# [What Should your App Include in Settings?](https://incipia.co/post/app-development/what-should-your-app-include-in-settings/)

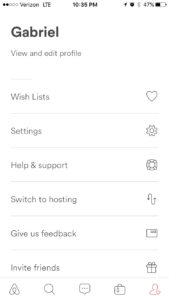
**Gabe Kwakyi | September 22, 2016**

One question that may not come up as a hot topic when planning out a mobile app roadmap, yet which matters very much to the user experience is:

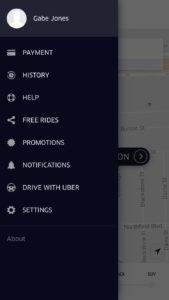
What should be include in an app's settings screen?

The settings screen is where users will go when they are confused and looking for guidance, making this screen vital to retaining users who are having trouble with an app. Providing a useful set of options or settings can produce improved retention, fewer angry reviews and more valuable feedback, all of which are required to improve an app's performance, from a design, development and marketing perspective.  
  
So what should go in your settings screen? Here are a 5 examples of settings screens in diverse, yet popular apps:

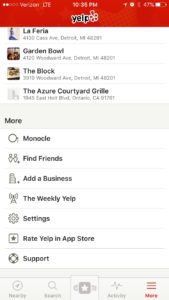
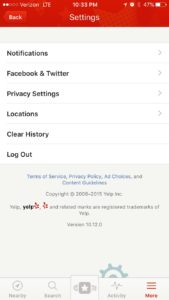
### Airbnb

[](https://incipia.co/wp-content/uploads/2016/09/airbnb1.jpeg) [](https://incipia.co/wp-content/uploads/2016/09/airbnb2.jpeg)

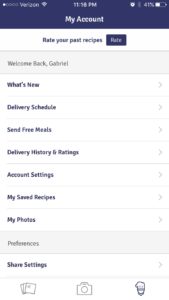
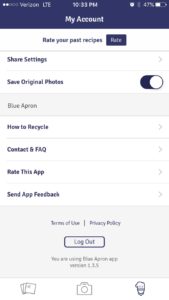
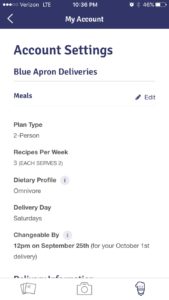
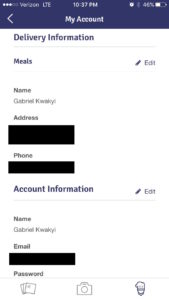
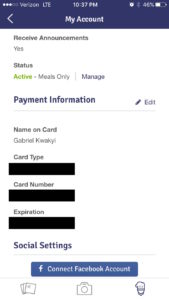
### Uber

[](https://incipia.co/wp-content/uploads/2016/09/uber1.jpeg) [](https://incipia.co/wp-content/uploads/2016/09/uber2.jpeg)

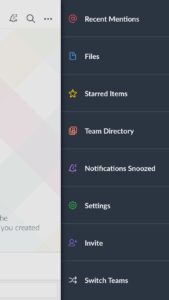
### Yelp

[](https://incipia.co/wp-content/uploads/2016/09/yelp1.jpeg) [](https://incipia.co/wp-content/uploads/2016/09/yelp2.jpeg)

### Blue Apron

[](https://incipia.co/wp-content/uploads/2016/09/blueaprona.jpeg) [](https://incipia.co/wp-content/uploads/2016/09/blueapron1.jpeg) [](https://incipia.co/wp-content/uploads/2016/09/blueapron2.jpeg) [](https://incipia.co/wp-content/uploads/2016/09/blueapron3.jpeg) [](https://incipia.co/wp-content/uploads/2016/09/blueapron4.jpeg)

### Slack

[](https://incipia.co/wp-content/uploads/2016/09/slack1.jpeg) [](https://incipia.co/wp-content/uploads/2016/09/slack2-1.jpeg)  
  
As you can see, many apps opt for a two-stage settings experience by presenting users with more important options relevant to the app after tapping a profile/hamburger menu UI element, with more general, less important options stored behind the actual settings cog or equivalent UI element. While the answer to the question posed at the start of this post will naturally depend based on what product/service the app offers, there are some generally applicable settings to keep in mind, such as:

* Cross your *t*s and dot your *i*s with a **terms of service/privacy policy**.
* **Account/login/logout/password** **recovery** – funny enough, sometimes apps forget to let people disconnect from their app or reset their password! If your app has a profile, it's also expected that users are able to edit their account details, and apps should provide access to **social logins** or **integrations** in settings as well.
  + For e-commerce apps, allowing users to edit/add/delete a **credit card** is a must.
* **Support/feedback/contact us/FAQ** – because the settings screen is one of the main locations that people will look to when they are seeking help, this is a great opportunity to help resolve their issues.
* While the conversion rate of a **rate us** or**share this app** callout in the settings screen is low, it is nonetheless an easy, low-risk place to encourage more app reviews and shares.
* Most apps also include information on the **current version** of an app and an **about** section to provide recognition for the team or any 3rd party citations.
* Allowing users to adjust **notifications** from within the app can help users customize their notifications and also allows apps trade the ability to send a few types of notifications, for retaining permissions in general and preventing users from revoking notification rights in Apple's settings.
* If the app has **sound effects**, the settings should provide an option to toggle these on/off.

# Configurations and Concurrent Access

Configuration objects are often central resources of an application and are accessed by multiple components. If multiple threads are involved which read or even update configuration data, care has to be taken that access to a Configuration object is properly synchronized to avoid data corruption or spurious exceptions. This section of the user's guide deals with concurrency and describes the actions necessary to make a Configuration work in a multi-threaded environment.

## Synchronizers

Whether a Configuration object has to be thread-safe or not strongly depends on a concrete use case. For an application which only reads some configuration properties in its main() method at startup, it does not matter whether this configuration can safely be accessed from multiple threads. In this case, the overhead of synchronizing access to the configuration is not needed, and thus operations on the Configuration object can be more efficient. On the other hand, if the Configuration object is accessed by multiple components running in different threads it should better be thread-safe.

To support these different use cases, Commons Configuration takes a similar approach as the Java Collections framework. Here collections are per default not thread-safe (and thus more efficient). If an application needs a thread-safe collection, it can "upgrade" an existing one by calling a method of the Collections class.

Objects implementing the Configuration interface can be associated with a [Synchronizer](https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/sync/Synchronizer.html) object. This synchronizer is triggered on each access to the configuration (distinguishing between read and write access). It can decide whether access is allowed or block the calling thread until it is safe to continue. Per default, a Configuration object uses a [NoOpSynchronizer](https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/sync/NoOpSynchronizer.html) instance. As the name implies, this class does nothing to protect its associated configuration against concurrent access; its methods are just empty dummies. It is appropriate for use cases in which a configuration is only accessed by a single thread.

If multiple threads are involved, Configuration objects have to be thread-safe. For this purpose, there is another implementation of Synchronizer: [ReadWriteSynchronizer](https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/sync/ReadWriteSynchronizer.html). This class is based on the ReentrantReadWriteLock class from the JDK. It implements the typical behavior desired when accessing a configuration in a multi-threaded environment:

* An arbitrary number of threads can read the configuration simultaneously.
* Updates of a configuration can only happen with an exclusive lock; so if a thread changes configuration data, all other threads (readers and writers) are blocked until the update operation is complete.

The synchronizer associated with a Configuration can be changed at any time by calling the setSynchronizer() method. The following example shows how this method is used to make a Configuration instance thread-safe:

config.setSynchronizer(new ReadWriteSynchronizer());

Rather than setting the synchronizer on an existing Configuration instance, it is usually better to configure the [configuration builder](https://commons.apache.org/proper/commons-configuration/userguide/howto_builder.html) responsible for the creation of the configuration to set the correct synchronizer directly after a new instance has been created. This is done in the usual way by setting the corresponding property of a parameters object passed to the builder's configure() method, for instance:

Parameters params = new Parameters();

BasicConfigurationBuilder<PropertiesConfiguration> builder =

new BasicConfigurationBuilder<PropertiesConfiguration>(

PropertiesConfiguration.class)

.configure(params.basic()

.setSynchronizer(new ReadWriteSynchronizer());

PropertiesConfiguration config = builder.getConfiguration();

It is also possible to set the synchronizer to **null**. In this case, the default NoOpSynchronizer is installed, which means that the configuration is no longer protected against concurrent access.

With the two classes NoOpSynchronizer and ReadWriteSynchronizer the Commons Configuration library covers the basic use cases of no protection and full protection of multi-threaded access. As the Synchronizer interface is pretty simple, applications are free to provide their own implementations according to their specific needs. However, this requires a certain understanding of internal mechanisms in Configuration implementations. Some caveats are provided in the remaining of this chapter.

## Basic operations and thread-safety

[AbstractConfiguration](https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/AbstractConfiguration.html) already provides a major part of the implementation of correctly interacting with a Synchronizer object. Methods for reading configuration data (such as getProperty(), isEmpty(), or getKeys()) and for changing properties (e.g. setProperty(), addProperty(), or clearProperty()) already call the correct methods of the Synchronizer. These methods are declared **final** to avoid that subclasses accidently break thread-safety by incorrectly usage of the Synchronizer.

Classes derived from AbstractConfiguration sometimes offer specific methods for accessing properties. For instance, hierarchical configurations offer operations on whole subtrees, or [INIConfiguration](https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/INIConfiguration.html) allows querying specific sections. These methods are also aware of the associated Synchronizer and invoke it correctly.

There is another pair of methods available for each Configuration object allowing direct control over the Synchronizer: lock() and unlock(). Both methods expect an argument of type [LockMode](https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/sync/LockMode.html) which tells them whether the configuration is to be locked for read or write access. These methods can be used to extend the locking behavior of standard methods. For instance, if multiple properties are to be added in an atomic way, lock() can be called first, then all properties are added, and finally unlock() is called. Provided that a corresponding Synchronizer implementation is used, other threads will not interfere with this sequence. Note that it is important to always call unlock() after a lock() call; this is done best in a **finally** block as shown in the following example:

config.lock(LockMode.WRITE);

try

{

config.addProperty("prop1", "value1");

...

config.addProperty("prop\_n", "value\_n");

}

finally

{

config.unlock(LockMode.WRITE);

}

So, in a nutshell: When accessing configuration data from standard configuration classes all operations are controlled via the configuration's Synchronizer object. Client code is only responsible for setting a correct Synchronizer object which is suitable for the intended use case.

## Other flags

In addition to the actual configuration data, each Configuration object has some flags controlling its behavior. One example for such a flag is the boolean throwExceptionOnMissing property. Other helper objects like the object responsible for interpolation or the expression engine for hierarchical configurations fall into the same category. The manipulation of those flags and helper objects is also related to thread-safety.

In contrast to configuration data, access to flags is **not** guarded by the Synchronizer. This means that when changing a flag in a multi-threaded environment, there is no guarantee that this change is visible to other threads.

The reason for this design is that the preferred way to create a Configuration object is using a *configuration builder*. The builder is responsible for fully initializing the configuration; afterwards, no behavioral changes should be performed any more. Because builders are always synchronized the values of all flags are safely published to all involved threads.

If there really is the need to change a flag later on in the life-cycle of a Configuration object, the lock() and unlock() methods described in the previous section should be used to do the change with a write lock held.

## Special cases

Thread-safety is certainly a complex topic. This section describes some corner cases which may occur when some of the more advanced configuration classes are involved.

* All hierarchical configurations derived from [BaseHierarchicalConfiguration](https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/BaseHierarchicalConfiguration.html) internally operate on a nodes structure implemented by immutable nodes. This is beneficial for concurrent access. It is even possible to share (sub) trees of configuration nodes between multiple configuration objects. Updates of these structures are implemented in a thread-safe and non-blocking way - even when using the default NoOpSynchronizer. So the point to take is that when using hierarchical configurations it is not required to set a special synchronizer because safe concurrent access is already a basic feature of these classes. The only exception is that change events caused by updates of a configuration's data are not guaranteed to be delivered in a specific order. For instance, if one thread clears a configuration and immediately afterwards another thread adds a property, it may be the case that the clear event arrives after the add property event at an event listener. If the listener relies on the fact that the configuration is empty now, it may be up for a surprise. In cases in which the sequence of generated configuration events is important, a fully functional synchronizer object should be set.
* [CombinedConfiguration](https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/CombinedConfiguration.html) is a bit special regarding lock handling. Although derived from BaseHierarchicalConfiguration, this class is not thread-safe per default. So if accessed by multiple threads, a suitable synchronizer has to be set. An instance manages a node tree which is constructed dynamically from the nodes of all contained configurations using the current *node combiner*. When one of the child configurations is changed the node tree is reset so that it has to be re-constructed on next access. Because this operation changes the configuration's internal state it is performed with a write lock held. So even if only data is read from a CombinedConfiguration, it may be the case that temporarily a write lock is obtained for constructing the combined node tree. Note that the synchronizers used for the children of a combined configuration are independent. For instance, if configuration objects derived from BaseHierarchicalConfiguration are added as children to a CombinedConfiguration, they can continue using a NoOpSynchronizer.
* Derived from CombinedConfiguration is [DynamicCombinedConfiguration](https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/DynamicCombinedConfiguration.html) which extends its base class by the ability to manage multiple combined configuration instances. The current instance is selected based on a key constructed by a [ConfigurationInterpolator](https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/interpol/ConfigurationInterpolator.html) instance. If this yields a key which has not been encountered before, a new CombinedConfiguration object is created. Here again it turns out that even a read access to a DynamicCombinedConfiguration may cause internal state changes which require a write lock to be held. When creating a new child combined configuration it is passed the Synchronizer of the owning DynamicCombinedConfiguration; so there is actually only a single Synchronizer controlling the access to all involved configurations.

## Read-only configurations

Objects that are not changed typically play well in an environment with multiple threads - provided that they are initialized in a safe way. For the safe initialization of Configuration objects specialized [builders](https://commons.apache.org/proper/commons-configuration/userguide/howto_builders.html) are responsible. These are classes derived from [BasicConfigurationBuilder](https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/builder/BasicConfigurationBuilder.html). Configuration builders are designed to be thread-safe: their getConfiguration() method is synchronized, so that configurations can be created and initialized in a safe way even if multiple threads are interacting with the builder. Synchronization also ensures that all values stored in member fields of newly created Configuration objects are safely published to all involved threads.

As long as a configuration returned freshly from a builder is not changed in any way, it can be used without a special Synchronizer (this means that the default NoOpSynchronizer is used). As was discussed in the previous section, there are special cases in which read-only access to Configuration objects causes internal state changes. This would be critical without a fully functional Synchronizer object. However, the builders dealing with affected classes are implemented in a way that they take care about these special cases and perform extra initialization steps which make write locks for later read operations unnecessary.

For instance, the builder for combined configurations explicitly accesses a newly created CombinedConfiguration object so that it is forced to construct its node tree. This happens in the builder's getConfiguration() method which is synchronized. So provided that the combined configuration is not changed (no other child configurations are added, no updates are performed on existing child configurations), no protection against concurrent access is needed - a simple NoOpSynchronizer can do the job.

Situation is similar for the other special cases described in the previous section. One exception is DynamicCombinedConfiguration: Whether an instance can be used in a read-only manner without a fully functional Synchronizer depends on the way it constructs its keys. If the keys remain constant during the life time of an instance (for instance, they are based on a system property specified as startup option of the Java virtual machine), NoOpSynchronizer is sufficient. If the keys are more dynamic, a fully functional Synchronizer is required for concurrent access - even if only reads are performed.

So to sum up, except for very few cases configurations can be read by multiple threads without having to use a special Synchronizer. For this to be safe, the configurations have to be created through a builder, and they must not be updated by any of these threads. A good way to prevent updates to a Configuration object is to wrap it by an immutable configuration.

javax.security.auth.login

# Class Configuration

* [java.lang.Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)
  + javax.security.auth.login.Configuration

public abstract class **Configuration**

extends [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

A Configuration object is responsible for specifying which LoginModules should be used for a particular application, and in what order the LoginModules should be invoked.

A login configuration contains the following information. Note that this example only represents the default syntax for the Configuration. Subclass implementations of this class may implement alternative syntaxes and may retrieve the Configuration from any source such as files, databases, or servers.

Name {

ModuleClass Flag ModuleOptions;

ModuleClass Flag ModuleOptions;

ModuleClass Flag ModuleOptions;

};

Name {

ModuleClass Flag ModuleOptions;

ModuleClass Flag ModuleOptions;

};

other {

ModuleClass Flag ModuleOptions;

ModuleClass Flag ModuleOptions;

};

Each entry in the Configuration is indexed via an application name, *Name*, and contains a list of LoginModules configured for that application. Each LoginModule is specified via its fully qualified class name. Authentication proceeds down the module list in the exact order specified. If an application does not have specific entry, it defaults to the specific entry for "*other*".

The *Flag* value controls the overall behavior as authentication proceeds down the stack. The following represents a description of the valid values for *Flag* and their respective semantics:

1) Required - The LoginModule is required to succeed.

If it succeeds or fails, authentication still continues

to proceed down the LoginModule list.

2) Requisite - The LoginModule is required to succeed.

If it succeeds, authentication continues down the

LoginModule list. If it fails,

control immediately returns to the application

(authentication does not proceed down the

LoginModule list).

3) Sufficient - The LoginModule is not required to

succeed. If it does succeed, control immediately

returns to the application (authentication does not

proceed down the LoginModule list).

If it fails, authentication continues down the

LoginModule list.

4) Optional - The LoginModule is not required to

succeed. If it succeeds or fails,

authentication still continues to proceed down the

LoginModule list.

The overall authentication succeeds only if all *Required* and *Requisite* LoginModules succeed. If a *Sufficient* LoginModule is configured and succeeds, then only the *Required* and *Requisite* LoginModules prior to that *Sufficient* LoginModule need to have succeeded for the overall authentication to succeed. If no *Required* or *Requisite* LoginModules are configured for an application, then at least one *Sufficient* or *Optional* LoginModule must succeed.

*ModuleOptions* is a space separated list of LoginModule-specific values which are passed directly to the underlying LoginModules. Options are defined by the LoginModule itself, and control the behavior within it. For example, a LoginModule may define options to support debugging/testing capabilities. The correct way to specify options in the Configuration is by using the following key-value pairing: *debug="true"*. The key and value should be separated by an 'equals' symbol, and the value should be surrounded by double quotes. If a String in the form, ${system.property}, occurs in the value, it will be expanded to the value of the system property. Note that there is no limit to the number of options a LoginModule may define.

