Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

SSLSocket

```
public abstract class SSLSocket
extends Socket (/reference/java/net/Socket)
```

java.lang.Object (/reference/java/lang/Object)
↳ java.net.Socket (/reference/java/net/Socket)
↳ javax.net.ssl.SSLSocket

This class extends `Socket`s 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 further 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 `SSLSocket`s 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. `SSLSocket`s are created by `SSLSocketFactory`s, or by `accept`ing 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.

Default configuration for different Android versions

`SSLSocket` instances obtained from default `SSLSocketFactory` (/reference/javax/net/ssl/SSLSocketFactory), `SSLServerSocketFactory` (/reference/javax/net/ssl/SSLServerSocketFactory), and `SSLContext` (/reference/javax/net/ssl/SSLContext) are configured as follows:

Protocols

Client socket:

Protocol	Supported (API Levels)	Enabled by default (API Levels)
SSLv3[1][2][3]		
TLSv1	1+	1+
TLSv1.1	16+	20+
TLSv1.2	16+	20+
TLSv1.3	29+	29+

Server socket:

Protocol	Supported (API Levels)	Enabled by default (API Levels)
SSLv31–251–22		
TLSv1	1+	1+
TLSv1.1	16+	16+
TLSv1.2	16+	16+
TLSv1.3	29+	29+

Cipher suites

Methods that operate with cipher suite names (for example, `getSupportedCipherSuites` (/reference/javax/net/ssl/SSLSocket#getSupportedCipherSuites()), `setEnabledCipherSuites` (/reference/javax/net/ssl/SSLSocket#setEnabledCipherSuites(java.lang.String[]))) have used standard names for cipher suites since API Level 9, as listed in the table below. Prior to API Level 9, non-standard (OpenSSL) names had been used (see the table following this table).

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA9-229-19		

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA9-229-19		
SSL_DHE_DSS_WITH_DES_CBC_SHA9-229-19		
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA9-229-19		
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA9-229-19		
SSL_DHE_RSA_WITH_DES_CBC_SHA9-229-19		
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA9-22		
SSL_DH_anon_EXPORT_WITH_RC4_40_MD59-22		

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA9-22		
SSL_DH_anon_WITH_DES_CBC_SHA9-22		
SSL_DH_anon_WITH_RC4_128_MD59-22		
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA9-229-19		
SSL_RSA_EXPORT_WITH_RC4_40_MD59-229-19		
SSL_RSA_WITH_3DES_EDE_CBC_SHA	9+	9-19
SSL_RSA_WITH_DES_CBC_SHA9-229-19		

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
SSL_RSA_WITH_NULL_MD59-22		
SSL_RSA_WITH_NULL_SHA9-22		
SSL_RSA_WITH_RC4_128_MD59-259-19		
SSL_RSA_WITH_RC4_128_SHA9-259-23		
TLS_AES_128_GCM_SHA256	29+	29+
TLS_AES_256_GCM_SHA384	29+	29+
TLS_CHACHA20_POLY1305_SHA256	29+	29+
TLS_DHE_DSS_WITH_AES_128_CBC_SHA9-229-22		

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
TLS_DHE_DSS_WITH_AES_128_CBC_SHA25620-22		
TLS_DHE_DSS_WITH_AES_128_GCM_SHA25620-22		
TLS_DHE_DSS_WITH_AES_256_CBC_SHA9-2211-22		
TLS_DHE_DSS_WITH_AES_256_CBC_SHA25620-22		
TLS_DHE_DSS_WITH_AES_256_GCM_SHA38420-22		
TLS_DHE_RSA_WITH_AES_128_CBC_SHA9-259-25		
TLS_DHE_RSA_WITH_AES_128_CBC_SHA25620-25		

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
TLS_DHE_RSA_WITH_AES_128_GCM_SHA25620-2520-25		
TLS_DHE_RSA_WITH_AES_256_CBC_SHA9-2511-25		
TLS_DHE_RSA_WITH_AES_256_CBC_SHA25620-25		
TLS_DHE_RSA_WITH_AES_256_GCM_SHA38420-2520-25		
TLS_DH_anon_WITH_AES_128_CBC_SHA9-22		
TLS_DH_anon_WITH_AES_128_CBC_SHA25620-22		
TLS_DH_anon_WITH_AES_128_GCM_SHA25620-22		

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
TLS_DH_anon_WITH_AES_256_CBC_SHA9-22		
TLS_DH_anon_WITH_AES_256_CBC_SHA25620-22		
TLS_DH_anon_WITH_AES_256_GCM_SHA38420-22		
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA11-2211-19		
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	11+	11+
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA25620-28		
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	20+	20+
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	11+	11+

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA38420-28		
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	20+	20+
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA25624+		24+
TLS_ECDHE_ECDSA_WITH_NULL_SHA11-22		
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA11-2511-23		
TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA	21+	21+
TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA	21+	21+
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	24+	24+
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA11-2211-19		

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	11+	11+
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA25620-28		
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	20+	20+
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	11+	11+
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA38420-28		
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	20+	20+
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	24+	24+
TLS_ECDHE_RSA_WITH_NULL_SHA11-22		
TLS_ECDHE_RSA_WITH_RC4_128_SHA11-2511-23		

