| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Sequencer.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/sound/midi/Sequence.html)   [**NEXT CLASS**](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sound/midi/Sequencer.html)    [**NO FRAMES**](http://docs.google.com/Sequencer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#tyjcwt) | CONSTR | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | CONSTR | [METHOD](#17dp8vu) |

## **javax.sound.midi**

Interface Sequencer

**All Superinterfaces:** [MidiDevice](http://docs.google.com/javax/sound/midi/MidiDevice.html)

public interface **Sequencer**extends [MidiDevice](http://docs.google.com/javax/sound/midi/MidiDevice.html)

A hardware or software device that plays back a MIDI [sequence](http://docs.google.com/javax/sound/midi/Sequence.html) is known as a *sequencer*. A MIDI sequence contains lists of time-stamped MIDI data, such as might be read from a standard MIDI file. Most sequencers also provide functions for creating and editing sequences.

The Sequencer interface includes methods for the following basic MIDI sequencer operations:

* obtaining a sequence from MIDI file data
* starting and stopping playback
* moving to an arbitrary position in the sequence
* changing the tempo (speed) of playback
* synchronizing playback to an internal clock or to received MIDI messages
* controlling the timing of another device

In addition, the following operations are supported, either directly, or indirectly through objects that the Sequencer has access to:

* editing the data by adding or deleting individual MIDI events or entire tracks
* muting or soloing individual tracks in the sequence
* notifying listener objects about any meta-events or control-change events encountered while playing back the sequence.

**See Also:**[Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html), [addMetaEventListener(javax.sound.midi.MetaEventListener)](http://docs.google.com/javax/sound/midi/Sequencer.html#addMetaEventListener(javax.sound.midi.MetaEventListener)), [ControllerEventListener](http://docs.google.com/javax/sound/midi/ControllerEventListener.html), [Receiver](http://docs.google.com/javax/sound/midi/Receiver.html), [Transmitter](http://docs.google.com/javax/sound/midi/Transmitter.html), [MidiDevice](http://docs.google.com/javax/sound/midi/MidiDevice.html)

| **Nested Class Summary** | |
| --- | --- |
| static class | [**Sequencer.SyncMode**](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html)            A SyncMode object represents one of the ways in which a MIDI sequencer's notion of time can be synchronized with a master or slave device. |

| **Nested classes/interfaces inherited from interface javax.sound.midi.**[**MidiDevice**](http://docs.google.com/javax/sound/midi/MidiDevice.html) |
| --- |
| [MidiDevice.Info](http://docs.google.com/javax/sound/midi/MidiDevice.Info.html) |

| **Field Summary** | |
| --- | --- |
| static int | [**LOOP\_CONTINUOUSLY**](http://docs.google.com/javax/sound/midi/Sequencer.html#LOOP_CONTINUOUSLY)            A value indicating that looping should continue indefinitely rather than complete after a specific number of loops. |

