 236

- Device Blog Web site
 - creating and posting entries to, 188–192
 - logging in to, 187–188
 - registering with, 186–187
 - selecting the entry type for posting to, 188
 - successful login to, 188
- device capabilities, MMAPI support for
 - discovery of, 2
- devices, querying the capabilities of, 127–133
- Digital Rights Management (DRM), of digital data, 42
- digitally recorded audio data. *See* sampled audio
- DistributedToneMIDlet, code that will load and play the `happybirthday_hex.jts` file, 91–92
- DRM. *See* Digital Rights Management (DRM)
- `dumpSequence()` method, provided by the RingToneConverter, 89
- duration
 - calculating using resolution and tempo, 82–83
 - using with `playTone()` method, 79–81

E

- EchoEventsMIDlet, example that echoes player events onscreen, 47–48
- EditableDisplay interface
 - code for creating, 195, 211–227
- EditCanvas class
 - providing the base class for editing media blog entries, 196
 - used as the base class to create editable canvases, 215–216
- EditForm class
 - for implementing the EditableDisplay interface, 211–213
 - provides the base class for editing textual blog entries, 196
- EffectControl interface, function of in `javax.microedition.amms.control` package, 253
- emulators, running ControllableMIDIMIDlet to control pitch, tempo, and volume, 104
- encoding formats, supported by Sun's DefaultColorPhone emulator, 132
- encoding strings. *See also* media encoding strings
 - types of, 132
- endofmedia event, delivered when no more media is left to play (or record/stream), 48
- event delivery mechanism, understanding in MMAPI, 50–51
- event handling class, creating, 51–55

event messages
 giving them time to process, 118–120
 sending short and long, 115

eventData
 function of when a particular event is fired, 49
 table of events and corresponding event data, 50

EventHandler class
 code for, 51–52
 converting to use reference checking instead of object equality, 56–57

events, and corresponding event data, 50

exceptions thrown, when playing a media file using MMAPI, 21–22

F

feature set, defined, 15

FileConnection API (JSR-75), using to store data on a device, 158

filename extensions
 for Java Tone sequence files, 89
 for MIDI Player based on a physical file, 11
 for SMF files distributed on the Internet, 101
 for Standard MIDI file format, 101
 for a tone Player instance, 11

files
 example of HTTP for accessing, 8
 playing tone sequences stored in, 91–92

flat, defined, 75–76

Form
 bundling together common functionalities of, 195
 code for displaying video in, 164–168

FormatControl interface, in `javax.microedition.amms.control` package, 252

formula
 for calculating tone duration using resolution and tempo, 82–83
 for SEMITONE_CONST, 76
 using a MMAPI to calculate note values, 76–78

FramePositioningControl
 in `javax.microedition.media.control` package, 242
 methods to convert media time to frame numbers, 181
 seeking video frames with, 181
 for video data that allows access to individual frames, 13

frames. *See* video frames

frequency. *See* note, pitch, and frequency

G

Gauge item, for controlling volume and volume level reflected on screen, 23

General MIDI (GM) specification, core features and function of, 101–102

GenericForm class, bundling together common functionalities of a Form in, 194

GenericList class, in Device Blog MIDlet application, 194

getBankList(boolean custom) method, for getting a list of all installed soundbanks, 114

getBankList(false) method, for getting a list of all soundbanks, 124–125

getControl("VolumeControl ") method, retrieving the VolumeControl with, 52

getDateTimePosted() method, 200

getKeyName(int bank, int program, int key) method, for getting the name of the device-assigned name of a key, 115

getKeys() method, for getting a list of keys, 141

getKeyValue() method, for getting the value assigned to each key, 141

getMaxRate() method, for querying for the maximum playback rate, 109

getMinRate() method, for querying for the minimum playback rate, 109

getName() method, must be a CLDC 1.1 method for the Thread class, 60–62

getPitch() method, using to query for the pitch value in milli-semitones, 108

getProgramList(int channel) method, for getting the current instrument/program assigned to a given channel, 115

getProgramList(selectedBank) method, using to display the programs installed on a soundbank, 125

getProgramName(int bank, int prog) method, for getting the name of an instrument/program, 115

getProgramName(selectedBank, programs[i], null) method, using to display the programs installed on a soundbank, 125

