| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SSLEngine.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/SSLContextSpi.html)   [**NEXT CLASS**](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) | [**FRAMES**](http://docs.google.com/index.html?javax/net/ssl/SSLEngine.html)    [**NO FRAMES**](http://docs.google.com/SSLEngine.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#2s8eyo1) |

## **javax.net.ssl**

Class SSLEngine

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

public abstract class **SSLEngine**extends [Object](http://docs.google.com/java/lang/Object.html)

A class which enables secure communications using protocols such as the Secure Sockets Layer (SSL) or  [IETF RFC 2246 "Transport Layer Security" (TLS)](http://www.ietf.org/rfc/rfc2246.txt)  protocols, but is transport independent.

The secure communications modes include:

* *Integrity Protection*. SSL/TLS protects against modification of messages by an active wiretapper.
* *Authentication*. In most modes, SSL/TLS provides peer authentication. Servers are usually authenticated, and clients may be authenticated as requested by servers.
* *Confidentiality (Privacy Protection)*. In most modes, SSL/TLS 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 cipher suite 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()](http://docs.google.com/javax/net/ssl/SSLEngine.html#getSession()) method.

The SSLSocket class provides much of the same security functionality, but all of the inbound and outbound data is automatically transported using the underlying [Socket](http://docs.google.com/java/net/Socket.html), which by design uses a blocking model. While this is appropriate for many applications, this model does not provide the scalability required by large servers.

The primary distinction of an SSLEngine is that it operates on inbound and outbound byte streams, independent of the transport mechanism. It is the responsibility of the SSLEngine user to arrange for reliable I/O transport to the peer. By separating the SSL/TLS abstraction from the I/O transport mechanism, the SSLEngine can be used for a wide variety of I/O types, such as [non-blocking I/O (polling)](http://docs.google.com/java/nio/channels/spi/AbstractSelectableChannel.html#configureBlocking(boolean)), [selectable non-blocking I/O](http://docs.google.com/java/nio/channels/Selector.html), [Socket](http://docs.google.com/java/net/Socket.html) and the traditional Input/OutputStreams, local [ByteBuffers](http://docs.google.com/java/nio/ByteBuffer.html) or byte arrays,  [future asynchronous I/O models](http://www.jcp.org/en/jsr/detail?id=203) , and so on.

At a high level, the SSLEngine appears thus:

app data  
  
 | ^  
 | | |  
 v | |  
 +----+-----|-----+----+  
 | | |  
 | SSL|Engine |  
 wrap() | | | unwrap()  
 | OUTBOUND | INBOUND |  
 | | |  
 +----+-----|-----+----+  
 | | ^  
 | | |  
 v |  
  
 net data

Application data (also known as plaintext or cleartext) is data which is produced or consumed by an application. Its counterpart is network data, which consists of either handshaking and/or ciphertext (encrypted) data, and destined to be transported via an I/O mechanism. Inbound data is data which has been received from the peer, and outbound data is destined for the peer.

(In the context of an SSLEngine, the term "handshake data" is taken to mean any data exchanged to establish and control a secure connection. Handshake data includes the SSL/TLS messages "alert", "change\_cipher\_spec," and "handshake.")

There are five distinct phases to an SSLEngine.

1. Creation - The SSLEngine has been created and initialized, but has not yet been used. During this phase, an application may set any SSLEngine-specific settings (enabled cipher suites, whether the SSLEngine should handshake in client or server mode, and so on). Once handshaking has begun, though, any new settings (except client/server mode, see below) will be used for the next handshake.
2. Initial Handshake - The initial handshake is a procedure by which the two peers exchange communication parameters until an SSLSession is established. Application data can not be sent during this phase.
3. Application Data - Once the communication parameters have been established and the handshake is complete, application data may flow through the SSLEngine. Outbound application messages are encrypted and integrity protected, and inbound messages reverse the process.
4. Rehandshaking - Either side may request a renegotiation of the session at any time during the Application Data phase. New handshaking data can be intermixed among the application data. Before starting the rehandshake phase, the application may reset the SSL/TLS communication parameters such as the list of enabled ciphersuites and whether to use client authentication, but can not change between client/server modes. As before, once handshaking has begun, any new SSLEngine configuration settings will not be used until the next handshake.
5. Closure - When the connection is no longer needed, the application should close the SSLEngine and should send/receive any remaining messages to the peer before closing the underlying transport mechanism. Once an engine is closed, it is not reusable: a new SSLEngine must be created.

An SSLEngine is created by calling [SSLContext.createSSLEngine()](http://docs.google.com/javax/net/ssl/SSLContext.html#createSSLEngine()) from an initialized SSLContext. Any configuration parameters should be set before making the first call to wrap(), unwrap(), or beginHandshake(). These methods all trigger the initial handshake.

Data moves through the engine by calling [wrap()](http://docs.google.com/javax/net/ssl/SSLEngine.html#wrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer)) or [unwrap()](http://docs.google.com/javax/net/ssl/SSLEngine.html#unwrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer)) on outbound or inbound data, respectively. Depending on the state of the SSLEngine, a wrap() call may consume application data from the source buffer and may produce network data in the destination buffer. The outbound data may contain application and/or handshake data. A call to unwrap() will examine the source buffer and may advance the handshake if the data is handshaking information, or may place application data in the destination buffer if the data is application. The state of the underlying SSL/TLS algorithm will determine when data is consumed and produced.