The following represents an example Configuration entry based on the syntax above:

Login {

com.sun.security.auth.module.UnixLoginModule required;

com.sun.security.auth.module.Krb5LoginModule optional

useTicketCache="true"

ticketCache="${user.home}${/}tickets";

};

This Configuration specifies that an application named, "Login", requires users to first authenticate to the *com.sun.security.auth.module.UnixLoginModule*, which is required to succeed. Even if the *UnixLoginModule* authentication fails, the *com.sun.security.auth.module.Krb5LoginModule* still gets invoked. This helps hide the source of failure. Since the *Krb5LoginModule* is *Optional*, the overall authentication succeeds only if the *UnixLoginModule* (*Required*) succeeds.

Also note that the LoginModule-specific options, *useTicketCache="true"* and *ticketCache=${user.home}${/}tickets"*, are passed to the *Krb5LoginModule*. These options instruct the *Krb5LoginModule* to use the ticket cache at the specified location. The system properties, *user.home* and */* (file.separator), are expanded to their respective values.

There is only one Configuration object installed in the runtime at any given time. A Configuration object can be installed by calling the setConfiguration method. The installed Configuration object can be obtained by calling the getConfiguration method.

If no Configuration object has been installed in the runtime, a call to getConfiguration installs an instance of the default Configuration implementation (a default subclass implementation of this abstract class). The default Configuration implementation can be changed by setting the value of the "login.configuration.provider" security property (in the Java security properties file) to the fully qualified name of the desired Configuration subclass implementation. The Java security properties file is located in the file named <JAVA\_HOME>/lib/security/java.security. <JAVA\_HOME> refers to the value of the java.home system property, and specifies the directory where the JRE is installed.

Application code can directly subclass Configuration to provide a custom implementation. In addition, an instance of a Configuration object can be constructed by invoking one of the getInstance factory methods with a standard type. The default policy type is "JavaLoginConfig". See the Configuration section in the [Java Cryptography Architecture Standard Algorithm Name Documentation](https://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#Configuration) for a list of standard Configuration types.

**See Also:**

[LoginContext](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginContext.html)

### Nested Class Summary

|  |  |
| --- | --- |
| **Nested Classes** | |
| **Modifier and Type** | **Class and Description** |
| static interface | [**Configuration.Parameters**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.Parameters.html)  This represents a marker interface for Configuration parameters. |

### Constructor Summary

|  |  |
| --- | --- |
| **Constructors** | |
| **Modifier** | **Constructor and Description** |
| protected | [**Configuration**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#Configuration())()  Sole constructor. |

### Method Summary

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| abstract **[AppConfigurationEntry](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/AppConfigurationEntry.html" \o "class in javax.security.auth.login)**[] | [**getAppConfigurationEntry**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#getAppConfigurationEntry(java.lang.String))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) name)  Retrieve the AppConfigurationEntries for the specified *name* from this Configuration. |
| static [**Configuration**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) | [**getConfiguration**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#getConfiguration())()  Get the installed login Configuration. |
| static [**Configuration**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) | [**getInstance**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#getInstance(java.lang.String,%20javax.security.auth.login.Configuration.Parameters))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) type, **[Configuration.Parameters](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.Parameters.html" \o "interface in javax.security.auth.login)** params)  Returns a Configuration object of the specified type. |
| static [**Configuration**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) | [**getInstance**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#getInstance(java.lang.String,%20javax.security.auth.login.Configuration.Parameters,%20java.security.Provider))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) type, [**Configuration.Parameters**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.Parameters.html) params, [**Provider**](https://docs.oracle.com/javase/7/docs/api/java/security/Provider.html) provider)  Returns a Configuration object of the specified type. |
| static [**Configuration**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) | [**getInstance**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#getInstance(java.lang.String,%20javax.security.auth.login.Configuration.Parameters,%20java.lang.String))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) type, [**Configuration.Parameters**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.Parameters.html) params, [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) provider)  Returns a Configuration object of the specified type. |
| [**Configuration.Parameters**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.Parameters.html) | [**getParameters**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#getParameters())()  Return Configuration parameters. |
| [**Provider**](https://docs.oracle.com/javase/7/docs/api/java/security/Provider.html) | [**getProvider**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#getProvider())()  Return the Provider of this Configuration. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**getType**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#getType())()  Return the type of this Configuration. |
| void | [**refresh**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#refresh())()  Refresh and reload the Configuration. |
| static void | [**setConfiguration**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#setConfiguration(javax.security.auth.login.Configuration))([**Configuration**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) configuration)  Set the login Configuration. |

### Methods inherited from class java.lang.[Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

[clone](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#clone()), [equals](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals(java.lang.Object)), [finalize](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#finalize()), [getClass](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#getClass()), [hashCode](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode()), [notify](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#notify()), [notifyAll](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#notifyAll()), [toString](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#toString()), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait()), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait(long)), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait(long,%20int))

### Constructor Detail

#### Configuration

protected Configuration()

Sole constructor. (For invocation by subclass constructors, typically implicit.)

### Method Detail

#### getConfiguration

public static [Configuration](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) getConfiguration()

Get the installed login Configuration.

**Returns:**

the login Configuration. If a Configuration object was set via the Configuration.setConfiguration method, then that object is returned. Otherwise, a default Configuration object is returned.

**Throws:**

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to retrieve the Configuration.

**See Also:**

[setConfiguration(javax.security.auth.login.Configuration)](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#setConfiguration(javax.security.auth.login.Configuration))

#### setConfiguration

public static void setConfiguration([Configuration](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) configuration)

Set the login Configuration.

**Parameters:**

configuration - the new Configuration

**Throws:**

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the current thread does not have Permission to set the Configuration.

**See Also:**

[getConfiguration()](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html#getConfiguration())

#### getInstance

* + - public static [Configuration](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) getInstance([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) type,
    - [Configuration.Parameters](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.Parameters.html) params)

throws [NoSuchAlgorithmException](https://docs.oracle.com/javase/7/docs/api/java/security/NoSuchAlgorithmException.html)

Returns a Configuration object of the specified type.

This method traverses the list of registered security providers, starting with the most preferred Provider. A new Configuration object encapsulating the ConfigurationSpi implementation from the first Provider that supports the specified type is returned.

Note that the list of registered providers may be retrieved via the [Security.getProviders()](https://docs.oracle.com/javase/7/docs/api/java/security/Security.html" \l "getProviders()) method.

**Parameters:**

type - the specified Configuration type. See the Configuration section in the [Java Cryptography Architecture Standard Algorithm Name Documentation](https://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#Configuration) for a list of standard Configuration types.

params - parameters for the Configuration, which may be null.

**Returns:**

the new Configuration object.

**Throws:**

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to get a Configuration instance for the specified type.

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the specified type is null.

[IllegalArgumentException](https://docs.oracle.com/javase/7/docs/api/java/lang/IllegalArgumentException.html) - if the specified parameters are not understood by the ConfigurationSpi implementation from the selected Provider.

[NoSuchAlgorithmException](https://docs.oracle.com/javase/7/docs/api/java/security/NoSuchAlgorithmException.html) - if no Provider supports a ConfigurationSpi implementation for the specified type.

**Since:**

1.6

**See Also:**

[Provider](https://docs.oracle.com/javase/7/docs/api/java/security/Provider.html)

#### getInstance

* + - public static [Configuration](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) getInstance([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) type,
    - [Configuration.Parameters](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.Parameters.html) params,
    - [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) provider)
    - throws [NoSuchProviderException](https://docs.oracle.com/javase/7/docs/api/java/security/NoSuchProviderException.html),

[NoSuchAlgorithmException](https://docs.oracle.com/javase/7/docs/api/java/security/NoSuchAlgorithmException.html)

Returns a Configuration object of the specified type.

A new Configuration object encapsulating the ConfigurationSpi implementation from the specified provider is returned. The specified provider must be registered in the provider list.

Note that the list of registered providers may be retrieved via the [Security.getProviders()](https://docs.oracle.com/javase/7/docs/api/java/security/Security.html" \l "getProviders()) method.

**Parameters:**

type - the specified Configuration type. See the Configuration section in the [Java Cryptography Architecture Standard Algorithm Name Documentation](https://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#Configuration) for a list of standard Configuration types.

params - parameters for the Configuration, which may be null.

provider - the provider.

**Returns:**

the new Configuration object.

**Throws:**

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to get a Configuration instance for the specified type.

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the specified type is null.

[IllegalArgumentException](https://docs.oracle.com/javase/7/docs/api/java/lang/IllegalArgumentException.html) - if the specified provider is null or empty, or if the specified parameters are not understood by the ConfigurationSpi implementation from the specified provider.

[NoSuchProviderException](https://docs.oracle.com/javase/7/docs/api/java/security/NoSuchProviderException.html) - if the specified provider is not registered in the security provider list.

[NoSuchAlgorithmException](https://docs.oracle.com/javase/7/docs/api/java/security/NoSuchAlgorithmException.html) - if the specified provider does not support a ConfigurationSpi implementation for the specified type.

**Since:**

1.6

**See Also:**

[Provider](https://docs.oracle.com/javase/7/docs/api/java/security/Provider.html)

#### getInstance

* + - public static [Configuration](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) getInstance([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) type,
    - [Configuration.Parameters](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.Parameters.html) params,
    - [Provider](https://docs.oracle.com/javase/7/docs/api/java/security/Provider.html) provider)

throws [NoSuchAlgorithmException](https://docs.oracle.com/javase/7/docs/api/java/security/NoSuchAlgorithmException.html)

Returns a Configuration object of the specified type.

A new Configuration object encapsulating the ConfigurationSpi implementation from the specified Provider object is returned. Note that the specified Provider object does not have to be registered in the provider list.

**Parameters:**

type - the specified Configuration type. See the Configuration section in the [Java Cryptography Architecture Standard Algorithm Name Documentation](https://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#Configuration) for a list of standard Configuration types.

params - parameters for the Configuration, which may be null.

provider - the Provider.

**Returns:**

the new Configuration object.

**Throws:**

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to get a Configuration instance for the specified type.

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the specified type is null.

[IllegalArgumentException](https://docs.oracle.com/javase/7/docs/api/java/lang/IllegalArgumentException.html) - if the specified Provider is null, or if the specified parameters are not understood by the ConfigurationSpi implementation from the specified Provider.

[NoSuchAlgorithmException](https://docs.oracle.com/javase/7/docs/api/java/security/NoSuchAlgorithmException.html) - if the specified Provider does not support a ConfigurationSpi implementation for the specified type.

**Since:**

1.6

**See Also:**

[Provider](https://docs.oracle.com/javase/7/docs/api/java/security/Provider.html)

#### getProvider

public [Provider](https://docs.oracle.com/javase/7/docs/api/java/security/Provider.html) getProvider()

Return the Provider of this Configuration.

This Configuration instance will only have a Provider if it was obtained via a call to Configuration.getInstance. Otherwise this method returns null.

**Returns:**

the Provider of this Configuration, or null.

**Since:**

1.6

#### getType

public [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) getType()

Return the type of this Configuration.

This Configuration instance will only have a type if it was obtained via a call to Configuration.getInstance. Otherwise this method returns null.

**Returns:**

the type of this Configuration, or null.

**Since:**

1.6

#### getParameters

public [Configuration.Parameters](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.Parameters.html" \o "interface in javax.security.auth.login) getParameters()

Return Configuration parameters.

This Configuration instance will only have parameters if it was obtained via a call to Configuration.getInstance. Otherwise this method returns null.

**Returns:**

Configuration parameters, or null.

**Since:**

1.6

#### getAppConfigurationEntry

public abstract [AppConfigurationEntry](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/AppConfigurationEntry.html)[] getAppConfigurationEntry([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) name)

Retrieve the AppConfigurationEntries for the specified *name* from this Configuration.

**Parameters:**

name - the name used to index the Configuration.

**Returns:**

an array of AppConfigurationEntries for the specified *name* from this Configuration, or null if there are no entries for the specified *name*

#### refresh

public void refresh()

Refresh and reload the Configuration.

This method causes this Configuration object to refresh/reload its contents in an implementation-dependent manner. For example, if this Configuration object stores its entries in a file, calling refresh may cause the file to be re-read.

The default implementation of this method does nothing. This method should be overridden if a refresh operation is supported by the implementation.

**Throws:**

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to refresh its Configuration.

javax.security.auth.spi

# Interface LoginModule

public interface ****LoginModule****

LoginModule describes the interface implemented by authentication technology providers. LoginModules are plugged in under applications to provide a particular type of authentication.

While applications write to the LoginContext API, authentication technology providers implement the LoginModule interface. A Configuration specifies the LoginModule(s) to be used with a particular login application. Therefore different LoginModules can be plugged in under the application without requiring any modifications to the application itself.

The LoginContext is responsible for reading the Configuration and instantiating the appropriate LoginModules. Each LoginModule is initialized with a Subject, a CallbackHandler, shared LoginModule state, and LoginModule-specific options. The Subject represents the Subject currently being authenticated and is updated with relevant Credentials if authentication succeeds. LoginModules use the CallbackHandler to communicate with users. The CallbackHandler may be used to prompt for usernames and passwords, for example. Note that the CallbackHandler may be null. LoginModules which absolutely require a CallbackHandler to authenticate the Subject may throw a LoginException. LoginModules optionally use the shared state to share information or data among themselves.

The LoginModule-specific options represent the options configured for this LoginModule by an administrator or user in the login Configuration. The options are defined by the LoginModule itself and control the behavior within it. For example, a LoginModule may define options to support debugging/testing capabilities. Options are defined using a key-value syntax, such as *debug=true*. The LoginModule stores the options as a Map so that the values may be retrieved using the key. Note that there is no limit to the number of options a LoginModule chooses to define.

The calling application sees the authentication process as a single operation. However, the authentication process within the LoginModule proceeds in two distinct phases. In the first phase, the LoginModule's login method gets invoked by the LoginContext's login method. The login method for the LoginModule then performs the actual authentication (prompt for and verify a password for example) and saves its authentication status as private state information. Once finished, the LoginModule's login method either returns true (if it succeeded) or false (if it should be ignored), or throws a LoginException to specify a failure. In the failure case, the LoginModule must not retry the authentication or introduce delays. The responsibility of such tasks belongs to the application. If the application attempts to retry the authentication, the LoginModule's login method will be called again.

In the second phase, if the LoginContext's overall authentication succeeded (the relevant REQUIRED, REQUISITE, SUFFICIENT and OPTIONAL LoginModules succeeded), then the commit method for the LoginModule gets invoked. The commit method for a LoginModule checks its privately saved state to see if its own authentication succeeded. If the overall LoginContext authentication succeeded and the LoginModule's own authentication succeeded, then the commit method associates the relevant Principals (authenticated identities) and Credentials (authentication data such as cryptographic keys) with the Subject located within the LoginModule.

If the LoginContext's overall authentication failed (the relevant REQUIRED, REQUISITE, SUFFICIENT and OPTIONAL LoginModules did not succeed), then the abort method for each LoginModule gets invoked. In this case, the LoginModule removes/destroys any authentication state originally saved.

Logging out a Subject involves only one phase. The LoginContext invokes the LoginModule's logout method. The logout method for the LoginModule then performs the logout procedures, such as removing Principals or Credentials from the Subject or logging session information.

A LoginModule implementation must have a constructor with no arguments. This allows classes which load the LoginModule to instantiate it.

****See Also:****

[LoginContext](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginContext.html), [Configuration](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html)

## Method Summary

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| boolean | [**abort**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/spi/LoginModule.html#abort())()  Method to abort the authentication process (phase 2). |
| boolean | [**commit**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/spi/LoginModule.html#commit())()  Method to commit the authentication process (phase 2). |
| void | [**initialize**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/spi/LoginModule.html#initialize(javax.security.auth.Subject,%20javax.security.auth.callback.CallbackHandler,%20java.util.Map,%20java.util.Map))([**Subject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) subject, [**CallbackHandler**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/CallbackHandler.html) callbackHandler, [**Map**](https://docs.oracle.com/javase/7/docs/api/java/util/Map.html)<[**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html),?> sharedState, [**Map**](https://docs.oracle.com/javase/7/docs/api/java/util/Map.html)<[**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html),?> options)  Initialize this LoginModule. |
| boolean | [**login**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/spi/LoginModule.html#login())()  Method to authenticate a Subject (phase 1). |
| boolean | [**logout**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/spi/LoginModule.html#logout())()  Method which logs out a Subject. |

## Method Detail

#### initialize

* + - void initialize([Subject](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html" \o "class in javax.security.auth) subject,
    - [CallbackHandler](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/CallbackHandler.html) callbackHandler,
    - [Map](https://docs.oracle.com/javase/7/docs/api/java/util/Map.html)<[String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html),?> sharedState,

[Map](https://docs.oracle.com/javase/7/docs/api/java/util/Map.html)<[String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html),?> options)

Initialize this LoginModule.

This method is called by the LoginContext after this LoginModule has been instantiated. The purpose of this method is to initialize this LoginModule with the relevant information. If this LoginModule does not understand any of the data stored in sharedState or options parameters, they can be ignored.

****Parameters:****

subject - the Subject to be authenticated.

callbackHandler - a CallbackHandler for communicating with the end user (prompting for usernames and passwords, for example).

sharedState - state shared with other configured LoginModules.

options - options specified in the login Configuration for this particular LoginModule.

#### login

* + - boolean login()

throws [LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html)

Method to authenticate a Subject (phase 1).

The implementation of this method authenticates a Subject. For example, it may prompt for Subject information such as a username and password and then attempt to verify the password. This method saves the result of the authentication attempt as private state within the LoginModule.

****Returns:****

true if the authentication succeeded, or false if this LoginModule should be ignored.

****Throws:****

[LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html) - if the authentication fails

#### commit

* + - boolean commit()

throws [LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html)

Method to commit the authentication process (phase 2).

This method is called if the LoginContext's overall authentication succeeded (the relevant REQUIRED, REQUISITE, SUFFICIENT and OPTIONAL LoginModules succeeded).

If this LoginModule's own authentication attempt succeeded (checked by retrieving the private state saved by the login method), then this method associates relevant Principals and Credentials with the Subject located in the LoginModule. If this LoginModule's own authentication attempted failed, then this method removes/destroys any state that was originally saved.

****Returns:****

true if this method succeeded, or false if this LoginModule should be ignored.

****Throws:****

[LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html) - if the commit fails

#### abort

* + - boolean abort()

throws [LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html)

Method to abort the authentication process (phase 2).

This method is called if the LoginContext's overall authentication failed. (the relevant REQUIRED, REQUISITE, SUFFICIENT and OPTIONAL LoginModules did not succeed).

If this LoginModule's own authentication attempt succeeded (checked by retrieving the private state saved by the login method), then this method cleans up any state that was originally saved.