getRate() method, using to get the current playback rate of a musical piece, 109

getSeekType() method, function of, 8

getSnapshot() method
 taking a snapshot of an image that is playing on a device screen with, 174
 using to take snapshots, 178–179

getState() method, determining the current state of a Player instance with, 40

`getStopTime()` method, returning the preset time with, 139
`getSupportedContentProtocols(String contentType)` method, for getting protocols over which it can be accessed, 11
`getSupportedContentTypes(null)` method, for getting the supported formats for all media types, 163–181
`getSupportedContentTypes(String protocol)` method, for a list of all supported content types for a particular protocol, 11
`getTimeBase()` method, accessing built in default `TimeBase` instance with, 9
`getTransferSize()` method, function of, 8
`GlobalManager`, function of, 252
`guessFileExtension()` method, 202
`GUIControl`
 for data that requires a display, 13
 in `javax.microedition.media.control` package, 242

H

half step. *See* semitone
 Helix streaming server, Web site address for, 182
 HTTP access, code for modifying the JAD manifest file for, 65
 HTTP/HTTPS access, required permissions for, 17
 HTTPS access, code for modifying the JAD manifest file for, 65

I

`IllegalStateException`
 thrown by `realize()` method, 42
 thrown if calling `start()` on an instance that is in the `CLOSED` state, 44
 thrown if `prefetch()` is called on a `CLOSED` instance, 43
`ImageBlogEntry`, in Device Blog MIDlet application, 193
`ImagePreviewCanvas`, showing the associated image as a preview, 223–224
 images, recording on Motorola C975 and BenQ M75 devices, 191–192
`initDisplayMode(int mode, Object arg)` method, providing basis on which video is displayed, 163
 Interactive MIDI feature set, function of, 15–16
`isBankQuerySupported()` method
 for finding whether a full or minimum `MIDIControl` is available, 120
 function of, 114–115

`itemStateChanged(Item item)` method
 using to raise the pitch value, 108

J

Java Community Process (JCP), Web site address for, 5, 251
 Java ME
 JVM in vs. JVM in Java SE, 59
 understanding threads in, 59–63
 Java ME development environment, showing MMAPI as an optional package, 4
 Java ME games, mixing sampled audio, MIDI, and/or tones in, 142
 Java Micro Edition (ME) platform, optional packages for developing applications for, 3–4
 Java SE, JVM in vs. JVM in Java ME, 59
 Java Specification Requests (JSR) 135, introduction of Mobile Media API (MMAPI) via, 1
 Java Tone Sequence files, filename extension for, 89
 Java Verified Program, Web site address for details about, 67
 Java Wireless Toolkit
 reference implementation of MMAPI installed with, 3
 Web site address for downloading, 3
 Java-enabled devices, using MMAPI to create applications for, 1–2
 Java-enabled mobile phones, MMAPI features and requirements for, 2
`javax.microedition.amms` package, new concepts defined in, 251–253
`javax.microedition.amms.control` package, control interfaces in, 252–253
 `audio3d` package, 253
 `audioeffect` package, 253
 `camera` package, 253
 `imageeffect` package, 253
 `tuner` package, 253
`javax.microedition.media` package
 alphabetical reference, 239–242
 `Player` interface defined in, 7
`javax.microedition.media.control` package, alphabetical reference, 242–246
`javax.microedition.media.control.Control` interface, new controls derived from, 251–253
`javax.microedition.media.protocol` package
 alphabetical reference, 246–247
 `DataSource` class defined in, 7
 JCP specification 135, development of by Sun, Nokia, and Beatnik, 5
 JSR 135, MMAPI released as, 251