Calls to wrap() and unwrap() return an SSLEngineResult which indicates the status of the operation, and (optionally) how to interact with the engine to make progress.

The SSLEngine produces/consumes complete SSL/TLS packets only, and does not store application data internally between calls to wrap()/unwrap(). Thus input and output ByteBuffers must be sized appropriately to hold the maximum record that can be produced. Calls to [SSLSession.getPacketBufferSize()](http://docs.google.com/javax/net/ssl/SSLSession.html#getPacketBufferSize()) and [SSLSession.getApplicationBufferSize()](http://docs.google.com/javax/net/ssl/SSLSession.html#getApplicationBufferSize()) should be used to determine the appropriate buffer sizes. The size of the outbound application data buffer generally does not matter. If buffer conditions do not allow for the proper consumption/production of data, the application must determine (via [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html)) and correct the problem, and then try the call again.

For example, unwrap() will return a [SSLEngineResult.Status.BUFFER\_OVERFLOW](http://docs.google.com/javax/net/ssl/SSLEngineResult.Status.html#BUFFER_OVERFLOW) result if the engine determines that there is not enough destination buffer space available. Applications should call [SSLSession.getApplicationBufferSize()](http://docs.google.com/javax/net/ssl/SSLSession.html#getApplicationBufferSize()) and compare that value with the space available in the destination buffer, enlarging the buffer if necessary. Similarly, if unwrap() were to return a [SSLEngineResult.Status.BUFFER\_UNDERFLOW](http://docs.google.com/javax/net/ssl/SSLEngineResult.Status.html#BUFFER_UNDERFLOW), the application should call [SSLSession.getPacketBufferSize()](http://docs.google.com/javax/net/ssl/SSLSession.html#getPacketBufferSize()) to ensure that the source buffer has enough room to hold a record (enlarging if necessary), and then obtain more inbound data.

SSLEngineResult r = engine.unwrap(src, dst);  
 switch (r.getStatus()) {  
 BUFFER\_OVERFLOW:  
 // Could attempt to drain the dst buffer of any already obtained  
 // data, but we'll just increase it to the size needed.  
 int appSize = engine.getSession().getApplicationBufferSize();  
 ByteBuffer b = ByteBuffer.allocate(appSize + dst.position());  
 dst.flip();  
 b.put(dst);  
 dst = b;  
 // retry the operation.  
 break;  
 BUFFER\_UNDERFLOW:  
 int netSize = engine.getSession().getPacketBufferSize();  
 // Resize buffer if needed.  
 if (netSize > dst.capacity()) {  
 ByteBuffer b = ByteBuffer.allocate(netSize);  
 src.flip();  
 b.put(src);  
 src = b;  
 }  
 // Obtain more inbound network data for src,  
 // then retry the operation.  
 break;  
 // other cases: CLOSED, OK.  
 }

Unlike SSLSocket, all methods of SSLEngine are non-blocking. SSLEngine implementations may require the results of tasks that may take an extended period of time to complete, or may even block. For example, a TrustManager may need to connect to a remote certificate validation service, or a KeyManager might need to prompt a user to determine which certificate to use as part of client authentication. Additionally, creating cryptographic signatures and verifying them can be slow, seemingly blocking.

For any operation which may potentially block, the SSLEngine will create a [Runnable](http://docs.google.com/java/lang/Runnable.html) delegated task. When SSLEngineResult indicates that a delegated task result is needed, the application must call [getDelegatedTask()](http://docs.google.com/javax/net/ssl/SSLEngine.html#getDelegatedTask()) to obtain an outstanding delegated task and call its [run()](http://docs.google.com/java/lang/Runnable.html#run()) method (possibly using a different thread depending on the compute strategy). The application should continue obtaining delegated tasks until no more exist, and try the original operation again.

At the end of a communication session, applications should properly close the SSL/TLS link. The SSL/TLS protocols have closure handshake messages, and these messages should be communicated to the peer before releasing the SSLEngine and closing the underlying transport mechanism. A close can be initiated by one of: an SSLException, an inbound closure handshake message, or one of the close methods. In all cases, closure handshake messages are generated by the engine, and wrap() should be repeatedly called until the resulting SSLEngineResult's status returns "CLOSED", or [isOutboundDone()](http://docs.google.com/javax/net/ssl/SSLEngine.html#isOutboundDone()) returns true. All data obtained from the wrap() method should be sent to the peer.

[closeOutbound()](http://docs.google.com/javax/net/ssl/SSLEngine.html#closeOutbound()) is used to signal the engine that the application will not be sending any more data.

A peer will signal its intent to close by sending its own closure handshake message. After this message has been received and processed by the local SSLEngine's unwrap() call, the application can detect the close by calling unwrap() and looking for a SSLEngineResult with status "CLOSED", or if [isInboundDone()](http://docs.google.com/javax/net/ssl/SSLEngine.html#isInboundDone()) returns true. If for some reason the peer closes the communication link without sending the proper SSL/TLS closure message, the application can detect the end-of-stream and can signal the engine via [closeInbound()](http://docs.google.com/javax/net/ssl/SSLEngine.html#closeInbound()) that there will no more inbound messages to process. Some applications might choose to require orderly shutdown messages from a peer, in which case they can check that the closure was generated by a handshake message and not by an end-of-stream condition.

