| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SocketOptions.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/java/net/SocketImplFactory.html)   [**NEXT CLASS**](http://docs.google.com/java/net/SocketPermission.html) | [**FRAMES**](http://docs.google.com/index.html?java/net/SocketOptions.html)    [**NO FRAMES**](http://docs.google.com/SocketOptions.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#tyjcwt) | CONSTR | [METHOD](#1y810tw) |

## **java.net**

Interface SocketOptions

**All Known Implementing Classes:** [DatagramSocketImpl](http://docs.google.com/java/net/DatagramSocketImpl.html), [SocketImpl](http://docs.google.com/java/net/SocketImpl.html)

public interface **SocketOptions**

Interface of methods to get/set socket options. This interface is implemented by: **SocketImpl** and **DatagramSocketImpl**. Subclasses of these should override the methods of this interface in order to support their own options.

The methods and constants which specify options in this interface are for implementation only. If you're not subclassing SocketImpl or DatagramSocketImpl, **you won't use these directly.** There are type-safe methods to get/set each of these options in Socket, ServerSocket, DatagramSocket and MulticastSocket.

| **Field Summary** | |
| --- | --- |
| static int | [**IP\_MULTICAST\_IF**](http://docs.google.com/java/net/SocketOptions.html#IP_MULTICAST_IF)            Set which outgoing interface on which to send multicast packets. |
| static int | [**IP\_MULTICAST\_IF2**](http://docs.google.com/java/net/SocketOptions.html#IP_MULTICAST_IF2)            Same as above. |
| static int | [**IP\_MULTICAST\_LOOP**](http://docs.google.com/java/net/SocketOptions.html#IP_MULTICAST_LOOP)            This option enables or disables local loopback of multicast datagrams. |
| static int | [**IP\_TOS**](http://docs.google.com/java/net/SocketOptions.html#IP_TOS)            This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket. |
| static int | [**SO\_BINDADDR**](http://docs.google.com/java/net/SocketOptions.html#SO_BINDADDR)            Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed). |
| static int | [**SO\_BROADCAST**](http://docs.google.com/java/net/SocketOptions.html#SO_BROADCAST)            Sets SO\_BROADCAST for a socket. |
| static int | [**SO\_KEEPALIVE**](http://docs.google.com/java/net/SocketOptions.html#SO_KEEPALIVE)            When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer. |
| static int | [**SO\_LINGER**](http://docs.google.com/java/net/SocketOptions.html#SO_LINGER)            Specify a linger-on-close timeout. |
| static int | [**SO\_OOBINLINE**](http://docs.google.com/java/net/SocketOptions.html#SO_OOBINLINE)            When the OOBINLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream. |
| static int | [**SO\_RCVBUF**](http://docs.google.com/java/net/SocketOptions.html#SO_RCVBUF)            Set a hint the size of the underlying buffers used by the platform for incoming network I/O. |
| static int | [**SO\_REUSEADDR**](http://docs.google.com/java/net/SocketOptions.html#SO_REUSEADDR)            Sets SO\_REUSEADDR for a socket. |
| static int | [**SO\_SNDBUF**](http://docs.google.com/java/net/SocketOptions.html#SO_SNDBUF)            Set a hint the size of the underlying buffers used by the platform for outgoing network I/O. |
| static int | [**SO\_TIMEOUT**](http://docs.google.com/java/net/SocketOptions.html#SO_TIMEOUT)            Set a timeout on blocking Socket operations: |
| static int | [**TCP\_NODELAY**](http://docs.google.com/java/net/SocketOptions.html#TCP_NODELAY)            Disable Nagle's algorithm for this connection. |

| **Method Summary** | |
| --- | --- |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getOption**](http://docs.google.com/java/net/SocketOptions.html#getOption(int))(int optID)            Fetch the value of an option. |
| void | [**setOption**](http://docs.google.com/java/net/SocketOptions.html#setOption(int,%20java.lang.Object))(int optID, [Object](http://docs.google.com/java/lang/Object.html) value)            Enable/disable the option specified by *optID*. |

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

### TCP\_NODELAY

static final int **TCP\_NODELAY**

Disable Nagle's algorithm for this connection. Written data to the network is not buffered pending acknowledgement of previously written data.