- JSR 234
 - implementations, Nokia Web site address
 - for reference implementation samples, 253
 - introduction to for mobile devices, 251–253
 - .jts filename extension, for Java Tone Sequence files, 11, 89
 - jts files, creating, 89–91
- K**
- .kar or .mid filename extension, for MIDI Player based on a physical file, 11
 - keyPressed(int keyCode) method, adding to the CaptureVideoCanvas Class, 179
 - Knudsen, Jonathan and Sing Li, *Beginning J2ME: From Novice to Professional*, Third Edition by, 3, 17
- L**
- laughter.wav, playing in Motorola C975 device, 23
 - LCDUI Image class, for manipulating image data, 174
 - Li, Sing and Jonathan Knudsen, *Beginning J2ME: From Novice to Professional*, Third Edition by, 3, 17
 - listRoots() method, for finding the root folder of a device, 158
 - loadVideo() method, for displaying video, 168
 - long MIDI event message, sending, 115
 - longMidiEvent(byte[] data, int offset, int length) method, for sending System Exclusive messages, 115
- M**
- Manager class
 - as bridge between a DataSource and a Player, 10
 - in javax.microedition.media package, 240
 - scaled-down version of in the MIDP 2.0 subset of MMAPI, 14–15
 - static method to play single tones provided by, 11
 - understanding, 9–12
 - using to create Player instances in MMAPI, 10
 - mapFrameToTime(int framenumbers) method, for converting frame numbers to media time, 181
 - mapTimeToFrame(long mediaTime) method, for converting media time to frame numbers, 181
 - media, accessing over the network, 59–71
 - media encoding strings. *See also* encoding strings
 - for specifying the format of media supported/desired for operations, 132–133
 - media encodings
 - understanding, 132–133
 - using for mixed formats, 132
 - media locators, URI syntax for, 249
 - media player, lifecycle and events, 39–58
 - media types, MMAPI support for
 - multiple, 2
 - MediaBlogEntry class
 - in Device Blog MIDlet application, 193
 - representing blog entries that have a media component, 200–201
 - MediaException
 - thrown by realize() method, 42
 - thrown if an error occurs when processing media for playback, 44
 - thrown if an error occurs when processing or decoding media data, 43
 - MediaException class, in javax.microedition.media package, 240
 - MediaProcessor interface, function of, 252
 - MetaDataControl
 - gathering information using, 139–141
 - in javax.microedition.media.control package, 242–243
 - predefined keys in, 141
 - used to determine metadata information stored within a media stream, 13
 - using to display meta information on Motorola C975 device, 141
 - .mid filename extension, for SMF files distributed on the Internet, 101
 - .mid or .kar filename extension, for MIDI Player based on a physical file, 11
 - Middle C (equal to 60), example for composing, 100
 - MIDI (Musical Instrument Digital Interface)
 - managing using MIDIControl, TempoControl and PitchControl, 95–125
 - playing without MIDIControl, 102–112
 - understanding, 95–102
 - using in MMAPI, 102–125
 - MIDI and tone Player constants, table of, 11
 - MIDI communication, illustration of, 96
 - MIDI feature set, function of, 15
 - MIDI files
 - controlling the pitch, tempo, and volume of, 104–109
 - playing, 102–103
 - MIDI message format
 - function of, 97–99
 - illustration of standard, 97
 - MIDI messages
 - classifications of, 98
 - sending simple, 116–120

