| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SSLSocket.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/net/ssl/SSLSessionContext.html)   [**NEXT CLASS**](http://docs.google.com/javax/net/ssl/SSLSocketFactory.html) | [**FRAMES**](http://docs.google.com/index.html?javax/net/ssl/SSLSocket.html)    [**NO FRAMES**](http://docs.google.com/SSLSocket.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#1t3h5sf) | [METHOD](#lnxbz9) |

## **javax.net.ssl**

Class SSLSocket

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.net.Socket](http://docs.google.com/java/net/Socket.html)  
 **javax.net.ssl.SSLSocket**

public abstract class **SSLSocket**extends [Socket](http://docs.google.com/java/net/Socket.html)

This class extends Sockets and provides secure socket using protocols such as the "Secure Sockets Layer" (SSL) or IETF "Transport Layer Security" (TLS) protocols.

Such sockets are normal stream sockets, but they add a layer of security protections over the underlying network transport protocol, such as TCP. Those protections include:

* *Integrity Protection*. SSL protects against modification of messages by an active wiretapper.
* *Authentication*. In most modes, SSL provides peer authentication. Servers are usually authenticated, and clients may be authenticated as requested by servers.
* *Confidentiality (Privacy Protection)*. In most modes, SSL encrypts data being sent between client and server. This protects the confidentiality of data, so that passive wiretappers won't see sensitive data such as financial information or personal information of many kinds.

These kinds of protection are specified by a "cipher suite", which is a combination of cryptographic algorithms used by a given SSL connection. During the negotiation process, the two endpoints must agree on a ciphersuite that is available in both environments. If there is no such suite in common, no SSL connection can be established, and no data can be exchanged.

The cipher suite used is established by a negotiation process called "handshaking". The goal of this process is to create or rejoin a "session", which may protect many connections over time. After handshaking has completed, you can access session attributes by using the *getSession* method. The initial handshake on this connection can be initiated in one of three ways:

* calling startHandshake which explicitly begins handshakes, or
* any attempt to read or write application data on this socket causes an implicit handshake, or
* a call to getSession tries to set up a session if there is no currently valid session, and an implicit handshake is done.

If handshaking fails for any reason, the SSLSocket is closed, and no futher communications can be done.

There are two groups of cipher suites which you will need to know about when managing cipher suites:

* *Supported* cipher suites: all the suites which are supported by the SSL implementation. This list is reported using *getSupportedCipherSuites*.
* *Enabled* cipher suites, which may be fewer than the full set of supported suites. This group is set using the *setEnabledCipherSuites* method, and queried using the *getEnabledCipherSuites* method. Initially, a default set of cipher suites will be enabled on a new socket that represents the minimum suggested configuration.

Implementation defaults require that only cipher suites which authenticate servers and provide confidentiality be enabled by default. Only if both sides explicitly agree to unauthenticated and/or non-private (unencrypted) communications will such a ciphersuite be selected.

When SSLSockets are first created, no handshaking is done so that applications may first set their communication preferences: what cipher suites to use, whether the socket should be in client or server mode, etc. However, security is always provided by the time that application data is sent over the connection.

You may register to receive event notification of handshake completion. This involves the use of two additional classes. *HandshakeCompletedEvent* objects are passed to *HandshakeCompletedListener* instances, which are registered by users of this API. SSLSockets are created by SSLSocketFactorys, or by accepting a connection from a SSLServerSocket.

A SSL socket must choose to operate in the client or server mode. This will determine who begins the handshaking process, as well as which messages should be sent by each party. Each connection must have one client and one server, or handshaking will not progress properly. Once the initial handshaking has started, a socket can not switch between client and server modes, even when performing renegotiations.