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()](http://docs.google.com/javax/net/ssl/SSLEngine.html#getSupportedCipherSuites()).
* *Enabled* cipher suites, which may be fewer than the full set of supported suites. This group is set using the [setEnabledCipherSuites(String [])](http://docs.google.com/javax/net/ssl/SSLEngine.html#setEnabledCipherSuites(java.lang.String%5B%5D)) method, and queried using the [getEnabledCipherSuites()](http://docs.google.com/javax/net/ssl/SSLEngine.html#getEnabledCipherSuites()) method. Initially, a default set of cipher suites will be enabled on a new engine 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 cipher suite be selected.

Each SSL/TLS connection must have one client and one server, thus each endpoint must decide which role to assume. This choice determines who begins the handshaking process as well as which type of messages should be sent by each party. The method [setUseClientMode(boolean)](http://docs.google.com/javax/net/ssl/SSLEngine.html#setUseClientMode(boolean)) configures the mode. Once the initial handshaking has started, an SSLEngine can not switch between client and server modes, even when performing renegotiations.

Applications might choose to process delegated tasks in different threads. When an SSLEngine is created, the current [AccessControlContext](http://docs.google.com/java/security/AccessControlContext.html) is saved. All future delegated tasks will be processed using this context: that is, all access control decisions will be made using the context captured at engine creation.

**Concurrency Notes**: There are two concurrency issues to be aware of:

1. The wrap() and unwrap() methods may execute concurrently of each other.
2. The SSL/TLS protocols employ ordered packets. Applications must take care to ensure that generated packets are delivered in sequence. If packets arrive out-of-order, unexpected or fatal results may occur.  
   For example:  
     
    synchronized (outboundLock) {  
    sslEngine.wrap(src, dst);  
    outboundQueue.put(dst);  
    }  
    As a corollary, two threads must not attempt to call the same method (either wrap() or unwrap()) concurrently, because there is no way to guarantee the eventual packet ordering.

**Since:** 1.5 **See Also:**[SSLContext](http://docs.google.com/javax/net/ssl/SSLContext.html), [SSLSocket](http://docs.google.com/javax/net/ssl/SSLSocket.html), [SSLServerSocket](http://docs.google.com/javax/net/ssl/SSLServerSocket.html), [SSLSession](http://docs.google.com/javax/net/ssl/SSLSession.html), [Socket](http://docs.google.com/java/net/Socket.html)

| **Constructor Summary** | |
| --- | --- |
| protected | [**SSLEngine**](http://docs.google.com/javax/net/ssl/SSLEngine.html#SSLEngine())()            Constructor for an SSLEngine providing no hints for an internal session reuse strategy. |
| protected | [**SSLEngine**](http://docs.google.com/javax/net/ssl/SSLEngine.html#SSLEngine(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) peerHost, int peerPort)            Constructor for an SSLEngine. |