- storing and distributing, 101–102
 - table showing how to spot the different messages, 98–99
 - types of, 97
- MIDI mixing, implemented by Motorola C975 device, 142
- MIDI Player instance, creating a standalone, 113
- MIDI sequences, varying the speed of playback of, 107–108
- MIDI sounds, creating, 11
- MIDI specification
 - first published in 1982, 95
 - understanding, 96–101
- MIDICapabilitiesMIDlet, for providing information about MIDIControl's capabilities, 113–114
- MIDIControl
 - in `javax.microedition.media.control` package, 243
 - Motorola C975 emulator support for a full, 120
 - playing MIDI with, 113–125
 - playing MIDI without, 102–112
 - querying for capabilities, 113–115
 - that enables access to a device's MIDI player, 13
- MIDlets
 - embedding multimedia capabilities in, 1
 - MIDP 2.0 as profile for the development of, 3–4
 - sources multimedia data can be received from, 7
- MIDP. *See* Mobile Information Device Profile (MIDP)
- MIDP 2.0, how MMAPI fits with, 3–4
- MIDP 2.0 subset, table of MMAPI subset features, 14–15
- mixed encoding, code example for, 132
- MMAPI. *See also* Mobile Media API (MMAPI)
 - example of BenQ M75 emulator and device MMAPI capabilities, 131
 - feature set implementations, 15–16
 - getting started with, 19–37
 - how it fits with MIDP 2.0, 3–4
 - issues required to look at for using across a network, 67
 - as an optional package for Java Micro Edition (ME) platform, 3–4
 - specification 135, 5
 - table of devices that support as an optional package, 5
 - table of exceptions thrown playing a media file using, 21–22
 - table of system properties, 128
 - understanding the event delivery mechanism in, 50–51
 - using MIDI in, 102–125
- MMAPI architecture, how it achieves platform and format neutrality, 7–17
- MMAPI controls, table of standard, 13
- MMAPI development, permissions that are interesting for, 63
- MMAPI format, converting RTTTL to, 84–86
- MMAPI formula
 - based on the frequency of desired note to be played as a tone, 76
 - for SEMITONE_CONST, 76
 - using to calculate note values, 76–78
- MMAPI implementations, supported protocols and content types, 11–12
- MMAPI MIDlets, creating, 19–37
- MMAPI system properties, table of, 128
- mobile blogging. *See also* blogging; Case Study: Device Blogging; Device Blog
- Mobile Information Device Profile (MIDP) 2.0, 1
- Mobile Media API (MMAPI). *See also* MMAPI
 - defined, 1–2
 - downloading implementation of supplied by Nokia, 3
 - features and requirements, 2
 - getting, 3
 - introduction to, 1–6
 - reference, 239–247
- Mode Channel messages, function of, 98
- model classes, creating for the Device Blog MIDlet application, 198–205
- Model-View-Controller (MVC) pattern, using to create the Device Blog MIDlet design, 192–198
- Module interface, function of, 252
- monophonic ring tones, defined, 73
- Motorola C975 device
 - audio/amr and audio/amr-wb formats supported by, 146
 - controlled audio recording MIDlet in action on, 157
 - example of MMAPI capabilities in, 130
 - image recording on, 191
 - lack of video recording support in, 188
 - looking through the viewfinder in, 178
 - MIDI mixing implemented by, 142
 - mixing MIDI and audio in, 146
 - moving video around and muting audio, 173
 - recording and previewing audio on, 191
 - starting the Device Blog MIDlet on, 186
 - video playback on a Canvas on, 169
- Motorola C975 emulator
 - example of installed banks and programs on, 121
 - example of MMAPI capabilities in, 130
 - example of RateControllableMIDIMIDlet running on, 112

- lack of TempoControl working in, 109
- running ControllableMIDIMIDlet to control pitch, tempo, and volume, 104
- running StopTimeControlMIDlet in, 135
- screen and console output for the ThreadTest MIDlet on, 62
- testing the NetworkTest MIDlet in, 67
- Motorola SDK V5.2, Web site address for downloading, 3
- multimedia capability, provided by Player and DataSource, 8
- multimedia player, building a simple, 19–22
- Musical Instrument Digital Interface (MIDI). *See* MIDI (Musical Instrument Digital Interface)
- Musical Instrument Digital Interface (MIDI) sounds, creating, 11

N

- Netbeans Integrated Development Environment (IDE), Web site address for, 3
- network
 - accessing media over, 59–71
 - code for playing an audio file over, 63–64
- network access
 - granting permissions for, 65
 - of a media file using a separate thread over Sun and Motorola emulators, 71
 - understanding permissions for, 63–67
- NetworkPlayerManager class, using to create a new thread, 67–69
- NetworkRunner class
 - code for, 205–209
 - function of, 198
- NetworkTest MIDlet, 65
- Note OFF message, function of, 100
- Note ON message, function of, 100
- note parameter, formula based on the frequency of desired note to be played as a tone, 76
- note, pitch, and frequency
 - table of frequency values for the basic notes, 74
 - understanding, 74–76
- note values
 - table of integer for the basic notes, 78
 - using a MMAPi formula to calculate, 76–78
- notes
 - defined, 73
 - specifying duration of using resolution, 82
 - table of frequency values for the basic, 74
 - table of integer values for the basic, 78

O

- object equality check, using to catch standard events in playerUpdate() method, 56
- octaves, 74–75
- OK command handler, for the AudioEditCanvas, 220