**Since:** 1.4 **See Also:**[Socket](http://docs.google.com/java/net/Socket.html), [SSLServerSocket](http://docs.google.com/javax/net/ssl/SSLServerSocket.html), [SSLSocketFactory](http://docs.google.com/javax/net/ssl/SSLSocketFactory.html)

| **Constructor Summary** | |
| --- | --- |
| protected | [**SSLSocket**](http://docs.google.com/javax/net/ssl/SSLSocket.html#SSLSocket())()            Used only by subclasses. |
| protected | [**SSLSocket**](http://docs.google.com/javax/net/ssl/SSLSocket.html#SSLSocket(java.net.InetAddress,%20int))([InetAddress](http://docs.google.com/java/net/InetAddress.html) address, int port)            Used only by subclasses. |
| protected | [**SSLSocket**](http://docs.google.com/javax/net/ssl/SSLSocket.html#SSLSocket(java.net.InetAddress,%20int,%20java.net.InetAddress,%20int))([InetAddress](http://docs.google.com/java/net/InetAddress.html) address, int port, [InetAddress](http://docs.google.com/java/net/InetAddress.html) clientAddress, int clientPort)            Used only by subclasses. |
| protected | [**SSLSocket**](http://docs.google.com/javax/net/ssl/SSLSocket.html#SSLSocket(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) host, int port)            Used only by subclasses. |
| protected | [**SSLSocket**](http://docs.google.com/javax/net/ssl/SSLSocket.html#SSLSocket(java.lang.String,%20int,%20java.net.InetAddress,%20int))([String](http://docs.google.com/java/lang/String.html) host, int port, [InetAddress](http://docs.google.com/java/net/InetAddress.html) clientAddress, int clientPort)            Used only by subclasses. |

| **Method Summary** | |
| --- | --- |
| abstract  void | [**addHandshakeCompletedListener**](http://docs.google.com/javax/net/ssl/SSLSocket.html#addHandshakeCompletedListener(javax.net.ssl.HandshakeCompletedListener))([HandshakeCompletedListener](http://docs.google.com/javax/net/ssl/HandshakeCompletedListener.html) listener)            Registers an event listener to receive notifications that an SSL handshake has completed on this connection. |
| abstract  [String](http://docs.google.com/java/lang/String.html)[] | [**getEnabledCipherSuites**](http://docs.google.com/javax/net/ssl/SSLSocket.html#getEnabledCipherSuites())()            Returns the names of the SSL cipher suites which are currently enabled for use on this connection. |
| abstract  [String](http://docs.google.com/java/lang/String.html)[] | [**getEnabledProtocols**](http://docs.google.com/javax/net/ssl/SSLSocket.html#getEnabledProtocols())()            Returns the names of the protocol versions which are currently enabled for use on this connection. |
| abstract  boolean | [**getEnableSessionCreation**](http://docs.google.com/javax/net/ssl/SSLSocket.html#getEnableSessionCreation())()            Returns true if new SSL sessions may be established by this socket. |
| abstract  boolean | [**getNeedClientAuth**](http://docs.google.com/javax/net/ssl/SSLSocket.html#getNeedClientAuth())()            Returns true if the socket will *require* client authentication. |
| abstract  [SSLSession](http://docs.google.com/javax/net/ssl/SSLSession.html) | [**getSession**](http://docs.google.com/javax/net/ssl/SSLSocket.html#getSession())()            Returns the SSL Session in use by this connection. |
| [SSLParameters](http://docs.google.com/javax/net/ssl/SSLParameters.html) | [**getSSLParameters**](http://docs.google.com/javax/net/ssl/SSLSocket.html#getSSLParameters())()            Returns the SSLParameters in effect for this SSLSocket. |
| abstract  [String](http://docs.google.com/java/lang/String.html)[] | [**getSupportedCipherSuites**](http://docs.google.com/javax/net/ssl/SSLSocket.html#getSupportedCipherSuites())()            Returns the names of the cipher suites which could be enabled for use on this connection. |
| abstract  [String](http://docs.google.com/java/lang/String.html)[] | [**getSupportedProtocols**](http://docs.google.com/javax/net/ssl/SSLSocket.html#getSupportedProtocols())()            Returns the names of the protocols which could be enabled for use on an SSL connection. |
| abstract  boolean | [**getUseClientMode**](http://docs.google.com/javax/net/ssl/SSLSocket.html#getUseClientMode())()            Returns true if the socket is set to use client mode when handshaking. |
| abstract  boolean | [**getWantClientAuth**](http://docs.google.com/javax/net/ssl/SSLSocket.html#getWantClientAuth())()            Returns true if the socket will *request* client authentication. |
| abstract  void | [**removeHandshakeCompletedListener**](http://docs.google.com/javax/net/ssl/SSLSocket.html#removeHandshakeCompletedListener(javax.net.ssl.HandshakeCompletedListener))([HandshakeCompletedListener](http://docs.google.com/javax/net/ssl/HandshakeCompletedListener.html) listener)            Removes a previously registered handshake completion listener. |
| abstract  void | [**setEnabledCipherSuites**](http://docs.google.com/javax/net/ssl/SSLSocket.html#setEnabledCipherSuites(java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html)[] suites)            Sets the cipher suites enabled for use on this connection. |
| abstract  void | [**setEnabledProtocols**](http://docs.google.com/javax/net/ssl/SSLSocket.html#setEnabledProtocols(java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html)[] protocols)            Sets the protocol versions enabled for use on this connection. |
| abstract  void | [**setEnableSessionCreation**](http://docs.google.com/javax/net/ssl/SSLSocket.html#setEnableSessionCreation(boolean))(boolean flag)            Controls whether new SSL sessions may be established by this socket. |
| abstract  void | [**setNeedClientAuth**](http://docs.google.com/javax/net/ssl/SSLSocket.html#setNeedClientAuth(boolean))(boolean need)            Configures the socket to *require* client authentication. |
| void | [**setSSLParameters**](http://docs.google.com/javax/net/ssl/SSLSocket.html#setSSLParameters(javax.net.ssl.SSLParameters))([SSLParameters](http://docs.google.com/javax/net/ssl/SSLParameters.html) params)            Applies SSLParameters to this socket. |
| abstract  void | [**setUseClientMode**](http://docs.google.com/javax/net/ssl/SSLSocket.html#setUseClientMode(boolean))(boolean mode)            Configures the socket to use client (or server) mode when handshaking. |
| abstract  void | [**setWantClientAuth**](http://docs.google.com/javax/net/ssl/SSLSocket.html#setWantClientAuth(boolean))(boolean want)            Configures the socket to *request* client authentication. |
| abstract  void | [**startHandshake**](http://docs.google.com/javax/net/ssl/SSLSocket.html#startHandshake())()            Starts an SSL handshake on this connection. |

