# Class java.net.Socket

```
java.lang.Object
  |
  +--java.net.Socket
```

public class **Socket**
extends Object

This class implements client sockets (also called just "sockets"). A socket is an endpoint for communication between two machines.

The actual work of the socket is performed by an instance of the `SocketImpl` class. An application, by changing the socket factory that creates the socket implementation, can configure itself to create sockets appropriate to the local firewall.

**Since:**
JDK1.0

**See Also:**
`setSocketImplFactory(java.net.SocketImplFactory)`, `SocketImpl`

## Constructor Summary

| |
|---|
| **Socket**()<br>Creates an unconnected socket, with the system-default type of SocketImpl. |
| **Socket**(InetAddress host, int port, boolean stream)<br>**Deprecated.** *Use DatagramSocket instead for UDP transport.* |
| **Socket**(InetAddress address, int port, InetAddress localAddr, int localPort)<br>Creates a socket and connects it to the specified remote address on the specified remote port. |
| **Socket**(InetAddress address, int port)<br>Creates a stream socket and connects it to the specified port number at the specified IP address. |
| **Socket**(SocketImpl impl)<br>Creates an unconnected Socket with a user-specified SocketImpl. |
| **Socket**(String host, int port, boolean stream)<br>**Deprecated.** *Use DatagramSocket instead for UDP transport.* |
| **Socket**(String host, int port, InetAddress localAddr, int localPort)<br>Creates a socket and connects it to the specified remote host on the specified remote port. |
| **Socket**(String host, int port)<br>Creates a stream socket and connects it to the specified port number on the named host. |

## Method Summary

| | |
|---|---|
| void | **close**()<br>Closes this socket. |
| InetAddress | **getInetAddress**()<br>Returns the address to which the socket is connected. |

| | |
|---|---|
| InputStream | **getInputStream**()<br>Returns an input stream for this socket. |
| InetAddress | **getLocalAddress**()<br>Gets the local address to which the socket is bound. |
| int | **getLocalPort**()<br>Returns the local port to which this socket is bound. |
| OutputStream | **getOutputStream**()<br>Returns an output stream for this socket. |
| int | **getPort**()<br>Returns the remote port to which this socket is connected. |
| int | **getReceiveBufferSize**()<br>Get value of the SO_RCVBUF option for this socket, that is the buffer size used by the platform for input on the this Socket. |
| int | **getSendBufferSize**()<br>Get value of the SO_SNDBUF option for this socket, that is the buffer size used by the platform for output on the this Socket. |
| int | **getSoLinger**()<br>Returns setting for SO_LINGER. -1 returns implies that the option is disabled. |
| int | **getSoTimeout**()<br>Returns setting for SO_TIMEOUT. 0 returns implies that the option is disabled (i.e. |
| boolean | **getTcpNoDelay**()<br>Tests if TCP_NODELAY is enabled. |
| void | **setReceiveBufferSize**(int size)<br>Sets the SO_RCVBUF option to the specified value for this DatagramSocket. |
| void | **setSendBufferSize**(int size)<br>Sets the SO_SNDBUF option to the specified value for this DatagramSocket. |
| static void | **setSocketImplFactory**(SocketImplFactory fac)<br>Sets the client socket implementation factory for the application. |
| void | **setSoLinger**(boolean on, int linger)<br>Enable/disable SO_LINGER with the specified linger time. |
| void | **setSoTimeout**(int timeout)<br>Enable/disable SO_TIMEOUT with the specified timeout, in milliseconds. |
| void | **setTcpNoDelay**(boolean on)<br>Enable/disable TCP_NODELAY (disable/enable Nagle's algorithm). |
| String | **toString**()<br>Converts this socket to a `String`. |

| **Methods inherited from class java.lang.Object** |
|---|
| clone , equals , finalize , getClass , hashCode , notify , notifyAll , wait , wait , wait |

# Constructor Detail

### Socket

```
protected Socket()
```

Creates an unconnected socket, with the system-default type of SocketImpl.

**Since:**
JDK1.1

---

### Socket

```
protected Socket(SocketImpl impl)
          throws SocketException
```

Creates an unconnected Socket with a user-specified SocketImpl.

The *impl* parameter is an instance of a **SocketImpl** the subclass wishes to use on the Socket.

**Since:**
JDK1.1

---

### Socket

```
public Socket(String host,
              int port)
       throws UnknownHostException,
              IOException
```

Creates a stream socket and connects it to the specified port number on the named host.

If the application has specified a server socket factory, that factory's `createSocketImpl` method is called to create the actual socket implementation. Otherwise a "plain" socket is created.