****Returns:****

true if this method succeeded, or false if this LoginModule should be ignored.

****Throws:****

[LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html) - if the abort fails

#### logout

* + - boolean logout()

throws [LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html)

Method which logs out a Subject.

An implementation of this method might remove/destroy a Subject's Principals and Credentials.

****Returns:****

true if this method succeeded, or false if this LoginModule should be ignored.

****Throws:****

[LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html) - if the logout fails

java.security

# Class Permission

* [java.lang.Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)
  + java.security.Permission
* **All Implemented Interfaces:**

[Serializable](https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html), [Guard](https://docs.oracle.com/javase/7/docs/api/java/security/Guard.html)

**Direct Known Subclasses:**

[AllPermission](https://docs.oracle.com/javase/7/docs/api/java/security/AllPermission.html), [BasicPermission](https://docs.oracle.com/javase/7/docs/api/java/security/BasicPermission.html), [FilePermission](https://docs.oracle.com/javase/7/docs/api/java/io/FilePermission.html), [MBeanPermission](https://docs.oracle.com/javase/7/docs/api/javax/management/MBeanPermission.html), [PrivateCredentialPermission](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/PrivateCredentialPermission.html), [ServicePermission](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/kerberos/ServicePermission.html), [SocketPermission](https://docs.oracle.com/javase/7/docs/api/java/net/SocketPermission.html), [UnresolvedPermission](https://docs.oracle.com/javase/7/docs/api/java/security/UnresolvedPermission.html)

public abstract class ****Permission****

extends [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

implements [Guard](https://docs.oracle.com/javase/7/docs/api/java/security/Guard.html), [Serializable](https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html)

Abstract class for representing access to a system resource. All permissions have a name (whose interpretation depends on the subclass), as well as abstract functions for defining the semantics of the particular Permission subclass.

Most Permission objects also include an "actions" list that tells the actions that are permitted for the object. For example, for a java.io.FilePermission object, the permission name is the pathname of a file (or directory), and the actions list (such as "read, write") specifies which actions are granted for the specified file (or for files in the specified directory). The actions list is optional for Permission objects, such as java.lang.RuntimePermission, that don't need such a list; you either have the named permission (such as "system.exit") or you don't.

An important method that must be implemented by each subclass is the implies method to compare Permissions. Basically, "permission p1 implies permission p2" means that if one is granted permission p1, one is naturally granted permission p2. Thus, this is not an equality test, but rather more of a subset test.

Permission objects are similar to String objects in that they are immutable once they have been created. Subclasses should not provide methods that can change the state of a permission once it has been created.

****See Also:****

[Permissions](https://docs.oracle.com/javase/7/docs/api/java/security/Permissions.html), [PermissionCollection](https://docs.oracle.com/javase/7/docs/api/java/security/PermissionCollection.html" \o "class in java.security), [Serialized Form](https://docs.oracle.com/javase/7/docs/api/serialized-form.html#java.security.Permission)

## Constructor Summary

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Permission**](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html#Permission(java.lang.String))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) name)  Constructs a permission with the specified name. |

## Method Summary

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| void | [**checkGuard**](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html#checkGuard(java.lang.Object))([**Object**](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) object)  Implements the guard interface for a permission. |
| abstract boolean | [**equals**](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html#equals(java.lang.Object))([**Object**](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) obj)  Checks two Permission objects for equality. |
| abstract [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**getActions**](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html#getActions())()  Returns the actions as a String. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**getName**](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html#getName())()  Returns the name of this Permission. |
| abstract int | [**hashCode**](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html#hashCode())()  Returns the hash code value for this Permission object. |
| abstract boolean | [**implies**](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html#implies(java.security.Permission))([**Permission**](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html) permission)  Checks if the specified permission's actions are "implied by" this object's actions. |
| [**PermissionCollection**](https://docs.oracle.com/javase/7/docs/api/java/security/PermissionCollection.html) | [**newPermissionCollection**](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html#newPermissionCollection())()  Returns an empty PermissionCollection for a given Permission object, or null if one is not defined. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**toString**](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html#toString())()  Returns a string describing this Permission. |

## Methods inherited from class java.lang.[Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

[clone](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#clone()), [finalize](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#finalize()), [getClass](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html" \l "getClass()), [notify](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#notify()), [notifyAll](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html" \l "notifyAll()), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait()), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait(long)), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait(long,%20int))

## Constructor Detail

#### Permission

public Permission([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html" \o "class in java.lang) name)

Constructs a permission with the specified name.

****Parameters:****

name - name of the Permission object being created.

### Method Detail

#### checkGuard

* + - public void checkGuard([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html" \o "class in java.lang) object)

throws [SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html)

Implements the guard interface for a permission. The SecurityManager.checkPermission method is called, passing this permission object as the permission to check. Returns silently if access is granted. Otherwise, throws a SecurityException.

**Specified by:**

[checkGuard](https://docs.oracle.com/javase/7/docs/api/java/security/Guard.html#checkGuard(java.lang.Object)) in interface [Guard](https://docs.oracle.com/javase/7/docs/api/java/security/Guard.html)

****Parameters:****

object - the object being guarded (currently ignored).

****Throws:****

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if a security manager exists and its checkPermission method doesn't allow access.

****See Also:****

[Guard](https://docs.oracle.com/javase/7/docs/api/java/security/Guard.html), [GuardedObject](https://docs.oracle.com/javase/7/docs/api/java/security/GuardedObject.html), [SecurityManager.checkPermission(java.security.Permission)](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityManager.html#checkPermission(java.security.Permission))

#### implies

public abstract boolean implies([Permission](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html" \o "class in java.security) permission)

Checks if the specified permission's actions are "implied by" this object's actions.

This must be implemented by subclasses of Permission, as they are the only ones that can impose semantics on a Permission object.

The implies method is used by the AccessController to determine whether or not a requested permission is implied by another permission that is known to be valid in the current execution context.

****Parameters:****

permission - the permission to check against.

****Returns:****

true if the specified permission is implied by this object, false if not.

#### equals

public abstract boolean equals([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html" \o "class in java.lang) obj)

Checks two Permission objects for equality.

Do not use the equals method for making access control decisions; use the implies method.

**Overrides:**

[equals](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals(java.lang.Object)) in class [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

****Parameters:****

obj - the object we are testing for equality with this object.

****Returns:****

true if both Permission objects are equivalent.

****See Also:****

[Object.hashCode()](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode()), [HashMap](https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html)

#### hashCode

public abstract int hashCode()

Returns the hash code value for this Permission object.

The required hashCode behavior for Permission Objects is the following:

* + - * Whenever it is invoked on the same Permission object more than once during an execution of a Java application, the hashCode method must consistently return the same integer. This integer need not remain consistent from one execution of an application to another execution of the same application.
      * If two Permission objects are equal according to the equals method, then calling the hashCode method on each of the two Permission objects must produce the same integer result.

**Overrides:**

[hashCode](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode()) in class [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

****Returns:****

a hash code value for this object.

****See Also:****

[Object.equals(java.lang.Object)](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals(java.lang.Object)), [System.identityHashCode(java.lang.Object)](https://docs.oracle.com/javase/7/docs/api/java/lang/System.html#identityHashCode(java.lang.Object))

#### getName

public final [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) getName()

Returns the name of this Permission. For example, in the case of a java.io.FilePermission, the name will be a pathname.

****Returns:****

the name of this Permission.

#### getActions

public abstract [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) getActions()

Returns the actions as a String. This is abstract so subclasses can defer creating a String representation until one is needed. Subclasses should always return actions in what they consider to be their canonical form. For example, two FilePermission objects created via the following:

perm1 = new FilePermission(p1,"read,write");

perm2 = new FilePermission(p2,"write,read");

both return "read,write" when the getActions method is invoked.

****Returns:****

the actions of this Permission.

#### newPermissionCollection

public [PermissionCollection](https://docs.oracle.com/javase/7/docs/api/java/security/PermissionCollection.html" \o "class in java.security) newPermissionCollection()

Returns an empty PermissionCollection for a given Permission object, or null if one is not defined. Subclasses of class Permission should override this if they need to store their permissions in a particular PermissionCollection object in order to provide the correct semantics when the PermissionCollection.implies method is called. If null is returned, then the caller of this method is free to store permissions of this type in any PermissionCollection they choose (one that uses a Hashtable, one that uses a Vector, etc).

****Returns:****

a new PermissionCollection object for this type of Permission, or null if one is not defined.

#### toString

public [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) toString()

Returns a string describing this Permission. The convention is to specify the class name, the permission name, and the actions in the following format: '("ClassName" "name" "actions")', or '("ClassName" "name")' if actions list is null or empty.

**Overrides:**

[toString](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#toString()) in class [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

****Returns:****

information about this Permission.

javax.security.auth

# Class Subject

* [java.lang.Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)
  + javax.security.auth.Subject
* **All Implemented Interfaces:**

[Serializable](https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html)

public final class ****Subject****

extends [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

implements [Serializable](https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html)

A Subject represents a grouping of related information for a single entity, such as a person. Such information includes the Subject's identities as well as its security-related attributes (passwords and cryptographic keys, for example).

Subjects may potentially have multiple identities. Each identity is represented as a Principal within the Subject. Principals simply bind names to a Subject. For example, a Subject that happens to be a person, Alice, might have two Principals: one which binds "Alice Bar", the name on her driver license, to the Subject, and another which binds, "999-99-9999", the number on her student identification card, to the Subject. Both Principals refer to the same Subject even though each has a different name.

A Subject may also own security-related attributes, which are referred to as credentials. Sensitive credentials that require special protection, such as private cryptographic keys, are stored within a private credential Set. Credentials intended to be shared, such as public key certificates or Kerberos server tickets are stored within a public credential Set. Different permissions are required to access and modify the different credential Sets.

To retrieve all the Principals associated with a Subject, invoke the getPrincipals method. To retrieve all the public or private credentials belonging to a Subject, invoke the getPublicCredentials method or getPrivateCredentials method, respectively. To modify the returned Set of Principals and credentials, use the methods defined in the Set class. For example:

Subject subject;

Principal principal;

Object credential;

// add a Principal and credential to the Subject

subject.getPrincipals().add(principal);

subject.getPublicCredentials().add(credential);

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| static <T> T | [**doAs**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#doAs(javax.security.auth.Subject,%20java.security.PrivilegedAction))([**Subject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) subject, **[PrivilegedAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedAction.html" \o "interface in java.security)**<T> action)  Perform work as a particular Subject. |
| static <T> T | [**doAs**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#doAs(javax.security.auth.Subject,%20java.security.PrivilegedExceptionAction))([**Subject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) subject, **[PrivilegedExceptionAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedExceptionAction.html" \o "interface in java.security)**<T> action)  Perform work as a particular Subject. |
| static <T> T | [**doAsPrivileged**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#doAsPrivileged(javax.security.auth.Subject,%20java.security.PrivilegedAction,%20java.security.AccessControlContext))([**Subject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) subject, [**PrivilegedAction**](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedAction.html)<T> action, [**AccessControlContext**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html) acc)  Perform privileged work as a particular Subject. |
| static <T> T | [**doAsPrivileged**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#doAsPrivileged(javax.security.auth.Subject,%20java.security.PrivilegedExceptionAction,%20java.security.AccessControlContext))([**Subject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) subject, [**PrivilegedExceptionAction**](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedExceptionAction.html)<T> action, [**AccessControlContext**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html) acc)  Perform privileged work as a particular Subject. |
| boolean | [**equals**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#equals(java.lang.Object))([**Object**](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) o)  Compares the specified Object with this Subject for equality. |
| [**Set**](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<[**Principal**](https://docs.oracle.com/javase/7/docs/api/java/security/Principal.html)> | [**getPrincipals**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#getPrincipals())()  Return the Set of Principals associated with this Subject. |
| <T extends [**Principal**](https://docs.oracle.com/javase/7/docs/api/java/security/Principal.html)>  [**Set**](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<T> | [**getPrincipals**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#getPrincipals(java.lang.Class))([**Class**](https://docs.oracle.com/javase/7/docs/api/java/lang/Class.html)<T> c)  Return a Set of Principals associated with this Subject that are instances or subclasses of the specified Class. |
| [**Set**](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<[**Object**](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)> | [**getPrivateCredentials**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#getPrivateCredentials())()  Return the Set of private credentials held by this Subject. |
| <T> [**Set**](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<T> | [**getPrivateCredentials**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#getPrivateCredentials(java.lang.Class))([**Class**](https://docs.oracle.com/javase/7/docs/api/java/lang/Class.html)<T> c)  Return a Set of private credentials associated with this Subject that are instances or subclasses of the specified Class. |
| [**Set**](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<[**Object**](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)> | [**getPublicCredentials**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#getPublicCredentials())()  Return the Set of public credentials held by this Subject. |
| <T> [**Set**](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<T> | [**getPublicCredentials**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#getPublicCredentials(java.lang.Class))([**Class**](https://docs.oracle.com/javase/7/docs/api/java/lang/Class.html)<T> c)  Return a Set of public credentials associated with this Subject that are instances or subclasses of the specified Class. |
| static [**Subject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) | [**getSubject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#getSubject(java.security.AccessControlContext))(**[AccessControlContext](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html" \o "class in java.security)** acc)  Get the Subject associated with the provided AccessControlContext. |
| int | [**hashCode**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#hashCode())()  Returns a hashcode for this Subject. |
| boolean | [**isReadOnly**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#isReadOnly())()  Query whether this Subject is read-only. |
| void | [**setReadOnly**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#setReadOnly())()  Set this Subject to be read-only. |
| [**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | [**toString**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#toString())()  Return the String representation of this Subject. |

This Subject class implements Serializable. While the Principals associated with the Subject are serialized, the credentials associated with the Subject are not. Note that the java.security.Principal class does not implement Serializable. Therefore all concrete Principal implementations associated with Subjects must implement Serializable.

****See Also:****

[Principal](https://docs.oracle.com/javase/7/docs/api/java/security/Principal.html), [DomainCombiner](https://docs.oracle.com/javase/7/docs/api/java/security/DomainCombiner.html" \o "interface in java.security), [Serialized Form](https://docs.oracle.com/javase/7/docs/api/serialized-form.html#javax.security.auth.Subject)

## Constructor Summary

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Subject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#Subject())()  Create an instance of a Subject with an empty Set of Principals and empty Sets of public and private credentials. |
| [**Subject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html#Subject(boolean,%20java.util.Set,%20java.util.Set,%20java.util.Set))(boolean readOnly, [**Set**](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<? extends [**Principal**](https://docs.oracle.com/javase/7/docs/api/java/security/Principal.html)> principals, [**Set**](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<?> pubCredentials, [**Set**](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<?> privCredentials)  Create an instance of a Subject with Principals and credentials. |

## Method Summary

### Methods inherited from class java.lang.[Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

[clone](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#clone()), [finalize](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#finalize()), [getClass](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html" \l "getClass()), [notify](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#notify()), [notifyAll](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html" \l "notifyAll()), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait()), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait(long)), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait(long,%20int))

### Constructor Detail

#### Subject

public Subject()

Create an instance of a Subject with an empty Set of Principals and empty Sets of public and private credentials.

The newly constructed Sets check whether this Subject has been set read-only before permitting subsequent modifications. The newly created Sets also prevent illegal modifications by ensuring that callers have sufficient permissions.

To modify the Principals Set, the caller must have AuthPermission("modifyPrincipals"). To modify the public credential Set, the caller must have AuthPermission("modifyPublicCredentials"). To modify the private credential Set, the caller must have AuthPermission("modifyPrivateCredentials").

#### Subject

* + - public Subject(boolean readOnly,
    - [Set](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<? extends [Principal](https://docs.oracle.com/javase/7/docs/api/java/security/Principal.html)> principals,
    - [Set](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<?> pubCredentials,

[Set](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<?> privCredentials)

Create an instance of a Subject with Principals and credentials.

The Principals and credentials from the specified Sets are copied into newly constructed Sets. These newly created Sets check whether this Subject has been set read-only before permitting subsequent modifications. The newly created Sets also prevent illegal modifications by ensuring that callers have sufficient permissions.

To modify the Principals Set, the caller must have AuthPermission("modifyPrincipals"). To modify the public credential Set, the caller must have AuthPermission("modifyPublicCredentials"). To modify the private credential Set, the caller must have AuthPermission("modifyPrivateCredentials").

****Parameters:****

readOnly - true if the Subject is to be read-only, and false otherwise.

principals - the Set of Principals to be associated with this Subject.

pubCredentials - the Set of public credentials to be associated with this Subject.

privCredentials - the Set of private credentials to be associated with this Subject.

****Throws:****

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the specified principals, pubCredentials, or privCredentials are null.

### Method Detail

#### setReadOnly

public void setReadOnly()

Set this Subject to be read-only.

Modifications (additions and removals) to this Subject's Principal Set and credential Sets will be disallowed. The destroy operation on this Subject's credentials will still be permitted.

Subsequent attempts to modify the Subject's Principal and credential Sets will result in an IllegalStateException being thrown. Also, once a Subject is read-only, it can not be reset to being writable again.

****Throws:****

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to set this Subject to be read-only.

#### isReadOnly

public boolean isReadOnly()

Query whether this Subject is read-only.

****Returns:****

true if this Subject is read-only, false otherwise.

#### getSubject

public static [Subject](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) getSubject([AccessControlContext](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html" \o "class in java.security) acc)

Get the Subject associated with the provided AccessControlContext.

The AccessControlContext may contain many Subjects (from nested doAs calls). In this situation, the most recent Subject associated with the AccessControlContext is returned.

****Parameters:****

acc - the AccessControlContext from which to retrieve the Subject.

****Returns:****

the Subject associated with the provided AccessControlContext, or null if no Subject is associated with the provided AccessControlContext.

****Throws:****

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to get the Subject.

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the provided AccessControlContext is null.

#### doAs

* + - public static <T> T doAs([Subject](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html" \o "class in javax.security.auth) subject,

[PrivilegedAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedAction.html)<T> action)

Perform work as a particular Subject.

This method first retrieves the current Thread's AccessControlContext via AccessController.getContext, and then instantiates a new AccessControlContext using the retrieved context along with a new SubjectDomainCombiner (constructed using the provided Subject). Finally, this method invokes AccessController.doPrivileged, passing it the provided PrivilegedAction, as well as the newly constructed AccessControlContext.

****Parameters:****

subject - the Subject that the specified action will run as. This parameter may be null.

action - the code to be run as the specified Subject.

****Returns:****

the value returned by the PrivilegedAction's run method.

****Throws:****

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the PrivilegedAction is null.

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to invoke this method.