| **Method Summary** | |
| --- | --- |
| int[] | [**addControllerEventListener**](http://docs.google.com/javax/sound/midi/Sequencer.html#addControllerEventListener(javax.sound.midi.ControllerEventListener,%20int%5B%5D))([ControllerEventListener](http://docs.google.com/javax/sound/midi/ControllerEventListener.html) listener, int[] controllers)            Registers a controller event listener to receive notification whenever the sequencer processes a control-change event of the requested type or types. |
| boolean | [**addMetaEventListener**](http://docs.google.com/javax/sound/midi/Sequencer.html#addMetaEventListener(javax.sound.midi.MetaEventListener))([MetaEventListener](http://docs.google.com/javax/sound/midi/MetaEventListener.html) listener)            Registers a meta-event listener to receive notification whenever a meta-event is encountered in the sequence and processed by the sequencer. |
| int | [**getLoopCount**](http://docs.google.com/javax/sound/midi/Sequencer.html#getLoopCount())()            Obtains the number of repetitions for playback. |
| long | [**getLoopEndPoint**](http://docs.google.com/javax/sound/midi/Sequencer.html#getLoopEndPoint())()            Obtains the end position of the loop, in MIDI ticks. |
| long | [**getLoopStartPoint**](http://docs.google.com/javax/sound/midi/Sequencer.html#getLoopStartPoint())()            Obtains the start position of the loop, in MIDI ticks. |
| [Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html) | [**getMasterSyncMode**](http://docs.google.com/javax/sound/midi/Sequencer.html#getMasterSyncMode())()            Obtains the current master synchronization mode for this sequencer. |
| [Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html)[] | [**getMasterSyncModes**](http://docs.google.com/javax/sound/midi/Sequencer.html#getMasterSyncModes())()            Obtains the set of master synchronization modes supported by this sequencer. |
| long | [**getMicrosecondLength**](http://docs.google.com/javax/sound/midi/Sequencer.html#getMicrosecondLength())()            Obtains the length of the current sequence, expressed in microseconds, or 0 if no sequence is set. |
| long | [**getMicrosecondPosition**](http://docs.google.com/javax/sound/midi/Sequencer.html#getMicrosecondPosition())()            Obtains the current position in the sequence, expressed in microseconds. |
| [Sequence](http://docs.google.com/javax/sound/midi/Sequence.html) | [**getSequence**](http://docs.google.com/javax/sound/midi/Sequencer.html#getSequence())()            Obtains the sequence on which the Sequencer is currently operating. |
| [Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html) | [**getSlaveSyncMode**](http://docs.google.com/javax/sound/midi/Sequencer.html#getSlaveSyncMode())()            Obtains the current slave synchronization mode for this sequencer. |
| [Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html)[] | [**getSlaveSyncModes**](http://docs.google.com/javax/sound/midi/Sequencer.html#getSlaveSyncModes())()            Obtains the set of slave synchronization modes supported by the sequencer. |
| float | [**getTempoFactor**](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoFactor())()            Returns the current tempo factor for the sequencer. |
| float | [**getTempoInBPM**](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoInBPM())()            Obtains the current tempo, expressed in beats per minute. |
| float | [**getTempoInMPQ**](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoInMPQ())()            Obtains the current tempo, expressed in microseconds per quarter note. |
| long | [**getTickLength**](http://docs.google.com/javax/sound/midi/Sequencer.html#getTickLength())()            Obtains the length of the current sequence, expressed in MIDI ticks, or 0 if no sequence is set. |
| long | [**getTickPosition**](http://docs.google.com/javax/sound/midi/Sequencer.html#getTickPosition())()            Obtains the current position in the sequence, expressed in MIDI ticks. |
| boolean | [**getTrackMute**](http://docs.google.com/javax/sound/midi/Sequencer.html#getTrackMute(int))(int track)            Obtains the current mute state for a track. |
| boolean | [**getTrackSolo**](http://docs.google.com/javax/sound/midi/Sequencer.html#getTrackSolo(int))(int track)            Obtains the current solo state for a track. |
| boolean | [**isRecording**](http://docs.google.com/javax/sound/midi/Sequencer.html#isRecording())()            Indicates whether the Sequencer is currently recording. |
| boolean | [**isRunning**](http://docs.google.com/javax/sound/midi/Sequencer.html#isRunning())()            Indicates whether the Sequencer is currently running. |
| void | [**recordDisable**](http://docs.google.com/javax/sound/midi/Sequencer.html#recordDisable(javax.sound.midi.Track))([Track](http://docs.google.com/javax/sound/midi/Track.html) track)            Disables recording to the specified track. |
| void | [**recordEnable**](http://docs.google.com/javax/sound/midi/Sequencer.html#recordEnable(javax.sound.midi.Track,%20int))([Track](http://docs.google.com/javax/sound/midi/Track.html) track, int channel)            Prepares the specified track for recording events received on a particular channel. |
| int[] | [**removeControllerEventListener**](http://docs.google.com/javax/sound/midi/Sequencer.html#removeControllerEventListener(javax.sound.midi.ControllerEventListener,%20int%5B%5D))([ControllerEventListener](http://docs.google.com/javax/sound/midi/ControllerEventListener.html) listener, int[] controllers)            Removes a controller event listener's interest in one or more types of controller event. |
| void | [**removeMetaEventListener**](http://docs.google.com/javax/sound/midi/Sequencer.html#removeMetaEventListener(javax.sound.midi.MetaEventListener))([MetaEventListener](http://docs.google.com/javax/sound/midi/MetaEventListener.html) listener)            Removes the specified meta-event listener from this sequencer's list of registered listeners, if in fact the listener is registered. |
| void | [**setLoopCount**](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopCount(int))(int count)            Sets the number of repetitions of the loop for playback. |
| void | [**setLoopEndPoint**](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopEndPoint(long))(long tick)            Sets the last MIDI tick that will be played in the loop. |
| void | [**setLoopStartPoint**](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopStartPoint(long))(long tick)            Sets the first MIDI tick that will be played in the loop. |
| void | [**setMasterSyncMode**](http://docs.google.com/javax/sound/midi/Sequencer.html#setMasterSyncMode(javax.sound.midi.Sequencer.SyncMode))([Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html) sync)            Sets the source of timing information used by this sequencer. |
| void | [**setMicrosecondPosition**](http://docs.google.com/javax/sound/midi/Sequencer.html#setMicrosecondPosition(long))(long microseconds)            Sets the current position in the sequence, expressed in microseconds |
| void | [**setSequence**](http://docs.google.com/javax/sound/midi/Sequencer.html#setSequence(java.io.InputStream))([InputStream](http://docs.google.com/java/io/InputStream.html) stream)            Sets the current sequence on which the sequencer operates. |
| void | [**setSequence**](http://docs.google.com/javax/sound/midi/Sequencer.html#setSequence(javax.sound.midi.Sequence))([Sequence](http://docs.google.com/javax/sound/midi/Sequence.html) sequence)            Sets the current sequence on which the sequencer operates. |
| void | [**setSlaveSyncMode**](http://docs.google.com/javax/sound/midi/Sequencer.html#setSlaveSyncMode(javax.sound.midi.Sequencer.SyncMode))([Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html) sync)            Sets the slave synchronization mode for the sequencer. |
| void | [**setTempoFactor**](http://docs.google.com/javax/sound/midi/Sequencer.html#setTempoFactor(float))(float factor)            Scales the sequencer's actual playback tempo by the factor provided. |
| void | [**setTempoInBPM**](http://docs.google.com/javax/sound/midi/Sequencer.html#setTempoInBPM(float))(float bpm)            Sets the tempo in beats per minute. |
| void | [**setTempoInMPQ**](http://docs.google.com/javax/sound/midi/Sequencer.html#setTempoInMPQ(float))(float mpq)            Sets the tempo in microseconds per quarter note. |
| void | [**setTickPosition**](http://docs.google.com/javax/sound/midi/Sequencer.html#setTickPosition(long))(long tick)            Sets the current sequencer position in MIDI ticks |
| void | [**setTrackMute**](http://docs.google.com/javax/sound/midi/Sequencer.html#setTrackMute(int,%20boolean))(int track, boolean mute)            Sets the mute state for a track. |
| void | [**setTrackSolo**](http://docs.google.com/javax/sound/midi/Sequencer.html#setTrackSolo(int,%20boolean))(int track, boolean solo)            Sets the solo state for a track. |
| void | [**start**](http://docs.google.com/javax/sound/midi/Sequencer.html#start())()            Starts playback of the MIDI data in the currently loaded sequence. |
| void | [**startRecording**](http://docs.google.com/javax/sound/midi/Sequencer.html#startRecording())()            Starts recording and playback of MIDI data. |
| void | [**stop**](http://docs.google.com/javax/sound/midi/Sequencer.html#stop())()            Stops recording, if active, and playback of the currently loaded sequence, if any. |
| void | [**stopRecording**](http://docs.google.com/javax/sound/midi/Sequencer.html#stopRecording())()            Stops recording, if active. |