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA11-2211-19		
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA11-2211-19		
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA25620-22		
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA25620-22		
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA11-2211-19		
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA38420-22		
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA38420-22		

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
TLS_ECDH_ECDSA_WITH_NULL_SHA11-22		
TLS_ECDH_ECDSA_WITH_RC4_128_SHA11-2211-19		
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA11-2211-19		
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA11-2211-19		
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA25620-22		
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA25620-22		
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA11-2211-19		

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA38420-22		
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA38420-22		
TLS_ECDH_RSA_WITH_NULL_SHA11-22		
TLS_ECDH_RSA_WITH_RC4_128_SHA11-2211-19		
TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA11-22		
TLS_ECDH_anon_WITH_AES_128_CBC_SHA11-22		
TLS_ECDH_anon_WITH_AES_256_CBC_SHA11-22		

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
TLS_ECDH_anon_WITH_NULL_SHA11-22		
TLS_ECDH_anon_WITH_RC4_128_SHA11-22		
TLS_EMPTY_RENEGOTIATION_INFO_SCSV	11+	11+
TLS_FALLBACK_SCSV	21+	
TLS_PSK_WITH_3DES_EDE_CBC_SHA21-22		
TLS_PSK_WITH_AES_128_CBC_SHA	21+	21+
TLS_PSK_WITH_AES_256_CBC_SHA	21+	21+
TLS_PSK_WITH_RC4_128_SHA21-25		
TLS_RSA_WITH_AES_128_CBC_SHA	9+	9+

Cipher suite	Supported (API Levels)	Enabled by default (API Levels)
TLS_RSA_WITH_AES_128_CBC_SHA25620-28		
TLS_RSA_WITH_AES_128_GCM_SHA256	20+	20+
TLS_RSA_WITH_AES_256_CBC_SHA	9+	11+
TLS_RSA_WITH_AES_256_CBC_SHA25620-28		
TLS_RSA_WITH_AES_256_GCM_SHA384	20+	20+
TLS_RSA_WITH_NULL_SHA25620-22		

NOTE: PSK cipher suites are enabled by default only if the `SSLContext` through which the socket was created has been initialized with a `PSKKeyManager`.

API Levels 1 to 8 use OpenSSL names for cipher suites. The table below lists these OpenSSL names and their corresponding standard names used in API Levels 9 and newer.

OpenSSL cipher suite	Standard cipher suite	Su (Al Le
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA	1+
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA	1+
DES-CBC-MD5	SSL_CK_DES_64_CBC_WITH_MD5	1–8
DES-CBC-SHA	*SSL_RSA_WITH_DES_CBC_SHA*	*1–22*
DES-CBC3-MD5	SSL_CK_DES_192_EDE3_CBC_WITH_MD5	1–8
DES-CBC3-SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA	1+
DHE-DSS-AES128-SHA	*TLS_DHE_DSS_WITH_AES_128_CBC_SHA*	*1–22*
DHE-DSS-AES256-SHA	*TLS_DHE_DSS_WITH_AES_256_CBC_SHA*	*1–22*

OpenSSL cipher suite	Standard cipher suite	Su (Al Le
DHE-RSA-AES128-SHA	TLS_DHE_RSA_WITH_AES_128_CBC_SHA	1+
DHE-RSA-AES256-SHA	TLS_DHE_RSA_WITH_AES_256_CBC_SHA	1+
EDH-DSS-DES-CBC-SHA	*SSL_DHE_DSS_WITH_DES_CBC_SHA*	*1–22*	*1–19*
EDH-DSS-DES-CBC3-SHA	*SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA*	*1–22*	*1–19*
EDH-RSA-DES-CBC-SHA	*SSL_DHE_RSA_WITH_DES_CBC_SHA*	*1–22*	*1–19*
EDH-RSA-DES-CBC3-SHA	*SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA*	*1–22*	*1–19*

OpenSSL cipher suite	Standard cipher suite	Su (Al Le
EXP-DES-CBC-SHA	*SSL_RSA_EXPORT_WITH_DES40_CBC_SHA*	*1–22* *1–19*
EXP-EDH-DSS-DES-CBC-SHA	*SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA*	*1–22* *19*
EXP-EDH-RSA-DES-CBC-SHA	*SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA*	*1–22* *19*
EXP-RC2-CBC-MD5	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	1–8

OpenSSL cipher suite	Standard cipher suite	Su (Al Le
EXP-RC4-MD5	*SSL_RSA_EXPORT_WITH_RC4_40_MD5*	*1–22*	*1–19*
RC2-CBC-MD5	SSL_CK_RC2_128_CBC_WITH_MD5	1–8
RC4-MD5	*SSL_RSA_WITH_RC4_128_MD5*	*1–25*	*1–19*
RC4-SHA	*SSL_RSA_WITH_RC4_128_SHA*	*1–25*	*1–23*

See also:

`Socket` (/reference/java/net/Socket)

`SSLServerSocket` (/reference/javax/net/ssl/SSLServerSocket)

`SSLSocketFactory` (/reference/javax/net/ssl/SSLSocketFactory)

Summary

Protected constructors

SSLSocket (/reference/javax/net/ssl/SSLSocket#SSLSocket())()

Used only by subclasses.