| **Methods inherited from class java.net.**[**Socket**](http://docs.google.com/java/net/Socket.html) |
| --- |
| [bind](http://docs.google.com/java/net/Socket.html#bind(java.net.SocketAddress)), [close](http://docs.google.com/java/net/Socket.html#close()), [connect](http://docs.google.com/java/net/Socket.html#connect(java.net.SocketAddress)), [connect](http://docs.google.com/java/net/Socket.html#connect(java.net.SocketAddress,%20int)), [getChannel](http://docs.google.com/java/net/Socket.html#getChannel()), [getInetAddress](http://docs.google.com/java/net/Socket.html#getInetAddress()), [getInputStream](http://docs.google.com/java/net/Socket.html#getInputStream()), [getKeepAlive](http://docs.google.com/java/net/Socket.html#getKeepAlive()), [getLocalAddress](http://docs.google.com/java/net/Socket.html#getLocalAddress()), [getLocalPort](http://docs.google.com/java/net/Socket.html#getLocalPort()), [getLocalSocketAddress](http://docs.google.com/java/net/Socket.html#getLocalSocketAddress()), [getOOBInline](http://docs.google.com/java/net/Socket.html#getOOBInline()), [getOutputStream](http://docs.google.com/java/net/Socket.html#getOutputStream()), [getPort](http://docs.google.com/java/net/Socket.html#getPort()), [getReceiveBufferSize](http://docs.google.com/java/net/Socket.html#getReceiveBufferSize()), [getRemoteSocketAddress](http://docs.google.com/java/net/Socket.html#getRemoteSocketAddress()), [getReuseAddress](http://docs.google.com/java/net/Socket.html#getReuseAddress()), [getSendBufferSize](http://docs.google.com/java/net/Socket.html#getSendBufferSize()), [getSoLinger](http://docs.google.com/java/net/Socket.html#getSoLinger()), [getSoTimeout](http://docs.google.com/java/net/Socket.html#getSoTimeout()), [getTcpNoDelay](http://docs.google.com/java/net/Socket.html#getTcpNoDelay()), [getTrafficClass](http://docs.google.com/java/net/Socket.html#getTrafficClass()), [isBound](http://docs.google.com/java/net/Socket.html#isBound()), [isClosed](http://docs.google.com/java/net/Socket.html#isClosed()), [isConnected](http://docs.google.com/java/net/Socket.html#isConnected()), [isInputShutdown](http://docs.google.com/java/net/Socket.html#isInputShutdown()), [isOutputShutdown](http://docs.google.com/java/net/Socket.html#isOutputShutdown()), [sendUrgentData](http://docs.google.com/java/net/Socket.html#sendUrgentData(int)), [setKeepAlive](http://docs.google.com/java/net/Socket.html#setKeepAlive(boolean)), [setOOBInline](http://docs.google.com/java/net/Socket.html#setOOBInline(boolean)), [setPerformancePreferences](http://docs.google.com/java/net/Socket.html#setPerformancePreferences(int,%20int,%20int)), [setReceiveBufferSize](http://docs.google.com/java/net/Socket.html#setReceiveBufferSize(int)), [setReuseAddress](http://docs.google.com/java/net/Socket.html#setReuseAddress(boolean)), [setSendBufferSize](http://docs.google.com/java/net/Socket.html#setSendBufferSize(int)), [setSocketImplFactory](http://docs.google.com/java/net/Socket.html#setSocketImplFactory(java.net.SocketImplFactory)), [setSoLinger](http://docs.google.com/java/net/Socket.html#setSoLinger(boolean,%20int)), [setSoTimeout](http://docs.google.com/java/net/Socket.html#setSoTimeout(int)), [setTcpNoDelay](http://docs.google.com/java/net/Socket.html#setTcpNoDelay(boolean)), [setTrafficClass](http://docs.google.com/java/net/Socket.html#setTrafficClass(int)), [shutdownInput](http://docs.google.com/java/net/Socket.html#shutdownInput()), [shutdownOutput](http://docs.google.com/java/net/Socket.html#shutdownOutput()), [toString](http://docs.google.com/java/net/Socket.html#toString()) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Constructor Detail** |
| --- |