| **Method Summary** | |
| --- | --- |
| abstract  void | [**beginHandshake**](http://docs.google.com/javax/net/ssl/SSLEngine.html#beginHandshake())()            Initiates handshaking (initial or renegotiation) on this SSLEngine. |
| abstract  void | [**closeInbound**](http://docs.google.com/javax/net/ssl/SSLEngine.html#closeInbound())()            Signals that no more inbound network data will be sent to this SSLEngine. |
| abstract  void | [**closeOutbound**](http://docs.google.com/javax/net/ssl/SSLEngine.html#closeOutbound())()            Signals that no more outbound application data will be sent on this SSLEngine. |
| abstract  [Runnable](http://docs.google.com/java/lang/Runnable.html) | [**getDelegatedTask**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getDelegatedTask())()            Returns a delegated Runnable task for this SSLEngine. |
| abstract  [String](http://docs.google.com/java/lang/String.html)[] | [**getEnabledCipherSuites**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getEnabledCipherSuites())()            Returns the names of the SSL cipher suites which are currently enabled for use on this engine. |
| abstract  [String](http://docs.google.com/java/lang/String.html)[] | [**getEnabledProtocols**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getEnabledProtocols())()            Returns the names of the protocol versions which are currently enabled for use with this SSLEngine. |
| abstract  boolean | [**getEnableSessionCreation**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getEnableSessionCreation())()            Returns true if new SSL sessions may be established by this engine. |
| abstract  [SSLEngineResult.HandshakeStatus](http://docs.google.com/javax/net/ssl/SSLEngineResult.HandshakeStatus.html) | [**getHandshakeStatus**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getHandshakeStatus())()            Returns the current handshake status for this SSLEngine. |
| abstract  boolean | [**getNeedClientAuth**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getNeedClientAuth())()            Returns true if the engine will *require* client authentication. |
| [String](http://docs.google.com/java/lang/String.html) | [**getPeerHost**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getPeerHost())()            Returns the host name of the peer. |
| int | [**getPeerPort**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getPeerPort())()            Returns the port number of the peer. |
| abstract  [SSLSession](http://docs.google.com/javax/net/ssl/SSLSession.html) | [**getSession**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getSession())()            Returns the SSLSession in use in this SSLEngine. |
| [SSLParameters](http://docs.google.com/javax/net/ssl/SSLParameters.html) | [**getSSLParameters**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getSSLParameters())()            Returns the SSLParameters in effect for this SSLEngine. |
| abstract  [String](http://docs.google.com/java/lang/String.html)[] | [**getSupportedCipherSuites**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getSupportedCipherSuites())()            Returns the names of the cipher suites which could be enabled for use on this engine. |
| abstract  [String](http://docs.google.com/java/lang/String.html)[] | [**getSupportedProtocols**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getSupportedProtocols())()            Returns the names of the protocols which could be enabled for use with this SSLEngine. |
| abstract  boolean | [**getUseClientMode**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getUseClientMode())()            Returns true if the engine is set to use client mode when handshaking. |
| abstract  boolean | [**getWantClientAuth**](http://docs.google.com/javax/net/ssl/SSLEngine.html#getWantClientAuth())()            Returns true if the engine will *request* client authentication. |
| abstract  boolean | [**isInboundDone**](http://docs.google.com/javax/net/ssl/SSLEngine.html#isInboundDone())()            Returns whether [unwrap(ByteBuffer, ByteBuffer)](http://docs.google.com/javax/net/ssl/SSLEngine.html#unwrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer)) will accept any more inbound data messages. |
| abstract  boolean | [**isOutboundDone**](http://docs.google.com/javax/net/ssl/SSLEngine.html#isOutboundDone())()            Returns whether [wrap(ByteBuffer, ByteBuffer)](http://docs.google.com/javax/net/ssl/SSLEngine.html#wrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer)) will produce any more outbound data messages. |
| abstract  void | [**setEnabledCipherSuites**](http://docs.google.com/javax/net/ssl/SSLEngine.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 engine. |
| abstract  void | [**setEnabledProtocols**](http://docs.google.com/javax/net/ssl/SSLEngine.html#setEnabledProtocols(java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html)[] protocols)            Set the protocol versions enabled for use on this engine. |
| abstract  void | [**setEnableSessionCreation**](http://docs.google.com/javax/net/ssl/SSLEngine.html#setEnableSessionCreation(boolean))(boolean flag)            Controls whether new SSL sessions may be established by this engine. |
| abstract  void | [**setNeedClientAuth**](http://docs.google.com/javax/net/ssl/SSLEngine.html#setNeedClientAuth(boolean))(boolean need)            Configures the engine to *require* client authentication. |
| void | [**setSSLParameters**](http://docs.google.com/javax/net/ssl/SSLEngine.html#setSSLParameters(javax.net.ssl.SSLParameters))([SSLParameters](http://docs.google.com/javax/net/ssl/SSLParameters.html) params)            Applies SSLParameters to this engine. |
| abstract  void | [**setUseClientMode**](http://docs.google.com/javax/net/ssl/SSLEngine.html#setUseClientMode(boolean))(boolean mode)            Configures the engine to use client (or server) mode when handshaking. |
| abstract  void | [**setWantClientAuth**](http://docs.google.com/javax/net/ssl/SSLEngine.html#setWantClientAuth(boolean))(boolean want)            Configures the engine to *request* client authentication. |
| [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) | [**unwrap**](http://docs.google.com/javax/net/ssl/SSLEngine.html#unwrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer))([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) src, [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) dst)            Attempts to decode SSL/TLS network data into a plaintext application data buffer. |
| [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) | [**unwrap**](http://docs.google.com/javax/net/ssl/SSLEngine.html#unwrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer%5B%5D))([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) src, [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html)[] dsts)            Attempts to decode SSL/TLS network data into a sequence of plaintext application data buffers. |
| abstract  [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) | [**unwrap**](http://docs.google.com/javax/net/ssl/SSLEngine.html#unwrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer%5B%5D,%20int,%20int))([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) src, [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html)[] dsts, int offset, int length)            Attempts to decode SSL/TLS network data into a subsequence of plaintext application data buffers. |
| [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) | [**wrap**](http://docs.google.com/javax/net/ssl/SSLEngine.html#wrap(java.nio.ByteBuffer%5B%5D,%20java.nio.ByteBuffer))([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html)[] srcs, [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) dst)            Attempts to encode plaintext bytes from a sequence of data buffers into SSL/TLS network data. |
| abstract  [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) | [**wrap**](http://docs.google.com/javax/net/ssl/SSLEngine.html#wrap(java.nio.ByteBuffer%5B%5D,%20int,%20int,%20java.nio.ByteBuffer))([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html)[] srcs, int offset, int length, [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) dst)            Attempts to encode plaintext bytes from a subsequence of data buffers into SSL/TLS network data. |
| [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) | [**wrap**](http://docs.google.com/javax/net/ssl/SSLEngine.html#wrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer))([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) src, [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) dst)            Attempts to encode a buffer of plaintext application data into SSL/TLS network data. |

| **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()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [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** |
| --- |

### SSLEngine

protected **SSLEngine**()

Constructor for an SSLEngine providing no hints for an internal session reuse strategy.

**See Also:**[SSLContext.createSSLEngine()](http://docs.google.com/javax/net/ssl/SSLContext.html#createSSLEngine()), [SSLSessionContext](http://docs.google.com/javax/net/ssl/SSLSessionContext.html)

### SSLEngine

protected **SSLEngine**([String](http://docs.google.com/java/lang/String.html) peerHost,  
 int peerPort)

Constructor for an SSLEngine.

SSLEngine implementations may use the peerHost and peerPort parameters as hints for their internal session reuse strategy.

Some cipher suites (such as Kerberos) require remote hostname information. Implementations of this class should use this constructor to use Kerberos.

The parameters are not authenticated by the SSLEngine.

