| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SourceDataLine.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/sampled/ReverbType.html)   [**NEXT CLASS**](http://docs.google.com/javax/sound/sampled/TargetDataLine.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sound/sampled/SourceDataLine.html)    [**NO FRAMES**](http://docs.google.com/SourceDataLine.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#tyjcwt) | DETAIL: FIELD | CONSTR | [METHOD](#4d34og8) |

## **javax.sound.sampled**

Interface SourceDataLine

**All Superinterfaces:** [DataLine](http://docs.google.com/javax/sound/sampled/DataLine.html), [Line](http://docs.google.com/javax/sound/sampled/Line.html)

public interface **SourceDataLine**extends [DataLine](http://docs.google.com/javax/sound/sampled/DataLine.html)

A source data line is a data line to which data may be written. It acts as a source to its mixer. An application writes audio bytes to a source data line, which handles the buffering of the bytes and delivers them to the mixer. The mixer may mix the samples with those from other sources and then deliver the mix to a target such as an output port (which may represent an audio output device on a sound card).

Note that the naming convention for this interface reflects the relationship between the line and its mixer. From the perspective of an application, a source data line may act as a target for audio data.

A source data line can be obtained from a mixer by invoking the [getLine](http://docs.google.com/javax/sound/sampled/Mixer.html#getLine(javax.sound.sampled.Line.Info)) method of Mixer with an appropriate [DataLine.Info](http://docs.google.com/javax/sound/sampled/DataLine.Info.html) object.

The SourceDataLine interface provides a method for writing audio data to the data line's buffer. Applications that play or mix audio should write data to the source data line quickly enough to keep the buffer from underflowing (emptying), which could cause discontinuities in the audio that are perceived as clicks. Applications can use the [available](http://docs.google.com/javax/sound/sampled/DataLine.html#available()) method defined in the DataLine interface to determine the amount of data currently queued in the data line's buffer. The amount of data which can be written to the buffer without blocking is the difference between the buffer size and the amount of queued data. If the delivery of audio output stops due to underflow, a [STOP](http://docs.google.com/javax/sound/sampled/LineEvent.Type.html#STOP) event is generated. A [START](http://docs.google.com/javax/sound/sampled/LineEvent.Type.html#START) event is generated when the audio output resumes.

**Since:** 1.3 **See Also:**[Mixer](http://docs.google.com/javax/sound/sampled/Mixer.html), [DataLine](http://docs.google.com/javax/sound/sampled/DataLine.html), [TargetDataLine](http://docs.google.com/javax/sound/sampled/TargetDataLine.html)

| **Nested Class Summary** | |
| --- | --- |

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

| **Method Summary** | |
| --- | --- |
| void | [**open**](http://docs.google.com/javax/sound/sampled/SourceDataLine.html#open(javax.sound.sampled.AudioFormat))([AudioFormat](http://docs.google.com/javax/sound/sampled/AudioFormat.html) format)            Opens the line with the specified format, causing the line to acquire any required system resources and become operational. |
| void | [**open**](http://docs.google.com/javax/sound/sampled/SourceDataLine.html#open(javax.sound.sampled.AudioFormat,%20int))([AudioFormat](http://docs.google.com/javax/sound/sampled/AudioFormat.html) format, int bufferSize)            Opens the line with the specified format and suggested buffer size, causing the line to acquire any required system resources and become operational. |
| int | [**write**](http://docs.google.com/javax/sound/sampled/SourceDataLine.html#write(byte%5B%5D,%20int,%20int))(byte[] b, int off, int len)            Writes audio data to the mixer via this source data line. |