### SSLSocket

protected **SSLSocket**()

Used only by subclasses. Constructs an uninitialized, unconnected TCP socket.

### SSLSocket

protected **SSLSocket**([String](http://docs.google.com/java/lang/String.html) host,  
 int port)  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [UnknownHostException](http://docs.google.com/java/net/UnknownHostException.html)

Used only by subclasses. Constructs a TCP connection to a named host at a specified port. This acts as the SSL client.

**Parameters:**host - name of the host with which to connectport - number of the server's port **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when creating the socket [UnknownHostException](http://docs.google.com/java/net/UnknownHostException.html) - if the host is not known

### SSLSocket

protected **SSLSocket**([InetAddress](http://docs.google.com/java/net/InetAddress.html) address,  
 int port)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Used only by subclasses. Constructs a TCP connection to a server at a specified address and port. This acts as the SSL client.

**Parameters:**address - the server's hostport - its port **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when creating the socket

### SSLSocket

protected **SSLSocket**([String](http://docs.google.com/java/lang/String.html) host,  
 int port,  
 [InetAddress](http://docs.google.com/java/net/InetAddress.html) clientAddress,  
 int clientPort)  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [UnknownHostException](http://docs.google.com/java/net/UnknownHostException.html)

Used only by subclasses. Constructs an SSL connection to a named host at a specified port, binding the client side of the connection a given address and port. This acts as the SSL client.