**Parameters:**
`host` - the host name.
`port` - the port number.
**Throws:**
IOException - if an I/O error occurs when creating the socket.
**See Also:**
`setSocketImplFactory(java.net.SocketImplFactory)`, `SocketImpl`, `SocketImplFactory.createSocketImpl()`

---

### Socket

```
public Socket(InetAddress address,
              int port)
       throws IOException
```

Creates a stream socket and connects it to the specified port number at the specified IP address.

If the application has specified a socket factory, that factory's `createSocketImpl` method is called to create the actual socket implementation. Otherwise a "plain" socket is created.

**Parameters:**
`address` - the IP address.
`port` - the port number.
**Throws:**
IOException - if an I/O error occurs when creating the socket.
**See Also:**
`setSocketImplFactory(java.net.SocketImplFactory)`, `SocketImpl`, `SocketImplFactory.createSocketImpl()`

---

### Socket

```
public Socket(String host,
              int port,
              InetAddress localAddr,
              int localPort)
       throws IOException
```

Creates a socket and connects it to the specified remote host on the specified remote port. The Socket will also bind() to the local address and port supplied.

**Parameters:**

`host` - the name of the remote host<br>
`port` - the remote port<br>
`localAddr` - the local address the socket is bound to<br>
`localPort` - the local port the socket is bound to

**Since:**

JDK1.1

---

### Socket

```
public Socket(InetAddress address,
              int port,
              InetAddress localAddr,
              int localPort)
       throws IOException
```

Creates a socket and connects it to the specified remote address on the specified remote port. The Socket will also bind() to the local address and port supplied.

**Parameters:**

`address` - the remote address<br>
`port` - the remote port<br>
`localAddr` - the local address the socket is bound to<br>
`localPort` - the local port the socket is bound to

**Since:**

JDK1.1

---

### Socket

```
public Socket(String host,
              int port,
              boolean stream)
       throws IOException
```

**Deprecated.** *Use DatagramSocket instead for UDP transport.*
Creates a stream socket and connects it to the specified port number on the named host.

If the stream argument is `true`, this creates a stream socket. If the stream argument is `false`, it creates a datagram socket.

If the application has specified a server socket factory, that factory's `createSocketImpl` method is called to create the actual socket implementation. Otherwise a "plain" socket is created.

**Parameters:**

`host` - the host name.<br>
`port` - the port number.<br>
`stream` - a `boolean` indicating whether this is a stream socket or a datagram socket.

**Throws:**

IOException - if an I/O error occurs when creating the socket.

**See Also:**

setSocketImplFactory(java.net.SocketImplFactory), SocketImpl, SocketImplFactory.createSocketImpl()

---

### Socket

```
public Socket(InetAddress host,
              int port,
              boolean stream)
       throws IOException
```

**Deprecated.** *Use DatagramSocket instead for UDP transport.*
Creates a socket and connects it to the specified port number at the specified IP address.

If the stream argument is `true`, this creates a stream socket. If the stream argument is `false`, it creates a datagram socket.

If the application has specified a server socket factory, that factory's `createSocketImpl` method is called to create the actual socket implementation. Otherwise a "plain" socket is created.

**Parameters:**

`address` - the IP address.
`port` - the port number.
`stream` - if `true`, create a stream socket; otherwise, create a datagram socket.

**Throws:**

IOException - if an I/O error occurs when creating the socket.

**See Also:**

setSocketImplFactory(java.net.SocketImplFactory), SocketImpl, SocketImplFactory.createSocketImpl()

## Method Detail

### getInetAddress

```
public InetAddress getInetAddress()
```

Returns the address to which the socket is connected.

**Returns:**

the remote IP address to which this socket is connected.

---

### getLocalAddress

```
public InetAddress getLocalAddress()
```

Gets the local address to which the socket is bound.

**Since:**

JDK1.1

---

### getPort

```
public int getPort()
```

Returns the remote port to which this socket is connected.

**Returns:**
the remote port number to which this socket is connected.

---

### getLocalPort

```
public int getLocalPort()
```

Returns the local port to which this socket is bound.

**Returns:**
the local port number to which this socket is connected.

---

### getInputStream

```
public InputStream getInputStream()
                           throws IOException
```

Returns an input stream for this socket.

**Returns:**
an input stream for reading bytes from this socket.

**Throws:**
IOException - if an I/O error occurs when creating the input stream.

---

### getOutputStream

```
public OutputStream getOutputStream()
                             throws IOException
```

Returns an output stream for this socket.

**Returns:**
an output stream for writing bytes to this socket.

**Throws:**
IOException - if an I/O error occurs when creating the output stream.

---

### setTcpNoDelay

```
public void setTcpNoDelay(boolean on)
                   throws SocketException
```

Enable/disable TCP_NODELAY (disable/enable Nagle's algorithm).