P

- pauseMedia() method, for pausing the Player instance, 26
- PCM (Pulse Code Manipulation), supported by Sun's DefaultColorPhone emulator, 132
- permissions, understanding for media access over the network, 63–67
- pitch. *See* note, pitch, and frequency
- Pitch Bend message, function of, 101
- PitchControl
 - for controlling the pitch (frequency) of audio data, 13
 - how it works, 108–109
 - in javax.microedition.media.control package, 243
 - using for simple changes, 102
- PLAY_BLOCK constant, using followed by the number identifying the block, 87
- player events, responding to, 47–57
- Player instances
 - creating for MIDI and tone sounds, 11
 - creating functional, 22–36
 - creating to play a MIDI file, 102–103
 - EchoEventsMIDlet acting as a listener for, 48
 - exploring the different states, 39–46
 - lifecycle and events, 39–58
 - methods for making usable, 40–41
 - overview of states it goes through during its lifetime, 39
 - resetting the TimeBase for, 9
 - to start playing the media file, 21
 - synchronizing two that use the same TimeBase, 9
 - using methods of the Manager class to create, 10
- Player interface
 - basics for playing and managing multimedia data, 9
 - function of, 7
 - in javax.microedition.media package, 240–241
- PlayerListener interface
 - implementing to receive notification of asynchronous events, 9
 - in javax.microedition.media package, 241
 - for responding to player events, 47–57
 - table of player events and when they are fired defined in, 49

Players and DataSources, understanding, 7–9

playerUpdate() method

- parameters taken by, 49
- using object equality check for catching standard events in, 56

playerUpdate(Player player, String event, Object eventData) method, invoking, 47

playMedia(String locator) method, function of, 26

playTone() method

- exceptions that can be thrown by, 81
- using, 79–81

playTone(int note, int duration, int volume) method

- provided by Manager class to play single tones, 11
- using to play a single note, 76

polyphonic ring tones, 73–74

- MIDI specification based on, 96

postEntry(BlogEntry entry) method, code for, 230–231

postentry.jsp file, 232–235

prefetch() method

- in a simple linear transition path for a Player instance, 40
- using instead of realize() method for quicker media playback startup, 34

PREFETCHED state

- function of, 43–44
- of Player instance, 39
- possible transitions, 44

PreviewCanvas abstract class

- as base abstract class for all preview canvases, 221–223
- providing commands to view the Device Blog MIDlet, 197

Program Change messages

- for changing the current instrument/program assigned to a channel, 117–118
- function of, 101

Q

quantization (or resolution), as measurement of sampled audio, 133

R

RateControl

- differentiating between it and TempoControl, 109–112
- in javax.microedition.media.control package, 243–244
- used to control the playback rate of a Player, 13
- using instead of TempoControl, 109

Real-time Transport Protocol (RTP), 181

realize() method

- exceptions thrown by, 42
- function of, 34
- in a simple linear transition path for a Player instance, 39–40

REALIZED state

- of Player instance, 39

- transitions, 41–42

RealTime Server, Web site address for, 182

RecordControl

- for controlling the recording of data from a capture device, 13

- in javax.microedition.media.control package, 244

- methods for setting the location of recorded data, 147

- using to capture media, 146

reference equality check, using to catch custom events in playerUpdate() method, 56

reference implementation (RI), of MMAPI, 3

removePlayerListener(PlayerListener listener) method, using to remove an instance, 47

RESET constant, removing a previously set preset time with, 139

resolution, using to specify duration of each note, 82

resource protocol, for creating a locator for a media file, 21

restartMedia() method, for restarting the Player instance, 26

ring tone, as example of synthetic tones, 73

Ringling Tone Text Transfer Language (RTTTL) format, converting to MMAPI format, 84–86

RingToneConverter class, using to convert RTTTL to MMAPI format, 84–86

rtc.dumpSequence() method, using to print the sequence on the standard out, 89

RTTTL format. *See* Ringling Tone Text Transfer Language (RTTTL) format

Run via OTA option, for testing the MIDlet after it is signed in Sun Java ME emulator, 66–67

