| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/MidiDevice.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/MidiChannel.html)   [**NEXT CLASS**](http://docs.google.com/javax/sound/midi/MidiDevice.Info.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sound/midi/MidiDevice.html)    [**NO FRAMES**](http://docs.google.com/MidiDevice.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | CONSTR | [METHOD](#2et92p0) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

## **javax.sound.midi**

Interface MidiDevice

**All Known Subinterfaces:** [Sequencer](http://docs.google.com/javax/sound/midi/Sequencer.html), [Synthesizer](http://docs.google.com/javax/sound/midi/Synthesizer.html)

public interface **MidiDevice**

MidiDevice is the base interface for all MIDI devices. Common devices include synthesizers, sequencers, MIDI input ports, and MIDI output ports.

A MidiDevice can be a transmitter or a receiver of MIDI events, or both. Therefore, it can provide [Transmitter](http://docs.google.com/javax/sound/midi/Transmitter.html) or [Receiver](http://docs.google.com/javax/sound/midi/Receiver.html) instances (or both). Typically, MIDI IN ports provide transmitters, MIDI OUT ports and synthesizers provide receivers. A Sequencer typically provides transmitters for playback and receivers for recording.

A MidiDevice can be opened and closed explicitly as well as implicitly. Explicit opening is accomplished by calling [open()](http://docs.google.com/javax/sound/midi/MidiDevice.html#open()), explicit closing is done by calling [close()](http://docs.google.com/javax/sound/midi/MidiDevice.html#close()) on the MidiDevice instance. If an application opens a MidiDevice explicitly, it has to close it explicitly to free system resources and enable the application to exit cleanly. Implicit opening is done by calling [MidiSystem.getReceiver](http://docs.google.com/javax/sound/midi/MidiSystem.html#getReceiver()) and [MidiSystem.getTransmitter](http://docs.google.com/javax/sound/midi/MidiSystem.html#getTransmitter()). The MidiDevice used by MidiSystem.getReceiver and MidiSystem.getTransmitter is implementation-dependant unless the properties javax.sound.midi.Receiver and javax.sound.midi.Transmitter are used (see the description of properties to select default providers in [MidiSystem](http://docs.google.com/javax/sound/midi/MidiSystem.html)). A MidiDevice that was opened implicitly, is closed implicitly by closing the Receiver or Transmitter that resulted in opening it. If more than one implicitly opening Receiver or Transmitter were obtained by the application, the decive is closed after the last Receiver or Transmitter has been closed. On the other hand, calling getReceiver or getTransmitter on the device instance directly does not open the device implicitly. Closing these Transmitters and Receivers does not close the device implicitly. To use a device with Receivers or Transmitters obtained this way, the device has to be opened and closed explicitly.

If implicit and explicit opening and closing are mixed on the same MidiDevice instance, the following rules apply:

* After an explicit open (either before or after implicit opens), the device will not be closed by implicit closing. The only way to close an explicitly opened device is an explicit close.
* An explicit close always closes the device, even if it also has been opened implicitly. A subsequent implicit close has no further effect.

To detect if a MidiDevice represents a hardware MIDI port, the following programming technique can be used:

MidiDevice device = ...;  
 if ( ! (device instanceof Sequencer) && ! (device instanceof Synthesizer)) {  
 // we're now sure that device represents a MIDI port  
 // ...  
 }

A MidiDevice includes a [MidiDevice.Info](http://docs.google.com/javax/sound/midi/MidiDevice.Info.html) object to provide manufacturer information and so on.

**See Also:**[Synthesizer](http://docs.google.com/javax/sound/midi/Synthesizer.html), [Sequencer](http://docs.google.com/javax/sound/midi/Sequencer.html), [Receiver](http://docs.google.com/javax/sound/midi/Receiver.html), [Transmitter](http://docs.google.com/javax/sound/midi/Transmitter.html)

| **Nested Class Summary** | |
| --- | --- |
| static class | [**MidiDevice.Info**](http://docs.google.com/javax/sound/midi/MidiDevice.Info.html)            A MidiDevice.Info object contains assorted data about a [MidiDevice](http://docs.google.com/javax/sound/midi/MidiDevice.html), including its name, the company who created it, and descriptive text. |