| **Methods inherited from interface javax.sound.midi.**[**MidiDevice**](http://docs.google.com/javax/sound/midi/MidiDevice.html) |
| --- |
| [close](http://docs.google.com/javax/sound/midi/MidiDevice.html#close()), [getDeviceInfo](http://docs.google.com/javax/sound/midi/MidiDevice.html#getDeviceInfo()), [getMaxReceivers](http://docs.google.com/javax/sound/midi/MidiDevice.html#getMaxReceivers()), [getMaxTransmitters](http://docs.google.com/javax/sound/midi/MidiDevice.html#getMaxTransmitters()), [getReceiver](http://docs.google.com/javax/sound/midi/MidiDevice.html#getReceiver()), [getReceivers](http://docs.google.com/javax/sound/midi/MidiDevice.html#getReceivers()), [getTransmitter](http://docs.google.com/javax/sound/midi/MidiDevice.html#getTransmitter()), [getTransmitters](http://docs.google.com/javax/sound/midi/MidiDevice.html#getTransmitters()), [isOpen](http://docs.google.com/javax/sound/midi/MidiDevice.html#isOpen()), [open](http://docs.google.com/javax/sound/midi/MidiDevice.html#open()) |

| **Field Detail** |
| --- |

### LOOP\_CONTINUOUSLY

static final int **LOOP\_CONTINUOUSLY**

A value indicating that looping should continue indefinitely rather than complete after a specific number of loops.

**Since:** 1.5 **See Also:**[setLoopCount(int)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopCount(int)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.sound.midi.Sequencer.LOOP_CONTINUOUSLY)

| **Method Detail** |
| --- |

### setSequence

void **setSequence**([Sequence](http://docs.google.com/javax/sound/midi/Sequence.html) sequence)  
 throws [InvalidMidiDataException](http://docs.google.com/javax/sound/midi/InvalidMidiDataException.html)

Sets the current sequence on which the sequencer operates.

This method can be called even if the Sequencer is closed.

**Parameters:**sequence - the sequence to be loaded. **Throws:** [InvalidMidiDataException](http://docs.google.com/javax/sound/midi/InvalidMidiDataException.html) - if the sequence contains invalid MIDI data, or is not supported.

### setSequence

void **setSequence**([InputStream](http://docs.google.com/java/io/InputStream.html) stream)  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [InvalidMidiDataException](http://docs.google.com/javax/sound/midi/InvalidMidiDataException.html)

Sets the current sequence on which the sequencer operates. The stream must point to MIDI file data.

This method can be called even if the Sequencer is closed.

**Parameters:**stream - stream containing MIDI file data. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O exception occurs during reading of the stream. [InvalidMidiDataException](http://docs.google.com/javax/sound/midi/InvalidMidiDataException.html) - if invalid data is encountered in the stream, or the stream is not supported.