**Parameters:**peerHost - the name of the peer hostpeerPort - the port number of the peer**See Also:**[SSLContext.createSSLEngine(String, int)](http://docs.google.com/javax/net/ssl/SSLContext.html#createSSLEngine(java.lang.String,%20int)), [SSLSessionContext](http://docs.google.com/javax/net/ssl/SSLSessionContext.html)

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

### getPeerHost

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

Returns the host name of the peer.

Note that the value is not authenticated, and should not be relied upon.

**Returns:**the host name of the peer, or null if nothing is available.

### getPeerPort

public int **getPeerPort**()

Returns the port number of the peer.

Note that the value is not authenticated, and should not be relied upon.

**Returns:**the port number of the peer, or -1 if nothing is available.

### wrap

public [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) **wrap**([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) src,  
 [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) dst)  
 throws [SSLException](http://docs.google.com/javax/net/ssl/SSLException.html)

Attempts to encode a buffer of plaintext application data into SSL/TLS network data.

An invocation of this method behaves in exactly the same manner as the invocation:

[engine.wrap(new ByteBuffer [] { src }, 0, 1, dst);](http://docs.google.com/javax/net/ssl/SSLEngine.html#wrap(java.nio.ByteBuffer%5B%5D,%20int,%20int,%20java.nio.ByteBuffer))

**Parameters:**src - a ByteBuffer containing outbound application datadst - a ByteBuffer to hold outbound network data  
**Returns:**an SSLEngineResult describing the result  
 of this operation.  
**Throws:**  
[SSLException](http://docs.google.com/javax/net/ssl/SSLException.html) - A problem was encountered while processing the  
 data that caused the SSLEngine to abort.  
 See the class description for more information on  
 engine closure.  
[ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - if the dst buffer is read-only.  
[IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if either src or dst  
 is null.  
[IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the client/server mode  
 has not yet been set.**See Also:**[wrap(ByteBuffer [], int, int, ByteBuffer)](http://docs.google.com/javax/net/ssl/SSLEngine.html#wrap(java.nio.ByteBuffer%5B%5D,%20int,%20int,%20java.nio.ByteBuffer))

### wrap

public [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) **wrap**([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html)[] srcs,  
 [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) dst)  
 throws [SSLException](http://docs.google.com/javax/net/ssl/SSLException.html)

Attempts to encode plaintext bytes from a sequence of data buffers into SSL/TLS network data.

An invocation of this method behaves in exactly the same manner as the invocation:

[engine.wrap(srcs, 0, srcs.length, dst);](http://docs.google.com/javax/net/ssl/SSLEngine.html#wrap(java.nio.ByteBuffer%5B%5D,%20int,%20int,%20java.nio.ByteBuffer))

**Parameters:**srcs - an array of ByteBuffers containing the  
 outbound application datadst - a ByteBuffer to hold outbound network data  
**Returns:**an SSLEngineResult describing the result  
 of this operation.  
**Throws:**  
[SSLException](http://docs.google.com/javax/net/ssl/SSLException.html) - A problem was encountered while processing the  
 data that caused the SSLEngine to abort.  
 See the class description for more information on  
 engine closure.  
[ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - if the dst buffer is read-only.  
[IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if either srcs or dst  
 is null, or if any element in srcs is null.  
[IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the client/server mode  
 has not yet been set.**See Also:**[wrap(ByteBuffer [], int, int, ByteBuffer)](http://docs.google.com/javax/net/ssl/SSLEngine.html#wrap(java.nio.ByteBuffer%5B%5D,%20int,%20int,%20java.nio.ByteBuffer))

### wrap

public abstract [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) **wrap**([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html)[] srcs,  
 int offset,  
 int length,  
 [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) dst)  
 throws [SSLException](http://docs.google.com/javax/net/ssl/SSLException.html)

Attempts to encode plaintext bytes from a subsequence of data buffers into SSL/TLS network data. This *"gathering"* operation encodes, in a single invocation, a sequence of bytes from one or more of a given sequence of buffers. Gathering wraps are often useful when implementing network protocols or file formats that, for example, group data into segments consisting of one or more fixed-length headers followed by a variable-length body. See [GatheringByteChannel](http://docs.google.com/java/nio/channels/GatheringByteChannel.html) for more information on gathering, and [GatheringByteChannel.write(ByteBuffer[], int, int)](http://docs.google.com/java/nio/channels/GatheringByteChannel.html#write(java.nio.ByteBuffer%5B%5D,%20int,%20int)) for more information on the subsequence behavior.

Depending on the state of the SSLEngine, this method may produce network data without consuming any application data (for example, it may generate handshake data.)

The application is responsible for reliably transporting the network data to the peer, and for ensuring that data created by multiple calls to wrap() is transported in the same order in which it was generated. The application must properly synchronize multiple calls to this method.

If this SSLEngine has not yet started its initial handshake, this method will automatically start the handshake.

This method will attempt to produce one SSL/TLS packet, and will consume as much source data as possible, but will never consume more than the sum of the bytes remaining in each buffer. Each ByteBuffer's position is updated to reflect the amount of data consumed or produced. The limits remain the same.

The underlying memory used by the srcs and dst ByteBuffers must not be the same.

See the class description for more information on engine closure.

