| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/KeyAgreementSpi.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/crypto/KeyAgreement.html)   [**NEXT CLASS**](http://docs.google.com/javax/crypto/KeyGenerator.html) | [**FRAMES**](http://docs.google.com/index.html?javax/crypto/KeyAgreementSpi.html)    [**NO FRAMES**](http://docs.google.com/KeyAgreementSpi.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

## **javax.crypto**

Class KeyAgreementSpi

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.crypto.KeyAgreementSpi**

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

This class defines the *Service Provider Interface* (**SPI**) for the KeyAgreement class. All the abstract methods in this class must be implemented by each cryptographic service provider who wishes to supply the implementation of a particular key agreement algorithm.

The keys involved in establishing a shared secret are created by one of the key generators (KeyPairGenerator or KeyGenerator), a KeyFactory, or as a result from an intermediate phase of the key agreement protocol ([engineDoPhase](http://docs.google.com/javax/crypto/KeyAgreementSpi.html#engineDoPhase(java.security.Key,%20boolean))).

For each of the correspondents in the key exchange, engineDoPhase needs to be called. For example, if the key exchange is with one other party, engineDoPhase needs to be called once, with the lastPhase flag set to true. If the key exchange is with two other parties, engineDoPhase needs to be called twice, the first time setting the lastPhase flag to false, and the second time setting it to true. There may be any number of parties involved in a key exchange.

**Since:** 1.4 **See Also:**[KeyGenerator](http://docs.google.com/javax/crypto/KeyGenerator.html), [SecretKey](http://docs.google.com/javax/crypto/SecretKey.html)

| **Constructor Summary** | |
| --- | --- |
| [**KeyAgreementSpi**](http://docs.google.com/javax/crypto/KeyAgreementSpi.html#KeyAgreementSpi())() |

| **Method Summary** | |
| --- | --- |
| protected abstract  [Key](http://docs.google.com/java/security/Key.html) | [**engineDoPhase**](http://docs.google.com/javax/crypto/KeyAgreementSpi.html#engineDoPhase(java.security.Key,%20boolean))([Key](http://docs.google.com/java/security/Key.html) key, boolean lastPhase)            Executes the next phase of this key agreement with the given key that was received from one of the other parties involved in this key agreement. |
| protected abstract  byte[] | [**engineGenerateSecret**](http://docs.google.com/javax/crypto/KeyAgreementSpi.html#engineGenerateSecret())()            Generates the shared secret and returns it in a new buffer. |
| protected abstract  int | [**engineGenerateSecret**](http://docs.google.com/javax/crypto/KeyAgreementSpi.html#engineGenerateSecret(byte%5B%5D,%20int))(byte[] sharedSecret, int offset)            Generates the shared secret, and places it into the buffer sharedSecret, beginning at offset inclusive. |
| protected abstract  [SecretKey](http://docs.google.com/javax/crypto/SecretKey.html) | [**engineGenerateSecret**](http://docs.google.com/javax/crypto/KeyAgreementSpi.html#engineGenerateSecret(java.lang.String))([String](http://docs.google.com/java/lang/String.html) algorithm)            Creates the shared secret and returns it as a secret key object of the requested algorithm type. |
| protected abstract  void | [**engineInit**](http://docs.google.com/javax/crypto/KeyAgreementSpi.html#engineInit(java.security.Key,%20java.security.spec.AlgorithmParameterSpec,%20java.security.SecureRandom))([Key](http://docs.google.com/java/security/Key.html) key, [AlgorithmParameterSpec](http://docs.google.com/java/security/spec/AlgorithmParameterSpec.html) params, [SecureRandom](http://docs.google.com/java/security/SecureRandom.html) random)            Initializes this key agreement with the given key, set of algorithm parameters, and source of randomness. |
| protected abstract  void | [**engineInit**](http://docs.google.com/javax/crypto/KeyAgreementSpi.html#engineInit(java.security.Key,%20java.security.SecureRandom))([Key](http://docs.google.com/java/security/Key.html) key, [SecureRandom](http://docs.google.com/java/security/SecureRandom.html) random)            Initializes this key agreement with the given key and source of randomness. |

| **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** |
| --- |

### KeyAgreementSpi

public **KeyAgreementSpi**()

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

### engineInit

protected abstract void **engineInit**([Key](http://docs.google.com/java/security/Key.html) key,  
 [SecureRandom](http://docs.google.com/java/security/SecureRandom.html) random)  
 throws [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html)

Initializes this key agreement with the given key and source of randomness. The given key is required to contain all the algorithm parameters required for this key agreement.

If the key agreement algorithm requires random bytes, it gets them from the given source of randomness, random. However, if the underlying algorithm implementation does not require any random bytes, random is ignored.