SSLSocket (/reference/javax/net/ssl/SSLSocket#SSLSocket(java.lang.String,%20int))(String (/reference/java/lang/String) host, int port)

Used only by subclasses.

SSLSocket (/reference/javax/net/ssl/SSLSocket#SSLSocket(java.net.InetAddress,%20int))(InetAddress (/reference/java/net/InetAddress) address, int port)

Used only by subclasses.

SSLSocket (/reference/javax/net/ssl/SSLSocket#SSLSocket(java.lang.String,%20int,%20java.net.InetAddress,%20int))(String (/reference/java/lang/String) host, int port, InetAddress (/reference/java/net/InetAddress) clientAddress, int clientPort)

Used only by subclasses.

SSLSocket (/reference/javax/net/ssl/SSLSocket#SSLSocket(java.net.InetAddress,%20int,%20java.net.InetAddress,%20int))(InetAddress (/reference/java/net/InetAddress) address, int port, InetAddress (/reference/java/net/InetAddress) clientAddress, int clientPort)

Used only by subclasses.

Public methods

abstract void	addHandshakeCompletedListener (/reference/javax/net/ssl/S (HandshakeCompletedListener (/reference/javax/net/ssl/Har Registers an event listener to receive notifications that an SSL ha

String (/reference/java/lang/String)	getApplicationProtocol (/reference/javax/net/ssl/SSLSocke Returns the most recent application protocol value negotiated fo
abstract boolean	getEnableSessionCreation (/reference/javax/net/ssl/SSLSoc Returns true if new SSL sessions may be established by this sock
abstract String[] (/reference/java/lang/String)	getEnabledCipherSuites (/reference/javax/net/ssl/SSLSocke Returns the names of the SSL cipher suites which are currently e
abstract String[] (/reference/java/lang/String)	getEnabledProtocols (/reference/javax/net/ssl/SSLSocket#ge Returns the names of the protocol versions which are currently e
String (/reference/java/lang/String)	getHandshakeApplicationProtocol (/reference/javax/net/ss Returns the application protocol value negotiated on a SSL/TLS h
BiFunction (/reference/java/util/function/BiFunction) <SSLSocket (/reference/javax/net/ssl/SSLSocket), List (/reference/java/util/List) <String (/reference/java/lang/String)>, String (/reference/java/lang/String)>	getHandshakeApplicationProtocolSelector (/reference/j Retrieves the callback function that selects an application protoc
SSLSession (/reference/javax/net/ssl/SSLSession)	getHandshakeSession (/reference/javax/net/ssl/SSLSocket#ge Returns the SSLSession being constructed during a SSL/TLS ha
abstract boolean	getNeedClientAuth (/reference/javax/net/ssl/SSLSocket#getN Returns true if the socket will *require* client authentication.
SSLParameters (/reference/javax/net/ssl/SSLParameters)	getSSLParameters (/reference/javax/net/ssl/SSLSocket#getSS Returns the SSLParameters in effect for this SSLSocket.

abstract SSLSession (/reference/javax/net/ssl/SSLSession)	getSession (/reference/javax/net/ssl/SSLSocket#getSession()) Returns the SSL Session in use by this connection.
abstract String[] (/reference/java/lang/String)	getSupportedCipherSuites (/reference/javax/net/ssl/SSLSoc Returns the names of the cipher suites which could be enabled f
abstract String[] (/reference/java/lang/String)	getSupportedProtocols (/reference/javax/net/ssl/SSLSocket# Returns the names of the protocols which could be enabled for u
abstract boolean	getUseClientMode (/reference/javax/net/ssl/SSLSocket#getUs Returns true if the socket is set to use client mode when handsha
abstract boolean	getWantClientAuth (/reference/javax/net/ssl/SSLSocket#getW Returns true if the socket will *request* client authentication.
abstract void	removeHandshakeCompletedListener (/reference/javax/net/s (HandshakeCompletedListener (/reference/javax/net/ssl/Har Removes a previously registered handshake completion listener.
abstract void	setEnableSessionCreation (/reference/javax/net/ssl/SSLSoc Controls whether new SSL sessions may be established by this s
abstract void	setEnabledCipherSuites (/reference/javax/net/ssl/SSLSocke Sets the cipher suites enabled for use on this connection.
abstract void	setEnabledProtocols (/reference/javax/net/ssl/SSLSocket#se Sets the protocol versions enabled for use on this connection.
void	setHandshakeApplicationProtocolSelector (/reference/javax/net/ssl/SSLSocket#setHandshakeApplicationP (BiFunction (/reference/java/util/function/BiFunction)<SSLSoc >, String (/reference/java/lang/String)> selector)

	Registers a callback function that selects an application protocol
`abstract void`	`setNeedClientAuth` (/reference/javax/net/ssl/SSLSocket#setN Configures the socket to *require* client authentication.
`void`	`setSSLParameters` (/reference/javax/net/ssl/SSLSocket#setSS Applies SSLParameters to this socket.
`abstract void`	`setUseClientMode` (/reference/javax/net/ssl/SSLSocket#setUs Configures the socket to use client (or server) mode when hands
`abstract void`	`setWantClientAuth` (/reference/javax/net/ssl/SSLSocket#setW Configures the socket to *request* client authentication.
`abstract void`	`startHandshake` (/reference/javax/net/ssl/SSLSocket#startHan Starts an SSL handshake on this connection.
`String` (/reference/java/lang/String)	`toString` (/reference/javax/net/ssl/SSLSocket#toString())`()` Converts this socket to a `String`.