#### doAs

* + - public static <T> T doAs([Subject](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html" \o "class in javax.security.auth) subject,
    - [PrivilegedExceptionAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedExceptionAction.html)<T> action)

throws [PrivilegedActionException](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedActionException.html)

Perform work as a particular Subject.

This method first retrieves the current Thread's AccessControlContext via AccessController.getContext, and then instantiates a new AccessControlContext using the retrieved context along with a new SubjectDomainCombiner (constructed using the provided Subject). Finally, this method invokes AccessController.doPrivileged, passing it the provided PrivilegedExceptionAction, as well as the newly constructed AccessControlContext.

****Parameters:****

subject - the Subject that the specified action will run as. This parameter may be null.

action - the code to be run as the specified Subject.

****Returns:****

the value returned by the PrivilegedExceptionAction's run method.

****Throws:****

[PrivilegedActionException](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedActionException.html) - if the PrivilegedExceptionAction.run method throws a checked exception.

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the specified PrivilegedExceptionAction is null.

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to invoke this method.

#### doAsPrivileged

* + - public static <T> T doAsPrivileged([Subject](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html" \o "class in javax.security.auth) subject,
    - [PrivilegedAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedAction.html)<T> action,

[AccessControlContext](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html) acc)

Perform privileged work as a particular Subject.

This method behaves exactly as Subject.doAs, except that instead of retrieving the current Thread's AccessControlContext, it uses the provided AccessControlContext. If the provided AccessControlContext is null, this method instantiates a new AccessControlContext with an empty collection of ProtectionDomains.

****Parameters:****

subject - the Subject that the specified action will run as. This parameter may be null.

action - the code to be run as the specified Subject.

acc - the AccessControlContext to be tied to the specified *subject* and *action*.

****Returns:****

the value returned by the PrivilegedAction's run method.

****Throws:****

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the PrivilegedAction is null.

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to invoke this method.

#### doAsPrivileged

* + - public static <T> T doAsPrivileged([Subject](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html" \o "class in javax.security.auth) subject,
    - [PrivilegedExceptionAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedExceptionAction.html)<T> action,
    - [AccessControlContext](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html) acc)

throws [PrivilegedActionException](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedActionException.html)

Perform privileged work as a particular Subject.

This method behaves exactly as Subject.doAs, except that instead of retrieving the current Thread's AccessControlContext, it uses the provided AccessControlContext. If the provided AccessControlContext is null, this method instantiates a new AccessControlContext with an empty collection of ProtectionDomains.

****Parameters:****

subject - the Subject that the specified action will run as. This parameter may be null.

action - the code to be run as the specified Subject.

acc - the AccessControlContext to be tied to the specified *subject* and *action*.

****Returns:****

the value returned by the PrivilegedExceptionAction's run method.

****Throws:****

[PrivilegedActionException](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedActionException.html) - if the PrivilegedExceptionAction.run method throws a checked exception.

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the specified PrivilegedExceptionAction is null.

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to invoke this method.

#### getPrincipals

public [Set](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<[Principal](https://docs.oracle.com/javase/7/docs/api/java/security/Principal.html)> getPrincipals()

Return the Set of Principals associated with this Subject. Each Principal represents an identity for this Subject.

The returned Set is backed by this Subject's internal Principal Set. Any modification to the returned Set affects the internal Principal Set as well.

****Returns:****

The Set of Principals associated with this Subject.

#### getPrincipals

public <T extends [Principal](https://docs.oracle.com/javase/7/docs/api/java/security/Principal.html)> [Set](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<T> getPrincipals([Class](https://docs.oracle.com/javase/7/docs/api/java/lang/Class.html)<T> c)

Return a Set of Principals associated with this Subject that are instances or subclasses of the specified Class.

The returned Set is not backed by this Subject's internal Principal Set. A new Set is created and returned for each method invocation. Modifications to the returned Set will not affect the internal Principal Set.

****Parameters:****

c - the returned Set of Principals will all be instances of this class.

****Returns:****

a Set of Principals that are instances of the specified Class.

****Throws:****

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the specified Class is null.

#### getPublicCredentials

public [Set](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<[Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)> getPublicCredentials()

Return the Set of public credentials held by this Subject.

The returned Set is backed by this Subject's internal public Credential Set. Any modification to the returned Set affects the internal public Credential Set as well.

****Returns:****

A Set of public credentials held by this Subject.

#### getPrivateCredentials

public [Set](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<[Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)> getPrivateCredentials()

Return the Set of private credentials held by this Subject.

The returned Set is backed by this Subject's internal private Credential Set. Any modification to the returned Set affects the internal private Credential Set as well.

A caller requires permissions to access the Credentials in the returned Set, or to modify the Set itself. A SecurityException is thrown if the caller does not have the proper permissions.

While iterating through the Set, a SecurityException is thrown if the caller does not have permission to access a particular Credential. The Iterator is nevertheless advanced to next element in the Set.

****Returns:****

A Set of private credentials held by this Subject.

#### getPublicCredentials

public <T> [Set](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<T> getPublicCredentials([Class](https://docs.oracle.com/javase/7/docs/api/java/lang/Class.html)<T> c)

Return a Set of public credentials associated with this Subject that are instances or subclasses of the specified Class.

The returned Set is not backed by this Subject's internal public Credential Set. A new Set is created and returned for each method invocation. Modifications to the returned Set will not affect the internal public Credential Set.

****Parameters:****

c - the returned Set of public credentials will all be instances of this class.

****Returns:****

a Set of public credentials that are instances of the specified Class.

****Throws:****

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the specified Class is null.

#### getPrivateCredentials

public <T> [Set](https://docs.oracle.com/javase/7/docs/api/java/util/Set.html)<T> getPrivateCredentials([Class](https://docs.oracle.com/javase/7/docs/api/java/lang/Class.html)<T> c)

Return a Set of private credentials associated with this Subject that are instances or subclasses of the specified Class.

The caller must have permission to access all of the requested Credentials, or a SecurityException will be thrown.

The returned Set is not backed by this Subject's internal private Credential Set. A new Set is created and returned for each method invocation. Modifications to the returned Set will not affect the internal private Credential Set.

****Parameters:****

c - the returned Set of private credentials will all be instances of this class.

****Returns:****

a Set of private credentials that are instances of the specified Class.

****Throws:****

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the specified Class is null.

#### equals

public boolean equals([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html" \o "class in java.lang) o)

Compares the specified Object with this Subject for equality. Returns true if the given object is also a Subject and the two Subject instances are equivalent. More formally, two Subject instances are equal if their Principal and Credential Sets are equal.

**Overrides:**

[equals](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals(java.lang.Object)) in class [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

****Parameters:****

o - Object to be compared for equality with this Subject.

****Returns:****

true if the specified Object is equal to this Subject.

****Throws:****

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to access the private credentials for this Subject, or if the caller does not have permission to access the private credentials for the provided Subject.

****See Also:****

[Object.hashCode()](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode()), [HashMap](https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html)

#### toString

public [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) toString()

Return the String representation of this Subject.

**Overrides:**

[toString](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#toString()) in class [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

****Returns:****

the String representation of this Subject.

#### hashCode

public int hashCode()

Returns a hashcode for this Subject.

**Overrides:**

[hashCode](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode()) in class [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

****Returns:****

a hashcode for this Subject.

****Throws:****

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if the caller does not have permission to access this Subject's private credentials.

****See Also:****

[Object.equals(java.lang.Object)](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals(java.lang.Object)), [System.identityHashCode(java.lang.Object)](https://docs.oracle.com/javase/7/docs/api/java/lang/System.html#identityHashCode(java.lang.Object))

java.security

# Class AccessController

* [java.lang.Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)
  + java.security.AccessController

public final class ****AccessController****

extends [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

The AccessController class is used for access control operations and decisions.

More specifically, the AccessController class is used for three purposes:

* + to decide whether an access to a critical system resource is to be allowed or denied, based on the security policy currently in effect,
  + to mark code as being "privileged", thus affecting subsequent access determinations, and
  + to obtain a "snapshot" of the current calling context so access-control decisions from a different context can be made with respect to the saved context.

The [checkPermission](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html" \l "checkPermission(java.security.Permission)) method determines whether the access request indicated by a specified permission should be granted or denied. A sample call appears below. In this example, checkPermission will determine whether or not to grant "read" access to the file named "testFile" in the "/temp" directory.

FilePermission perm = new FilePermission("/temp/testFile", "read");

AccessController.checkPermission(perm);

If a requested access is allowed, checkPermission returns quietly. If denied, an AccessControlException is thrown. AccessControlException can also be thrown if the requested permission is of an incorrect type or contains an invalid value. Such information is given whenever possible. Suppose the current thread traversed m callers, in the order of caller 1 to caller 2 to caller m. Then caller m invoked the checkPermission method. The checkPermission method determines whether access is granted or denied based on the following algorithm:

for (int i = m; i > 0; i--) {

if (caller i's domain does not have the permission)

throw AccessControlException

else if (caller i is marked as privileged) {

if (a context was specified in the call to doPrivileged)

context.checkPermission(permission)

return;

}

};

// Next, check the context inherited when the thread was created.

// Whenever a new thread is created, the AccessControlContext at

// that time is stored and associated with the new thread, as the

// "inherited" context.

inheritedContext.checkPermission(permission);

A caller can be marked as being "privileged" (see [doPrivileged](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html" \l "doPrivileged(java.security.PrivilegedAction)) and below). When making access control decisions, the checkPermission method stops checking if it reaches a caller that was marked as "privileged" via a doPrivileged call without a context argument (see below for information about a context argument). If that caller's domain has the specified permission, no further checking is done and checkPermission returns quietly, indicating that the requested access is allowed. If that domain does not have the specified permission, an exception is thrown, as usual.

The normal use of the "privileged" feature is as follows. If you don't need to return a value from within the "privileged" block, do the following:

somemethod() {

...normal code here...

AccessController.doPrivileged(new PrivilegedAction<Void>() {

public Void run() {

// privileged code goes here, for example:

System.loadLibrary("awt");

return null; // nothing to return

}

});

...normal code here...

}

PrivilegedAction is an interface with a single method, named run. The above example shows creation of an implementation of that interface; a concrete implementation of the run method is supplied. When the call to doPrivileged is made, an instance of the PrivilegedAction implementation is passed to it. The doPrivileged method calls the run method from the PrivilegedAction implementation after enabling privileges, and returns the run method's return value as the doPrivileged return value (which is ignored in this example).

If you need to return a value, you can do something like the following:

somemethod() {

...normal code here...

String user = AccessController.doPrivileged(

new PrivilegedAction<String>() {

public String run() {

return System.getProperty("user.name");

}

});

...normal code here...

}

If the action performed in your run method could throw a "checked" exception (those listed in the throws clause of a method), then you need to use the PrivilegedExceptionAction interface instead of the PrivilegedAction interface:

somemethod() throws FileNotFoundException {

...normal code here...

try {

FileInputStream fis = AccessController.doPrivileged(

new PrivilegedExceptionAction<FileInputStream>() {

public FileInputStream run() throws FileNotFoundException {

return new FileInputStream("someFile");

}

});

} catch (PrivilegedActionException e) {

// e.getException() should be an instance of FileNotFoundException,

// as only "checked" exceptions will be "wrapped" in a

// PrivilegedActionException.

throw (FileNotFoundException) e.getException();

}

...normal code here...

}

Be \*very\* careful in your use of the "privileged" construct, and always remember to make the privileged code section as small as possible.

Note that checkPermission always performs security checks within the context of the currently executing thread. Sometimes a security check that should be made within a given context will actually need to be done from within a *different* context (for example, from within a worker thread). The [getContext](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html" \l "getContext()) method and AccessControlContext class are provided for this situation. The getContext method takes a "snapshot" of the current calling context, and places it in an AccessControlContext object, which it returns. A sample call is the following:

AccessControlContext acc = AccessController.getContext()

AccessControlContext itself has a checkPermission method that makes access decisions based on the context *it* encapsulates, rather than that of the current execution thread. Code within a different context can thus call that method on the previously-saved AccessControlContext object. A sample call is the following:

acc.checkPermission(permission)

There are also times where you don't know a priori which permissions to check the context against. In these cases you can use the doPrivileged method that takes a context:

somemethod() {

AccessController.doPrivileged(new PrivilegedAction<Object>() {

public Object run() {

// Code goes here. Any permission checks within this

// run method will require that the intersection of the

// callers protection domain and the snapshot's

// context have the desired permission.

}

}, acc);

...normal code here...

}

****See Also:****

[AccessControlContext](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html)

## Method Summary

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| static void | [**checkPermission**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#checkPermission(java.security.Permission))([**Permission**](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html) perm)  Determines whether the access request indicated by the specified permission should be allowed or denied, based on the current AccessControlContext and security policy. |
| static <T> T | [**doPrivileged**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedAction))(**[PrivilegedAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedAction.html" \o "interface in java.security)**<T> action)  Performs the specified PrivilegedAction with privileges enabled. |
| static <T> T | [**doPrivileged**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedAction,%20java.security.AccessControlContext))([**PrivilegedAction**](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedAction.html)<T> action, [**AccessControlContext**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html) context)  Performs the specified PrivilegedAction with privileges enabled and restricted by the specified AccessControlContext. |
| static <T> T | [**doPrivileged**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedExceptionAction))(**[PrivilegedExceptionAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedExceptionAction.html" \o "interface in java.security)**<T> action)  Performs the specified PrivilegedExceptionAction with privileges enabled. |
| static <T> T | [**doPrivileged**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedExceptionAction,%20java.security.AccessControlContext))([**PrivilegedExceptionAction**](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedExceptionAction.html)<T> action, [**AccessControlContext**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html) context)  Performs the specified PrivilegedExceptionAction with privileges enabled and restricted by the specified AccessControlContext. |
| static <T> T | [**doPrivilegedWithCombiner**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivilegedWithCombiner(java.security.PrivilegedAction))(**[PrivilegedAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedAction.html" \o "interface in java.security)**<T> action)  Performs the specified PrivilegedAction with privileges enabled. |
| static <T> T | [**doPrivilegedWithCombiner**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivilegedWithCombiner(java.security.PrivilegedExceptionAction))(**[PrivilegedExceptionAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedExceptionAction.html" \o "interface in java.security)**<T> action)  Performs the specified PrivilegedExceptionAction with privileges enabled. |
| static **[AccessControlContext](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html" \o "class in java.security)** | [**getContext**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#getContext())()  This method takes a "snapshot" of the current calling context, which includes the current Thread's inherited AccessControlContext, and places it in an AccessControlContext object. |

### Methods inherited from class java.lang.[Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

[clone](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#clone()), [equals](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals(java.lang.Object)), [finalize](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#finalize()), [getClass](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#getClass()), [hashCode](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode()), [notify](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#notify()), [notifyAll](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#notifyAll()), [toString](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#toString()), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait()), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait(long)), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait(long,%20int))

## Method Detail

#### doPrivileged

public static <T> T doPrivileged([PrivilegedAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedAction.html" \o "interface in java.security)<T> action)

Performs the specified PrivilegedAction with privileges enabled. The action is performed with *all* of the permissions possessed by the caller's protection domain.

If the action's run method throws an (unchecked) exception, it will propagate through this method.

Note that any DomainCombiner associated with the current AccessControlContext will be ignored while the action is performed.

****Parameters:****

action - the action to be performed.

****Returns:****

the value returned by the action's run method.

****Throws:****

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the action is null

****See Also:****

[doPrivileged(PrivilegedAction,AccessControlContext)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedAction,%20java.security.AccessControlContext)), [doPrivileged(PrivilegedExceptionAction)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedExceptionAction)), [doPrivilegedWithCombiner(PrivilegedAction)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivilegedWithCombiner(java.security.PrivilegedAction)), [DomainCombiner](https://docs.oracle.com/javase/7/docs/api/java/security/DomainCombiner.html)

#### doPrivilegedWithCombiner

public static <T> T doPrivilegedWithCombiner([PrivilegedAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedAction.html)<T> action)

Performs the specified PrivilegedAction with privileges enabled. The action is performed with *all* of the permissions possessed by the caller's protection domain.

If the action's run method throws an (unchecked) exception, it will propagate through this method.

This method preserves the current AccessControlContext's DomainCombiner (which may be null) while the action is performed.

****Parameters:****

action - the action to be performed.

****Returns:****

the value returned by the action's run method.

****Throws:****

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the action is null

****Since:****

1.6

****See Also:****

[doPrivileged(PrivilegedAction)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedAction)), [DomainCombiner](https://docs.oracle.com/javase/7/docs/api/java/security/DomainCombiner.html" \o "interface in java.security)

#### doPrivileged

* + - public static <T> T doPrivileged([PrivilegedAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedAction.html" \o "interface in java.security)<T> action,

[AccessControlContext](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html) context)

Performs the specified PrivilegedAction with privileges enabled and restricted by the specified AccessControlContext. The action is performed with the intersection of the permissions possessed by the caller's protection domain, and those possessed by the domains represented by the specified AccessControlContext.

If the action's run method throws an (unchecked) exception, it will propagate through this method.

****Parameters:****

action - the action to be performed.

context - an *access control context* representing the restriction to be applied to the caller's domain's privileges before performing the specified action. If the context is null, then no additional restriction is applied.

****Returns:****

the value returned by the action's run method.

****Throws:****

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the action is null

****See Also:****

[doPrivileged(PrivilegedAction)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedAction)), [doPrivileged(PrivilegedExceptionAction,AccessControlContext)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedExceptionAction,%20java.security.AccessControlContext))

#### doPrivileged

* + - public static <T> T doPrivileged([PrivilegedExceptionAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedExceptionAction.html" \o "interface in java.security)<T> action)

throws [PrivilegedActionException](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedActionException.html)

Performs the specified PrivilegedExceptionAction with privileges enabled. The action is performed with *all* of the permissions possessed by the caller's protection domain.

If the action's run method throws an *unchecked* exception, it will propagate through this method.

Note that any DomainCombiner associated with the current AccessControlContext will be ignored while the action is performed.

****Parameters:****

action - the action to be performed

****Returns:****

the value returned by the action's run method

****Throws:****

[PrivilegedActionException](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedActionException.html) - if the specified action's run method threw a *checked* exception

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the action is null

****See Also:****

[doPrivileged(PrivilegedAction)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedAction)), [doPrivileged(PrivilegedExceptionAction,AccessControlContext)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedExceptionAction,%20java.security.AccessControlContext)), [doPrivilegedWithCombiner(PrivilegedExceptionAction)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivilegedWithCombiner(java.security.PrivilegedExceptionAction)), [DomainCombiner](https://docs.oracle.com/javase/7/docs/api/java/security/DomainCombiner.html)

#### doPrivilegedWithCombiner

* + - public static <T> T doPrivilegedWithCombiner([PrivilegedExceptionAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedExceptionAction.html)<T> action)

throws [PrivilegedActionException](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedActionException.html)

Performs the specified PrivilegedExceptionAction with privileges enabled. The action is performed with *all* of the permissions possessed by the caller's protection domain.