Valid for TCP only: SocketImpl.

**See Also:**[Socket.setTcpNoDelay(boolean)](http://docs.google.com/java/net/Socket.html#setTcpNoDelay(boolean)), [Socket.getTcpNoDelay()](http://docs.google.com/java/net/Socket.html#getTcpNoDelay()), [Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.TCP_NODELAY)

### SO\_BINDADDR

static final int **SO\_BINDADDR**

Fetch the local address binding of a socket (this option cannot be "set" only "gotten", since sockets are bound at creation time, and so the locally bound address cannot be changed). The default local address of a socket is INADDR\_ANY, meaning any local address on a multi-homed host. A multi-homed host can use this option to accept connections to only one of its addresses (in the case of a ServerSocket or DatagramSocket), or to specify its return address to the peer (for a Socket or DatagramSocket). The parameter of this option is an InetAddress.

This option **must** be specified in the constructor.

Valid for: SocketImpl, DatagramSocketImpl

**See Also:**[Socket.getLocalAddress()](http://docs.google.com/java/net/Socket.html#getLocalAddress()), [DatagramSocket.getLocalAddress()](http://docs.google.com/java/net/DatagramSocket.html#getLocalAddress()), [Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.SO_BINDADDR)

### SO\_REUSEADDR

static final int **SO\_REUSEADDR**

Sets SO\_REUSEADDR for a socket. This is used only for MulticastSockets in java, and it is set by default for MulticastSockets.

Valid for: DatagramSocketImpl

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.SO_REUSEADDR)

### SO\_BROADCAST

static final int **SO\_BROADCAST**

Sets SO\_BROADCAST for a socket. This option enables and disables the ability of the process to send broadcast messages. It is supported for only datagram sockets and only on networks that support the concept of a broadcast message (e.g. Ethernet, token ring, etc.), and it is set by default for DatagramSockets.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.SO_BROADCAST)

### IP\_MULTICAST\_IF

static final int **IP\_MULTICAST\_IF**

Set which outgoing interface on which to send multicast packets. Useful on hosts with multiple network interfaces, where applications want to use other than the system default. Takes/returns an InetAddress.

Valid for Multicast: DatagramSocketImpl

**See Also:**[MulticastSocket.setInterface(InetAddress)](http://docs.google.com/java/net/MulticastSocket.html#setInterface(java.net.InetAddress)), [MulticastSocket.getInterface()](http://docs.google.com/java/net/MulticastSocket.html#getInterface()), [Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.IP_MULTICAST_IF)

### IP\_MULTICAST\_IF2

static final int **IP\_MULTICAST\_IF2**

Same as above. This option is introduced so that the behaviour with IP\_MULTICAST\_IF will be kept the same as before, while this new option can support setting outgoing interfaces with either IPv4 and IPv6 addresses. NOTE: make sure there is no conflict with this

**Since:** 1.4 **See Also:**[MulticastSocket.setNetworkInterface(NetworkInterface)](http://docs.google.com/java/net/MulticastSocket.html#setNetworkInterface(java.net.NetworkInterface)), [MulticastSocket.getNetworkInterface()](http://docs.google.com/java/net/MulticastSocket.html#getNetworkInterface()), [Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.IP_MULTICAST_IF2)

### IP\_MULTICAST\_LOOP

static final int **IP\_MULTICAST\_LOOP**

This option enables or disables local loopback of multicast datagrams. This option is enabled by default for Multicast Sockets.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.IP_MULTICAST_LOOP)

### IP\_TOS

static final int **IP\_TOS**

This option sets the type-of-service or traffic class field in the IP header for a TCP or UDP socket.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.IP_TOS)

### SO\_LINGER

static final int **SO\_LINGER**

Specify a linger-on-close timeout. This option disables/enables immediate return from a **close()** of a TCP Socket. Enabling this option with a non-zero Integer *timeout* means that a **close()** will block pending the transmission and acknowledgement of all data written to the peer, at which point the socket is closed *gracefully*. Upon reaching the linger timeout, the socket is closed *forcefully*, with a TCP RST. Enabling the option with a timeout of zero does a forceful close immediately. If the specified timeout value exceeds 65,535 it will be reduced to 65,535.