### getSequence

[Sequence](http://docs.google.com/javax/sound/midi/Sequence.html) **getSequence**()

Obtains the sequence on which the Sequencer is currently operating.

This method can be called even if the Sequencer is closed.

**Returns:**the current sequence, or null if no sequence is currently set.

### start

void **start**()

Starts playback of the MIDI data in the currently loaded sequence. Playback will begin from the current position. If the playback position reaches the loop end point, and the loop count is greater than 0, playback will resume at the loop start point for the number of repetitions set with setLoopCount. After that, or if the loop count is 0, playback will continue to play to the end of the sequence.

The implementation ensures that the synthesizer is brought to a consistent state when jumping to the loop start point by sending appropriate controllers, pitch bend, and program change events.

**Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the Sequencer is closed.**See Also:**[setLoopStartPoint(long)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopStartPoint(long)), [setLoopEndPoint(long)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopEndPoint(long)), [setLoopCount(int)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopCount(int)), [stop()](http://docs.google.com/javax/sound/midi/Sequencer.html#stop())

### stop

void **stop**()

Stops recording, if active, and playback of the currently loaded sequence, if any.

**Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the Sequencer is closed.**See Also:**[start()](http://docs.google.com/javax/sound/midi/Sequencer.html#start()), [isRunning()](http://docs.google.com/javax/sound/midi/Sequencer.html#isRunning())

### isRunning

boolean **isRunning**()

Indicates whether the Sequencer is currently running. The default is false. The Sequencer starts running when either [start()](http://docs.google.com/javax/sound/midi/Sequencer.html#start()) or [startRecording()](http://docs.google.com/javax/sound/midi/Sequencer.html#startRecording()) is called. isRunning then returns true until playback of the sequence completes or [stop()](http://docs.google.com/javax/sound/midi/Sequencer.html#stop()) is called.

**Returns:**true if the Sequencer is running, otherwise false

### startRecording

void **startRecording**()

Starts recording and playback of MIDI data. Data is recorded to all enabled tracks, on the channel(s) for which they were enabled. Recording begins at the current position of the sequencer. Any events already in the track are overwritten for the duration of the recording session. Events from the currently loaded sequence, if any, are delivered to the sequencer's transmitter(s) along with messages received during recording.

Note that tracks are not by default enabled for recording. In order to record MIDI data, at least one track must be specifically enabled for recording.

**Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the Sequencer is closed.**See Also:**[startRecording()](http://docs.google.com/javax/sound/midi/Sequencer.html#startRecording()), [recordEnable(javax.sound.midi.Track, int)](http://docs.google.com/javax/sound/midi/Sequencer.html#recordEnable(javax.sound.midi.Track,%20int)), [recordDisable(javax.sound.midi.Track)](http://docs.google.com/javax/sound/midi/Sequencer.html#recordDisable(javax.sound.midi.Track))

### stopRecording

void **stopRecording**()

Stops recording, if active. Playback of the current sequence continues.

**Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the Sequencer is closed.**See Also:**[startRecording()](http://docs.google.com/javax/sound/midi/Sequencer.html#startRecording()), [isRecording()](http://docs.google.com/javax/sound/midi/Sequencer.html#isRecording())

### isRecording

boolean **isRecording**()

Indicates whether the Sequencer is currently recording. The default is false. The Sequencer begins recording when [startRecording()](http://docs.google.com/javax/sound/midi/Sequencer.html#startRecording()) is called, and then returns true until [stop()](http://docs.google.com/javax/sound/midi/Sequencer.html#stop()) or [stopRecording()](http://docs.google.com/javax/sound/midi/Sequencer.html#stopRecording()) is called.

**Returns:**true if the Sequencer is recording, otherwise false

### recordEnable

void **recordEnable**([Track](http://docs.google.com/javax/sound/midi/Track.html) track,  
 int channel)

Prepares the specified track for recording events received on a particular channel. Once enabled, a track will receive events when recording is active.

**Parameters:**track - the track to which events will be recordedchannel - the channel on which events will be received. If -1 is specified for the channel value, the track will receive data from all channels. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - thrown if the track is not part of the current sequence.

### recordDisable

void **recordDisable**([Track](http://docs.google.com/javax/sound/midi/Track.html) track)

Disables recording to the specified track. Events will no longer be recorded into this track.

**Parameters:**track - the track to disable for recording, or null to disable recording for all tracks.

### getTempoInBPM

float **getTempoInBPM**()

Obtains the current tempo, expressed in beats per minute. The actual tempo of playback is the product of the returned value and the tempo factor.

**Returns:**the current tempo in beats per minute**See Also:**[getTempoFactor()](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoFactor()), [setTempoInBPM(float)](http://docs.google.com/javax/sound/midi/Sequencer.html#setTempoInBPM(float)), [getTempoInMPQ()](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoInMPQ())

### setTempoInBPM

void **setTempoInBPM**(float bpm)

Sets the tempo in beats per minute. The actual tempo of playback is the product of the specified value and the tempo factor.