**Parameters:**srcs - an array of ByteBuffers containing the outbound application dataoffset - The offset within the buffer array of the first buffer from which bytes are to be retrieved; it must be non-negative and no larger than srcs.lengthlength - The maximum number of buffers to be accessed; it must be non-negative and no larger than srcs.length - offsetdst - a ByteBuffer to hold outbound network data **Returns:**an SSLEngineResult describing the result of this operation. **Throws:** [SSLException](http://docs.google.com/javax/net/ssl/SSLException.html) - A problem was encountered while processing the data that caused the SSLEngine to abort. See the class description for more information on engine closure. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the preconditions on the offset and length parameters do not hold. [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - if the dst buffer is read-only. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if either srcs or dst is null, or if any element in the srcs subsequence specified is null. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the client/server mode has not yet been set.**See Also:**[GatheringByteChannel](http://docs.google.com/java/nio/channels/GatheringByteChannel.html), [GatheringByteChannel.write( ByteBuffer[], int, int)](http://docs.google.com/java/nio/channels/GatheringByteChannel.html#write(java.nio.ByteBuffer%5B%5D,%20int,%20int))

### unwrap

public [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) **unwrap**([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) src,  
 [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) dst)  
 throws [SSLException](http://docs.google.com/javax/net/ssl/SSLException.html)

Attempts to decode SSL/TLS network data into a plaintext application data buffer.

An invocation of this method behaves in exactly the same manner as the invocation:

[engine.unwrap(src, new ByteBuffer [] { dst }, 0, 1);](http://docs.google.com/javax/net/ssl/SSLEngine.html#unwrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer%5B%5D,%20int,%20int))

**Parameters:**src - a ByteBuffer containing inbound network data.dst - a ByteBuffer to hold inbound application data.  
**Returns:**an SSLEngineResult describing the result  
 of this operation.  
**Throws:**  
[SSLException](http://docs.google.com/javax/net/ssl/SSLException.html) - A problem was encountered while processing the  
 data that caused the SSLEngine to abort.  
 See the class description for more information on  
 engine closure.  
[ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - if the dst buffer is read-only.  
[IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if either src or dst  
 is null.  
[IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the client/server mode  
 has not yet been set.**See Also:**[unwrap(ByteBuffer, ByteBuffer [], int, int)](http://docs.google.com/javax/net/ssl/SSLEngine.html#unwrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer%5B%5D,%20int,%20int))

### unwrap

public [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) **unwrap**([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) src,  
 [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html)[] dsts)  
 throws [SSLException](http://docs.google.com/javax/net/ssl/SSLException.html)

Attempts to decode SSL/TLS network data into a sequence of plaintext application data buffers.

An invocation of this method behaves in exactly the same manner as the invocation:

[engine.unwrap(src, dsts, 0, dsts.length);](http://docs.google.com/javax/net/ssl/SSLEngine.html#unwrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer%5B%5D,%20int,%20int))

**Parameters:**src - a ByteBuffer containing inbound network data.dsts - an array of ByteBuffers to hold inbound  
 application data.  
**Returns:**an SSLEngineResult describing the result  
 of this operation.  
**Throws:**  
[SSLException](http://docs.google.com/javax/net/ssl/SSLException.html) - A problem was encountered while processing the  
 data that caused the SSLEngine to abort.  
 See the class description for more information on  
 engine closure.  
[ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - if any of the dst buffers are read-only.  
[IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if either src or dsts  
 is null, or if any element in dsts is null.  
[IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the client/server mode  
 has not yet been set.**See Also:**[unwrap(ByteBuffer, ByteBuffer [], int, int)](http://docs.google.com/javax/net/ssl/SSLEngine.html#unwrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer%5B%5D,%20int,%20int))

### unwrap

public abstract [SSLEngineResult](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) **unwrap**([ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html) src,  
 [ByteBuffer](http://docs.google.com/java/nio/ByteBuffer.html)[] dsts,  
 int offset,  
 int length)  
 throws [SSLException](http://docs.google.com/javax/net/ssl/SSLException.html)

Attempts to decode SSL/TLS network data into a subsequence of plaintext application data buffers. This *"scattering"* operation decodes, in a single invocation, a sequence of bytes into one or more of a given sequence of buffers. Scattering unwraps are often useful when implementing network protocols or file formats that, for example, group data into segments consisting of one or more fixed-length headers followed by a variable-length body. See [ScatteringByteChannel](http://docs.google.com/java/nio/channels/ScatteringByteChannel.html) for more information on scattering, and [ScatteringByteChannel.read(ByteBuffer[], int, int)](http://docs.google.com/java/nio/channels/ScatteringByteChannel.html#read(java.nio.ByteBuffer%5B%5D,%20int,%20int)) for more information on the subsequence behavior.

Depending on the state of the SSLEngine, this method may consume network data without producing any application data (for example, it may consume handshake data.)

The application is responsible for reliably obtaining the network data from the peer, and for invoking unwrap() on the data in the order it was received. The application must properly synchronize multiple calls to this method.

If this SSLEngine has not yet started its initial handshake, this method will automatically start the handshake.

This method will attempt to consume one complete SSL/TLS network packet, but will never consume more than the sum of the bytes remaining in the buffers. Each ByteBuffer's position is updated to reflect the amount of data consumed or produced. The limits remain the same.

The underlying memory used by the src and dsts ByteBuffers must not be the same.

The inbound network buffer may be modified as a result of this call: therefore if the network data packet is required for some secondary purpose, the data should be duplicated before calling this method. Note: the network data will not be useful to a second SSLEngine, as each SSLEngine contains unique random state which influences the SSL/TLS messages.

See the class description for more information on engine closure.