**Parameters:**host - name of the host with which to connectport - number of the server's portclientAddress - the client's hostclientPort - number of the client's port **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when creating the socket [UnknownHostException](http://docs.google.com/java/net/UnknownHostException.html) - if the host is not known

### SSLSocket

protected **SSLSocket**([InetAddress](http://docs.google.com/java/net/InetAddress.html) address,  
 int port,  
 [InetAddress](http://docs.google.com/java/net/InetAddress.html) clientAddress,  
 int clientPort)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Used only by subclasses. Constructs an SSL connection to a server at a specified address and TCP port, binding the client side of the connection a given address and port. This acts as the SSL client.

**Parameters:**address - the server's hostport - its portclientAddress - the client's hostclientPort - number of the client's port **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs when creating the socket

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

### getSupportedCipherSuites

public abstract [String](http://docs.google.com/java/lang/String.html)[] **getSupportedCipherSuites**()

Returns the names of the cipher suites which could be enabled for use on this connection. Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites might be useful in specialized applications.

**Returns:**an array of cipher suite names**See Also:**[getEnabledCipherSuites()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getEnabledCipherSuites()), [setEnabledCipherSuites(String [])](http://docs.google.com/javax/net/ssl/SSLSocket.html#setEnabledCipherSuites(java.lang.String%5B%5D))

### getEnabledCipherSuites

public abstract [String](http://docs.google.com/java/lang/String.html)[] **getEnabledCipherSuites**()

Returns the names of the SSL cipher suites which are currently enabled for use on this connection. When an SSLSocket is first created, all enabled cipher suites support a minimum quality of service. Thus, in some environments this value might be empty.

Even if a suite has been enabled, it might never be used. (For example, the peer does not support it, the requisite certificates (and private keys) for the suite are not available, or an anonymous suite is enabled but authentication is required.

**Returns:**an array of cipher suite names**See Also:**[getSupportedCipherSuites()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getSupportedCipherSuites()), [setEnabledCipherSuites(String [])](http://docs.google.com/javax/net/ssl/SSLSocket.html#setEnabledCipherSuites(java.lang.String%5B%5D))

### setEnabledCipherSuites

public abstract void **setEnabledCipherSuites**([String](http://docs.google.com/java/lang/String.html)[] suites)

Sets the cipher suites enabled for use on this connection.

Each cipher suite in the suites parameter must have been listed by getSupportedCipherSuites(), or the method will fail. Following a successful call to this method, only suites listed in the suites parameter are enabled for use.

See [getEnabledCipherSuites()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getEnabledCipherSuites()) for more information on why a specific ciphersuite may never be used on a connection.

**Parameters:**suites - Names of all the cipher suites to enable **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - when one or more of the ciphers named by the parameter is not supported, or when the parameter is null.**See Also:**[getSupportedCipherSuites()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getSupportedCipherSuites()), [getEnabledCipherSuites()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getEnabledCipherSuites())

### getSupportedProtocols

public abstract [String](http://docs.google.com/java/lang/String.html)[] **getSupportedProtocols**()

Returns the names of the protocols which could be enabled for use on an SSL connection.

**Returns:**an array of protocols supported

### getEnabledProtocols

public abstract [String](http://docs.google.com/java/lang/String.html)[] **getEnabledProtocols**()

Returns the names of the protocol versions which are currently enabled for use on this connection.

**Returns:**an array of protocols**See Also:**[setEnabledProtocols(String [])](http://docs.google.com/javax/net/ssl/SSLSocket.html#setEnabledProtocols(java.lang.String%5B%5D))

### setEnabledProtocols

public abstract void **setEnabledProtocols**([String](http://docs.google.com/java/lang/String.html)[] protocols)

Sets the protocol versions enabled for use on this connection.

The protocols must have been listed by getSupportedProtocols() as being supported. Following a successful call to this method, only protocols listed in the protocols parameter are enabled for use.

**Parameters:**protocols - Names of all the protocols to enable. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - when one or more of the protocols named by the parameter is not supported or when the protocols parameter is null.**See Also:**[getEnabledProtocols()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getEnabledProtocols())

### getSession

public abstract [SSLSession](http://docs.google.com/javax/net/ssl/SSLSession.html) **getSession**()

Returns the SSL Session in use by this connection. These can be long lived, and frequently correspond to an entire login session for some user. The session specifies a particular cipher suite which is being actively used by all connections in that session, as well as the identities of the session's client and server.