Inherited methods

From class `java.net.Socket` (/reference/java/net/Socket)

`void`	`bind` (/reference/ `(SocketAddres` Binds the socket
`void`	`close` (/referenc Closes this socke

void	connect (/refere (SocketAddres timeout) Connects this soc
void	connect (/refere (SocketAddres Connects this soc
SocketChannel (/reference/java/nio/channels/SocketChannel)	getChannel (/re Returns the uniqu (/reference/java/ socket, if any.
InetAddress (/reference/java/net/InetAddress)	getInetAddres Returns the addre
InputStream (/reference/java/io/InputStream)	getInputStrea Returns an input s
boolean	getKeepAlive (Tests if SO_KEEP enabled.
InetAddress (/reference/java/net/InetAddress)	getLocalAddre Gets the local ad
int	getLocalPort (Returns the local
SocketAddress (/reference/java/net/SocketAddress)	getLocalSocke (/reference/java/

	Returns the addre
`boolean`	`getOOBInline` (Tests if `SO_OOBI` enabled.
`<T> T`	`getOption` (/refe (`SocketOption` Returns the value
`OutputStream` (/reference/java/io/OutputStream)	`getOutputStre` Returns an outpu
`int`	`getPort` (/refere Returns the remo
`int`	`getReceiveBuf` () Gets the value of (/reference/java/ is the buffer size
`SocketAddress` (/reference/java/net/SocketAddress)	`getRemoteSock` (/reference/java/ Returns the addre unconnected.
`boolean`	`getReuseAddre` Tests if `SO_REUS` enabled.
`int`	`getSendBuffer` Get value of the `S` option for this `So`

	this `Socket`.
`int`	`getSoLinger` (/r Returns setting fc .
`int`	`getSoTimeout` (Returns setting fc (/reference/java/
`boolean`	`getTcpNoDelay` Tests if `TCP_NOD` enabled.
`int`	`getTrafficCla` Gets traffic class Socket As the underlying of-service set usi (/reference/java/ different value tha (/reference/java/
`boolean`	`isBound` (/refere Returns the bindi
`boolean`	`isClosed` (/refer Returns the close
`boolean`	`isConnected` (/r Returns the conn
`boolean`	`isInputShutdo`

	Returns whether
boolean	isOutputShutd Returns whether
void	sendUrgentDat data) Send one byte of
void	setKeepAlive (on) Enable/disable SO (/reference/java/
void	setOOBInline (on) Enable/disable SO (/reference/java/ data) By default, socket is silently
<T> Socket (/reference/java/net/Socket)	setOption (/reference/java/ (SocketOption Sets the value of
void	setPerformanc (/reference/java/ (int connecti Sets performance
void	setReceiveBuf (/reference/java/ Sets the SO_RCVI the specified valu

void	setReuseAddre(boolean on) Enable/disable th (/reference/java/
void	setSendBuffer(size) Sets the SO_SNDI to the specified v
void	setSoLinger (/r(boolean on, Enable/disable SC with the specifiec
void	setSoTimeout (timeout) Enable/disable SC with the specifiec
static void	setSocketImpl (/reference/java/ (SocketImplFa Sets the client so
void	setTcpNoDelay(boolean on) Enable/disable TC (/reference/java/ algorithm).
void	setTrafficCla Sets traffic class this Socket.

void	shutdownInput Places the input s
void	shutdownOutpu Disables the outp
Set (/reference/java/util/Set)<SocketOption (/reference/java/net/SocketOption)<?>>	supportedOpti Returns a set of t
String (/reference/java/lang/String)	toString (/refer Converts this soc

From class java.lang.Object (/reference/java/lang/Object)

Object (/reference/java/lang/Object)	clone (/reference/java/lang/Object#clone())() Creates and returns a copy of this object.
boolean	equals (/reference/java/lang/Object#equals(java.lang.Ob Indicates whether some other object is "equal to" this one
void	finalize (/reference/java/lang/Object#finalize())() Called by the garbage collector on an object when garbag the object.
final Class (/reference/java/lang/Class)<?>	getClass (/reference/java/lang/Object#getClass())() Returns the runtime class of this Object.
int	hashCode (/reference/java/lang/Object#hashCode())() Returns a hash code value for the object.
final void	notify (/reference/java/lang/Object#notify())()

	Wakes up a single thread that is waiting on this object's m
`final void`	`notifyAll` (/reference/java/lang/Object#notifyAll())() Wakes up all threads that are waiting on this object's mon
`String` (/reference/java/lang/String)	`toString` (/reference/java/lang/Object#toString())() Returns a string representation of the object.
`final void`	`wait` (/reference/java/lang/Object#wait(long,%20int))(`lo` Causes the current thread to wait until it is awakened, typ of real time has elapsed.
`final void`	`wait` (/reference/java/lang/Object#wait(long))(`long ti` Causes the current thread to wait until it is awakened, typ of real time has elapsed.
`final void`	`wait` (/reference/java/lang/Object#wait())() Causes the current thread to wait until it is awakened, typ

From interface `java.io.Closeable` (/reference/java/io/Closeable)

`abstract void`	`close` (/reference/java/io/Closeable#close())() Closes this stream and releases any system resources associated with

From interface `java.lang.AutoCloseable` (/reference/java/lang/AutoCloseable)

`abstract void`	`close` (/reference/java/lang/AutoCloseable#close())() Closes this resource, relinquishing any underlying resources.

