# **Java Keytool**

The *Java Keytool* is a command line tool which can generate public key / private key pairs and store them in a [**Java KeyStore**](http://tutorials.jenkov.com/java-cryptography/keystore.html). The Keytool executable is distributed with the Java SDK (or JRE), so if you have an SDK installed you will also have the Keytool executable.

The Keytool executable is called keytool. To execute it, open a command line (cmd, console, shell etc.). and change directory into the bin directory of your Java SDK installation. Type keytool followed by pressing the Enter key. You should see something similar to this:

C:\Program Files\Java\jdk1.8.0\_111\bin>keytool

Key and Certificate Management Tool

Commands:

-certreq Generates a certificate request

-changealias Changes an entry's alias

-delete Deletes an entry

-exportcert Exports certificate

-genkeypair Generates a key pair

-genseckey Generates a secret key

-gencert Generates certificate from a certificate request

-importcert Imports a certificate or a certificate chain

-importpass Imports a password

-importkeystore Imports one or all entries from another keystore

-keypasswd Changes the key password of an entry

-list Lists entries in a keystore

-printcert Prints the content of a certificate

-printcertreq Prints the content of a certificate request

-printcrl Prints the content of a CRL file

-storepasswd Changes the store password of a keystore

Use "keytool -command\_name -help" for usage of command\_name

C:\Program Files\Java\jdk1.8.0\_111\bin>

As you can see, keytool supports a set of commands to work with keys, certificates and key stores. This Java Keytool tutorial will cover the most commonly used of these commands.

**Keytool Scripts**

Keytool commands take a lot of arguments which may be hard to remember to set correctly. Therefore it is a good idea to create some Keytool CMD or Shell scripts with the Keytool commands in. The scripts makes it easier to re-execute the keytool commands later on, and makes it possible to go back later and see how a KeyStore was generated.

**Generate Key Pair**

Generating a public key / private key pair is one of the most common tasks to use the Java Keytool for. The generated key pair is inserted into a Java KeyStore file as a self signed key pair. Here is the general command line format for generating a key pair with the Keytool:

-genkeypair

-alias alias

-keyalg keyalg

-keysize keysize

-sigalg sigalg

-dname dname

-keypass keypass

-validity valDays

-storetype storetype

-keystore keystore

-storepass storepass

-providerClass provider\_class\_name

-providerArg provider\_arg

-v

-protected

-Jjavaoption

The arguments are explained in the **[Keytool Arguments](http://tutorials.jenkov.com/java-cryptography/keytool.html" \l "keytool-arguments)** section. Not all of these arguments are needed. Many are optional. The Keytool will tell you if you are missing a required argument.

The line breaks should not be included in the final command. The line breaks are only there to make the command format easier to read here.

Here is an example keytool -genkeypair command. Remember to remove the line breaks before trying it out!

"C:\\Program Files\Java\jdk1.8.0\_111\bin\keytool" -genkeypair

-alias testkey

-keyalg RSA

-keysize 2048

-dname "CN=Jakob Jenkov, OU=Jenkov Aps, O=Jenkov Aps, L=Copenhagen, ST=Unknown, C=DK"

-keypass 123456

-validity 100

-storetype JKS

-keystore keystore.jks

-storepass abcdef

**Export Certificate**

The Java Keytool can also export certificates stored in a KeyStore. Here is how the Keytool command looks for exporting certificates:

-exportcert

-alias alias

-file cert\_file

-storetype storetype

-keystore keystore

-storepass storepass

-providerName provider\_name

-providerClass provider\_class\_name

-providerArg provider\_arg

-rfc

-v

-protected

-Jjavaoption

The arguments are explained in the **[Keytool Arguments](http://tutorials.jenkov.com/java-cryptography/keytool.html" \l "keytool-arguments)** section. Not all of these arguments are needed. Many are optional. The Keytool will tell you if you are missing a required argument.

Here is a Keytool command example that exports the certificate for a key pair. Remember to remove the line breaks when entering the command on the command line.