This method will initiate the initial handshake if necessary and then block until the handshake has been established.

If an error occurs during the initial handshake, this method returns an invalid session object which reports an invalid cipher suite of "SSL\_NULL\_WITH\_NULL\_NULL".

**Returns:**the SSLSession

### addHandshakeCompletedListener

public abstract void **addHandshakeCompletedListener**([HandshakeCompletedListener](http://docs.google.com/javax/net/ssl/HandshakeCompletedListener.html) listener)

Registers an event listener to receive notifications that an SSL handshake has completed on this connection.

**Parameters:**listener - the HandShake Completed event listener **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the argument is null.**See Also:**[startHandshake()](http://docs.google.com/javax/net/ssl/SSLSocket.html#startHandshake()), [removeHandshakeCompletedListener(HandshakeCompletedListener)](http://docs.google.com/javax/net/ssl/SSLSocket.html#removeHandshakeCompletedListener(javax.net.ssl.HandshakeCompletedListener))

### removeHandshakeCompletedListener

public abstract void **removeHandshakeCompletedListener**([HandshakeCompletedListener](http://docs.google.com/javax/net/ssl/HandshakeCompletedListener.html) listener)

Removes a previously registered handshake completion listener.

**Parameters:**listener - the HandShake Completed event listener **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the listener is not registered, or the argument is null.**See Also:**[addHandshakeCompletedListener(HandshakeCompletedListener)](http://docs.google.com/javax/net/ssl/SSLSocket.html#addHandshakeCompletedListener(javax.net.ssl.HandshakeCompletedListener))

### startHandshake

public abstract void **startHandshake**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Starts an SSL handshake on this connection. Common reasons include a need to use new encryption keys, to change cipher suites, or to initiate a new session. To force complete reauthentication, the current session could be invalidated before starting this handshake.

If data has already been sent on the connection, it continues to flow during this handshake. When the handshake completes, this will be signaled with an event. This method is synchronous for the initial handshake on a connection and returns when the negotiated handshake is complete. Some protocols may not support multiple handshakes on an existing socket and may throw an IOException.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - on a network level error**See Also:**[addHandshakeCompletedListener(HandshakeCompletedListener)](http://docs.google.com/javax/net/ssl/SSLSocket.html#addHandshakeCompletedListener(javax.net.ssl.HandshakeCompletedListener))

### setUseClientMode

public abstract void **setUseClientMode**(boolean mode)

Configures the socket to use client (or server) mode when handshaking.

This method must be called before any handshaking occurs. Once handshaking has begun, the mode can not be reset for the life of this socket.

Servers normally authenticate themselves, and clients are not required to do so.

**Parameters:**mode - true if the socket should start its handshaking in "client" mode **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if a mode change is attempted after the initial handshake has begun.**See Also:**[getUseClientMode()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getUseClientMode())

### getUseClientMode

public abstract boolean **getUseClientMode**()

Returns true if the socket is set to use client mode when handshaking.

**Returns:**true if the socket should do handshaking in "client" mode**See Also:**[setUseClientMode(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setUseClientMode(boolean))

### setNeedClientAuth

public abstract void **setNeedClientAuth**(boolean need)

Configures the socket to *require* client authentication. This option is only useful for sockets in the server mode.

A socket's client authentication setting is one of the following:

* client authentication required
* client authentication requested
* no client authentication desired

Unlike [setWantClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setWantClientAuth(boolean)), if this option is set and the client chooses not to provide authentication information about itself, *the negotiations will stop and the connection will be dropped*.

Calling this method overrides any previous setting made by this method or [setWantClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setWantClientAuth(boolean)).

**Parameters:**need - set to true if client authentication is required, or false if no client authentication is desired.**See Also:**[getNeedClientAuth()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getNeedClientAuth()), [setWantClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setWantClientAuth(boolean)), [getWantClientAuth()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getWantClientAuth()), [setUseClientMode(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setUseClientMode(boolean))

### getNeedClientAuth

public abstract boolean **getNeedClientAuth**()

Returns true if the socket will *require* client authentication. This option is only useful to sockets in the server mode.