If the action's run method throws an *unchecked* exception, it will propagate through this method.

This method preserves the current AccessControlContext's DomainCombiner (which may be null) while the action is performed.

****Parameters:****

action - the action to be performed.

****Returns:****

the value returned by the action's run method

****Throws:****

[PrivilegedActionException](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedActionException.html) - if the specified action's run method threw a *checked* exception

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the action is null

****Since:****

1.6

****See Also:****

[doPrivileged(PrivilegedAction)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedAction)), [doPrivileged(PrivilegedExceptionAction,AccessControlContext)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedExceptionAction,%20java.security.AccessControlContext)), [DomainCombiner](https://docs.oracle.com/javase/7/docs/api/java/security/DomainCombiner.html)

#### doPrivileged

* + - public static <T> T doPrivileged([PrivilegedExceptionAction](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedExceptionAction.html" \o "interface in java.security)<T> action,
    - [AccessControlContext](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html) context)

throws [PrivilegedActionException](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedActionException.html)

Performs the specified PrivilegedExceptionAction with privileges enabled and restricted by the specified AccessControlContext. The action is performed with the intersection of the permissions possessed by the caller's protection domain, and those possessed by the domains represented by the specified AccessControlContext.

If the action's run method throws an *unchecked* exception, it will propagate through this method.

****Parameters:****

action - the action to be performed

context - an *access control context* representing the restriction to be applied to the caller's domain's privileges before performing the specified action. If the context is null, then no additional restriction is applied.

****Returns:****

the value returned by the action's run method

****Throws:****

[PrivilegedActionException](https://docs.oracle.com/javase/7/docs/api/java/security/PrivilegedActionException.html) - if the specified action's run method threw a *checked* exception

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the action is null

****See Also:****

[doPrivileged(PrivilegedAction)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedAction)), [doPrivileged(PrivilegedExceptionAction,AccessControlContext)](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html#doPrivileged(java.security.PrivilegedExceptionAction,%20java.security.AccessControlContext))

#### getContext

public static [AccessControlContext](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html" \o "class in java.security) getContext()

This method takes a "snapshot" of the current calling context, which includes the current Thread's inherited AccessControlContext, and places it in an AccessControlContext object. This context may then be checked at a later point, possibly in another thread.

****Returns:****

the AccessControlContext based on the current context.

****See Also:****

[AccessControlContext](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html)

#### checkPermission

* + - public static void checkPermission([Permission](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html" \o "class in java.security) perm)

throws [AccessControlException](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlException.html)

Determines whether the access request indicated by the specified permission should be allowed or denied, based on the current AccessControlContext and security policy. This method quietly returns if the access request is permitted, or throws an AccessControlException otherwise. The getPermission method of the AccessControlException returns the perm Permission object instance.

****Parameters:****

perm - the requested permission.

****Throws:****

[AccessControlException](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlException.html) - if the specified permission is not permitted, based on the current security policy.

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the specified permission is null and is checked based on the security policy currently in effect.

java.security

# Class AccessControlContext

* [java.lang.Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)
  + java.security.AccessControlContext

public final class ****AccessControlContext****

extends [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

An AccessControlContext is used to make system resource access decisions based on the context it encapsulates.

More specifically, it encapsulates a context and has a single method, checkPermission, that is equivalent to the checkPermission method in the AccessController class, with one difference: The AccessControlContext checkPermission method makes access decisions based on the context it encapsulates, rather than that of the current execution thread.

Thus, the purpose of AccessControlContext is for those situations where a security check that should be made within a given context actually needs to be done from within a *different* context (for example, from within a worker thread).

An AccessControlContext is created by calling the AccessController.getContext method. The getContext method takes a "snapshot" of the current calling context, and places it in an AccessControlContext object, which it returns. A sample call is the following:

AccessControlContext acc = AccessController.getContext()

Code within a different context can subsequently call the checkPermission method on the previously-saved AccessControlContext object. A sample call is the following:

acc.checkPermission(permission)

****See Also:****

[AccessController](https://docs.oracle.com/javase/7/docs/api/java/security/AccessController.html)

## Constructor Summary

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**AccessControlContext**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html#AccessControlContext(java.security.AccessControlContext,%20java.security.DomainCombiner))([**AccessControlContext**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html) acc, [**DomainCombiner**](https://docs.oracle.com/javase/7/docs/api/java/security/DomainCombiner.html) combiner)  Create a new AccessControlContext with the given AccessControlContext and DomainCombiner. |
| [**AccessControlContext**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html#AccessControlContext(java.security.ProtectionDomain[]))(**[ProtectionDomain](https://docs.oracle.com/javase/7/docs/api/java/security/ProtectionDomain.html" \o "class in java.security)**[] context)  Create an AccessControlContext with the given array of ProtectionDomains. |

## Method Summary

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| void | [**checkPermission**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html#checkPermission(java.security.Permission))([**Permission**](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html) perm)  Determines whether the access request indicated by the specified permission should be allowed or denied, based on the security policy currently in effect, and the context in this object. |
| boolean | [**equals**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html#equals(java.lang.Object))([**Object**](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html) obj)  Checks two AccessControlContext objects for equality. |
| [**DomainCombiner**](https://docs.oracle.com/javase/7/docs/api/java/security/DomainCombiner.html) | [**getDomainCombiner**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html#getDomainCombiner())()  Get the DomainCombiner associated with this AccessControlContext. |
| int | [**hashCode**](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html#hashCode())()  Returns the hash code value for this context. |

## Methods inherited from class java.lang.[Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

[clone](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#clone()), [finalize](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#finalize()), [getClass](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#getClass()), [notify](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#notify()), [notifyAll](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#notifyAll()), [toString](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#toString()), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait()), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait(long)), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait(long,%20int))

## Constructor Detail

#### AccessControlContext

public AccessControlContext([ProtectionDomain](https://docs.oracle.com/javase/7/docs/api/java/security/ProtectionDomain.html" \o "class in java.security)[] context)

Create an AccessControlContext with the given array of ProtectionDomains. Context must not be null. Duplicate domains will be removed from the context.

****Parameters:****

context - the ProtectionDomains associated with this context. The non-duplicate domains are copied from the array. Subsequent changes to the array will not affect this AccessControlContext.

****Throws:****

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if context is null

#### AccessControlContext

* + - public AccessControlContext([AccessControlContext](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlContext.html" \o "class in java.security) acc,

[DomainCombiner](https://docs.oracle.com/javase/7/docs/api/java/security/DomainCombiner.html) combiner)

Create a new AccessControlContext with the given AccessControlContext and DomainCombiner. This constructor associates the provided DomainCombiner with the provided AccessControlContext.

****Parameters:****

acc - the AccessControlContext associated with the provided DomainCombiner.

combiner - the DomainCombiner to be associated with the provided AccessControlContext.

****Throws:****

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the provided context is null.

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if a security manager is installed and the caller does not have the "createAccessControlContext" [SecurityPermission](https://docs.oracle.com/javase/7/docs/api/java/security/SecurityPermission.html" \o "class in java.security)

****Since:****

1.3

### Method Detail

#### getDomainCombiner

public [DomainCombiner](https://docs.oracle.com/javase/7/docs/api/java/security/DomainCombiner.html" \o "interface in java.security) getDomainCombiner()

Get the DomainCombiner associated with this AccessControlContext.

****Returns:****

the DomainCombiner associated with this AccessControlContext, or null if there is none.

****Throws:****

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if a security manager is installed and the caller does not have the "getDomainCombiner" [SecurityPermission](https://docs.oracle.com/javase/7/docs/api/java/security/SecurityPermission.html" \o "class in java.security)

****Since:****

1.3

#### checkPermission

* + - public void checkPermission([Permission](https://docs.oracle.com/javase/7/docs/api/java/security/Permission.html" \o "class in java.security) perm)

throws [AccessControlException](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlException.html)

Determines whether the access request indicated by the specified permission should be allowed or denied, based on the security policy currently in effect, and the context in this object. The request is allowed only if every ProtectionDomain in the context implies the permission. Otherwise the request is denied.

This method quietly returns if the access request is permitted, or throws a suitable AccessControlException otherwise.

****Parameters:****

perm - the requested permission.

****Throws:****

[AccessControlException](https://docs.oracle.com/javase/7/docs/api/java/security/AccessControlException.html) - if the specified permission is not permitted, based on the current security policy and the context encapsulated by this object.

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - if the permission to check for is null.

#### equals

public boolean equals([Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html" \o "class in java.lang) obj)

Checks two AccessControlContext objects for equality. Checks that *obj* is an AccessControlContext and has the same set of ProtectionDomains as this context.

**Overrides:**

[equals](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals(java.lang.Object)) in class [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

****Parameters:****

obj - the object we are testing for equality with this object.

****Returns:****

true if *obj* is an AccessControlContext, and has the same set of ProtectionDomains as this context, false otherwise.

****See Also:****

[Object.hashCode()](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode()), [HashMap](https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html)

#### hashCode

public int hashCode()

Returns the hash code value for this context. The hash code is computed by exclusive or-ing the hash code of all the protection domains in the context together.

**Overrides:**

[hashCode](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode()) in class [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

****Returns:****

a hash code value for this context.

****See Also:****

[Object.equals(java.lang.Object)](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals(java.lang.Object)), [System.identityHashCode(java.lang.Object)](https://docs.oracle.com/javase/7/docs/api/java/lang/System.html#identityHashCode(java.lang.Object))

# **Java Code Examples for javax.security.auth.login.Configuration**

The following examples show how to use javax.security.auth.login.Configuration. These examples are extracted from open source projects. You can vote up the ones you like or vote down the ones you don't like, and go to the original project or source file by following the links above each example. You may check out the related API usage on the sidebar.

## Example 1

|  |  |  |
| --- | --- | --- |
| Source Project: *[swellrt](https://www.programcreek.com/java-api-examples/?project_name=SwellRT%2Fswellrt)* Source File: [AuthenticationServlet.java](https://www.programcreek.com/java-api-examples/?code=SwellRT%2Fswellrt%2Fswellrt-master%2Fwave%2Fsrc%2Fmain%2Fjava%2Forg%2Fwaveprotocol%2Fbox%2Fserver%2Frpc%2FAuthenticationServlet.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=SwellRT%2Fswellrt%2Fswellrt-master%2FLICENSE) | 6 votes |  |

@Inject

public AuthenticationServlet(AccountStore accountStore,

Configuration configuration,

SessionManager sessionManager,

@Named(CoreSettingsNames.WAVE\_SERVER\_DOMAIN) String domain,

Config config

/\* ,WelcomeRobot welcomeBot \*/) {

Preconditions.checkNotNull(accountStore, "AccountStore is null");

Preconditions.checkNotNull(configuration, "Configuration is null");

Preconditions.checkNotNull(sessionManager, "Session manager is null");

this.accountStore = accountStore;

this.configuration = configuration;

this.sessionManager = sessionManager;

this.domain = domain.toLowerCase();

this.isClientAuthEnabled = config.getBoolean("security.enable\_clientauth");

this.clientAuthCertDomain = config.getString("security.clientauth\_cert\_domain").toLowerCase();

this.isRegistrationDisabled = config.getBoolean("administration.disable\_registration");

this.isLoginPageDisabled = config.getBoolean("administration.disable\_loginpage");

// this.welcomeBot = welcomeBot;

this.analyticsAccount = config.getString("administration.analytics\_account");

}

## Example 2

|  |  |  |
| --- | --- | --- |
| Source Project: [*openjdk-8*](https://www.programcreek.com/java-api-examples/?project_name=bpupadhyaya%2Fopenjdk-8) Source File: [LoginConfigImpl.java](https://www.programcreek.com/java-api-examples/?code=bpupadhyaya%2Fopenjdk-8%2Fopenjdk-8-master%2Fjdk%2Fsrc%2Fshare%2Fclasses%2Fsun%2Fsecurity%2Fjgss%2FLoginConfigImpl.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=bpupadhyaya%2Fopenjdk-8%2Fopenjdk-8-master%2FLICENSE) | 6 votes |  |

/\*\*

\* A new instance of LoginConfigImpl must be created for each login request

\* since it's only used by a single (caller, mech) pair

\* @param caller defined in GSSUtil as CALLER\_XXX final fields

\* @param oid defined in GSSUtil as XXX\_MECH\_OID final fields

\*/

public LoginConfigImpl(GSSCaller caller, Oid mech) {

this.caller = caller;

if (mech.equals(GSSUtil.GSS\_KRB5\_MECH\_OID)) {

mechName = "krb5";

} else {

throw new IllegalArgumentException(mech.toString() + " not supported");

}

config = java.security.AccessController.doPrivileged

(new java.security.PrivilegedAction <Configuration> () {

public Configuration run() {

return Configuration.getConfiguration();

}

});

}

ADVERTISING

## Example 3

|  |  |  |
| --- | --- | --- |
| Source Project: *[blazingcache](https://www.programcreek.com/java-api-examples/?project_name=diennea%2Fblazingcache)* Source File: [SaslNettyServer.java](https://www.programcreek.com/java-api-examples/?code=diennea%2Fblazingcache%2Fblazingcache-master%2Fblazingcache-core%2Fsrc%2Fmain%2Fjava%2Fblazingcache%2Fsecurity%2Fsasl%2FSaslNettyServer.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=diennea%2Fblazingcache%2Fblazingcache-master%2FLICENSE) | 6 votes |  |

public SaslServerCallbackHandler(Configuration configuration) throws IOException {

AppConfigurationEntry configurationEntries[] = configuration.getAppConfigurationEntry(SERVER\_JAAS\_SECTION);

if (configurationEntries == null) {

String errorMessage = "Could not find a '" + SERVER\_JAAS\_SECTION + "' entry in this configuration: Server cannot start.";

throw new IOException(errorMessage);

}

credentials.clear();

for (AppConfigurationEntry entry : configurationEntries) {

Map<String, ?> options = entry.getOptions();

// Populate DIGEST-MD5 user -> password map with JAAS configuration entries from the "Server" section.

// Usernames are distinguished from other options by prefixing the username with a "user\_" prefix.

for (Map.Entry<String, ?> pair : options.entrySet()) {

String key = pair.getKey();

if (key.startsWith(USER\_PREFIX)) {

String userName = key.substring(USER\_PREFIX.length());

credentials.put(userName, (String) pair.getValue());

}

}

}

}

## Example 4

|  |  |  |
| --- | --- | --- |
| Source Project: *[knox](https://www.programcreek.com/java-api-examples/?project_name=apache%2Fknox)* Source File: [AuthUtils.java](https://www.programcreek.com/java-api-examples/?code=apache%2Fknox%2Fknox-master%2Fgateway-discovery-cm%2Fsrc%2Fmain%2Fjava%2Forg%2Fapache%2Fknox%2Fgateway%2Ftopology%2Fdiscovery%2Fcm%2Fauth%2FAuthUtils.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=apache%2Fknox%2Fknox-master%2FLICENSE) | 6 votes |  |

public static Subject getKerberosSubject() {

Subject subject = null;

try {

Configuration jaasConf = getKerberosJAASConfiguration();

if (jaasConf != null) {

log.attemptingKerberosLogin(getKerberosLoginConfigLocation());

LoginContext lc = new LoginContext(JGSS\_LOGIN\_MODULE, null, null, jaasConf);

lc.login();

subject = lc.getSubject();

}

} catch (Exception e) {

log.failedKerberosLogin(getKerberosLoginConfigLocation(), JGSS\_LOGIN\_MODULE, e);

}

return subject;

}

## Example 5

|  |  |  |
| --- | --- | --- |
| Source Project: *[herddb](https://www.programcreek.com/java-api-examples/?project_name=diennea%2Fherddb)* Source File: [SaslNettyServer.java](https://www.programcreek.com/java-api-examples/?code=diennea%2Fherddb%2Fherddb-master%2Fherddb-core%2Fsrc%2Fmain%2Fjava%2Fherddb%2Fsecurity%2Fsasl%2FSaslNettyServer.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=diennea%2Fherddb%2Fherddb-master%2FLICENSE) | 6 votes |  |

public SaslServerCallbackHandler(Configuration configuration) throws IOException {

AppConfigurationEntry[] configurationEntries = configuration.getAppConfigurationEntry(JASS\_SERVER\_SECTION);

if (configurationEntries == null) {

String errorMessage = "Could not find a '" + JASS\_SERVER\_SECTION + "' entry in this configuration: Server cannot start.";

throw new IOException(errorMessage);

}

credentials.clear();

for (AppConfigurationEntry entry : configurationEntries) {

Map<String, ?> options = entry.getOptions();

// Populate DIGEST-MD5 user -> password map with JAAS configuration entries from the "Server" section.