**Parameters:**bpm - desired new tempo in beats per minute**See Also:**[getTempoFactor()](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoFactor()), [setTempoInMPQ(float)](http://docs.google.com/javax/sound/midi/Sequencer.html#setTempoInMPQ(float)), [getTempoInBPM()](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoInBPM())

### getTempoInMPQ

float **getTempoInMPQ**()

Obtains the current tempo, expressed in microseconds per quarter note. The actual tempo of playback is the product of the returned value and the tempo factor.

**Returns:**the current tempo in microseconds per quarter note**See Also:**[getTempoFactor()](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoFactor()), [setTempoInMPQ(float)](http://docs.google.com/javax/sound/midi/Sequencer.html#setTempoInMPQ(float)), [getTempoInBPM()](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoInBPM())

### setTempoInMPQ

void **setTempoInMPQ**(float mpq)

Sets the tempo in microseconds per quarter note. The actual tempo of playback is the product of the specified value and the tempo factor.

**Parameters:**mpq - desired new tempo in microseconds per quarter note.**See Also:**[getTempoFactor()](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoFactor()), [setTempoInBPM(float)](http://docs.google.com/javax/sound/midi/Sequencer.html#setTempoInBPM(float)), [getTempoInMPQ()](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoInMPQ())

### setTempoFactor

void **setTempoFactor**(float factor)

Scales the sequencer's actual playback tempo by the factor provided. The default is 1.0. A value of 1.0 represents the natural rate (the tempo specified in the sequence), 2.0 means twice as fast, etc. The tempo factor does not affect the values returned by [getTempoInMPQ()](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoInMPQ()) and [getTempoInBPM()](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoInBPM()). Those values indicate the tempo prior to scaling.

Note that the tempo factor cannot be adjusted when external synchronization is used. In that situation, setTempoFactor always sets the tempo factor to 1.0.

**Parameters:**factor - the requested tempo scalar**See Also:**[getTempoFactor()](http://docs.google.com/javax/sound/midi/Sequencer.html#getTempoFactor())

### getTempoFactor

float **getTempoFactor**()

Returns the current tempo factor for the sequencer. The default is 1.0.

**Returns:**tempo factor.**See Also:**[setTempoFactor(float)](http://docs.google.com/javax/sound/midi/Sequencer.html#setTempoFactor(float))

### getTickLength

long **getTickLength**()

Obtains the length of the current sequence, expressed in MIDI ticks, or 0 if no sequence is set.

**Returns:**length of the sequence in ticks

### getTickPosition

long **getTickPosition**()

Obtains the current position in the sequence, expressed in MIDI ticks. (The duration of a tick in seconds is determined both by the tempo and by the timing resolution stored in the [Sequence](http://docs.google.com/javax/sound/midi/Sequence.html).)

**Returns:**current tick**See Also:**[setTickPosition(long)](http://docs.google.com/javax/sound/midi/Sequencer.html#setTickPosition(long))

### setTickPosition

void **setTickPosition**(long tick)

Sets the current sequencer position in MIDI ticks

**Parameters:**tick - the desired tick position**See Also:**[getTickPosition()](http://docs.google.com/javax/sound/midi/Sequencer.html#getTickPosition())

### getMicrosecondLength

long **getMicrosecondLength**()

Obtains the length of the current sequence, expressed in microseconds, or 0 if no sequence is set.

**Returns:**length of the sequence in microseconds.

### getMicrosecondPosition

long **getMicrosecondPosition**()

Obtains the current position in the sequence, expressed in microseconds.

**Specified by:**[getMicrosecondPosition](http://docs.google.com/javax/sound/midi/MidiDevice.html#getMicrosecondPosition()) in interface [MidiDevice](http://docs.google.com/javax/sound/midi/MidiDevice.html) **Returns:**the current position in microseconds**See Also:**[setMicrosecondPosition(long)](http://docs.google.com/javax/sound/midi/Sequencer.html#setMicrosecondPosition(long))

### setMicrosecondPosition

void **setMicrosecondPosition**(long microseconds)

Sets the current position in the sequence, expressed in microseconds

**Parameters:**microseconds - desired position in microseconds**See Also:**[getMicrosecondPosition()](http://docs.google.com/javax/sound/midi/Sequencer.html#getMicrosecondPosition())

### setMasterSyncMode

void **setMasterSyncMode**([Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html) sync)

Sets the source of timing information used by this sequencer. The sequencer synchronizes to the master, which is the internal clock, MIDI clock, or MIDI time code, depending on the value of sync. The sync argument must be one of the supported modes, as returned by [getMasterSyncModes()](http://docs.google.com/javax/sound/midi/Sequencer.html#getMasterSyncModes()).