| **Method Summary** | |
| --- | --- |
| void | [**close**](http://docs.google.com/javax/sound/midi/MidiDevice.html#close())()            Closes the device, indicating that the device should now release any system resources it is using. |
| [MidiDevice.Info](http://docs.google.com/javax/sound/midi/MidiDevice.Info.html) | [**getDeviceInfo**](http://docs.google.com/javax/sound/midi/MidiDevice.html#getDeviceInfo())()            Obtains information about the device, including its Java class and Strings containing its name, vendor, and description. |
| int | [**getMaxReceivers**](http://docs.google.com/javax/sound/midi/MidiDevice.html#getMaxReceivers())()            Obtains the maximum number of MIDI IN connections available on this MIDI device for receiving MIDI data. |
| int | [**getMaxTransmitters**](http://docs.google.com/javax/sound/midi/MidiDevice.html#getMaxTransmitters())()            Obtains the maximum number of MIDI OUT connections available on this MIDI device for transmitting MIDI data. |
| long | [**getMicrosecondPosition**](http://docs.google.com/javax/sound/midi/MidiDevice.html#getMicrosecondPosition())()            Obtains the current time-stamp of the device, in microseconds. |
| [Receiver](http://docs.google.com/javax/sound/midi/Receiver.html) | [**getReceiver**](http://docs.google.com/javax/sound/midi/MidiDevice.html#getReceiver())()            Obtains a MIDI IN receiver through which the MIDI device may receive MIDI data. |
| [List](http://docs.google.com/java/util/List.html)<[Receiver](http://docs.google.com/javax/sound/midi/Receiver.html)> | [**getReceivers**](http://docs.google.com/javax/sound/midi/MidiDevice.html#getReceivers())()            Returns all currently active, non-closed receivers connected with this MidiDevice. |
| [Transmitter](http://docs.google.com/javax/sound/midi/Transmitter.html) | [**getTransmitter**](http://docs.google.com/javax/sound/midi/MidiDevice.html#getTransmitter())()            Obtains a MIDI OUT connection from which the MIDI device will transmit MIDI data The returned transmitter must be closed when the application has finished using it. |
| [List](http://docs.google.com/java/util/List.html)<[Transmitter](http://docs.google.com/javax/sound/midi/Transmitter.html)> | [**getTransmitters**](http://docs.google.com/javax/sound/midi/MidiDevice.html#getTransmitters())()            Returns all currently active, non-closed transmitters connected with this MidiDevice. |
| boolean | [**isOpen**](http://docs.google.com/javax/sound/midi/MidiDevice.html#isOpen())()            Reports whether the device is open. |
| void | [**open**](http://docs.google.com/javax/sound/midi/MidiDevice.html#open())()            Opens the device, indicating that it should now acquire any system resources it requires and become operational. |

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

### getDeviceInfo

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

Obtains information about the device, including its Java class and Strings containing its name, vendor, and description.

**Returns:**device info

### open

void **open**()  
 throws [MidiUnavailableException](http://docs.google.com/javax/sound/midi/MidiUnavailableException.html)

Opens the device, indicating that it should now acquire any system resources it requires and become operational.

An application opening a device explicitly with this call has to close the device by calling [close()](http://docs.google.com/javax/sound/midi/MidiDevice.html#close()). This is necessary to release system resources and allow applications to exit cleanly.

Note that some devices, once closed, cannot be reopened. Attempts to reopen such a device will always result in a MidiUnavailableException.

**Throws:** [MidiUnavailableException](http://docs.google.com/javax/sound/midi/MidiUnavailableException.html) - thrown if the device cannot be opened due to resource restrictions. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - thrown if the device cannot be opened due to security restrictions.**See Also:**[close()](http://docs.google.com/javax/sound/midi/MidiDevice.html#close()), [isOpen()](http://docs.google.com/javax/sound/midi/MidiDevice.html#isOpen())

### close

void **close**()

Closes the device, indicating that the device should now release any system resources it is using.

All Receiver and Transmitter instances open from this device are closed. This includes instances retrieved via MidiSystem.

**See Also:**[open()](http://docs.google.com/javax/sound/midi/MidiDevice.html#open()), [isOpen()](http://docs.google.com/javax/sound/midi/MidiDevice.html#isOpen())

### isOpen

boolean **isOpen**()

Reports whether the device is open.