// Usernames are distinguished from other options by prefixing the username with a "user\_" prefix.

for (Map.Entry<String, ?> pair : options.entrySet()) {

String key = pair.getKey();

if (key.startsWith(USER\_PREFIX)) {

String userName = key.substring(USER\_PREFIX.length());

credentials.put(userName, (String) pair.getValue());

}

}

}

}

## Example 6

|  |  |  |
| --- | --- | --- |
| Source Project: [*dragonwell8\_jdk*](https://www.programcreek.com/java-api-examples/?project_name=alibaba%2Fdragonwell8_jdk) Source File: [DynamicConfigurationTest.java](https://www.programcreek.com/java-api-examples/?code=alibaba%2Fdragonwell8_jdk%2Fdragonwell8_jdk-master%2Ftest%2Fjavax%2Fsecurity%2Fauth%2Flogin%2FLoginContext%2FDynamicConfigurationTest.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=alibaba%2Fdragonwell8_jdk%2Fdragonwell8_jdk-master%2FLICENSE) | 6 votes |  |

public static void testLogin(String confName, char[] passwd,

Configuration cf, boolean expectException) {

try {

CallbackHandler ch = new MyCallbackHandler("testUser", passwd);

LoginContext lc = new LoginContext(confName, new Subject(),

ch, cf);

lc.login();

if (expectException) {

throw new RuntimeException("Login Test failed: "

+ "expected LoginException not thrown");

}

} catch (LoginException le) {

if (!expectException) {

System.out.println("Login Test failed: "

+ "received Unexpected exception.");

throw new RuntimeException(le);

}

}

}

## Example 7

|  |  |  |
| --- | --- | --- |
| Source Project: *[hottub](https://www.programcreek.com/java-api-examples/?project_name=dsrg-uoft%2Fhottub)* Source File: [DynamicConfigurationTest.java](https://www.programcreek.com/java-api-examples/?code=dsrg-uoft%2Fhottub%2Fhottub-master%2Fjdk%2Ftest%2Fjavax%2Fsecurity%2Fauth%2Flogin%2FLoginContext%2FDynamicConfigurationTest.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=dsrg-uoft%2Fhottub%2Fhottub-master%2FLICENSE) | 6 votes |  |

public static void testLogin(String confName, char[] passwd,

Configuration cf, boolean expectException) {

try {

CallbackHandler ch = new MyCallbackHandler("testUser", passwd);

LoginContext lc = new LoginContext(confName, new Subject(),

ch, cf);

lc.login();

if (expectException) {

throw new RuntimeException("Login Test failed: "

+ "expected LoginException not thrown");

}

} catch (LoginException le) {

if (!expectException) {

System.out.println("Login Test failed: "

+ "received Unexpected exception.");

throw new RuntimeException(le);

}

}

}

## Example 8

|  |  |  |
| --- | --- | --- |
| Source Project: [*openjdk-8-source*](https://www.programcreek.com/java-api-examples/?project_name=keerath%2Fopenjdk-8-source) Source File: [LoginConfigImpl.java](https://www.programcreek.com/java-api-examples/?code=keerath%2Fopenjdk-8-source%2Fopenjdk-8-source-master%2Fjdk%2Fsrc%2Fshare%2Fclasses%2Fsun%2Fsecurity%2Fjgss%2FLoginConfigImpl.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=keerath%2Fopenjdk-8-source%2Fopenjdk-8-source-master%2FLICENSE) | 6 votes |  |

/\*\*

\* A new instance of LoginConfigImpl must be created for each login request

\* since it's only used by a single (caller, mech) pair

\* @param caller defined in GSSUtil as CALLER\_XXX final fields

\* @param oid defined in GSSUtil as XXX\_MECH\_OID final fields

\*/

public LoginConfigImpl(GSSCaller caller, Oid mech) {

this.caller = caller;

if (mech.equals(GSSUtil.GSS\_KRB5\_MECH\_OID)) {

mechName = "krb5";

} else {

throw new IllegalArgumentException(mech.toString() + " not supported");

}

config = java.security.AccessController.doPrivileged

(new java.security.PrivilegedAction <Configuration> () {

public Configuration run() {

return Configuration.getConfiguration();

}

});

}

## Example 9

|  |  |  |
| --- | --- | --- |
| Source Project: [*openjdk-jdk8u*](https://www.programcreek.com/java-api-examples/?project_name=AdoptOpenJDK%2Fopenjdk-jdk8u) Source File: [DynamicConfigurationTest.java](https://www.programcreek.com/java-api-examples/?code=AdoptOpenJDK%2Fopenjdk-jdk8u%2Fopenjdk-jdk8u-master%2Fjdk%2Ftest%2Fjavax%2Fsecurity%2Fauth%2Flogin%2FLoginContext%2FDynamicConfigurationTest.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=AdoptOpenJDK%2Fopenjdk-jdk8u%2Fopenjdk-jdk8u-master%2FLICENSE) | 6 votes |  |

public static void testLogin(String confName, char[] passwd,

Configuration cf, boolean expectException) {

try {

CallbackHandler ch = new MyCallbackHandler("testUser", passwd);

LoginContext lc = new LoginContext(confName, new Subject(),

ch, cf);

lc.login();

if (expectException) {

throw new RuntimeException("Login Test failed: "

+ "expected LoginException not thrown");

}

} catch (LoginException le) {

if (!expectException) {

System.out.println("Login Test failed: "

+ "received Unexpected exception.");

throw new RuntimeException(le);

}

}

}

## Example 10

|  |  |  |
| --- | --- | --- |
| Source Project: *[hadoop](https://www.programcreek.com/java-api-examples/?project_name=naver%2Fhadoop)* Source File: [ZKSignerSecretProvider.java](https://www.programcreek.com/java-api-examples/?code=naver%2Fhadoop%2Fhadoop-master%2Fhadoop-common-project%2Fhadoop-auth%2Fsrc%2Fmain%2Fjava%2Forg%2Fapache%2Fhadoop%2Fsecurity%2Fauthentication%2Futil%2FZKSignerSecretProvider.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=naver%2Fhadoop%2Fhadoop-master%2FLICENSE) | 6 votes |  |

private String setJaasConfiguration(Properties config) throws Exception {

String keytabFile = config.getProperty(ZOOKEEPER\_KERBEROS\_KEYTAB).trim();

if (keytabFile == null || keytabFile.length() == 0) {

throw new IllegalArgumentException(ZOOKEEPER\_KERBEROS\_KEYTAB

+ " must be specified");

}

String principal = config.getProperty(ZOOKEEPER\_KERBEROS\_PRINCIPAL)

.trim();

if (principal == null || principal.length() == 0) {

throw new IllegalArgumentException(ZOOKEEPER\_KERBEROS\_PRINCIPAL

+ " must be specified");

}

// This is equivalent to writing a jaas.conf file and setting the system

// property, "java.security.auth.login.config", to point to it

JaasConfiguration jConf =

new JaasConfiguration(JAAS\_LOGIN\_ENTRY\_NAME, principal, keytabFile);

Configuration.setConfiguration(jConf);

return principal.split("[/@]")[0];

}

## Example 11

|  |  |  |
| --- | --- | --- |
| Source Project: *[jstorm](https://www.programcreek.com/java-api-examples/?project_name=alibaba%2Fjstorm)* Source File: [AuthUtils.java](https://www.programcreek.com/java-api-examples/?code=alibaba%2Fjstorm%2Fjstorm-master%2Fjstorm-core%2Fsrc%2Fmain%2Fjava%2Fbacktype%2Fstorm%2Fsecurity%2Fauth%2FAuthUtils.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=alibaba%2Fjstorm%2Fjstorm-master%2FLICENSE) | 6 votes |  |

/\*\*

\* Construct a JAAS configuration object per storm configuration file

\*

\* @param storm\_conf Storm configuration

\* @return JAAS configuration object

\*/

public static Configuration GetConfiguration(Map storm\_conf) {

Configuration login\_conf = null;

// find login file configuration from Storm configuration

String loginConfigurationFile = (String) storm\_conf.get("java.security.auth.login.config");

if ((loginConfigurationFile != null) && (loginConfigurationFile.length() > 0)) {

File config\_file = new File(loginConfigurationFile);

if (!config\_file.canRead()) {

throw new RuntimeException("File " + loginConfigurationFile + " cannot be read.");

}

try {

URI config\_uri = config\_file.toURI();

login\_conf = Configuration.getInstance("JavaLoginConfig", new URIParameter(config\_uri));

} catch (Exception ex) {

throw new RuntimeException(ex);

}

}

return login\_conf;

}

## Example 12

|  |  |  |
| --- | --- | --- |
| Source Project: [*jdk8u-jdk*](https://www.programcreek.com/java-api-examples/?project_name=lambdalab-mirror%2Fjdk8u-jdk) Source File: [DynamicConfigurationTest.java](https://www.programcreek.com/java-api-examples/?code=lambdalab-mirror%2Fjdk8u-jdk%2Fjdk8u-jdk-master%2Ftest%2Fjavax%2Fsecurity%2Fauth%2Flogin%2FLoginContext%2FDynamicConfigurationTest.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=lambdalab-mirror%2Fjdk8u-jdk%2Fjdk8u-jdk-master%2FLICENSE) | 6 votes |  |

public static void testLogin(String confName, char[] passwd,

Configuration cf, boolean expectException) {

try {

CallbackHandler ch = new MyCallbackHandler("testUser", passwd);

LoginContext lc = new LoginContext(confName, new Subject(),

ch, cf);

lc.login();

if (expectException) {

throw new RuntimeException("Login Test failed: "

+ "expected LoginException not thrown");

}

} catch (LoginException le) {

if (!expectException) {

System.out.println("Login Test failed: "

+ "received Unexpected exception.");

throw new RuntimeException(le);

}

}

}

## Example 13

|  |  |  |
| --- | --- | --- |
| Source Project: [*jdk8u\_jdk*](https://www.programcreek.com/java-api-examples/?project_name=JetBrains%2Fjdk8u_jdk) Source File: [DynamicConfigurationTest.java](https://www.programcreek.com/java-api-examples/?code=JetBrains%2Fjdk8u_jdk%2Fjdk8u_jdk-master%2Ftest%2Fjavax%2Fsecurity%2Fauth%2Flogin%2FLoginContext%2FDynamicConfigurationTest.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=JetBrains%2Fjdk8u_jdk%2Fjdk8u_jdk-master%2FLICENSE) | 6 votes |  |

public static void testLogin(String confName, char[] passwd,

Configuration cf, boolean expectException) {

try {

CallbackHandler ch = new MyCallbackHandler("testUser", passwd);

LoginContext lc = new LoginContext(confName, new Subject(),

ch, cf);

lc.login();

if (expectException) {

throw new RuntimeException("Login Test failed: "

+ "expected LoginException not thrown");

}

} catch (LoginException le) {

if (!expectException) {

System.out.println("Login Test failed: "

+ "received Unexpected exception.");

throw new RuntimeException(le);

}

}

}

## Example 14

|  |  |  |
| --- | --- | --- |
| Source Project: [*light-oauth2*](https://www.programcreek.com/java-api-examples/?project_name=networknt%2Flight-oauth2) Source File: [KerberosKDCUtil.java](https://www.programcreek.com/java-api-examples/?code=networknt%2Flight-oauth2%2Flight-oauth2-master%2Fauthhub%2Fsrc%2Fmain%2Fjava%2Fcom%2Fnetworknt%2Foauth%2Fspnego%2FKerberosKDCUtil.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=networknt%2Flight-oauth2%2Flight-oauth2-master%2FLICENSE) | 6 votes |  |

private static Configuration createJaasConfiguration() {

return new Configuration() {

@Override

public AppConfigurationEntry[] getAppConfigurationEntry(String name) {

if (!"KDC".equals(name)) {

throw new IllegalArgumentException("Unexpected name '" + name + "'");

}

AppConfigurationEntry[] entries = new AppConfigurationEntry[1];

Map<String, Object> options = new HashMap<>();

options.put("debug", config.getDebug());

options.put("refreshKrb5Config", "true");

options.put("storeKey", "true");

if("true".equalsIgnoreCase(config.getUseKeyTab())) {

options.put("useKeyTab", config.getUseKeyTab());

options.put("keyTab", config.getKeyTab());

options.put("principal", config.getPrincipal());

}

options.put("isInitiator", "true");

entries[0] = new AppConfigurationEntry("com.sun.security.auth.module.Krb5LoginModule", REQUIRED, options);

return entries;

}

};

}

## Example 15

|  |  |  |
| --- | --- | --- |
| Source Project: *[keycloak](https://www.programcreek.com/java-api-examples/?project_name=keycloak%2Fkeycloak)* Source File: [LoginModulesTest.java](https://www.programcreek.com/java-api-examples/?code=keycloak%2Fkeycloak%2Fkeycloak-master%2Ftestsuite%2Fintegration-arquillian%2Ftests%2Fbase%2Fsrc%2Ftest%2Fjava%2Forg%2Fkeycloak%2Ftestsuite%2Fjaas%2FLoginModulesTest.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=keycloak%2Fkeycloak%2Fkeycloak-master%2FLICENSE) | 6 votes |  |

private Configuration createJaasConfigurationForDirectGrant(String scope) {

return new Configuration() {

@Override

public AppConfigurationEntry[] getAppConfigurationEntry(String name) {

Map<String, Object> options = new HashMap<>();

options.put(AbstractKeycloakLoginModule.KEYCLOAK\_CONFIG\_FILE\_OPTION, DIRECT\_GRANT\_CONFIG\_FILE.getAbsolutePath());

if (scope != null) {

options.put(DirectAccessGrantsLoginModule.SCOPE\_OPTION, scope);

}

AppConfigurationEntry LMConfiguration = new AppConfigurationEntry(DirectAccessGrantsLoginModule.class.getName(), AppConfigurationEntry.LoginModuleControlFlag.REQUIRED, options);

return new AppConfigurationEntry[] { LMConfiguration };

}

};

}

## Example 16

|  |  |  |
| --- | --- | --- |
| Source Project: [*openjdk-jdk9*](https://www.programcreek.com/java-api-examples/?project_name=AdoptOpenJDK%2Fopenjdk-jdk9) Source File: [LoginConfigImpl.java](https://www.programcreek.com/java-api-examples/?code=AdoptOpenJDK%2Fopenjdk-jdk9%2Fopenjdk-jdk9-master%2Fjdk%2Fsrc%2Fjava.security.jgss%2Fshare%2Fclasses%2Fsun%2Fsecurity%2Fjgss%2FLoginConfigImpl.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=AdoptOpenJDK%2Fopenjdk-jdk9%2Fopenjdk-jdk9-master%2FLICENSE) | 6 votes |  |

/\*\*

\* A new instance of LoginConfigImpl must be created for each login request

\* since it's only used by a single (caller, mech) pair

\* @param caller defined in GSSUtil as CALLER\_XXX final fields

\* @param mech defined in GSSUtil as XXX\_MECH\_OID final fields

\*/

public LoginConfigImpl(GSSCaller caller, Oid mech) {

this.caller = caller;

if (mech.equals(GSSUtil.GSS\_KRB5\_MECH\_OID)) {

mechName = "krb5";

} else {

throw new IllegalArgumentException(mech.toString() + " not supported");

}

config = java.security.AccessController.doPrivileged

(new java.security.PrivilegedAction <Configuration> () {

public Configuration run() {

return Configuration.getConfiguration();

}

});

}

## Example 17

|  |  |  |
| --- | --- | --- |
| Source Project: [*openjdk-jdk9*](https://www.programcreek.com/java-api-examples/?project_name=AdoptOpenJDK%2Fopenjdk-jdk9) Source File: [GetInstance.java](https://www.programcreek.com/java-api-examples/?code=AdoptOpenJDK%2Fopenjdk-jdk9%2Fopenjdk-jdk9-master%2Fjdk%2Ftest%2Fjavax%2Fsecurity%2Fauth%2Flogin%2FConfiguration%2FGetInstance.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=AdoptOpenJDK%2Fopenjdk-jdk9%2Fopenjdk-jdk9-master%2FLICENSE) | 6 votes |  |

private int testStringProvider(int testnum) throws Exception {

// get an instance of JavaLoginConfig from SUN

Configuration c = Configuration.getInstance(JAVA\_CONFIG, null, "SUN");

doTest(c, testnum++);

// get an instance of JavaLoginConfig from SunRsaSign

try {

c = Configuration.getInstance(JAVA\_CONFIG, null, "SunRsaSign");

throw new SecurityException("test " + testnum++ + " failed");

} catch (NoSuchAlgorithmException nsae) {

// good

System.out.println("test " + testnum++ + " passed");

}

// get an instance of JavaLoginConfig from FOO

try {

c = Configuration.getInstance(JAVA\_CONFIG, null, "FOO");

throw new SecurityException("test " + testnum++ + " failed");

} catch (NoSuchProviderException nspe) {

// good

System.out.println("test " + testnum++ + " passed");

}

return testnum;

}

## Example 18

|  |  |  |
| --- | --- | --- |
| Source Project: [*openjdk-jdk9*](https://www.programcreek.com/java-api-examples/?project_name=AdoptOpenJDK%2Fopenjdk-jdk9) Source File: [GetInstance.java](https://www.programcreek.com/java-api-examples/?code=AdoptOpenJDK%2Fopenjdk-jdk9%2Fopenjdk-jdk9-master%2Fjdk%2Ftest%2Fjavax%2Fsecurity%2Fauth%2Flogin%2FConfiguration%2FGetInstance.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=AdoptOpenJDK%2Fopenjdk-jdk9%2Fopenjdk-jdk9-master%2FLICENSE) | 6 votes |  |

private int testProvider(int testnum) throws Exception {

// get an instance of JavaLoginConfig from SUN

Configuration c = Configuration.getInstance(JAVA\_CONFIG,

null,

Security.getProvider("SUN"));

doTest(c, testnum++);

// get an instance of JavaLoginConfig from SunRsaSign

try {

c = Configuration.getInstance(JAVA\_CONFIG,

null,

Security.getProvider("SunRsaSign"));

throw new SecurityException("test " + testnum++ + " failed");

} catch (NoSuchAlgorithmException nsae) {

// good

System.out.println("test " + testnum++ + " passed");

}

return testnum;

}

## Example 19

|  |  |  |
| --- | --- | --- |
| Source Project: *[datacollector](https://www.programcreek.com/java-api-examples/?project_name=streamsets%2Fdatacollector)* Source File: [TestSaslEnabledKafka.java](https://www.programcreek.com/java-api-examples/?code=streamsets%2Fdatacollector%2Fdatacollector-master%2Fsdc-kafka_0_9%2Fsrc%2Ftest%2Fjava%2Fcom%2Fstreamsets%2Fpipeline%2Fkafka%2Fimpl%2FTestSaslEnabledKafka.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=streamsets%2Fdatacollector%2Fdatacollector-master%2FLICENSE) | 6 votes |  |

@BeforeClass

public static void beforeClass() throws Exception {

testDir = new File("target", UUID.randomUUID().toString()).getAbsoluteFile();

Assert.assertTrue(testDir.mkdirs());

File kdcDir = new File(testDir, KDC);

Assert.assertTrue(kdcDir.mkdirs());

keytabFile = new File(testDir, TEST\_KEYTAB);

jaasConfigFile = new File(testDir, KAFKA\_JAAS\_CONF);

jaasConfigFile.createNewFile();

jaasConfigFile.setReadable(true);

String jaasConf = JAAS\_CONF.replaceAll("keyTabFile", keytabFile.getAbsolutePath());

FileOutputStream outputStream = new FileOutputStream(jaasConfigFile);

IOUtils.write(jaasConf, outputStream);

outputStream.close();

plainTextPort = NetworkUtils.getRandomPort();

securePort = NetworkUtils.getRandomPort();

// reload configuration when getConfiguration is called next

Configuration.setConfiguration(null);

System.setProperty(JAVA\_SECURITY\_AUTH\_LOGIN\_CONFIG, jaasConfigFile.getAbsolutePath());

SecureKafkaBase.beforeClass();

}

## Example 20

|  |  |  |
| --- | --- | --- |
| Source Project: *[jstorm](https://www.programcreek.com/java-api-examples/?project_name=alibaba%2Fjstorm)* Source File: [ClientCallbackHandler.java](https://www.programcreek.com/java-api-examples/?code=alibaba%2Fjstorm%2Fjstorm-master%2Fjstorm-core%2Fsrc%2Fmain%2Fjava%2Fbacktype%2Fstorm%2Fsecurity%2Fauth%2Fdigest%2FClientCallbackHandler.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=alibaba%2Fjstorm%2Fjstorm-master%2FLICENSE) | 6 votes |  |

/\*\*

\* Constructor based on a JAAS configuration

\*

\* For digest, you should have a pair of user name and password defined.