**Parameters:**sync - the desired master synchronization mode**See Also:**[Sequencer.SyncMode.INTERNAL\_CLOCK](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html#INTERNAL_CLOCK), [Sequencer.SyncMode.MIDI\_SYNC](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html#MIDI_SYNC), [Sequencer.SyncMode.MIDI\_TIME\_CODE](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html#MIDI_TIME_CODE), [getMasterSyncMode()](http://docs.google.com/javax/sound/midi/Sequencer.html#getMasterSyncMode())

### getMasterSyncMode

[Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html) **getMasterSyncMode**()

Obtains the current master synchronization mode for this sequencer.

**Returns:**the current master synchronization mode**See Also:**[setMasterSyncMode(Sequencer.SyncMode)](http://docs.google.com/javax/sound/midi/Sequencer.html#setMasterSyncMode(javax.sound.midi.Sequencer.SyncMode)), [getMasterSyncModes()](http://docs.google.com/javax/sound/midi/Sequencer.html#getMasterSyncModes())

### getMasterSyncModes

[Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html)[] **getMasterSyncModes**()

Obtains the set of master synchronization modes supported by this sequencer.

**Returns:**the available master synchronization modes**See Also:**[Sequencer.SyncMode.INTERNAL\_CLOCK](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html#INTERNAL_CLOCK), [Sequencer.SyncMode.MIDI\_SYNC](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html#MIDI_SYNC), [Sequencer.SyncMode.MIDI\_TIME\_CODE](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html#MIDI_TIME_CODE), [getMasterSyncMode()](http://docs.google.com/javax/sound/midi/Sequencer.html#getMasterSyncMode()), [setMasterSyncMode(Sequencer.SyncMode)](http://docs.google.com/javax/sound/midi/Sequencer.html#setMasterSyncMode(javax.sound.midi.Sequencer.SyncMode))

### setSlaveSyncMode

void **setSlaveSyncMode**([Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html) sync)

Sets the slave synchronization mode for the sequencer. This indicates the type of timing information sent by the sequencer to its receiver. The sync argument must be one of the supported modes, as returned by [getSlaveSyncModes()](http://docs.google.com/javax/sound/midi/Sequencer.html#getSlaveSyncModes()).

**Parameters:**sync - the desired slave synchronization mode**See Also:**[Sequencer.SyncMode.MIDI\_SYNC](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html#MIDI_SYNC), [Sequencer.SyncMode.MIDI\_TIME\_CODE](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html#MIDI_TIME_CODE), [Sequencer.SyncMode.NO\_SYNC](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html#NO_SYNC), [getSlaveSyncModes()](http://docs.google.com/javax/sound/midi/Sequencer.html#getSlaveSyncModes())

### getSlaveSyncMode

[Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html) **getSlaveSyncMode**()

Obtains the current slave synchronization mode for this sequencer.

**Returns:**the current slave synchronization mode**See Also:**[setSlaveSyncMode(Sequencer.SyncMode)](http://docs.google.com/javax/sound/midi/Sequencer.html#setSlaveSyncMode(javax.sound.midi.Sequencer.SyncMode)), [getSlaveSyncModes()](http://docs.google.com/javax/sound/midi/Sequencer.html#getSlaveSyncModes())

### getSlaveSyncModes

[Sequencer.SyncMode](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html)[] **getSlaveSyncModes**()

Obtains the set of slave synchronization modes supported by the sequencer.

**Returns:**the available slave synchronization modes**See Also:**[Sequencer.SyncMode.MIDI\_SYNC](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html#MIDI_SYNC), [Sequencer.SyncMode.MIDI\_TIME\_CODE](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html#MIDI_TIME_CODE), [Sequencer.SyncMode.NO\_SYNC](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html#NO_SYNC)

### setTrackMute

void **setTrackMute**(int track,  
 boolean mute)

Sets the mute state for a track. This method may fail for a number of reasons. For example, the track number specified may not be valid for the current sequence, or the sequencer may not support this functionality. An application which needs to verify whether this operation succeeded should follow this call with a call to [getTrackMute(int)](http://docs.google.com/javax/sound/midi/Sequencer.html#getTrackMute(int)).

**Parameters:**track - the track number. Tracks in the current sequence are numbered from 0 to the number of tracks in the sequence minus 1.mute - the new mute state for the track. true implies the track should be muted, false implies the track should be unmuted.**See Also:**[getSequence()](http://docs.google.com/javax/sound/midi/Sequencer.html#getSequence())

### getTrackMute

boolean **getTrackMute**(int track)

Obtains the current mute state for a track. The default mute state for all tracks which have not been muted is false. In any case where the specified track has not been muted, this method should return false. This applies if the sequencer does not support muting of tracks, and if the specified track index is not valid.

**Parameters:**track - the track number. Tracks in the current sequence are numbered from 0 to the number of tracks in the sequence minus 1. **Returns:**true if muted, false if not.