**Returns:**true if the device is open, otherwise false**See Also:**[open()](http://docs.google.com/javax/sound/midi/MidiDevice.html#open()), [close()](http://docs.google.com/javax/sound/midi/MidiDevice.html#close())

### getMicrosecondPosition

long **getMicrosecondPosition**()

Obtains the current time-stamp of the device, in microseconds. If a device supports time-stamps, it should start counting at 0 when the device is opened and continue incrementing its time-stamp in microseconds until the device is closed. If it does not support time-stamps, it should always return -1.

**Returns:**the current time-stamp of the device in microseconds, or -1 if time-stamping is not supported by the device.

### getMaxReceivers

int **getMaxReceivers**()

Obtains the maximum number of MIDI IN connections available on this MIDI device for receiving MIDI data.

**Returns:**maximum number of MIDI IN connections, or -1 if an unlimited number of connections is available.

### getMaxTransmitters

int **getMaxTransmitters**()

Obtains the maximum number of MIDI OUT connections available on this MIDI device for transmitting MIDI data.

**Returns:**maximum number of MIDI OUT connections, or -1 if an unlimited number of connections is available.

### getReceiver

[Receiver](http://docs.google.com/javax/sound/midi/Receiver.html) **getReceiver**()  
 throws [MidiUnavailableException](http://docs.google.com/javax/sound/midi/MidiUnavailableException.html)

Obtains a MIDI IN receiver through which the MIDI device may receive MIDI data. The returned receiver must be closed when the application has finished using it.

Obtaining a Receiver with this method does not open the device. To be able to use the device, it has to be opened explicitly by calling [open()](http://docs.google.com/javax/sound/midi/MidiDevice.html#open()). Also, closing the Receiver does not close the device. It has to be closed explicitly by calling [close()](http://docs.google.com/javax/sound/midi/MidiDevice.html#close()).

**Returns:**a receiver for the device. **Throws:** [MidiUnavailableException](http://docs.google.com/javax/sound/midi/MidiUnavailableException.html) - thrown if a receiver is not available due to resource restrictions**See Also:**[Receiver.close()](http://docs.google.com/javax/sound/midi/Receiver.html#close())

### getReceivers

[List](http://docs.google.com/java/util/List.html)<[Receiver](http://docs.google.com/javax/sound/midi/Receiver.html)> **getReceivers**()

Returns all currently active, non-closed receivers connected with this MidiDevice. A receiver can be removed from the device by closing it.

**Returns:**an unmodifiable list of the open receivers**Since:** 1.5

### getTransmitter

[Transmitter](http://docs.google.com/javax/sound/midi/Transmitter.html) **getTransmitter**()  
 throws [MidiUnavailableException](http://docs.google.com/javax/sound/midi/MidiUnavailableException.html)

Obtains a MIDI OUT connection from which the MIDI device will transmit MIDI data The returned transmitter must be closed when the application has finished using it.

Obtaining a Transmitter with this method does not open the device. To be able to use the device, it has to be opened explicitly by calling [open()](http://docs.google.com/javax/sound/midi/MidiDevice.html#open()). Also, closing the Transmitter does not close the device. It has to be closed explicitly by calling [close()](http://docs.google.com/javax/sound/midi/MidiDevice.html#close()).

**Returns:**a MIDI OUT transmitter for the device. **Throws:** [MidiUnavailableException](http://docs.google.com/javax/sound/midi/MidiUnavailableException.html) - thrown if a transmitter is not available due to resource restrictions**See Also:**[Transmitter.close()](http://docs.google.com/javax/sound/midi/Transmitter.html#close())

### getTransmitters

[List](http://docs.google.com/java/util/List.html)<[Transmitter](http://docs.google.com/javax/sound/midi/Transmitter.html)> **getTransmitters**()

Returns all currently active, non-closed transmitters connected with this MidiDevice. A transmitter can be removed from the device by closing it.

**Returns:**an unmodifiable list of the open transmitters**Since:** 1.5

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/MidiDevice.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/MidiChannel.html)   [**NEXT CLASS**](http://docs.google.com/javax/sound/midi/MidiDevice.Info.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sound/midi/MidiDevice.html)    [**NO FRAMES**](http://docs.google.com/MidiDevice.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | CONSTR | [METHOD](#2et92p0) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

[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).