Valid only for TCP: SocketImpl

**See Also:**[Socket.setSoLinger(boolean, int)](http://docs.google.com/java/net/Socket.html#setSoLinger(boolean,%20int)), [Socket.getSoLinger()](http://docs.google.com/java/net/Socket.html#getSoLinger()), [Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.SO_LINGER)

### SO\_TIMEOUT

static final int **SO\_TIMEOUT**

Set a timeout on blocking Socket operations:

ServerSocket.accept();  
 SocketInputStream.read();  
 DatagramSocket.receive();

The option must be set prior to entering a blocking operation to take effect. If the timeout expires and the operation would continue to block, **java.io.InterruptedIOException** is raised. The Socket is not closed in this case.

Valid for all sockets: SocketImpl, DatagramSocketImpl

**See Also:**[Socket.setSoTimeout(int)](http://docs.google.com/java/net/Socket.html#setSoTimeout(int)), [ServerSocket.setSoTimeout(int)](http://docs.google.com/java/net/ServerSocket.html#setSoTimeout(int)), [DatagramSocket.setSoTimeout(int)](http://docs.google.com/java/net/DatagramSocket.html#setSoTimeout(int)), [Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.SO_TIMEOUT)

### SO\_SNDBUF

static final int **SO\_SNDBUF**

Set a hint the size of the underlying buffers used by the platform for outgoing network I/O. When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be sent over the socket. When used in get, this must return the size of the buffer actually used by the platform when sending out data on this socket. Valid for all sockets: SocketImpl, DatagramSocketImpl

**See Also:**[Socket.setSendBufferSize(int)](http://docs.google.com/java/net/Socket.html#setSendBufferSize(int)), [Socket.getSendBufferSize()](http://docs.google.com/java/net/Socket.html#getSendBufferSize()), [DatagramSocket.setSendBufferSize(int)](http://docs.google.com/java/net/DatagramSocket.html#setSendBufferSize(int)), [DatagramSocket.getSendBufferSize()](http://docs.google.com/java/net/DatagramSocket.html#getSendBufferSize()), [Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.SO_SNDBUF)

### SO\_RCVBUF

static final int **SO\_RCVBUF**

Set a hint the size of the underlying buffers used by the platform for incoming network I/O. When used in set, this is a suggestion to the kernel from the application about the size of buffers to use for the data to be received over the socket. When used in get, this must return the size of the buffer actually used by the platform when receiving in data on this socket. Valid for all sockets: SocketImpl, DatagramSocketImpl

**See Also:**[Socket.setReceiveBufferSize(int)](http://docs.google.com/java/net/Socket.html#setReceiveBufferSize(int)), [Socket.getReceiveBufferSize()](http://docs.google.com/java/net/Socket.html#getReceiveBufferSize()), [DatagramSocket.setReceiveBufferSize(int)](http://docs.google.com/java/net/DatagramSocket.html#setReceiveBufferSize(int)), [DatagramSocket.getReceiveBufferSize()](http://docs.google.com/java/net/DatagramSocket.html#getReceiveBufferSize()), [Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.SO_RCVBUF)

### SO\_KEEPALIVE

static final int **SO\_KEEPALIVE**

When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours (NOTE: the actual value is implementation dependent), TCP automatically sends a keepalive probe to the peer. This probe is a TCP segment to which the peer must respond. One of three responses is expected: 1. The peer responds with the expected ACK. The application is not notified (since everything is OK). TCP will send another probe following another 2 hours of inactivity. 2. The peer responds with an RST, which tells the local TCP that the peer host has crashed and rebooted. The socket is closed. 3. There is no response from the peer. The socket is closed. The purpose of this option is to detect if the peer host crashes. Valid only for TCP socket: SocketImpl

**See Also:**[Socket.setKeepAlive(boolean)](http://docs.google.com/java/net/Socket.html#setKeepAlive(boolean)), [Socket.getKeepAlive()](http://docs.google.com/java/net/Socket.html#getKeepAlive()), [Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.SO_KEEPALIVE)

### SO\_OOBINLINE

static final int **SO\_OOBINLINE**

When the OOBINLINE option is set, any TCP urgent data received on the socket will be received through the socket input stream. When the option is disabled (which is the default) urgent data is silently discarded.