\*

\* @param configuration

\* @throws IOException

\*/

public ClientCallbackHandler(Configuration configuration) throws IOException {

if (configuration == null)

return;

AppConfigurationEntry configurationEntries[] = configuration.getAppConfigurationEntry(AuthUtils.LOGIN\_CONTEXT\_CLIENT);

if (configurationEntries == null) {

String errorMessage = "Could not find a '" + AuthUtils.LOGIN\_CONTEXT\_CLIENT + "' entry in this configuration: Client cannot start.";

throw new IOException(errorMessage);

}

\_password = "";

for (AppConfigurationEntry entry : configurationEntries) {

if (entry.getOptions().get(USERNAME) != null) {

\_username = (String) entry.getOptions().get(USERNAME);

}

if (entry.getOptions().get(PASSWORD) != null) {

\_password = (String) entry.getOptions().get(PASSWORD);

}

}

}

## Example 21

|  |  |  |
| --- | --- | --- |
| Source Project: [*openjdk-jdk9*](https://www.programcreek.com/java-api-examples/?project_name=AdoptOpenJDK%2Fopenjdk-jdk9) Source File: [GetInstance.java](https://www.programcreek.com/java-api-examples/?code=AdoptOpenJDK%2Fopenjdk-jdk9%2Fopenjdk-jdk9-master%2Fjdk%2Ftest%2Fjavax%2Fsecurity%2Fauth%2Flogin%2FConfiguration%2FGetInstance.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=AdoptOpenJDK%2Fopenjdk-jdk9%2Fopenjdk-jdk9-master%2FLICENSE) | 6 votes |  |

private void doCustomTest(Configuration c,

int testnum,

Provider custom) throws Exception {

testnum = doCommon(c, testnum);

// test getProvider

if (custom == c.getProvider() &&

"GetInstanceProvider".equals(c.getProvider().getName())) {

System.out.println("test " + testnum + " (getProvider) passed");

} else {

throw new SecurityException

("test " + testnum + " (getProvider) failed");

}

// test getType

if ("GetInstanceConfigSpi".equals(c.getType())) {

System.out.println("test " + testnum + "(getType) passed");

} else {

throw new SecurityException("test " + testnum +

" (getType) failed");

}

}

## Example 22

|  |  |  |
| --- | --- | --- |
| Source Project: [*openjdk-jdk9*](https://www.programcreek.com/java-api-examples/?project_name=AdoptOpenJDK%2Fopenjdk-jdk9) Source File: [GetInstance.java](https://www.programcreek.com/java-api-examples/?code=AdoptOpenJDK%2Fopenjdk-jdk9%2Fopenjdk-jdk9-master%2Fjdk%2Ftest%2Fjavax%2Fsecurity%2Fauth%2Flogin%2FConfiguration%2FGetInstance.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=AdoptOpenJDK%2Fopenjdk-jdk9%2Fopenjdk-jdk9-master%2FLICENSE) | 6 votes |  |

private void doTest(Configuration c, int testnum) throws Exception {

testnum = doCommon(c, testnum);

// test getProvider

if ("SUN".equals(c.getProvider().getName())) {

System.out.println("test " + testnum + " (getProvider) passed");

} else {

throw new SecurityException("test " + testnum +

" (getProvider) failed");

}

// test getType

if (JAVA\_CONFIG.equals(c.getType())) {

System.out.println("test " + testnum + " (getType) passed");

} else {

throw new SecurityException("test " + testnum +

" (getType) failed");

}

}

## Example 23

|  |  |  |
| --- | --- | --- |
| Source Project: [*deprecated-security-advanced-modules*](https://www.programcreek.com/java-api-examples/?project_name=opendistro-for-elasticsearch%2Fdeprecated-security-advanced-modules) Source File: [JaasKrbUtil.java](https://www.programcreek.com/java-api-examples/?code=opendistro-for-elasticsearch%2Fdeprecated-security-advanced-modules%2Fdeprecated-security-advanced-modules-master%2Fsrc%2Fmain%2Fjava%2Fcom%2Famazon%2Fdlic%2Fauth%2Fhttp%2Fkerberos%2Futil%2FJaasKrbUtil.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=opendistro-for-elasticsearch%2Fdeprecated-security-advanced-modules%2Fdeprecated-security-advanced-modules-master%2FLICENSE) | 5 votes |  |

public static Subject loginUsingTicketCache(final String principal, final Path cachePath) throws LoginException {

final Set<Principal> principals = new HashSet<Principal>();

principals.add(new KerberosPrincipal(principal));

final Subject subject = new Subject(false, principals, new HashSet<Object>(), new HashSet<Object>());

final Configuration conf = useTicketCache(principal, cachePath);

final String confName = "TicketCacheConf";

final LoginContext loginContext = new LoginContext(confName, subject, null, conf);

loginContext.login();

return loginContext.getSubject();

}

## Example 24

|  |  |  |
| --- | --- | --- |
| Source Project: [*presto*](https://www.programcreek.com/java-api-examples/?project_name=prestosql%2Fpresto) Source File: [KerberosAuthentication.java](https://www.programcreek.com/java-api-examples/?code=prestosql%2Fpresto%2Fpresto-master%2Fpresto-plugin-toolkit%2Fsrc%2Fmain%2Fjava%2Fio%2Fprestosql%2Fplugin%2Fbase%2Fauthentication%2FKerberosAuthentication.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=prestosql%2Fpresto%2Fpresto-master%2FLICENSE) | 5 votes |  |

private static Configuration createConfiguration(String principal, String keytabLocation)

{

ImmutableMap.Builder<String, String> optionsBuilder = ImmutableMap.<String, String>builder()

.put("useKeyTab", "true")

.put("storeKey", "true")

.put("doNotPrompt", "true")

.put("isInitiator", "true")

.put("principal", principal)

.put("keyTab", keytabLocation);

if (log.isDebugEnabled()) {

optionsBuilder.put("debug", "true");

}

Map<String, String> options = optionsBuilder.build();

return new Configuration()

{

@Override

public AppConfigurationEntry[] getAppConfigurationEntry(String name)

{

return new AppConfigurationEntry[] {

new AppConfigurationEntry(

KERBEROS\_LOGIN\_MODULE,

AppConfigurationEntry.LoginModuleControlFlag.REQUIRED,

options)};

}

};

}

## Example 25

|  |  |  |
| --- | --- | --- |
| Source Project: *[elasticsearch-shield-kerberos-realm](https://www.programcreek.com/java-api-examples/?project_name=codecentric%2Felasticsearch-shield-kerberos-realm)* Source File: [JaasKrbUtil.java](https://www.programcreek.com/java-api-examples/?code=codecentric%2Felasticsearch-shield-kerberos-realm%2Felasticsearch-shield-kerberos-realm-master%2Fsrc%2Fmain%2Fjava%2Fde%2Fcodecentric%2Felasticsearch%2Fplugin%2Fkerberosrealm%2Fsupport%2FJaasKrbUtil.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=codecentric%2Felasticsearch-shield-kerberos-realm%2Felasticsearch-shield-kerberos-realm-master%2FLICENSE) | 5 votes |  |

public static Subject loginUsingKeytab(final String principal, final Path keytabPath, final boolean initiator) throws LoginException {

final Set<Principal> principals = new HashSet<Principal>();

principals.add(new KerberosPrincipal(principal));

final Subject subject = new Subject(false, principals, new HashSet<Object>(), new HashSet<Object>());

final Configuration conf = useKeytab(principal, keytabPath, initiator);

final String confName = "KeytabConf";

final LoginContext loginContext = new LoginContext(confName, subject, null, conf);

loginContext.login();

return loginContext.getSubject();

}

## Example 26

|  |  |  |
| --- | --- | --- |
| Source Project: *[kareldb](https://www.programcreek.com/java-api-examples/?project_name=rayokota%2Fkareldb)* Source File: [RemoteClusterHttpAuthTestHarness.java](https://www.programcreek.com/java-api-examples/?code=rayokota%2Fkareldb%2Fkareldb-master%2Fkareldb-server%2Fsrc%2Ftest%2Fjava%2Fio%2Fkareldb%2Fserver%2Futils%2FRemoteClusterHttpAuthTestHarness.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=rayokota%2Fkareldb%2Fkareldb-master%2FLICENSE) | 5 votes |  |

@Before

public void setUp() throws Exception {

tempDir = Files.createTempDir();

passwordFile = new File(tempDir, "password-file");

jaasConfigFile = new File(tempDir, "jaas\_config.file");

writePasswordFile(passwordFile);

writeJaasFile(jaasConfigFile, passwordFile);

Configuration.setConfiguration(null);

System.setProperty("java.security.auth.login.config", jaasConfigFile.getAbsolutePath());

super.setUp();

}

## Example 27

|  |  |  |
| --- | --- | --- |
| Source Project: [*incubator-atlas*](https://www.programcreek.com/java-api-examples/?project_name=apache%2Fincubator-atlas) Source File: [InMemoryJAASConfiguration.java](https://www.programcreek.com/java-api-examples/?code=apache%2Fincubator-atlas%2Fincubator-atlas-master%2Fcommon%2Fsrc%2Fmain%2Fjava%2Forg%2Fapache%2Fatlas%2Fsecurity%2FInMemoryJAASConfiguration.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=apache%2Fincubator-atlas%2Fincubator-atlas-master%2FLICENSE) | 5 votes |  |

public static void init(org.apache.commons.configuration.Configuration atlasConfiguration) throws AtlasException {

LOG.debug("==> InMemoryJAASConfiguration.init()");

if (atlasConfiguration != null && !atlasConfiguration.isEmpty()) {

Properties properties = ConfigurationConverter.getProperties(atlasConfiguration);

init(properties);

} else {

throw new AtlasException("Failed to load JAAS application properties: configuration NULL or empty!");

}

LOG.debug("<== InMemoryJAASConfiguration.init()");

}

## Example 28

|  |  |  |
| --- | --- | --- |
| Source Project: *[knox](https://www.programcreek.com/java-api-examples/?project_name=apache%2Fknox)* Source File: [RemoteConfigurationRegistryJAASConfig.java](https://www.programcreek.com/java-api-examples/?code=apache%2Fknox%2Fknox-master%2Fgateway-service-remoteconfig%2Fsrc%2Fmain%2Fjava%2Forg%2Fapache%2Fknox%2Fgateway%2Fservice%2Fconfig%2Fremote%2Fzk%2FRemoteConfigurationRegistryJAASConfig.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=apache%2Fknox%2Fknox-master%2FLICENSE) | 5 votes |  |

private RemoteConfigurationRegistryJAASConfig(List<RemoteConfigurationRegistryConfig> configs, AliasService aliasService) {

try {

delegate = Configuration.getConfiguration();

} catch(Exception e) {

//populate the original error with a meaningful message; logging will happen later in the call hierarchy

final String message = String.format(Locale.ROOT, "%s: %s", JAAS\_CONFIG\_ERRROR\_PREFIX, System.getProperty(GatewayConfig.KRB5\_LOGIN\_CONFIG, "Undefined"));

throw new ConfigurationException(message, e);

}

validateKeytabFile();

this.aliasService = aliasService;

// Populate context entries

for (RemoteConfigurationRegistryConfig config : configs) {

if (config.isSecureRegistry()) {

contextEntries.put(config.getName(), createEntries(config));

}

}

// If there is at least one context entry, then set this as the client configuration

if (!contextEntries.isEmpty()) {

// TODO: PJZ: ZooKeeper 3.6.0 will have per-client JAAS Configuration support; Upgrade ASAP!!

// For now, set this as the static JAAS configuration

Configuration.setConfiguration(this);

}

}

## Example 29

|  |  |  |
| --- | --- | --- |
| Source Project: *[swellrt](https://www.programcreek.com/java-api-examples/?project_name=SwellRT%2Fswellrt)* Source File: [ServerModule.java](https://www.programcreek.com/java-api-examples/?code=SwellRT%2Fswellrt%2Fswellrt-master%2Fwave%2Fsrc%2Fmain%2Fjava%2Forg%2Fwaveprotocol%2Fbox%2Fserver%2FServerModule.java)   License: [Apache License 2.0](https://www.programcreek.com/java-api-examples/?code=SwellRT%2Fswellrt%2Fswellrt-master%2FLICENSE) | 5 votes |  |

@Override

protected void configure() {

bind(WaveServerImpl.class).in(Singleton.class);

// Receive updates from the outside world, and push them into our local Wave

// Server.

bind(WaveletFederationListener.Factory.class).annotatedWith(FederationRemoteBridge.class).to(

WaveServerImpl.class);

// Provide history and respond to submits about our own local waves.

bind(WaveletFederationProvider.class).annotatedWith(FederationHostBridge.class).to(

WaveServerImpl.class);

install(waveServerModule);

TypeLiteral<List<String>> certs = new TypeLiteral<List<String>>() {};

bind(certs).annotatedWith(Names.named("certs")).toInstance(Arrays.<String> asList());

bind(ProtoSerializer.class).in(Singleton.class);

bind(Configuration.class).toInstance(Configuration.getConfiguration());

bind(SessionManager.class).to(SessionManagerImpl.class).in(Singleton.class);

bind(ServerRpcProvider.class).in(Singleton.class);

// bind(RobotRegistrar.class).to(RobotRegistrarImpl.class);

requestStaticInjection(WebSocketChannel.class);

}

## Example 30

|  |  |  |
| --- | --- | --- |
| Source Project: [*jdk8u\_jdk*](https://www.programcreek.com/java-api-examples/?project_name=JetBrains%2Fjdk8u_jdk) Source File: [DynamicConfigurationTest.java](https://www.programcreek.com/java-api-examples/?code=JetBrains%2Fjdk8u_jdk%2Fjdk8u_jdk-master%2Ftest%2Fjavax%2Fsecurity%2Fauth%2Flogin%2FLoginContext%2FDynamicConfigurationTest.java)   License: [GNU General Public License v2.0](https://www.programcreek.com/java-api-examples/?code=JetBrains%2Fjdk8u_jdk%2Fjdk8u_jdk-master%2FLICENSE) | 5 votes |  |

public static void main(String... args) {

String rightConfigName = "PT";

String wrongConfigName = "NT";

char[] rightPwd = new char[]{'t', 'e', 's', 't', 'P', 'a', 's', 's',

'w', 'o', 'r', 'd', '1'};

char[] wrongPwd = new char[]{'w', 'r', 'o', 'n', 'g', 'P', 'a', 's',

's','w', 'o', 'r', 'd'};

// Test with wrong configuration name

// Expect LoginException when initiate a new LoginContext object

testConfigName(wrongConfigName, true);

System.out.println("Wrong Config Name Test passed ");

// Spedify two loginModules: SmartLoginModule and DummyLoginModule

// Flags: required-required

// Test with right password for SmartLoginModule

// No exception is expected

Configuration cf = new MyConfiguration();

testLogin(rightConfigName, rightPwd, cf, false);

System.out.println("Positive test passed");

// Spedify two loginModules: SmartLoginModule and DummyLoginModule

// Flags: required-required

// Test with wrong password for SmartLoginModule

// Expect LoginException by calling LoginContext.login() method

testLogin(rightConfigName, wrongPwd, cf, true);

System.out.println("Should fail test passed");

// Spedify two loginModules: SmartLoginModule and DummyLoginModule

// Change the flags from required-required to optional-sufficient

// Test with wrong password for SmartLoginModule, while DummyLoginModule

// always passes

// No Exception is expected

cf = new MyConfiguration(true);

testLogin(rightConfigName, wrongPwd, cf, false);

System.out.println("One module fails where are other module succeeeds "

+ "Test passed with optional-sufficient flags");

}

# JAAS (JAVA Authentication and Authorization Service) Login Configuration File

JAAS authentication is performed in a pluggable fashion, so Java applications can remain independent from underlying authentication technologies. Configuration information such as the desired authentication technology is specified at runtime. The source of the configuration information (for example, a file or a database) is up to the current [javax.security.auth.login.Configuration](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) implementation. The default Configuration implementation from Sun Microsystems reads configuration information from configuration files, which are described in this document.

## Login Configuration File Structure and Contents

A login configuration file consists of one or more entries, each specifying which underlying authentication technology should be used for a particular application or applications. The structure of each entry is the following:

<entry name> {

<LoginModule> <flag> <LoginModule options>;

<LoginModule> <flag> <LoginModule options>;

. . .

};

Thus, each login configuration file entry consists of a name followed by one or more LoginModule-specific items. Each LoginModule-specific item specifies a LoginModule, a flag value, and options to be passed to the LoginModule. (These are described futher below.) Each LoginModule-specific item is terminated by a semicolon and the entire group of items is enclosed in braces. Each configuration file entry is terminated by a semicolon.

As an example, the login configuration file used for the [JAAS Authentication](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html) tutorial contains just one entry, which is

**JaasSample {**

**com.sun.security.auth.module.Krb5LoginModule required;**

**};**

Here, the entry is named "JaasSample" and that is the name that the JAAS Authentication tutorial application (JaasAcn.java) uses to refer to this entry. The entry specifies that the LoginModule to be used to do the user authentication is the Krb5LoginModule in the com.sun.security.auth.module package and that this Krb5LoginModule is required to "succeed" in order for authentication to be considered successful. The Krb5LoginModule succeeds only if the name and password supplied by the user are successfully used to log the user into the Kerberos KDC.

The **name** for an entry in a login configuration file is the name that applications use to refer to the entry when they instantiate a LoginContext, as described in [Instantiating a LoginContext](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html#ClientLC) in the JAAS authentication tutorial. The name can be whatever name the application developer wishes to use. Here, the term "application" refers to whatever code does the JAAS login, whether it is your application (as shown in the [JAAS Authentication](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html) and [JAAS Authorization](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnAndAzn.html) tutorials) or a Login utility that does the JAAS operations for you (as shown in the [Use of JAAS Login Utility](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/LoginSample.html) and [Use of Java GSS-API for Secure Message Exchanges Using JAAS Login Utility](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/ClientServer.html) tutorials.)

The subparts of each LoginModule-specific item are described by the following. See the [Configuration](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) documentation for more information.