**Parameters:**src - a ByteBuffer containing inbound network data.dsts - an array of ByteBuffers to hold inbound application data.offset - The offset within the buffer array of the first buffer from which bytes are to be transferred; it must be non-negative and no larger than dsts.length.length - The maximum number of buffers to be accessed; it must be non-negative and no larger than dsts.length - offset. **Returns:**an SSLEngineResult describing the result of this operation. **Throws:** [SSLException](http://docs.google.com/javax/net/ssl/SSLException.html) - A problem was encountered while processing the data that caused the SSLEngine to abort. See the class description for more information on engine closure. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If the preconditions on the offset and length parameters do not hold. [ReadOnlyBufferException](http://docs.google.com/java/nio/ReadOnlyBufferException.html) - if any of the dst buffers are read-only. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if either src or dsts is null, or if any element in the dsts subsequence specified is null. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the client/server mode has not yet been set.**See Also:**[ScatteringByteChannel](http://docs.google.com/java/nio/channels/ScatteringByteChannel.html), [ScatteringByteChannel.read( ByteBuffer[], int, int)](http://docs.google.com/java/nio/channels/ScatteringByteChannel.html#read(java.nio.ByteBuffer%5B%5D,%20int,%20int))

### getDelegatedTask

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

Returns a delegated Runnable task for this SSLEngine.

SSLEngine operations may require the results of operations that block, or may take an extended period of time to complete. This method is used to obtain an outstanding [Runnable](http://docs.google.com/java/lang/Runnable.html) operation (task). Each task must be assigned a thread (possibly the current) to perform the [run](http://docs.google.com/java/lang/Runnable.html#run()) operation. Once the run method returns, the Runnable object is no longer needed and may be discarded.

Delegated tasks run in the AccessControlContext in place when this object was created.

A call to this method will return each outstanding task exactly once.

Multiple delegated tasks can be run in parallel.

**Returns:**a delegated Runnable task, or null if none are available.

### closeInbound

public abstract void **closeInbound**()  
 throws [SSLException](http://docs.google.com/javax/net/ssl/SSLException.html)

Signals that no more inbound network data will be sent to this SSLEngine.

If the application initiated the closing process by calling [closeOutbound()](http://docs.google.com/javax/net/ssl/SSLEngine.html#closeOutbound()), under some circumstances it is not required that the initiator wait for the peer's corresponding close message. (See section 7.2.1 of the TLS specification ([RFC 2246](http://www.ietf.org/rfc/rfc2246.txt)) for more information on waiting for closure alerts.) In such cases, this method need not be called.

But if the application did not initiate the closure process, or if the circumstances above do not apply, this method should be called whenever the end of the SSL/TLS data stream is reached. This ensures closure of the inbound side, and checks that the peer followed the SSL/TLS close procedure properly, thus detecting possible truncation attacks.

This method is idempotent: if the inbound side has already been closed, this method does not do anything.

[wrap()](http://docs.google.com/javax/net/ssl/SSLEngine.html#wrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer)) should be called to flush any remaining handshake data.

**Throws:** [SSLException](http://docs.google.com/javax/net/ssl/SSLException.html) - if this engine has not received the proper SSL/TLS close notification message from the peer.**See Also:**[isInboundDone()](http://docs.google.com/javax/net/ssl/SSLEngine.html#isInboundDone()), [isOutboundDone()](http://docs.google.com/javax/net/ssl/SSLEngine.html#isOutboundDone())

### isInboundDone

public abstract boolean **isInboundDone**()

Returns whether [unwrap(ByteBuffer, ByteBuffer)](http://docs.google.com/javax/net/ssl/SSLEngine.html#unwrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer)) will accept any more inbound data messages.

**Returns:**true if the SSLEngine will not consume anymore network data (and by implication, will not produce any more application data.)**See Also:**[closeInbound()](http://docs.google.com/javax/net/ssl/SSLEngine.html#closeInbound())

### closeOutbound

public abstract void **closeOutbound**()

Signals that no more outbound application data will be sent on this SSLEngine.

This method is idempotent: if the outbound side has already been closed, this method does not do anything.

[wrap(ByteBuffer, ByteBuffer)](http://docs.google.com/javax/net/ssl/SSLEngine.html#wrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer)) should be called to flush any remaining handshake data.

**See Also:**[isOutboundDone()](http://docs.google.com/javax/net/ssl/SSLEngine.html#isOutboundDone())

### isOutboundDone

public abstract boolean **isOutboundDone**()

Returns whether [wrap(ByteBuffer, ByteBuffer)](http://docs.google.com/javax/net/ssl/SSLEngine.html#wrap(java.nio.ByteBuffer,%20java.nio.ByteBuffer)) will produce any more outbound data messages.

Note that during the closure phase, a SSLEngine may generate handshake closure data that must be sent to the peer. wrap() must be called to generate this data. When this method returns true, no more outbound data will be created.

**Returns:**true if the SSLEngine will not produce any more network data**See Also:**[closeOutbound()](http://docs.google.com/javax/net/ssl/SSLEngine.html#closeOutbound()), [closeInbound()](http://docs.google.com/javax/net/ssl/SSLEngine.html#closeInbound())

### 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 engine. 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/SSLEngine.html#getEnabledCipherSuites()), [setEnabledCipherSuites(String [])](http://docs.google.com/javax/net/ssl/SSLEngine.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 engine. When an SSLEngine 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/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/SSLEngine.html#getSupportedCipherSuites()), [setEnabledCipherSuites(String [])](http://docs.google.com/javax/net/ssl/SSLEngine.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 engine.