Protected constructors

SSLSocket

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected SSLSocket ()
```

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

SSLSocket

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected SSLSocket (String (/reference/java/lang/String) host,
                int port)
```

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

If there is a security manager, its `checkConnect` method is called with the host address and `port` as its arguments. This could result in a SecurityException.

Parameters

`host`	`String`: name of the host with which to connect, or `null` for the loopback address.
`port`	`int`: number of the server's port

Throws

`IOException` (/reference/java/io/IOException)	if an I/O error occurs when creating the socket
`SecurityException` (/reference/java/lang/SecurityException)	if a security manager exists and its `checkConnect` method doesn't allow the operation.

`UnknownHostException` (/reference/java/net/UnknownHostException)	if the host is not known
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the port parameter is outside the specified range of valid port values, which is between 0 and 65535, inclusive.

See also:

`SecurityManager.checkConnect(String, int)` (/reference/java/lang/SecurityManager#checkConnect(java.lang.String,%20int))

SSLSocket

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected SSLSocket (InetAddress (/reference/java/net/InetAddress) address,
                int port)
```

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

If there is a security manager, its `checkConnect` method is called with the host address and `port` as its arguments. This could result in a SecurityException.

Parameters

`address`	`InetAddress`: the server's host
`port`	`int`: its port

Throws

IOException (/reference/java/io/IOException)	if an I/O error occurs when creating the socket
SecurityException (/reference/java/lang/SecurityException)	if a security manager exists and its `checkConnect` method doesn't allow the operation.
IllegalArgumentException (/reference/java/lang/IllegalArgumentException)	if the port parameter is outside the specified range of valid port values, which is between 0 and 65535, inclusive.
NullPointerException (/reference/java/lang/NullPointerException)	if `address` is null.

See also:

`SecurityManager.checkConnect(String, int)` (/reference/java/lang/SecurityManager#checkConnect(java.lang.String,%20int))

SSLSocket

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected SSLSocket (String (/reference/java/lang/String) host,
                int port,
                InetAddress (/reference/java/net/InetAddress) clientAddress,
                int clientPort)
```

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.

If there is a security manager, its `checkConnect` method is called with the host address and `port` as its arguments. This could result in a SecurityException.

Parameters

`host`	`String`: name of the host with which to connect, or `null` for the loopback address.
`port`	`int`: number of the server's port
`clientAddress`	`InetAddress`: the client's address the socket is bound to, or `null` for the `anyLocal` address.
`clientPort`	`int`: the client's port the socket is bound to, or `zero` for a system selected free port.

Throws

`IOException` (/reference/java/io/IOException)	if an I/O error occurs when creating the socket
`SecurityException` (/reference/java/lang/SecurityException)	if a security manager exists and its `checkConnect` method doesn't allow the operation.
`UnknownHostException` (/reference/java/net/UnknownHostException)	if the host is not known
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the port parameter or clientPort parameter is outside the specified range of valid port values, which is between 0 and 65535, inclusive.

See also:

`SecurityManager.checkConnect(String, int)` (/reference/java/lang/SecurityManager#checkConnect(java.lang.String,%20int))

SSLSocket

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected SSLSocket (InetAddress (/reference/java/net/InetAddress) address,
                int port,
                InetAddress (/reference/java/net/InetAddress) clientAddress,
                int clientPort)
```

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.

If there is a security manager, its `checkConnect` method is called with the host address and `port` as its arguments. This could result in a SecurityException.

Parameters

`address`	`InetAddress`: the server's host
`port`	`int`: its port
`clientAddress`	`InetAddress`: the client's address the socket is bound to, or `null` for the `anyLocal` address.
`clientPort`	`int`: the client's port the socket is bound to, or `zero` for a system selected free port.

Throws

`IOException` (/reference/java/io/IOException)	if an I/O error occurs when creating the socket
`SecurityException`	if a security manager exists and its `checkConnect`

(/reference/java/lang/SecurityException)	method doesn't allow the operation.
IllegalArgumentException (/reference/java/lang/IllegalArgumentException)	if the port parameter or clientPort parameter is outside the specified range of valid port values, which is between 0 and 65535, inclusive.
NullPointerException (/reference/java/lang/NullPointerException)	if **address** is null.

See also:

`SecurityManager.checkConnect(String, int)` (/reference/java/lang/SecurityManager#checkConnect(java.lang.String,%20int))

Public methods

addHandshakeCompletedListener

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public abstract void addHandshakeCompletedListener (HandshakeCompletedListener (/r
```

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

Parameters

listener	**HandshakeCompletedListener**: the HandShake Completed event listener

Throws

`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the argument is null.