| **Methods inherited from interface javax.sound.sampled.**[**DataLine**](http://docs.google.com/javax/sound/sampled/DataLine.html) |
| --- |
| [available](http://docs.google.com/javax/sound/sampled/DataLine.html#available()), [drain](http://docs.google.com/javax/sound/sampled/DataLine.html#drain()), [flush](http://docs.google.com/javax/sound/sampled/DataLine.html#flush()), [getBufferSize](http://docs.google.com/javax/sound/sampled/DataLine.html#getBufferSize()), [getFormat](http://docs.google.com/javax/sound/sampled/DataLine.html#getFormat()), [getFramePosition](http://docs.google.com/javax/sound/sampled/DataLine.html#getFramePosition()), [getLevel](http://docs.google.com/javax/sound/sampled/DataLine.html#getLevel()), [getLongFramePosition](http://docs.google.com/javax/sound/sampled/DataLine.html#getLongFramePosition()), [getMicrosecondPosition](http://docs.google.com/javax/sound/sampled/DataLine.html#getMicrosecondPosition()), [isActive](http://docs.google.com/javax/sound/sampled/DataLine.html#isActive()), [isRunning](http://docs.google.com/javax/sound/sampled/DataLine.html#isRunning()), [start](http://docs.google.com/javax/sound/sampled/DataLine.html#start()), [stop](http://docs.google.com/javax/sound/sampled/DataLine.html#stop()) |

| **Methods inherited from interface javax.sound.sampled.**[**Line**](http://docs.google.com/javax/sound/sampled/Line.html) |
| --- |
| [addLineListener](http://docs.google.com/javax/sound/sampled/Line.html#addLineListener(javax.sound.sampled.LineListener)), [close](http://docs.google.com/javax/sound/sampled/Line.html#close()), [getControl](http://docs.google.com/javax/sound/sampled/Line.html#getControl(javax.sound.sampled.Control.Type)), [getControls](http://docs.google.com/javax/sound/sampled/Line.html#getControls()), [getLineInfo](http://docs.google.com/javax/sound/sampled/Line.html#getLineInfo()), [isControlSupported](http://docs.google.com/javax/sound/sampled/Line.html#isControlSupported(javax.sound.sampled.Control.Type)), [isOpen](http://docs.google.com/javax/sound/sampled/Line.html#isOpen()), [open](http://docs.google.com/javax/sound/sampled/Line.html#open()), [removeLineListener](http://docs.google.com/javax/sound/sampled/Line.html#removeLineListener(javax.sound.sampled.LineListener)) |

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

### open

void **open**([AudioFormat](http://docs.google.com/javax/sound/sampled/AudioFormat.html) format,  
 int bufferSize)  
 throws [LineUnavailableException](http://docs.google.com/javax/sound/sampled/LineUnavailableException.html)

Opens the line with the specified format and suggested buffer size, causing the line to acquire any required system resources and become operational.

The buffer size is specified in bytes, but must represent an integral number of sample frames. Invoking this method with a requested buffer size that does not meet this requirement may result in an IllegalArgumentException. The actual buffer size for the open line may differ from the requested buffer size. The value actually set may be queried by subsequently calling [DataLine.getBufferSize()](http://docs.google.com/javax/sound/sampled/DataLine.html#getBufferSize()).

If this operation succeeds, the line is marked as open, and an [OPEN](http://docs.google.com/javax/sound/sampled/LineEvent.Type.html#OPEN) event is dispatched to the line's listeners.

Invoking this method on a line which is already open is illegal and may result in an IllegalStateException.

Note that some lines, once closed, cannot be reopened. Attempts to reopen such a line will always result in a LineUnavailableException.

**Parameters:**format - the desired audio formatbufferSize - the desired buffer size **Throws:** [LineUnavailableException](http://docs.google.com/javax/sound/sampled/LineUnavailableException.html) - if the line cannot be opened due to resource restrictions [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the buffer size does not represent an integral number of sample frames, or if format is not fully specified or invalid [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the line is already open [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the line cannot be opened due to security restrictions**See Also:**[open(AudioFormat)](http://docs.google.com/javax/sound/sampled/SourceDataLine.html#open(javax.sound.sampled.AudioFormat)), [Line.open()](http://docs.google.com/javax/sound/sampled/Line.html#open()), [Line.close()](http://docs.google.com/javax/sound/sampled/Line.html#close()), [Line.isOpen()](http://docs.google.com/javax/sound/sampled/Line.html#isOpen()), [LineEvent](http://docs.google.com/javax/sound/sampled/LineEvent.html)

### open

void **open**([AudioFormat](http://docs.google.com/javax/sound/sampled/AudioFormat.html) format)  
 throws [LineUnavailableException](http://docs.google.com/javax/sound/sampled/LineUnavailableException.html)

Opens the line with the specified format, causing the line to acquire any required system resources and become operational.