Runnable interface, implemented by the NetworkPlayerManager class, 69–70

S

sampled audio

- a brief overview of, 133–134

- code for mixing MIDI and, 143–146

- controlling, 134–141

- mixing with MIDI and tones, 142–146

- storing, 133–134

sampled audio feature set, function of, 15

- sampling rate, as measurement of sampled audio, 133
- Scalable Polyphony MIDI (SP-MIDI), mobile phone ring tones based on, 96
- security architecture, of devices using MMAPi, 16–17
- security conscious methods, table of classes/interfaces and permissions required, 16
- SecurityException
 - reason it is thrown by the realize() method, 42–43
 - thrown by calling prefetch() method, 43
 - thrown if there is not enough permission to start the media, 44
- seek(int frameNumber) method, for seeking video frames, 181
- seek(long where) method, function of, 8
- semitone, defined, 75, 108
- SEMITONE_CONST, formula for, 76
- sequences. *See also* tone sequences
 - creating, 83–84
 - creating, defining, and playing blocks of, 87
 - playing using ToneControl and Player, 88
- server side, code for, 232–235
- setChannelVolume(int channel, int volume) method
 - for changing volume of a channel, 117
 - for setting the volume for a channel, 115
- setMediaTime() method, setting media time to play back from file starting point with, 34
- setPitch(int milliSemitones) method, setting actual pitch value with, 108
- setProgram(int channel, int bank, int program) method
 - for setting the program/instrument to use on a particular channel, 115
 - using to change a program, 125
- setRate() method, using where a relative value in percentages is specified, 109
- setRecordLocation(String locator) method, for directing recorded data to a location, 147
- setRecordStream(OutputStream stream) method, directs data to an OutputStream, 147
- setStopTime() method, exception thrown when calling on the started Player instance, 139
- setTempo(int milliTempo) method
 - using where an absolute value of the desired tempo is required, 109
 - varying the speed of playback of MIDI sequences using, 107–108
- setTimeBase(TimeBase base) method, overriding the default TimeBase instance with, 9
- setVisible(true) method, calling so the video will be displayed in a Canvas, 169
- sharp, defined, 75–76
- short MIDI event message, sending, 115
- shortMidiEvent(int type, int data1, int data2) method
 - for sending a short MIDI event message, 115
 - sending simple MIDI messages with, 116–117
- showCanvas() method, for displaying a preview of the blog entry to the user, 197
- showDisplay() method, of EditableDisplay interface, 195
- skip(int framesToSkip) method, using to skip video frames, 181
- .smf filename extension, for Standard MIDI file format, 101
- Smith, David, original MIDI standard proposed by, 95
- snapshots, taking, 178–179
- soundbanks. *See also* bank (or soundbanks)
 - getting a list of all installed on your device, 114
 - working with, 120–125
- SourceStream
 - for abstracting a single stream of media data, 8–9
 - in javax.microedition.media.protocol package, 247
 - table of constants to test it for seekability, 8
- Spectator class, function of, 252
- SP-MIDI. *See* Scalable Polyphony MIDI (SP-MIDI)
- Standard MIDI file (SMF) format, common format for storing MIDI messages, 101
- start() method
 - effect of calling on an UNREALIZED instance, 45
 - in a simple linear transition path for a Player instance, 40
 - using to restart paused media file, 21
- started event, representing when an instance has entered the STARTED state, 48
- STARTED state
 - function of, 44–46
 - of Player instance, 39
 - transitions, 46
- Status byte, differentiating System from Channel messages by, 98