See also:

`startHandshake()` (/reference/javax/net/ssl/SSLSocket#startHandshake())

`removeHandshakeCompletedListener(HandshakeCompletedListener)` (/reference/javax/net/ssl/SSLSocket#removeHandshakeCompletedListener(javax.net.ssl.HandshakeCompletedListener))

getApplicationProtocol

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String (/reference/java/lang/String) getApplicationProtocol ()
```

Returns the most recent application protocol value negotiated for this connection.

If supported by the underlying SSL/TLS implementation, application name negotiation mechanisms such as RFC 7301 (http://www.ietf.org/rfc/rfc7301.txt), the Application-Layer Protocol Negotiation (ALPN), can negotiate application-level values between peers.

Implementation Requirements:

The implementation in this class throws `UnsupportedOperationException` and performs no other action.

Returns

`String` (/reference/java/lang/String)	null if it has not yet been determined if application protocols might be used for this connection, an empty `String` if application protocols values will not be used, or a non-empty application protocol `String` if a value was successfully negotiated.

Throws

UnsupportedOperationException (/reference/java/lang/UnsupportedOperationException)	if the underlying provider does not implement the operation.

getEnableSessionCreation

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public abstract boolean getEnableSessionCreation ()
```

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

Returns

boolean	true indicates that sessions may be created; this is the default. false indicates that an existing session must be resumed

See also:

setEnableSessionCreation(boolean) (/reference/javax/net/ssl/SSLSocket#setEnableSessionCreation(boolean))

getEnabledCipherSuites

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public abstract String[] (/reference/java/lang/String) 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

String[] (/reference/java/lang/String)	an array of cipher suite names

See also:

getSupportedCipherSuites() (/reference/javax/net/ssl/SSLSocket#getSupportedCipherSuites())

setEnabledCipherSuites(String[]) (/reference/javax/net/ssl/SSLSocket#setEnabledCipherSuites(java.lang.String[]))

getEnabledProtocols

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public abstract String[] (/reference/java/lang/String) getEnabledProtocols ()
```

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

Returns

String[] (/reference/java/lang/String)	an array of protocols

See also:

setEnabledProtocols(String[]) (/reference/javax/net/ssl/SSLSocket#setEnabledProtocols(java.lang.String[]))

getHandshakeApplicationProtocol

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String (/reference/java/lang/String) getHandshakeApplicationProtocol ()
```

Returns the application protocol value negotiated on a SSL/TLS handshake currently in progress.

Like `getHandshakeSession()` (/reference/javax/net/ssl/SSLSocket#getHandshakeSession()), a connection may be in the middle of a handshake. The application protocol may or may not yet be available.

Implementation Requirements:

The implementation in this class throws `UnsupportedOperationException` and performs no other action.

Returns

String (/reference/java/lang/String)	null if it has not yet been determined if application protocols might be used for this handshake, an empty `String` if application protocols values will not be used, or a non-empty application protocol `String` if a value was successfully negotiated.

Throws

UnsupportedOperationException (/reference/java/lang/UnsupportedOperationException)	if the underlying provider does not implement the operation.

getHandshakeApplicationProtocolSelector

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public BiFunction (/reference/java/util/function/BiFunction)<SSLSocket (/reference/javax/net/ssl/SSL
```

Retrieves the callback function that selects an application protocol value during a SSL/TLS handshake. See `setHandshakeApplicationProtocolSelector` (/reference/javax/net/ssl/SSLSocket#setHandshakeApplicationProtocolSelector(java.util.function.BiFunction<javax.net.ssl.SSLSocket,java.util.List<java.lang.String>,java.lang.String>)) for the function's type parameters.

Implementation Requirements:

The implementation in this class throws `UnsupportedOperationException` and performs no other action.

Returns

BiFunction (/reference/java/util/function/BiFunction)<SSLSocket (/reference/javax/net/ssl/SSLSocket), List (/reference/java/util/List)<String (/reference/java/lang/String)>, String (/reference/java/lang/String)>	the callback function, or null if none has been set.

Throws

UnsupportedOperationException (/reference/java/lang/UnsupportedOperationException)	if the underlying provider does not implement the operation.

---

getHandshakeSession

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public SSLSession (/reference/javax/net/ssl/SSLSession) getHandshakeSession ()
```

Returns the `SSLSession` being constructed during a SSL/TLS handshake.

TLS protocols may negotiate parameters that are needed when using an instance of this class, but before the `SSLSession` has been completely initialized and made available via `getSession`. For example, the list of valid signature algorithms may restrict the type of certificates that can used during TrustManager decisions, or the maximum TLS fragment packet sizes can be resized to better support the network environment.

This method provides early access to the `SSLSession` being constructed. Depending on how far the handshake has progressed, some data may not yet be available for use. For example, if a remote server will be sending a Certificate chain, but that chain has yet not been processed, the `getPeerCertificates` method of `SSLSession` will throw a SSLPeerUnverifiedException. Once that chain has been processed, `getPeerCertificates` will return the proper value.

Unlike `getSession()` (/reference/javax/net/ssl/SSLSocket#getSession()), this method does not initiate the initial handshake and does not block until handshaking is complete.

Returns

SSLSession (/reference/javax/net/ssl/SSLSession)	null if this instance is not currently handshaking, or if the current handshake has not progressed far enough to create a basic SSLSession. Otherwise, this method returns the **SSLSession** currently being negotiated.