The implementation chooses a buffer size, which is measured in bytes but which encompasses an integral number of sample frames. The buffer size that the system has chosen may be queried by subsequently calling [DataLine.getBufferSize()](http://docs.google.com/javax/sound/sampled/DataLine.html#getBufferSize()).

If this operation succeeds, the line is marked as open, and an [OPEN](http://docs.google.com/javax/sound/sampled/LineEvent.Type.html#OPEN) event is dispatched to the line's listeners.

Invoking this method on a line which is already open is illegal and may result in an IllegalStateException.

Note that some lines, once closed, cannot be reopened. Attempts to reopen such a line will always result in a LineUnavailableException.

**Parameters:**format - the desired audio format **Throws:** [LineUnavailableException](http://docs.google.com/javax/sound/sampled/LineUnavailableException.html) - if the line cannot be opened due to resource restrictions [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if format is not fully specified or invalid [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the line is already open [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the line cannot be opened due to security restrictions**See Also:**[open(AudioFormat, int)](http://docs.google.com/javax/sound/sampled/SourceDataLine.html#open(javax.sound.sampled.AudioFormat,%20int)), [Line.open()](http://docs.google.com/javax/sound/sampled/Line.html#open()), [Line.close()](http://docs.google.com/javax/sound/sampled/Line.html#close()), [Line.isOpen()](http://docs.google.com/javax/sound/sampled/Line.html#isOpen()), [LineEvent](http://docs.google.com/javax/sound/sampled/LineEvent.html)

### write

int **write**(byte[] b,  
 int off,  
 int len)

Writes audio data to the mixer via this source data line. The requested number of bytes of data are read from the specified array, starting at the given offset into the array, and written to the data line's buffer. If the caller attempts to write more data than can currently be written (see [available](http://docs.google.com/javax/sound/sampled/DataLine.html#available())), this method blocks until the requested amount of data has been written. This applies even if the requested amount of data to write is greater than the data line's buffer size. However, if the data line is closed, stopped, or flushed before the requested amount has been written, the method no longer blocks, but returns the number of bytes written thus far.

The number of bytes that can be written without blocking can be ascertained using the [available](http://docs.google.com/javax/sound/sampled/DataLine.html#available()) method of the DataLine interface. (While it is guaranteed that this number of bytes can be written without blocking, there is no guarantee that attempts to write additional data will block.)

The number of bytes to write must represent an integral number of sample frames, such that:

[ bytes written ] % [frame size in bytes ] == 0

The return value will always meet this requirement. A request to write a number of bytes representing a non-integral number of sample frames cannot be fulfilled and may result in an IllegalArgumentException.

**Parameters:**b - a byte array containing data to be written to the data linelen - the length, in bytes, of the valid data in the array (in other words, the requested amount of data to write, in bytes)off - the offset from the beginning of the array, in bytes **Returns:**the number of bytes actually written **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the requested number of bytes does not represent an integral number of sample frames, or if len is negative [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if off is negative, or off+len is greater than the length of the array b.**See Also:**[TargetDataLine.read(byte[], int, int)](http://docs.google.com/javax/sound/sampled/TargetDataLine.html#read(byte%5B%5D,%20int,%20int)), [DataLine.available()](http://docs.google.com/javax/sound/sampled/DataLine.html#available())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SourceDataLine.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/sampled/ReverbType.html)   [**NEXT CLASS**](http://docs.google.com/javax/sound/sampled/TargetDataLine.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sound/sampled/SourceDataLine.html)    [**NO FRAMES**](http://docs.google.com/SourceDataLine.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#tyjcwt) | DETAIL: FIELD | CONSTR | [METHOD](#4d34og8) |

[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).