### setTrackSolo

void **setTrackSolo**(int track,  
 boolean solo)

Sets the solo state for a track. If solo is true only this track and other solo'd tracks will sound. If solo is false then only other solo'd tracks will sound, unless no tracks are solo'd in which case all un-muted tracks will sound.

This method may fail for a number of reasons. For example, the track number specified may not be valid for the current sequence, or the sequencer may not support this functionality. An application which needs to verify whether this operation succeeded should follow this call with a call to [getTrackSolo(int)](http://docs.google.com/javax/sound/midi/Sequencer.html#getTrackSolo(int)).

**Parameters:**track - the track number. Tracks in the current sequence are numbered from 0 to the number of tracks in the sequence minus 1.solo - the new solo state for the track. true implies the track should be solo'd, false implies the track should not be solo'd.**See Also:**[getSequence()](http://docs.google.com/javax/sound/midi/Sequencer.html#getSequence())

### getTrackSolo

boolean **getTrackSolo**(int track)

Obtains the current solo state for a track. The default mute state for all tracks which have not been solo'd is false. In any case where the specified track has not been solo'd, this method should return false. This applies if the sequencer does not support soloing of tracks, and if the specified track index is not valid.

**Parameters:**track - the track number. Tracks in the current sequence are numbered from 0 to the number of tracks in the sequence minus 1. **Returns:**true if solo'd, false if not.

### addMetaEventListener

boolean **addMetaEventListener**([MetaEventListener](http://docs.google.com/javax/sound/midi/MetaEventListener.html) listener)

Registers a meta-event listener to receive notification whenever a meta-event is encountered in the sequence and processed by the sequencer. This method can fail if, for instance,this class of sequencer does not support meta-event notification.

**Parameters:**listener - listener to add **Returns:**true if the listener was successfully added, otherwise false**See Also:**[removeMetaEventListener(javax.sound.midi.MetaEventListener)](http://docs.google.com/javax/sound/midi/Sequencer.html#removeMetaEventListener(javax.sound.midi.MetaEventListener)), [MetaEventListener](http://docs.google.com/javax/sound/midi/MetaEventListener.html), [MetaMessage](http://docs.google.com/javax/sound/midi/MetaMessage.html)

### removeMetaEventListener

void **removeMetaEventListener**([MetaEventListener](http://docs.google.com/javax/sound/midi/MetaEventListener.html) listener)

Removes the specified meta-event listener from this sequencer's list of registered listeners, if in fact the listener is registered.

**Parameters:**listener - the meta-event listener to remove**See Also:**[addMetaEventListener(javax.sound.midi.MetaEventListener)](http://docs.google.com/javax/sound/midi/Sequencer.html#addMetaEventListener(javax.sound.midi.MetaEventListener))

### addControllerEventListener

int[] **addControllerEventListener**([ControllerEventListener](http://docs.google.com/javax/sound/midi/ControllerEventListener.html) listener,  
 int[] controllers)

Registers a controller event listener to receive notification whenever the sequencer processes a control-change event of the requested type or types. The types are specified by the controllers argument, which should contain an array of MIDI controller numbers. (Each number should be between 0 and 127, inclusive. See the MIDI 1.0 Specification for the numbers that correspond to various types of controllers.)

The returned array contains the MIDI controller numbers for which the listener will now receive events. Some sequencers might not support controller event notification, in which case the array has a length of 0. Other sequencers might support notification for some controllers but not all. This method may be invoked repeatedly. Each time, the returned array indicates all the controllers that the listener will be notified about, not only the controllers requested in that particular invocation.

**Parameters:**listener - the controller event listener to add to the list of registered listenerscontrollers - the MIDI controller numbers for which change notification is requested **Returns:**the numbers of all the MIDI controllers whose changes will now be reported to the specified listener**See Also:**[removeControllerEventListener(javax.sound.midi.ControllerEventListener, int[])](http://docs.google.com/javax/sound/midi/Sequencer.html#removeControllerEventListener(javax.sound.midi.ControllerEventListener,%20int%5B%5D)), [ControllerEventListener](http://docs.google.com/javax/sound/midi/ControllerEventListener.html)

### removeControllerEventListener

int[] **removeControllerEventListener**([ControllerEventListener](http://docs.google.com/javax/sound/midi/ControllerEventListener.html) listener,  
 int[] controllers)

Removes a controller event listener's interest in one or more types of controller event. The controllers argument is an array of MIDI numbers corresponding to the controllers for which the listener should no longer receive change notifications. To completely remove this listener from the list of registered listeners, pass in null for controllers. The returned array contains the MIDI controller numbers for which the listener will now receive events. The array has a length of 0 if the listener will not receive change notifications for any controllers.