**Returns:**true if client authentication is required, or false if no client authentication is desired.**See Also:**[setNeedClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setNeedClientAuth(boolean)), [setWantClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setWantClientAuth(boolean)), [getWantClientAuth()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getWantClientAuth()), [setUseClientMode(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setUseClientMode(boolean))

### setWantClientAuth

public abstract void **setWantClientAuth**(boolean want)

Configures the socket to *request* client authentication. This option is only useful for sockets in the server mode.

A socket's client authentication setting is one of the following:

* client authentication required
* client authentication requested
* no client authentication desired

Unlike [setNeedClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setNeedClientAuth(boolean)), if this option is set and the client chooses not to provide authentication information about itself, *the negotiations will continue*.

Calling this method overrides any previous setting made by this method or [setNeedClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setNeedClientAuth(boolean)).

**Parameters:**want - set to true if client authentication is requested, or false if no client authentication is desired.**See Also:**[getWantClientAuth()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getWantClientAuth()), [setNeedClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setNeedClientAuth(boolean)), [getNeedClientAuth()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getNeedClientAuth()), [setUseClientMode(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setUseClientMode(boolean))

### getWantClientAuth

public abstract boolean **getWantClientAuth**()

Returns true if the socket will *request* client authentication. This option is only useful for sockets in the server mode.

**Returns:**true if client authentication is requested, or false if no client authentication is desired.**See Also:**[setNeedClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setNeedClientAuth(boolean)), [getNeedClientAuth()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getNeedClientAuth()), [setWantClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setWantClientAuth(boolean)), [setUseClientMode(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setUseClientMode(boolean))

### setEnableSessionCreation

public abstract void **setEnableSessionCreation**(boolean flag)

Controls whether new SSL sessions may be established by this socket. If session creations are not allowed, and there are no existing sessions to resume, there will be no successful handshaking.

**Parameters:**flag - true indicates that sessions may be created; this is the default. false indicates that an existing session must be resumed**See Also:**[getEnableSessionCreation()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getEnableSessionCreation())

### getEnableSessionCreation

public abstract boolean **getEnableSessionCreation**()

Returns true if new SSL sessions may be established by this socket.

**Returns:**true indicates that sessions may be created; this is the default. false indicates that an existing session must be resumed**See Also:**[setEnableSessionCreation(boolean)](http://docs.google.com/javax/net/ssl/SSLSocket.html#setEnableSessionCreation(boolean))

### getSSLParameters

public [SSLParameters](http://docs.google.com/javax/net/ssl/SSLParameters.html) **getSSLParameters**()

Returns the SSLParameters in effect for this SSLSocket. The ciphersuites and protocols of the returned SSLParameters are always non-null.

**Returns:**the SSLParameters in effect for this SSLSocket.**Since:** 1.6

### setSSLParameters

public void **setSSLParameters**([SSLParameters](http://docs.google.com/javax/net/ssl/SSLParameters.html) params)

Applies SSLParameters to this socket.

This means:

* if params.getCipherSuites() is non-null, setEnabledCipherSuites() is called with that value
* if params.getProtocols() is non-null, setEnabledProtocols() is called with that value
* if params.getNeedClientAuth() or params.getWantClientAuth() return true, setNeedClientAuth(true) and setWantClientAuth(true) are called, respectively; otherwise setWantClientAuth(false) is called.

**Parameters:**params - the parameters **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the setEnabledCipherSuites() or the setEnabledProtocols() call fails**Since:** 1.6

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SSLSocket.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/net/ssl/SSLSessionContext.html)   [**NEXT CLASS**](http://docs.google.com/javax/net/ssl/SSLSocketFactory.html) | [**FRAMES**](http://docs.google.com/index.html?javax/net/ssl/SSLSocket.html)    [**NO FRAMES**](http://docs.google.com/SSLSocket.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#1t3h5sf) | [METHOD](#lnxbz9) |

[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).