**Since:**
JDK1.1

---

### getTcpNoDelay

```
public boolean getTcpNoDelay()
                      throws SocketException
```

Tests if TCP_NODELAY is enabled.

**Since:**
JDK1.1

---

### setSoLinger

```
public void setSoLinger(boolean on,
                        int linger)
                 throws SocketException
```

Enable/disable SO_LINGER with the specified linger time.

**Parameters:**
`on` - whether or not to linger on.
`linger` - how to linger for, if on is true.
**Throws:**
IllegalArgumentException - if the linger value is negative.
**Since:**
JDK1.1

---

### getSoLinger

```
public int getSoLinger()
                throws SocketException
```

Returns setting for SO_LINGER. -1 returns implies that the option is disabled.

**Since:**
JDK1.1

---

### setSoTimeout

```
public void setSoTimeout(int timeout)
                  throws SocketException
```

Enable/disable SO_TIMEOUT with the specified timeout, in milliseconds. With this option set to a non-zero timeout, a read() call on the InputStream associated with this Socket will block for only this amount of time. If the timeout expires, a **java.io.InterruptedIOException** is raised, though the Socket is still valid. The option **must** be enabled prior to entering the blocking operation to have effect. The timeout must be > 0. A timeout of zero is interpreted as an infinite timeout.

**Since:**
JDK 1.1

---

### getSoTimeout

```
public int getSoTimeout()
                 throws SocketException
```

Returns setting for SO_TIMEOUT. 0 returns implies that the option is disabled (i.e., timeout of infinity).

**Since:**
JDK1.1

---

### setSendBufferSize

```
public void setSendBufferSize(int size)
                       throws SocketException
```

Sets the SO_SNDBUF option to the specified value for this DatagramSocket. The SO_SNDBUF option is used by the platform's networking code as a hint for the size to use to allocate set the underlying network I/O buffers.

Increasing buffer size can increase the performance of network I/O for high-volume connection, while decreasing it can help reduce the backlog of incoming data. For UDP, this sets the maximum size of a packet that may be sent on this socket.

Because SO_SNDBUF is a hint, applications that want to verify what size the buffers were set to should call getSendBufferSize.

**Parameters:**

`size` - the size to which to set the send buffer size. This value must be greater than 0.

**Throws:**

IllegalArgumentException - if the value is 0 or is negative.

---

### getSendBufferSize

```
public int getSendBufferSize()
                      throws SocketException
```

Get value of the SO_SNDBUF option for this socket, that is the buffer size used by the platform for output on the this Socket.

**See Also:**

`setSendBufferSize(int)`

---

### setReceiveBufferSize

```
public void setReceiveBufferSize(int size)
                          throws SocketException
```

Sets the SO_RCVBUF option to the specified value for this DatagramSocket. The SO_RCVBUF option is used by the platform's networking code as a hint for the size to use to allocate set the underlying network I/O buffers.

Increasing buffer size can increase the performance of network I/O for high-volume connection, while decreasing it can help reduce the backlog of incoming data. For UDP, this sets the maximum size of a packet that may be sent on this socket.

Because SO_RCVBUF is a hint, applications that want to verify what size the buffers were set to should call getReceiveBufferSize.

**Parameters:**

`size` - the size to which to set the receive buffer size. This value must be greater than 0.

**Throws:**

IllegalArgumentException - if the value is 0 or is negative.

---

### getReceiveBufferSize

```
public int getReceiveBufferSize()
                         throws SocketException
```

Get value of the SO_RCVBUF option for this socket, that is the buffer size used by the platform for input on the this Socket.

**See Also:**

`setReceiveBufferSize(int)`

## close

```
public void close()
           throws IOException
```

Closes this socket.

**Throws:**

IOException - if an I/O error occurs when closing this socket.

## toString

```
public String toString()
```

Converts this socket to a `String`.

**Returns:**

a string representation of this socket.

**Overrides:**

toString in class Object

## setSocketImplFactory

```
public static void setSocketImplFactory(SocketImplFactory fac)
                                 throws IOException
```

Sets the client socket implementation factory for the application. The factory can be specified only once.

When an application creates a new client socket, the socket implementation factory's `createSocketImpl` method is called to create the actual socket implementation.

**Parameters:**

`fac` - the desired factory.

**Throws:**

IOException - if an I/O error occurs when setting the socket factory.

SocketException - if the factory is already defined.

**See Also:**

`SocketImplFactory.createSocketImpl()`

Submit a bug or feature
Submit comments/suggestions about new javadoc look

This documentation was generated with a post-Beta4 version of Javadoc.