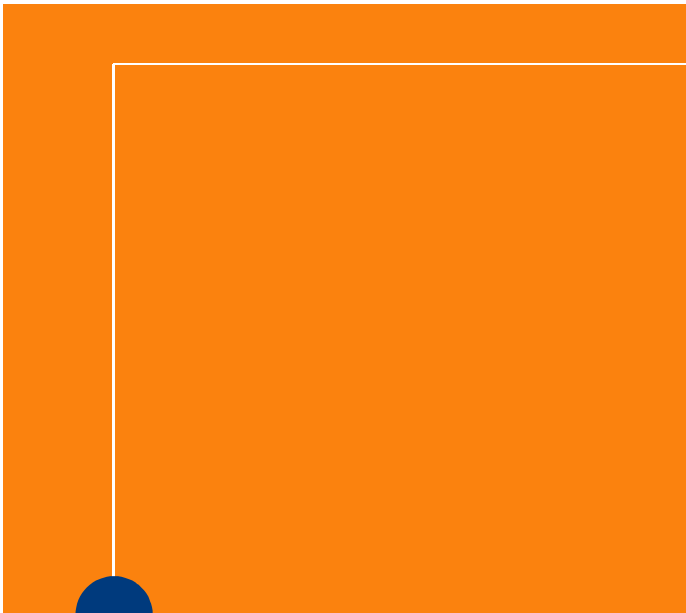


***Using Sound on the  
Motorola V300, V500,  
and V600***

*November 26, 2003*



*Tip of the Month*

---

## *Using Sound on the Motorola V300, V500, and V600*

By

Motocoders Technical Support Group

---

**H**ow are sounds created on the V300, V500, and V600? This has been a hot topic of discussion among developers recently, especially as it pertains to creating game applications. The most direct answer to this question is simple; sounds are created using the MIDP 2.0 profile which consists of a group of APIs useful for creating applications. This group of APIs relate to user interfaces, persistent storage, and most importantly media.

For developers concerned with using/creating sound on the V300, V500, and V600, the most important API is the one relating to media. This media API is a directly compatible subset of the Mobile Media API (JSR-135) specification and is known as the MIDP 2.0 Media API. This API contains two classes that are used to play sounds - the Manager class and the Player class.

### ***Manager Class***

The Manager class, as the name implies, is the organizer of the audio resources. This class does not play the audio data, but simply delegates the task to its Players. These Players are created using the Manager's createPlayer method and are responsible for the processing and playing of sounds.

### ***Player Class***

When a Player is alive and carrying out its tasks, it goes through a number of states known as unrealized, realized, prefetched, started, and closed. A player starts its life in the unrealized state, implying the Player has merely been created and has not done anything useful such as acquire audio data or attempt to acquire resources such as the audio hardware. The second state is known as the realized state and occurs after a Player's realize () method is called. The realized state implies that the Player has located the media data it wants to play, for example communicating with a file system or a server. The next state is the prefetched state, transitioned through the prefetch () method, and indicates the Player has performed all the necessary work in order to start playing the sound. This kind of work could include filling up the buffers and acquiring control of the audio hardware. When the Player actually started rendering audio data (via the start () method), it is in the started state. The final state is known as the closed state and occurs when the Player has freed all its resources, closed all network connections, and cannot be used again. Please note that it is not necessary to explicitly execute the methods corresponding to each of the states. If a step is skipped, the intervening states are implied. For example, it is possible to call start () on a freshly created Player where the missing methods would be implicitly called.

---

### ***Additional Rules***

In order to successfully play sounds on the V300, V500, and V600, the following rules must also be followed:

#### ***Playing sounds simultaneously***

Unfortunately, it is not possible to play more than 2 sounds at once and only a particular combination of media is supported. The only simultaneous combination supported is midi sounds and wav sounds. The sounds must be played in a specific order – midi then wav – to work successfully.

### ***Prefetching***

It is not possible to have more than 1 Player in the prefetch state unless they are instances of midi, imelody, mix, or basetrack sessions, in which case 2 Players can be in the prefetched state. Clearly, the other exception involves midi and wav.

### ***Playing sounds in sequence***

When an application is going to play a new sound, the previous Player must be stopped and deallocated (i.e stop () and deallocate () respectively) before the new media is played.

---

## ***Conclusion***

Creating sounds for the Motorola V300, V500, and V600 is performed through Players which are created by Manager classes that are defined in the MIDP 2.0 Media API. As long as the rules for playing sounds described in this whitepaper are followed, playing sounds should not be difficult.

---

## ***References***

Wireless Java: Developing with J2ME, Second Edition; Knudsen, Jonathan; Feb 2003 ISBN: 1-59059-077-5