"C:\\Program Files\Java\jdk1.8.0\_111\bin\keytool"

-exportcert

-alias testkey

-keypass 123456

-storetype JKS

-keystore keystore.jks

-file cert.cert

-rfc

-storepass abcdef

**Import Certificate**

The Java Keytool can also import certificates into a KeyStore. Here is how the Keytool command looks for importing certificates:

-importcert

-alias alias

-file cert\_file

-keypass keypass

-noprompt

-trustcacerts

-storetype storetype

-keystore keystore

-storepass storepass

-providerName provider\_name

-providerClass provider\_class\_name

-providerArg provider\_arg

-v

-protected

-Jjavaoption

The arguments are explained in the **[Keytool Arguments](http://tutorials.jenkov.com/java-cryptography/keytool.html" \l "keytool-arguments)** section. Not all of these arguments are needed. Many are optional. The Keytool will tell you if you are missing a required argument.

Here is an example Keytool command that imports a certificate into a KeyStore. Remember to remove the line breaks when entering the command on the command line.

"C:\\Program Files\Java\jdk1.8.0\_111\bin\keytool"

-importcert

-alias testkey

-keypass 123456

-storetype JKS

-keystore keystore2.jks

-file cert.cert

-rfc

-storepass abcdef

**List KeyStore Entries**

To list the entries in a Java KeyStore you can use the Keytool -list command. Here is the format for the Keytool -list command. The line breaks are only here to make the command format easier to read. Remove the line breaks before running the command.

-list

-alias alias

-storetype storetype

-keystore keystore

-storepass storepass

-providerName provider\_name

-providerClass provider\_class\_name

-providerArg provider\_arg

-v

-rfc

-protected

-Jjavaoption

The arguments are explained in the **[Keytool Arguments](http://tutorials.jenkov.com/java-cryptography/keytool.html" \l "keytool-arguments)** section. Not all of these arguments are needed. Many are optional. The Keytool will tell you if you are missing a required argument.

Here is a Keytool -list command example. Remember to remove the line breaks!

"C:\\Program Files\Java\jdk1.8.0\_111\bin\keytool"

-list

-storetype JKS

-keystore keystore.jks

-storepass abcdef

This Keytool -list command will list all entries in the given KeyStore. The output of running this Keytool -list command will look similar to this:

Keystore type: JKS

Keystore provider: SUN

Your keystore contains 1 entry

testkey, 19-Dec-2017, PrivateKeyEntry,

Certificate fingerprint (SHA1): 4F:4C:E2:C5:DA:36:E6:A9:93:6F:10:36:9E:E5:E8:5A:6E:F2:11:16

If you include an -alias argument in the Keytool -list command, then only the entry matching the given alias will get listed. Here is an example Keytool -list command with an -alias argument:

"C:\\Program Files\Java\jdk1.8.0\_111\bin\keytool"

-list

-alias testkey

-storetype JKS

-keystore keystore.jks

-storepass abcdef

The output of running the above Keytool -list command will look similar to this:

y -storetype JKS -keystore keystore.jks -storepass abcdef

testkey, 15-Dec-2017, PrivateKeyEntry,

Certificate fingerprint (SHA1): 71:B0:6E:F1:E9:5A:E7:F5:5E:78:71:DC:08:80:47:E9:5F:F8:6D:25

**Delete KeyStore Entry**

The Keytool has a command that can delete a key entry in a Java KeyStore. The Keytool command for deleting keys is -delete. Here is the format of the Keytool -delete command:

-delete

-alias alias

-storetype storetype

-keystore keystore

-storepass storepass

-providerName provider\_name

-providerClass provider\_class\_name

-providerArg provider\_arg

-v

-protected

-Jjavaoption

The arguments are explained in the **[Keytool Arguments](http://tutorials.jenkov.com/java-cryptography/keytool.html" \l "keytool-arguments)** section. Not all of these arguments are needed. Many are optional. The Keytool will tell you if you are missing a required argument.

Here is a Keytool -delete command example. Remember to remove the line breaks before running it!

"C:\\Program Files\Java\jdk1.8.0\_111\bin\keytool"

-delete

-alias testkey

-storetype JKS

-keystore keystore.jks

-storepass abcdef

This Keytool -delete command will remove the KeyStore entry with the alias testkey from the KeyStore stored in the file keystore.jks .