**Parameters:**key - the party's private information. For example, in the case of the Diffie-Hellman key agreement, this would be the party's own Diffie-Hellman private key.random - the source of randomness **Throws:** [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html) - if the given key is inappropriate for this key agreement, e.g., is of the wrong type or has an incompatible algorithm type.

### engineInit

protected abstract void **engineInit**([Key](http://docs.google.com/java/security/Key.html) key,  
 [AlgorithmParameterSpec](http://docs.google.com/java/security/spec/AlgorithmParameterSpec.html) params,  
 [SecureRandom](http://docs.google.com/java/security/SecureRandom.html) random)  
 throws [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html),  
 [InvalidAlgorithmParameterException](http://docs.google.com/java/security/InvalidAlgorithmParameterException.html)

Initializes this key agreement with the given key, set of algorithm parameters, and source of randomness.

**Parameters:**key - the party's private information. For example, in the case of the Diffie-Hellman key agreement, this would be the party's own Diffie-Hellman private key.params - the key agreement parametersrandom - the source of randomness **Throws:** [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html) - if the given key is inappropriate for this key agreement, e.g., is of the wrong type or has an incompatible algorithm type. [InvalidAlgorithmParameterException](http://docs.google.com/java/security/InvalidAlgorithmParameterException.html) - if the given parameters are inappropriate for this key agreement.

### engineDoPhase

protected abstract [Key](http://docs.google.com/java/security/Key.html) **engineDoPhase**([Key](http://docs.google.com/java/security/Key.html) key,  
 boolean lastPhase)  
 throws [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html),  
 [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html)

Executes the next phase of this key agreement with the given key that was received from one of the other parties involved in this key agreement.

**Parameters:**key - the key for this phase. For example, in the case of Diffie-Hellman between 2 parties, this would be the other party's Diffie-Hellman public key.lastPhase - flag which indicates whether or not this is the last phase of this key agreement. **Returns:**the (intermediate) key resulting from this phase, or null if this phase does not yield a key **Throws:** [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html) - if the given key is inappropriate for this phase. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this key agreement has not been initialized.

### engineGenerateSecret

protected abstract byte[] **engineGenerateSecret**()  
 throws [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html)

Generates the shared secret and returns it in a new buffer.

This method resets this KeyAgreementSpi object, so that it can be reused for further key agreements. Unless this key agreement is reinitialized with one of the engineInit methods, the same private information and algorithm parameters will be used for subsequent key agreements.

**Returns:**the new buffer with the shared secret **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this key agreement has not been completed yet

### engineGenerateSecret

protected abstract int **engineGenerateSecret**(byte[] sharedSecret,  
 int offset)  
 throws [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html),  
 [ShortBufferException](http://docs.google.com/javax/crypto/ShortBufferException.html)

Generates the shared secret, and places it into the buffer sharedSecret, beginning at offset inclusive.

If the sharedSecret buffer is too small to hold the result, a ShortBufferException is thrown. In this case, this call should be repeated with a larger output buffer.

This method resets this KeyAgreementSpi object, so that it can be reused for further key agreements. Unless this key agreement is reinitialized with one of the engineInit methods, the same private information and algorithm parameters will be used for subsequent key agreements.

**Parameters:**sharedSecret - the buffer for the shared secretoffset - the offset in sharedSecret where the shared secret will be stored **Returns:**the number of bytes placed into sharedSecret **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this key agreement has not been completed yet [ShortBufferException](http://docs.google.com/javax/crypto/ShortBufferException.html) - if the given output buffer is too small to hold the secret

### engineGenerateSecret

protected abstract [SecretKey](http://docs.google.com/javax/crypto/SecretKey.html) **engineGenerateSecret**([String](http://docs.google.com/java/lang/String.html) algorithm)  
 throws [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html),  
 [NoSuchAlgorithmException](http://docs.google.com/java/security/NoSuchAlgorithmException.html),  
 [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html)

Creates the shared secret and returns it as a secret key object of the requested algorithm type.

This method resets this KeyAgreementSpi object, so that it can be reused for further key agreements. Unless this key agreement is reinitialized with one of the engineInit methods, the same private information and algorithm parameters will be used for subsequent key agreements.

**Parameters:**algorithm - the requested secret key algorithm **Returns:**the shared secret key **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this key agreement has not been completed yet [NoSuchAlgorithmException](http://docs.google.com/java/security/NoSuchAlgorithmException.html) - if the requested secret key algorithm is not available [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html) - if the shared secret key material cannot be used to generate a secret key of the requested algorithm type (e.g., the key material is too short)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/KeyAgreementSpi.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/crypto/KeyAgreement.html)   [**NEXT CLASS**](http://docs.google.com/javax/crypto/KeyGenerator.html) | [**FRAMES**](http://docs.google.com/index.html?javax/crypto/KeyAgreementSpi.html)    [**NO FRAMES**](http://docs.google.com/KeyAgreementSpi.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

[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).