**The AlgorithmParameters Class**

The AlgorithmParameters class is an [engine class](https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html#Engine) that provides an opaque representation of cryptographic parameters. You can initialize the AlgorithmParameters class using a specific AlgorithmParameterSpec object, or by encoding the parameters in a recognized format. You can retrieve the resulting specification with the getParameterSpec method (see the following section).

**Creating an AlgorithmParameters Object**

AlgorithmParameters objects are obtained by using one of the AlgorithmParameters [getInstance() static factory methods](https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html#ProviderImplReq).

**Initializing an AlgorithmParameters Object**

Once an AlgorithmParameters object is instantiated, it must be initialized via a call to init, using an appropriate parameter specification or parameter encoding:

void init(AlgorithmParameterSpec paramSpec)

void init(byte[] params)

void init(byte[] params, String format)

In these init methods, params is an array containing the encoded parameters, and format is the name of the decoding format. In the init method with a params argument but no format argument, the primary decoding format for parameters is used. The primary decoding format is ASN.1, if an ASN.1 specification for the parameters exists.

**NOTE:** AlgorithmParameters objects can be initialized only once. They are not reusable.

**Obtaining the Encoded Parameters**

A byte encoding of the parameters represented in an AlgorithmParameters object may be obtained via a call to getEncoded:

byte[] getEncoded()

This method returns the parameters in their primary encoding format. The primary encoding format for parameters is ASN.1, if an ASN.1 specification for this type of parameters exists.

If you want the parameters returned in a specified encoding format, use

byte[] getEncoded(String format)

If format is null, the primary encoding format for parameters is used, as in the other getEncoded method.

**NOTE:** In the default AlgorithmParameters implementation, supplied by the SUN provider, the format argument is currently ignored.

**Converting an AlgorithmParameters Object to a Transparent Specification**

A transparent parameter specification for the algorithm parameters may be obtained from an AlgorithmParameters object via a call to getParameterSpec:

AlgorithmParameterSpec getParameterSpec(Class paramSpec)

paramSpec identifies the specification class in which the parameters should be returned. The specification class could be, for example, DSAParameterSpec.class to indicate that the parameters should be returned in an instance of the DSAParameterSpec class. (This class is in the java.security.spec package.)

**The AlgorithmParameterGenerator Class**

The AlgorithmParameterGenerator class is an [engine class](https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html#Engine) used to generate a set of **brand-new** parameters suitable for a certain algorithm (the algorithm is specified when an AlgorithmParameterGenerator instance is created). This object is used when you do not have an existing set of algorithm parameters, and want to generate one from scratch.

**Creating an AlgorithmParameterGenerator Object**

AlgorithmParameterGenerator objects are obtained by using one of the AlgorithmParameterGenerator [getInstance() static factory methods](https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html#ProviderImplReq).

**Initializing an AlgorithmParameterGenerator Object**

The AlgorithmParameterGenerator object can be initialized in two different ways: an algorithm-independent manner or an algorithm-specific manner.

The algorithm-independent approach uses the fact that all parameter generators share the concept of a "size" and a source of randomness. The measure of size is universally shared by all algorithm parameters, though it is interpreted differently for different algorithms. For example, in the case of parameters for the DSA algorithm, "size" corresponds to the size of the prime modulus, in bits. (See the [Standard Names](https://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html) document for information about the sizes for specific algorithms.) When using this approach, algorithm-specific parameter generation values--if any--default to some standard values. One init method that takes these two universally shared types of arguments:

void init(int size, SecureRandom random);

Another init method takes only a size argument and uses a system-provided source of randomness:

void init(int size)

A third approach initializes a parameter generator object using algorithm-specific semantics, which are represented by a set of algorithm-specific parameter generation values supplied in an AlgorithmParameterSpec object:

void init(AlgorithmParameterSpec genParamSpec,

SecureRandom random)

void init(AlgorithmParameterSpec genParamSpec)

To generate Diffie-Hellman system parameters, for example, the parameter generation values usually consist of the size of the prime modulus and the size of the random exponent, both specified in number of bits.

**Generating Algorithm Parameters**

Once you have created and initialized an AlgorithmParameterGenerator object, you can use the generateParameters method to generate the algorithm parameters:

AlgorithmParameters generateParameters()