**Generate a Certificate Request**

The Java Keytool can generate a certificate request using the -certreq command. A certificate request is a request for a certificate authority (CA) to create a public certificate for your organization. Once generated, the certificate request should be sent to the CA you want to create a certificate for you (e.g. Verisign, Thawte, or some other CA).

Before you can generate a certificate request for a private key, public key pair, you must have generated that private key, public key pair into the Keystore (or imported it). See elsewhere in this Java Keytool tutorial to see how to do that.

Here is the command format for generating a certificate request. Remember to remove all line breaks when trying out this command:

-certreq

-alias alias

-sigalg sigalg

-file certreq\_file

-keypass keypass

-storetype storetype

-keystore keystore

-storepass storepass

-providerName provider\_name

-providerClass provider\_class\_name

-providerArg provider\_arg

-v

-protected

-Jjavaoption

The arguments are explained in the **[Keytool Arguments](http://tutorials.jenkov.com/java-cryptography/keytool.html" \l "keytool-arguments)** section. Not all of these arguments are needed. Many are optional. The Keytool will tell you if you are missing a required argument.

Here is a Java Keytool -certreq command example:

"C:\\Program Files\Java\jdk1.8.0\_111\bin\keytool"

-certreq

-alias testkey

-keypass 123456

-storetype JKS

-keystore keystore.jks

-storepass abcdef

-file certreq.certreq

This command will generate a certificate request for the key stored with alias testkey in the keystore file keystore.jks, and write the certificate request into the file named certreq.certreq .

Remember, the line breaks are only included to make the command easier to read. Omit them when typing in the command on the command line yourself.

**Keytool Arguments**

Below is a list of the arguments the various Keytool commands take. Please keep in mind that not all commands accept all of these arguments. Look at the concrete command to see what arguments it takes.

|  |  |
| --- | --- |
| **Argument** | **Description** |
| -alias | The name in the Java KeyStore the generated key should be identified by. Remember, an alias can only point to one key. |
| -keyalg | The name of the algorithm used to generate the key. A common value is RSA meaning the RSA algorithm should be used to generate the key pair. |
| -keysize | The size in bits of the key to generate. Normally key sizes are multiples of 8 which aligns with a number of bytes. Additionally, different algorithms may only support certain preset key sizes. You will need to check what the key size should be for the key you want to generate. |
| -sigalg | The signature algorithm used to sign the key pair. |
| -dname | The Distinguished Name from the X.500 standard. This name will be associated with the alias for this key pair in the KeyStore. The dname is also used as the "issuer" and "subject" fields in the self signed certificate. |
| -keypass | The key pair password needed to access this specific key pair within the KeyStore. |
| -validity | The number of days the certificate attached to the key pair should be valid. |
| -storetype | The file format the KeyStore should be saved in. The default is JKS. Another option is the value PKCS11 which represents the standard PKCS11 format. |
| -keystore | The name of the KeyStore file to store the generated key pair in. If the file does not exist, it will be created. |
| -file | The name of the file to read from or write to (certificate or certificate request). |
| -storepass | The password for the whole KeyStore. Anyone who wants to open this KeyStore later will need this password. The storepass is not the same as the keypass. The keypasspassword only counts for a single key. You will need both the KeyStore password and the key password to access any given key stored in a KeyStore. |
| -rfc | If this flag is included (it has no value following it) then Keytool will use a textual format rather than binary format e.g. for export or import of certificates. The value -rfc refers to the RFC 1421 standard. |
| -providerName | The name of the cryptographic API provider you want to use (if any) when generating the key pair. The provider name must be listed in the Java security property files for this to work. |
| -providerClass | The name of the root class of the cryptographic API provider you want to use. Use this when the provider name is not listed in the Java security property files. |
| -providerArg | Arguments you can pass to your cryptographic provider at initialization (if needed by the provider). |
| -v | Short for "verbose" (?!?), meaning the Keytool will print out a lot of extra information into the command line in a humanly readable format. |
| -protected | Specifies whether or not the KeyStore password should be provided by some external mechanism like a pin reader. Valid values are true and false. |
| -Jjavaoption | A Java option string (Java VM options) which can be passed to the Java VM that generates the key pair and creates the KeyStore. |