Throws

UnsupportedOperationException (/reference/java/lang/UnsupportedOperationException)	if the underlying provider does not implement the operation.

See also:

`SSLEngine` (/reference/javax/net/ssl/SSLEngine)

`SSLSession` (/reference/javax/net/ssl/SSLSession)

`ExtendedSSLSession` (/reference/javax/net/ssl/ExtendedSSLSession)

`X509ExtendedKeyManager` (/reference/javax/net/ssl/X509ExtendedKeyManager)

`X509ExtendedTrustManager` (/reference/javax/net/ssl/X509ExtendedTrustManager)

getNeedClientAuth

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
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

`boolean`	true if client authentication is required, or false if no client authentication is desired.

See also:

`setNeedClientAuth(boolean)` (/reference/javax/net/ssl/SSLSocket#setNeedClientAuth(boolean))

`setWantClientAuth(boolean)` (/reference/javax/net/ssl/SSLSocket#setWantClientAuth(boolean))

`getWantClientAuth()` (/reference/javax/net/ssl/SSLSocket#getWantClientAuth())

`setUseClientMode(boolean)` (/reference/javax/net/ssl/SSLSocket#setUseClientMode(boolean))

getSSLParameters

Added in API level 9 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public SSLParameters (/reference/javax/net/ssl/SSLParameters) getSSLParameters ()
```

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

Returns

SSLParameters (/reference/javax/net/ssl/SSLParameters)	the SSLParameters in effect for this SSLSocket.

getSession

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public abstract SSLSession (/reference/javax/net/ssl/SSLSession) 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

SSLSession (/reference/javax/net/ssl/SSLSession)	the `SSLSession`

getSupportedCipherSuites

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public abstract String[] (/reference/java/lang/String) 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.

Applications should not blindly enable all supported cipher suites. The supported cipher suites can include signaling cipher suite values that can cause connection problems if enabled inappropriately.

The proper way to use this method is to either check if a specific cipher suite is supported via `Arrays.asList(getSupportedCipherSuites()).contains(...)` or to filter a desired list of cipher suites to only the supported ones via `desiredSuiteSet.retainAll(Arrays.asList(getSupportedCipherSuites()))`.

Returns

String[] (/reference/java/lang/String)	an array of cipher suite names

See also:

`getEnabledCipherSuites()` (/reference/javax/net/ssl/SSLSocket#getEnabledCipherSuites())

`setEnabledCipherSuites(String[])` (/reference/javax/net/ssl/SSLSocket#setEnabledCipherSuites(java.lang.String[]))

getSupportedProtocols

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public abstract String[] (/reference/java/lang/String) getSupportedProtocols ()
```

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

Returns

String[] (/reference/java/lang/String)	an array of protocols supported

getUseClientMode

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public abstract boolean getUseClientMode ()
```

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

Returns

boolean	true if the socket should do handshaking in "client" mode

See also:

setUseClientMode(boolean) (/reference/javax/net/ssl/SSLSocket#setUseClientMode(boolean))

getWantClientAuth

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
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

boolean	true if client authentication is requested, or false if no client authentication is desired.

See also:

`setNeedClientAuth(boolean)` (/reference/javax/net/ssl/SSLSocket#setNeedClientAuth(boolean))

`getNeedClientAuth()` (/reference/javax/net/ssl/SSLSocket#getNeedClientAuth())

`setWantClientAuth(boolean)` (/reference/javax/net/ssl/SSLSocket#setWantClientAuth(boolean))

`setUseClientMode(boolean)` (/reference/javax/net/ssl/SSLSocket#setUseClientMode(boolean))

removeHandshakeCompletedListener

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public abstract void removeHandshakeCompletedListener (HandshakeCompletedListene
```

Removes a previously registered handshake completion listener.

Parameters

`listener`	`HandshakeCompletedListener`: the HandShake Completed event listener

Throws

`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the listener is not registered, or the argument is null.

See also:

`addHandshakeCompletedListener(HandshakeCompletedListener)` (/reference/javax/net/ssl/SSLSocket#addHandshakeCompletedListener(javax.net.ssl.HandshakeCompletedListener))

setEnableSessionCreation

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
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`	**`boolean`**: true indicates that sessions may be created; this is the default. false indicates that an existing session must be resumed

See also:

`getEnableSessionCreation()` (/reference/javax/net/ssl/SSLSocket#getEnableSessionCreation())

setEnabledCipherSuites

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public abstract void setEnabledCipherSuites (String[] (/reference/java/lang/String) suit
```

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()` (/reference/javax/net/ssl/SSLSocket#getEnabledCipherSuites()) for more information on why a specific ciphersuite may never be used on a connection.

Parameters

`suites`	`String`: Names of all the cipher suites to enable

Throws

`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	when one or more of the ciphers named by the parameter is not supported, or when the parameter is null.

See also:

`getSupportedCipherSuites()` (/reference/javax/net/ssl/SSLSocket#getSupportedCipherSuites())

`getEnabledCipherSuites()` (/reference/javax/net/ssl/SSLSocket#getEnabledCipherSuites())

setEnabledProtocols

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public abstract void setEnabledProtocols (String[] (/reference/java/lang/String) protoco
```

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.