**See Also:**[Socket.setOOBInline(boolean)](http://docs.google.com/java/net/Socket.html#setOOBInline(boolean)), [Socket.getOOBInline()](http://docs.google.com/java/net/Socket.html#getOOBInline()), [Constant Field Values](http://docs.google.com/constant-values.html#java.net.SocketOptions.SO_OOBINLINE)

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

### setOption

void **setOption**(int optID,  
 [Object](http://docs.google.com/java/lang/Object.html) value)  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Enable/disable the option specified by *optID*. If the option is to be enabled, and it takes an option-specific "value", this is passed in *value*. The actual type of value is option-specific, and it is an error to pass something that isn't of the expected type:

SocketImpl s;  
 ...  
 s.setOption(SO\_LINGER, new Integer(10));  
 // OK - set SO\_LINGER w/ timeout of 10 sec.  
 s.setOption(SO\_LINGER, new Double(10));  
 // ERROR - expects java.lang.Integer

If the requested option is binary, it can be set using this method by a java.lang.Boolean:

s.setOption(TCP\_NODELAY, new Boolean(true));  
 // OK - enables TCP\_NODELAY, a binary option

Any option can be disabled using this method with a Boolean(false):

s.setOption(TCP\_NODELAY, new Boolean(false));  
 // OK - disables TCP\_NODELAY  
 s.setOption(SO\_LINGER, new Boolean(false));  
 // OK - disables SO\_LINGER

For an option that has a notion of on and off, and requires a non-boolean parameter, setting its value to anything other than *Boolean(false)* implicitly enables it.

Throws SocketException if the option is unrecognized, the socket is closed, or some low-level error occurred

**Parameters:**optID - identifies the optionvalue - the parameter of the socket option **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if the option is unrecognized, the socket is closed, or some low-level error occurred**See Also:**[getOption(int)](http://docs.google.com/java/net/SocketOptions.html#getOption(int))

### getOption

[Object](http://docs.google.com/java/lang/Object.html) **getOption**(int optID)  
 throws [SocketException](http://docs.google.com/java/net/SocketException.html)

Fetch the value of an option. Binary options will return java.lang.Boolean(true) if enabled, java.lang.Boolean(false) if disabled, e.g.:

SocketImpl s;  
 ...  
 Boolean noDelay = (Boolean)(s.getOption(TCP\_NODELAY));  
 if (noDelay.booleanValue()) {  
 // true if TCP\_NODELAY is enabled...  
 ...  
 }

For options that take a particular type as a parameter, getOption(int) will return the paramter's value, else it will return java.lang.Boolean(false):

Object o = s.getOption(SO\_LINGER);  
 if (o instanceof Integer) {  
 System.out.print("Linger time is " + ((Integer)o).intValue());  
 } else {  
 // the true type of o is java.lang.Boolean(false);  
 }

**Parameters:**optID - an int identifying the option to fetch **Returns:**the value of the option **Throws:** [SocketException](http://docs.google.com/java/net/SocketException.html) - if the socket is closed [SocketException](http://docs.google.com/java/net/SocketException.html) - if *optID* is unknown along the protocol stack (including the SocketImpl)**See Also:**[setOption(int, java.lang.Object)](http://docs.google.com/java/net/SocketOptions.html#setOption(int,%20java.lang.Object))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SocketOptions.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/java/net/SocketImplFactory.html)   [**NEXT CLASS**](http://docs.google.com/java/net/SocketPermission.html) | [**FRAMES**](http://docs.google.com/index.html?java/net/SocketOptions.html)    [**NO FRAMES**](http://docs.google.com/SocketOptions.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#tyjcwt) | CONSTR | [METHOD](#1y810tw) |

[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).