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/SSLEngine.html#getEnabledCipherSuites()) for more information on why a specific cipher suite may never be used on a engine.

**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/SSLEngine.html#getSupportedCipherSuites()), [getEnabledCipherSuites()](http://docs.google.com/javax/net/ssl/SSLEngine.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 with this SSLEngine.

**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 with this SSLEngine.

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

### setEnabledProtocols

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

Set the protocol versions enabled for use on this engine.

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/SSLEngine.html#getEnabledProtocols())

### getSession

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

Returns the SSLSession in use in this SSLEngine.

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.

Unlike [SSLSocket.getSession()](http://docs.google.com/javax/net/ssl/SSLSocket.html#getSession()) this method does not block until handshaking is complete.

Until the initial handshake has completed, this method returns a session object which reports an invalid cipher suite of "SSL\_NULL\_WITH\_NULL\_NULL".

**Returns:**the SSLSession for this SSLEngine**See Also:**[SSLSession](http://docs.google.com/javax/net/ssl/SSLSession.html)

### beginHandshake

public abstract void **beginHandshake**()  
 throws [SSLException](http://docs.google.com/javax/net/ssl/SSLException.html)

Initiates handshaking (initial or renegotiation) on this SSLEngine.

This method is not needed for the initial handshake, as the wrap() and unwrap() methods will implicitly call this method if handshaking has not already begun.

Note that the peer may also request a session renegotiation with this SSLEngine by sending the appropriate session renegotiate handshake message.

Unlike the [SSLSocket#startHandshake()](http://docs.google.com/javax/net/ssl/SSLSocket.html#startHandshake()) method, this method does not block until handshaking is completed.

To force a complete SSL/TLS session renegotiation, the current session should be invalidated prior to calling this method.

Some protocols may not support multiple handshakes on an existing engine and may throw an SSLException.

**Throws:** [SSLException](http://docs.google.com/javax/net/ssl/SSLException.html) - if a problem was encountered while signaling the SSLEngine to begin a new handshake. See the class description for more information on engine closure. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the client/server mode has not yet been set.**See Also:**[SSLSession.invalidate()](http://docs.google.com/javax/net/ssl/SSLSession.html#invalidate())

### getHandshakeStatus

public abstract [SSLEngineResult.HandshakeStatus](http://docs.google.com/javax/net/ssl/SSLEngineResult.HandshakeStatus.html) **getHandshakeStatus**()

Returns the current handshake status for this SSLEngine.

**Returns:**the current SSLEngineResult.HandshakeStatus.

### setUseClientMode

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

Configures the engine 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 engine.

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

**Parameters:**mode - true if the engine 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/SSLEngine.html#getUseClientMode())

### getUseClientMode

public abstract boolean **getUseClientMode**()

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

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

### setNeedClientAuth

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

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

An engine'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/SSLEngine.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 engine will begin its closure procedure*.

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

### getNeedClientAuth

public abstract boolean **getNeedClientAuth**()

Returns true if the engine will *require* client authentication. This option is only useful to engines 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/SSLEngine.html#setNeedClientAuth(boolean)), [setWantClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLEngine.html#setWantClientAuth(boolean)), [getWantClientAuth()](http://docs.google.com/javax/net/ssl/SSLEngine.html#getWantClientAuth()), [setUseClientMode(boolean)](http://docs.google.com/javax/net/ssl/SSLEngine.html#setUseClientMode(boolean))

### setWantClientAuth

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

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

An engine'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/SSLEngine.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/SSLEngine.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/SSLEngine.html#getWantClientAuth()), [setNeedClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLEngine.html#setNeedClientAuth(boolean)), [getNeedClientAuth()](http://docs.google.com/javax/net/ssl/SSLEngine.html#getNeedClientAuth()), [setUseClientMode(boolean)](http://docs.google.com/javax/net/ssl/SSLEngine.html#setUseClientMode(boolean))

### getWantClientAuth

public abstract boolean **getWantClientAuth**()

Returns true if the engine will *request* client authentication. This option is only useful for engines 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/SSLEngine.html#setNeedClientAuth(boolean)), [getNeedClientAuth()](http://docs.google.com/javax/net/ssl/SSLEngine.html#getNeedClientAuth()), [setWantClientAuth(boolean)](http://docs.google.com/javax/net/ssl/SSLEngine.html#setWantClientAuth(boolean)), [setUseClientMode(boolean)](http://docs.google.com/javax/net/ssl/SSLEngine.html#setUseClientMode(boolean))

### setEnableSessionCreation

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

Controls whether new SSL sessions may be established by this engine. 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/SSLEngine.html#getEnableSessionCreation())

### getEnableSessionCreation

public abstract boolean **getEnableSessionCreation**()

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

**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/SSLEngine.html#setEnableSessionCreation(boolean))

### getSSLParameters

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

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

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

### setSSLParameters

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

Applies SSLParameters to this engine.

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/SSLEngine.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/SSLContextSpi.html)   [**NEXT CLASS**](http://docs.google.com/javax/net/ssl/SSLEngineResult.html) | [**FRAMES**](http://docs.google.com/index.html?javax/net/ssl/SSLEngine.html)    [**NO FRAMES**](http://docs.google.com/SSLEngine.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#2s8eyo1) |

[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).