Because of the way the protocol version is negotiated, connections will only be able to use a member of the lowest set of contiguous enabled protocol versions. For example, enabling TLSv1.2 and TLSv1 will result in connections only being able to use TLSv1.

Parameters

`protocols`	`String`: Names of all the protocols to enable.

Throws

`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	when one or more of the protocols named by the parameter is not supported or when the protocols parameter is null.

See also:

`getEnabledProtocols()` (/reference/javax/net/ssl/SSLSocket#getEnabledProtocols())

setHandshakeApplicationProtocolSelector

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setHandshakeApplicationProtocolSelector (BiFunction (/reference/java/util/fu
```

Registers a callback function that selects an application protocol value for a SSL/TLS handshake. The function overrides any values supplied using `SSLParameters.setApplicationProtocols` (/reference/javax/net/ssl/SSLParameters#setApplicationProtocols(java.lang.String[])) and it supports the following type parameters:

> `SSLSocket`
>
> The function's first argument allows the current `SSLSocket` to be inspected, including the handshake session and configuration settings.
>
> `List<String>`
>
> The function's second argument lists the application protocol names advertised by the TLS peer.
>
> `String`

> The function's result is an application protocol name, or null to indicate that none of the advertised names are acceptable. If the return value is an empty `String` then application protocol indications will not be used. If the return value is null (no value chosen) or is a value that was not advertised by the peer, the underlying protocol will determine what action to take. (For example, ALPN will send a "no_application_protocol" alert and terminate the connection.)

For example, the following call registers a callback function that examines the TLS handshake parameters and selects an application protocol name:

```
serverSocket.setHandshakeApplicationProtocolSelector(
        (serverSocket, clientProtocols) -> {
            SSLSession session = serverSocket.getHandshakeSession();
            return chooseApplicationProtocol(
                serverSocket,
                clientProtocols,
                session.getProtocol(),
                session.getCipherSuite());
        });
```

API Note:

This method should be called by TLS server applications before the TLS handshake begins. Also, this `SSLSocket` should be configured with parameters that are compatible with the application protocol selected by the callback function. For example, enabling a poor choice of cipher suites could result in no suitable application protocol. See `SSLParameters` (/reference/javax/net/ssl/SSLParameters).

Implementation Requirements:

The implementation in this class throws `UnsupportedOperationException` and performs no other action.

Parameters

`selector`	`BiFunction`: the callback function, or null to de-register.

Throws

`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if the underlying provider does not implement the operation.

setNeedClientAuth

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
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)` (/reference/javax/net/ssl/SSLSocket#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)` (/reference/javax/net/ssl/SSLSocket#setWantClientAuth(boolean)).

Parameters

need	**boolean**: set to true if client authentication is required, or false if no client authentication is desired.

See also:

`getNeedClientAuth()` (/reference/javax/net/ssl/SSLSocket#getNeedClientAuth())

`setWantClientAuth(boolean)` (/reference/javax/net/ssl/SSLSocket#setWantClientAuth(boolean))

`getWantClientAuth()` (/reference/javax/net/ssl/SSLSocket#getWantClientAuth())

`setUseClientMode(boolean)` (/reference/javax/net/ssl/SSLSocket#setUseClientMode(boolean))

setSSLParameters

Added in API level 9 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setSSLParameters (SSLParameters (/reference/javax/net/ssl/SSLParameters) param
```

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.
- If `params.getServerNames()` is non-null, the socket will configure its server names with that value.
- If `params.getSNIMatchers()` is non-null, the socket will configure its SNI matchers with that value.

Parameters

`params`	`SSLParameters`: the parameters

Throws

`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the setEnabledCipherSuites() or the setEnabledProtocols() call fails

setUseClientMode

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
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`	`boolean`: true if the socket should start its handshaking in "client" mode

Throws

`IllegalArgumentException`	if a mode change is attempted after the initial handshake

(/reference/java/lang/IllegalArgumentException)has begun.

See also:

`getUseClientMode()` (/reference/javax/net/ssl/SSLSocket#getUseClientMode())

setWantClientAuth

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
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)` (/reference/javax/net/ssl/SSLSocket#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)` (/reference/javax/net/ssl/SSLSocket#setNeedClientAuth(boolean)).

Parameters

`want`	**`boolean`**: set to true if client authentication is requested, or false if no client authentication is desired.

See also:

`getWantClientAuth()` (/reference/javax/net/ssl/SSLSocket#getWantClientAuth())

`setNeedClientAuth(boolean)` (/reference/javax/net/ssl/SSLSocket#setNeedClientAuth(boolean))

`getNeedClientAuth()` (/reference/javax/net/ssl/SSLSocket#getNeedClientAuth())

`setUseClientMode(boolean)` (/reference/javax/net/ssl/SSLSocket#setUseClientMode(boolean))

startHandshake

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public abstract void startHandshake ()
```

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` (/reference/java/io/IOException)	on a network level error

See also:

`addHandshakeCompletedListener(HandshakeCompletedListener)` (/reference/javax/net/ssl/SSLSocket#addHandshakeCompletedListener(javax.net.ssl.HandshakeCompletedListener))

toString

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String (/reference/java/lang/String) toString ()
```

Converts this socket to a `String`.

Returns

`String` (/reference/java/lang/String)	a string representation of this socket.

Last updated 2024-02-16 UTC.