- stop() method, effect of, 45
- StopTimeControl
 - illustration showing stopped after a preset time, 138
 - in javax.microedition.media.control package, 244
 - setting preset stop times for media data with, 134–139
 - that allows setting of a preset time, 13
- StopTimeControlMIDlet, code that allows you to play with the StopTimeControl, 135–138
- streaming media, over the network, 181–183
- streaming servers, Web site addresses for, 182
- Sun Java ME DefaultColorPhone emulator, example of output from playing all tones in, 81
- Sun Java ME emulator, signing the NetworkTest MIDlet with trustedkey key in, 66
- Sun Wireless Toolkit, integrated in the Netbeans IDE, 3
- Sun's DefaultColorPhone emulator
 - blog entry choices presented by, 190
 - example of MMAPi capabilities in, 130
 - lack of video recording support in, 188
 - running ControllableMIDlet to control pitch, tempo, and volume, 104
 - simulated video capture on, 178
 - specification of the encoding formats for property audio.encodings, 132
 - starting the Device Blog MIDlet on, 186
 - video capture formats supported by, 132
- synthetic tones, understanding, 73–78
- System Common messages, in MIDI, 99
- System Exclusive messages, in MIDI, 99
- System messages, 97
 - Common, Real-Time, or Exclusive System message types, 97
 - how they are differentiated from Channel messages, 98
 - table of formats for different types of in MIDI, 99
- System Real Time messages, in MIDI, 99
- System.getProperty(String key) method, for querying the capabilities of your device, 127–133
 - how it works, 107–108
 - in javax.microedition.media.control package, 244
 - using for simple changes, 102
- TextBlogEntry class
 - in Device Blog MIDlet application, 193
 - representing all entries that have a textual component, 199–200
- TextEditForm class, code for creating, 213–214
- ThreadedMIDlet, code creating a new thread to play back a media file, 67–69
- ThreadTest MIDlet
 - example of system and new thread activity and actions, 62
 - screen and console output for on a Motorola emulator, 62
- time signature, understanding, 82
- TimeBase interface
 - in javax.microedition.media package, 242
 - resetting a Player instance's, 9
- timed capture and playback, of audio, 147–151
- Tone Sequence feature set, function of, 15
- tone sequences. *See also* sequences
 - creating, defining, and playing blocks of, 87
 - creating to play using the ToneControl interface, 83–84
 - defining, 82–87
 - distributing, 89–92
 - playing those stored in files, 91–92
 - playing using ToneControl and Player, 88
 - using with ToneControl interface, 81–92
- ToneControl interface
 - creating and playing tones with, 73–93
 - in javax.microedition.media.control package, 245
 - table of constants, 86
 - that allows playing of montonic tone sequences, 13
 - using to create a tone sequence to play, 83–84
 - using tone sequences with, 81–92
- True Audio (TTA) format, 134
- trust, establishment of, 65
- trustedkey key, used to sign the NetworkTest MIDlet, 66
- TTA format. *See* True Audio (TTA) format

T

- Technology compatibility kit (TCK), 5
- TempoControl
 - for changing the tempo of playback for an audio player, 13
 - differentiating between it and RateControl, 109–112

U

- Unified Modeling Language (UML)
 - diagram for common functionalities of a Form, 195
 - diagram for common functionalities of a List, 194
 - model for the Device Blog MIDlet application, 193

- UNREALIZED Player instance, making usable, 40–41
- UNREALIZED state
 - and its transitions, 41
 - of Player instance, 39
- URI syntax, media locators, 249
- URLEncoder class
 - code containing a single method to encode URLs, 209–211
 - function of, 198
- USE_DIRECT_VIDEO argument, using to embed video in a Canvas, 164
- USE_GUI_PRIMITIVE argument, using to embed video in a Form, 164
- User class
 - in Device Blog MIDlet application, 194
 - represents the user of the Device Blog MIDlet, 202–204
- utility classes, creating, 205–211

V

- video
 - positioning and controlling volume, 169–173
 - working with, 163–181
- video and audio, working with, 127–184
- video and images, capturing, 173–181
- video capture
 - formats supported by Sun's DefaultColorPhone emulator, 132
 - on the BenQ M75 device, 192
- video clips, capturing, 179–181
- video feature set, function of, 16

- video frames
 - seeking with FramePositioningControl, 181
 - skipping using the skip(int framesToSkip) method, 181
- VideoBlogEntry
 - in Device Blog MIDlet application, 193
 - that extends MediaBlogEntry, 202
- VideoControl
 - extends GUIControl and controls the display of video, 13
 - in javax.microedition.media.control package, 245
 - as primary control for displaying video, 163–181
 - to take a snapshot of a video, 173–174
- VideoPreviewCanvas, previews, 225–227
- viewfinder, looking through, 174–178
- views
 - creating, 209–211
 - in Device Blog MIDlet application, 194–197
- Voice Channel messages
 - function of, 98
 - more information about, 99–101
 - table of all possibilities along with a description, 100–101
- volumeChanged event, 48
- VolumeControl
 - in javax.microedition.media.control package, 246
 - for providing volume control to audio files, 13–14
 - using for simple volume changes, 102

W

- WAV file, mixing with a MIDI file, 142