**Parameters:**listener - old listenercontrollers - the MIDI controller numbers for which change notification should be cancelled, or null to cancel for all controllers **Returns:**the numbers of all the MIDI controllers whose changes will now be reported to the specified listener**See Also:**[addControllerEventListener(javax.sound.midi.ControllerEventListener, int[])](http://docs.google.com/javax/sound/midi/Sequencer.html#addControllerEventListener(javax.sound.midi.ControllerEventListener,%20int%5B%5D))

### setLoopStartPoint

void **setLoopStartPoint**(long tick)

Sets the first MIDI tick that will be played in the loop. If the loop count is greater than 0, playback will jump to this point when reaching the loop end point.

A value of 0 for the starting point means the beginning of the loaded sequence. The starting point must be lower than or equal to the ending point, and it must fall within the size of the loaded sequence.

A sequencer's loop start point defaults to start of the sequence.

**Parameters:**tick - the loop's starting position, in MIDI ticks (zero-based) **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the requested loop start point cannot be set, usually because it falls outside the sequence's duration or because the start point is after the end point**Since:** 1.5 **See Also:**[setLoopEndPoint(long)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopEndPoint(long)), [setLoopCount(int)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopCount(int)), [getLoopStartPoint()](http://docs.google.com/javax/sound/midi/Sequencer.html#getLoopStartPoint()), [start()](http://docs.google.com/javax/sound/midi/Sequencer.html#start())

### getLoopStartPoint

long **getLoopStartPoint**()

Obtains the start position of the loop, in MIDI ticks.

**Returns:**the start position of the loop, in MIDI ticks (zero-based)**Since:** 1.5 **See Also:**[setLoopStartPoint(long)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopStartPoint(long))

### setLoopEndPoint

void **setLoopEndPoint**(long tick)

Sets the last MIDI tick that will be played in the loop. If the loop count is 0, the loop end point has no effect and playback continues to play when reaching the loop end point.

A value of -1 for the ending point indicates the last tick of the sequence. Otherwise, the ending point must be greater than or equal to the starting point, and it must fall within the size of the loaded sequence.

A sequencer's loop end point defaults to -1, meaning the end of the sequence.

**Parameters:**tick - the loop's ending position, in MIDI ticks (zero-based), or -1 to indicate the final tick **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the requested loop point cannot be set, usually because it falls outside the sequence's duration or because the ending point is before the starting point**Since:** 1.5 **See Also:**[setLoopStartPoint(long)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopStartPoint(long)), [setLoopCount(int)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopCount(int)), [getLoopEndPoint()](http://docs.google.com/javax/sound/midi/Sequencer.html#getLoopEndPoint()), [start()](http://docs.google.com/javax/sound/midi/Sequencer.html#start())

### getLoopEndPoint

long **getLoopEndPoint**()

Obtains the end position of the loop, in MIDI ticks.

**Returns:**the end position of the loop, in MIDI ticks (zero-based), or -1 to indicate the end of the sequence**Since:** 1.5 **See Also:**[setLoopEndPoint(long)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopEndPoint(long))

### setLoopCount

void **setLoopCount**(int count)

Sets the number of repetitions of the loop for playback. When the playback position reaches the loop end point, it will loop back to the loop start point count times, after which playback will continue to play to the end of the sequence.

If the current position when this method is invoked is greater than the loop end point, playback continues to the end of the sequence without looping, unless the loop end point is changed subsequently.

A count value of 0 disables looping: playback will continue at the loop end point, and it will not loop back to the loop start point. This is a sequencer's default.

If playback is stopped during looping, the current loop status is cleared; subsequent start requests are not affected by an interrupted loop operation.

**Parameters:**count - the number of times playback should loop back from the loop's end position to the loop's start position, or [LOOP\_CONTINUOUSLY](http://docs.google.com/javax/sound/midi/Sequencer.html#LOOP_CONTINUOUSLY) to indicate that looping should continue until interrupted **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if count is negative and not equal to [LOOP\_CONTINUOUSLY](http://docs.google.com/javax/sound/midi/Sequencer.html#LOOP_CONTINUOUSLY)**Since:** 1.5 **See Also:**[setLoopStartPoint(long)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopStartPoint(long)), [setLoopEndPoint(long)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopEndPoint(long)), [getLoopCount()](http://docs.google.com/javax/sound/midi/Sequencer.html#getLoopCount()), [start()](http://docs.google.com/javax/sound/midi/Sequencer.html#start())

### getLoopCount

int **getLoopCount**()

Obtains the number of repetitions for playback.

**Returns:**the number of loops after which playback plays to the end of the sequence**Since:** 1.5 **See Also:**[setLoopCount(int)](http://docs.google.com/javax/sound/midi/Sequencer.html#setLoopCount(int)), [start()](http://docs.google.com/javax/sound/midi/Sequencer.html#start())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Sequencer.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/sound/midi/Sequence.html)   [**NEXT CLASS**](http://docs.google.com/javax/sound/midi/Sequencer.SyncMode.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sound/midi/Sequencer.html)    [**NO FRAMES**](http://docs.google.com/Sequencer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#tyjcwt) | CONSTR | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | CONSTR | [METHOD](#17dp8vu) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).