* **LoginModule**

This specifies the fully qualified class name for a class that implements a particular authentication technology. Specifically, the class must implement the javax.security.auth.spi.LoginModule interface. A typical LoginModule may prompt for and verify a user name and password. Any vendor can provide a LoginModule implementation that you can use. Some implementations are supplied with the JRE from Sun Microsystems. Throughout these tutorials we use the Krb5LoginModule in the com.sun.security.auth.module package. The Krb5LoginModule uses Kerberos as the underlying authentication technology. You can view the reference documentation for the various LoginModules, all in the com.sun.security.auth package:

* + [JndiLoginModule](https://docs.oracle.com/javase/7/docs/jre/api/security/jaas/spec/com/sun/security/auth/module/JndiLoginModule.html)
  + [KeyStoreLoginModule](https://docs.oracle.com/javase/7/docs/jre/api/security/jaas/spec/com/sun/security/auth/module/KeyStoreLoginModule.html)
  + [Krb5LoginModule](https://docs.oracle.com/javase/7/docs/jre/api/security/jaas/spec/com/sun/security/auth/module/Krb5LoginModule.html)
  + [NTLoginModule](https://docs.oracle.com/javase/7/docs/jre/api/security/jaas/spec/com/sun/security/auth/module/NTLoginModule.html)
  + [UnixLoginModule](https://docs.oracle.com/javase/7/docs/jre/api/security/jaas/spec/com/sun/security/auth/module/UnixLoginModule.html)
* **flag**

The flag value indicates whether success of the LoginModule is "required", "requisite", "sufficient", or "optional".

* **LoginModule options**

If the specified LoginModule implementation has options that can be set, you specify any desired option values here. This is a space-separated list of values which are passed directly to the underlying LoginModule. Options are defined by the LoginModule itself, and control the behavior within it. For example, a LoginModule may define options to support debugging/testing capabilities. See the [Krb5LoginModule documentation](https://docs.oracle.com/javase/7/docs/jre/api/security/jaas/spec/com/sun/security/auth/module/Krb5LoginModule.html) for information about the options defined for the Krb5LoginModule used for all the tutorials in this series.

The correct way to specify options in the configuration file is by using a name-value pairing, for example *debug=true*, where the option name (in this case, "debug") and value (in this case, "true") should be separated by an "equals" symbol.

## Where to Specify Which Login Configuration File Should Be Used

The configuration file to be used can be specified in one of two ways:

1. On the command line.

You can use a -Djava.security.auth.login.config command line argument to specify the login configuration file that should be used. We use this approach for all the tutorials. For example, we run our JaasAcn application in the JAAS Authentication tutorial using the following command, which specifies that the configuration file is the jaas.conf file in the current directory:

java -Djava.security.auth.login.config=jaas.conf JaasAcn

1. In the Java security properties file.

An alternate approach to specifying the location of the login configuration file is to indicate its URL as the value of a login.config.url.*n* property in the security properties file. The security properties file is the java.security file located in the lib/security directory of the JRE.

Here, *n* indicates a consecutively-numbered integer starting with 1. Thus, if desired, you can specify more than one login configuration file by indicating one file's URL for the login.config.url.1 property, a second file's URL for the login.config.url.2 property, and so on. If more than one login configuration file is specified (that is, if n > 1), then the files are read and concatenated into a single configuration.

Here is an example of what would need to be added to the security properties file in order to indicate the jaas.conf login configuration file used by this tutorial. This example assumes the file is in the C:\AcnTest directory on a Microsoft Windows system:

login.config.url.1=file:C:/AcnTest/jaas.conf

(Note that URLs always use forward slashes, regardless of what operating system the user is running.)

# JAAS Authentication

The Java Authentication and Authorization Service (JAAS) was introduced as an optional package to the Java 2 SDK, Standard Edition (J2SDK), v 1.3. JAAS has now been integrated into the J2SDK 1.4.

JAAS can be used for two purposes:

* for *authentication* of users, to reliably and securely determine who is currently executing Java code, regardless of whether the code is running as an application, an applet, a bean, or a servlet; and
* for *authorization* of users to ensure they have the access control rights (permissions) required to do the actions performed.

This section provides a basic tutorial for the authentication component. The authorization component will be described in the [JAAS Authorization](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnAndAzn.html) tutorial.

JAAS authentication is performed in a *pluggable* fashion. This permits Java applications to remain independent from underlying authentication technologies. New or updated technologies can be plugged in without requiring modifications to the application itself. An implementation for a particular authentication technology to be used is determined at runtime. The implementation is specified in a login configuration file. The authentication technology used for this tutorial is Kerberos. (See [Kerberos Requirements](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/KerberosReq.html).)

The rest of this tutorial consists of the following sections:

1. [The Authentication Tutorial Code](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html#AcCode)
2. [The Login Configuration](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html#ConfigFile)
3. [Running the Code](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html#RunAc)
4. [Running the Code with a Security Manager](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html#RunAcWSM)

If you want to first see the tutorial code in action, you can skip directly to [Running the Code](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html#RunAc) and then go back to the other sections to learn about coding and configuration file details.

## The Authentication Tutorial Code

Our authentication tutorial code is contained in a single source file, [JaasAcn.java](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/JaasAcn.java). This file's main method performs the authentication and then reports whether or not authentication succeeded.

The code for authenticating the user is very simple, consisting of just two steps:

1. [Instantiate a LoginContext.](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html#ClientLC)
2. [Call the LoginContext's login method.](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html#ClientLogin)

First the basic code is shown, followed by a [complete listing](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html#AcnFullCode) of the JaasAcn.java source file, complete with import statements and error handling.

### Instantiating a LoginContext

In order to authenticate a user, you first need a javax.security.auth.login.LoginContext. Here is the basic way to instantiate a LoginContext:

import javax.security.auth.login.\*;

. . .

LoginContext lc =

new LoginContext(<config file entry name>,

<CallbackHandler to be used for user interaction>);

and here is the specific way our tutorial code does the instantiation:

import javax.security.auth.login.\*;

import com.sun.security.auth.callback.TextCallbackHandler;

. . .

**LoginContext lc =**

**new LoginContext("JaasSample",**

**new TextCallbackHandler());**

The arguments are the following:

1. **The name of an entry in the JAAS login configuration file**

This is the name for the LoginContext to use to look up an entry for this application in the JAAS login configuration file, described [here](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html#ConfigFile). Such an entry specifies the class(es) that implement the desired underlying authentication technology(ies). The class(es) must implement the LoginModule interface, which is in the javax.security.auth.spi package.

In our sample code, we use the Krb5LoginModule in the com.sun.security.auth.module package, which performs Kerberos authentication.

The entry in the login configuration file we use for this tutorial (see [jaas.conf](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/jaas.conf)) has the name "JaasSample", so that is the name we specify as the first argument to the LoginContext constructor.

1. **A CallbackHandler instance.**

When a LoginModule needs to communicate with the user, for example to ask for a user name and password, it does not do so directly. That is because there are various ways of communicating with a user, and it is desirable for LoginModules to remain independent of the different types of user interaction. Rather, the LoginModule invokes a CallbackHandler to perform the user interaction and obtain the requested information, such as the user name and password. (CallbackHandler is an interface in the javax.security.auth.callback pkg.)

An instance of the particular CallbackHandler to be used is specified as the second argument to the LoginContext constructor. The LoginContext forwards that instance to the underlying LoginModule (in our case Krb5LoginModule). An application typically provides its own CallbackHandler implementation. Two simple CallbackHandlers, TextCallbackHandler and DialogCallbackHandler, are provided in the com.sun.security.auth.callback package as sample implementations. Our code uses the TextCallbackHandler, which outputs information to and reads input from the command line.

### Calling the LoginContext's login Method

Once we have a LoginContext lc, we can call its login method to carry out the authentication process:

**lc.login();**

The LoginContext instantiates a new empty [javax.security.auth.Subject](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) object (which represents the user or service being authenticated). The LoginContext constructs the configured LoginModule (in our case Krb5LoginModule) and initializes it with this new Subject and TextCallbackHandler.

The LoginContext's login method then calls methods in the Krb5LoginModule to perform the login and authentication. The Krb5LoginModule will utilize the TextCallbackHandler to obtain the user name and password. Then the Krb5LoginModule will use this information to get the user credentials from the Kerberos KDC. See the [Kerberos reference documentation](http://web.mit.edu/kerberos/www/index.html).

If authentication is successful, the Krb5LoginModule populates the Subject with (1) a Kerberos Principal representing the user and (2) the user's credentials (TGT). See [Subjects, Principals, Authentication, and Credentials](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/glossary.html) for information on these terms.

The calling application can subsequently retrieve the authenticated Subject by calling the LoginContext's getSubject method, although doing so is not necessary for this tutorial.

### The Complete JaasAcn.java Code

Now that you have seen the basic code required to authenticate the user, we can put it all together into the full class in [JaasAcn.java](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/JaasAcn.java), which includes relevant import statements and error handling:

import javax.security.auth.\*;

import javax.security.auth.callback.\*;

import javax.security.auth.login.\*;

import com.sun.security.auth.callback.TextCallbackHandler;

/\*\*

\* This JaasAcn application attempts to authenticate a user

\* and reports whether or not the authentication was successful.

\*/

public class **JaasAcn** {

public static void main(String[] args) {

// Obtain a LoginContext, needed for authentication. Tell

// it to use the LoginModule implementation specified by

// the entry named "JaasSample" in the JAAS login

// configuration file and to also use the specified

// CallbackHandler.

LoginContext lc = null;

try {

**lc = new LoginContext("JaasSample",**

**new TextCallbackHandler());**

} catch (LoginException le) {

System.err.println("Cannot create LoginContext. "

+ le.getMessage());

System.exit(-1);

} catch (SecurityException se) {

System.err.println("Cannot create LoginContext. "

+ se.getMessage());

System.exit(-1);

}

try {

// attempt authentication

**lc.login();**

} catch (LoginException le) {

System.err.println("Authentication failed: "

System.err.println(" " + le.getMessage());

System.exit(-1);

}

System.out.println("Authentication succeeded!");

}

}

## The Login Configuration

JAAS authentication is performed in a pluggable fashion, so applications can remain independent from underlying authentication technologies. A system administrator determines the authentication technologies, or LoginModules, to be used for each application and configures them in a login Configuration. The source of the configuration information (for example, a file or a database) is up to the current [javax.security.auth.login.Configuration](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) implementation. The default Configuration implementation from Sun Microsystems reads configuration information from configuration files, as described in [com.sun.security.auth.login.ConfigFile.html](https://docs.oracle.com/javase/7/docs/jre/api/security/jaas/spec/com/sun/security/auth/login/ConfigFile.html).

See [JAAS Login Configuration File](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/LoginConfigFile.html) for information as to what a login configuration file is, what it contains, and how to specify which login configuration file should be used.

### The Login Configuration File for This Tutorial

As noted, the login configuration file we use for this tutorial, [jaas.conf](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/jaas.conf), contains just one entry, which is

**JaasSample {**

**com.sun.security.auth.module.Krb5LoginModule required;**

**};**

This entry is named "JaasSample" and that is the name that our tutorial application, JaasAcn, uses to refer to this entry. The entry specifies that the LoginModule to be used to do the user authentication is the Krb5LoginModule in the com.sun.security.auth.module package and that this Krb5LoginModule is required to "succeed" in order for authentication to be considered successful. The Krb5LoginModule succeeds only if the name and password supplied by the user are successfully used to log the user into the Kerberos KDC.

For information about all the possible options that can be passed to Krb5LoginModule, see the [Krb5LoginModule documentation](https://docs.oracle.com/javase/7/docs/jre/api/security/jaas/spec/com/sun/security/auth/module/Krb5LoginModule.html).

## Running the Code

To execute our JAAS authentication tutorial code, all you have to do is

1. Place the [JaasAcn.java](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/JaasAcn.java) application source file and the [jaas.conf](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/jaas.conf) login configuration file into a directory.
2. Compile JaasAcn.java:
3. javac JaasAcn.java
4. Execute the JaasAcn application, specifying
   * by -Djava.security.krb5.realm=<your\_realm> that your Kerberos realm is the one specified. For example, if your realm is "KRBNT-OPERATIONS.EXAMPLE.COM" you'd put -Djava.security.krb5.realm=KRBNT-OPERATIONS.EXAMPLE.COM.
   * by -Djava.security.krb5.kdc=<your\_kdc> that your Kerberos KDC is the one specified. For example, if your KDC is "samplekdc.example.com" you'd put -Djava.security.krb5.kdc=samplekdc.example.com.
   * by -Djava.security.auth.login.config=jaas.conf that the login configuration file to be used is jaas.conf.

The full command is below. **Be sure to replace <your\_realm> with your Kerberos realm, and <your\_kdc> with your Kerberos KDC.**

java -Djava.security.krb5.realm=<your\_realm>

-Djava.security.krb5.kdc=<your\_kdc>

-Djava.security.auth.login.config=jaas.conf JaasAcn

Type all that on one line. Multiple lines are used here for legibility.

You will be prompted for your Kerberos user name and password, and the underlying Kerberos authentication mechanism specified in the login configuration file will log you into Kerberos. If your login is successful, you will see the following message:

Authentication succeeded!

If the login is not successful (for example, if you misspell your password), you will see

Authentication failed:

followed by a reason for the failure. For example, if you mistype your user name, you may see a message like the following (where the formatting is slightly modified here to increase legibility):

Authentication failed:

Kerberos Authentication Failed:

javax.security.auth.login.LoginException:

KrbException: Client not found in Kerberos database

For login troubleshooting suggestions, see [Troubleshooting](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/Troubleshooting.html).

After fixing any problems, re-run the program to try again.

## Running the Code with a Security Manager

When a Java program is run with a security manager installed, the program is not allowed to access resources or otherwise perform security-sensitive operations unless it is explicitly granted [permission](https://docs.oracle.com/javase/7/docs/technotes/guides/security/permissions.html) to do so by the security policy in effect. In Java platforms that are compatible with J2SE v 1.2 and later, the permission must be granted by an entry in a [policy file](https://docs.oracle.com/javase/7/docs/technotes/guides/security/PolicyFiles.html).

Most browsers install a security manager, so *applets* typically run under the scrutiny of a security manager. *Applications*, on the other hand, do not, since a security manager is not automatically installed when an application is running. Thus an application, like our JaasAcn application, by default has full access to resources.

**To run an application with a security manager**, simply invoke the interpreter with a -Djava.security.manager argument included on the command line.

If you try invoking JaasAcn with a security manager but without specifying any policy file, you will get the following (unless you have a default policy setup elsewhere that grants the required permissions or grants AllPermission):

% java -Djava.security.manager \

-Djava.security.krb5.realm=<your\_realm> \

-Djava.security.krb5.kdc=<your\_kdc> \

-Djava.security.auth.login.config=jaas.conf JaasAcn

Exception in thread "main" java.security.AccessControlException:

access denied (

javax.security.auth.AuthPermission createLoginContext.JaasSample)

As you can see, you get an AccessControlException, because we haven't created and used a policy file granting our code the permission that is required in order to be allowed to create a LoginContext.

Here are the complete steps required in order to be able to run our JaasAcn application with a security manager installed. You can skip the first two steps if you have already done them, as described in [Running the Code](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/AcnOnly.html#RunAc).

1. Place the [JaasAcn.java](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/JaasAcn.java) application source file and the [jaas.conf](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/jaas.conf) login configuration file into a directory.
2. Compile JaasAcn.java:
3. javac JaasAcn.java
4. Create a JAR file containing JaasAcn.class:
5. jar -cvf JaasAcn.jar JaasAcn.class

This command creates a JAR file, JaasAcn.jar, and places the JaasAcn.class file inside it.

1. Create a policy file granting the code in the JAR file the required permission.

The permission that is needed by code attempting to instantiate a LoginContext is a javax.security.auth.AuthPermission with target "createLoginContext.<entry name>". Here, <entry name> refers to the name of the login configuration file entry that the application references in its instantiation of LoginContext. The name used by our JaasAcn application's LoginContext instantiation is "JaasSample", as you can see in the code:

LoginContext lc =

new LoginContext("JaasSample",

new TextCallbackHandler());

Thus, the permission that needs to be granted to JaasAcn.jar is

permission javax.security.auth.AuthPermission

"createLoginContext.JaasSample";

Copy the policy file [jaasacn.policy](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/jaasacn.policy) to the same directory as that in which you stored JaasAcn.java, etc. This is a text file containing the following grant statement to grant JaasAcn.jar (in the current directory) the required permission:

grant codebase "file:./JaasAcn.jar" {

permission javax.security.auth.AuthPermission

"createLoginContext.JaasSample";

};

Note: Policy files and the structure of entries within them are described in [Default Policy Implementation and Policy File Syntax](https://docs.oracle.com/javase/7/docs/technotes/guides/security/PolicyFiles.html). Permissions are described [here](https://docs.oracle.com/javase/7/docs/technotes/guides/security/permissions.html).

1. Execute the JaasAcn application, specifying
   1. by an appropriate -classpath clause that classes should be searched for in the JaasAcn.jar JAR file,
   2. by -Djava.security.manager that a security manager should be installed,
   3. by -Djava.security.krb5.realm=<your\_realm> that your Kerberos realm is the one specified. For example, if your realm is "KRBNT-OPERATIONS.EXAMPLE.COM" you'd put -Djava.security.krb5.realm=KRBNT-OPERATIONS.EXAMPLE.COM.
   4. by -Djava.security.krb5.kdc=<your\_kdc> that your Kerberos KDC is the one specified. For example, if your KDC is "samplekdc.example.com" you'd put -Djava.security.krb5.kdc=samplekdc.example.com.
   5. by -Djava.security.policy=jaasacn.policy that the policy file to be used is jaasacn.policy, and
   6. by -Djava.security.auth.login.config=jaas.conf that the login configuration file to be used is jaas.conf.

The full command is below. **Be sure to replace <your\_realm> with your Kerberos realm, and <your\_kdc> with your Kerberos KDC.**

java -classpath JaasAcn.jar -Djava.security.manager

-Djava.security.krb5.realm=<your\_realm>

-Djava.security.krb5.kdc=<your\_kdc>

-Djava.security.policy=jaasacn.policy

-Djava.security.auth.login.config=jaas.conf JaasAcn

Type all that on one line. Multiple lines are used here for legibility. If the command is too long for your system, you may need to place it in a .bat file (for Windows) or a .sh file (for UNIX) and then run that file to execute the command.

Since the specified policy file contains an entry granting the code the required permission, JaasAcn will be allowed to instantiate a LoginContext and continue execution. You will be prompted for your Kerberos user name and password, and the underlying Kerberos authentication mechanism specified in the login configuration file will log you into Kerberos. If your login is successful, you will see the message "Authentication succeeded!" and if not, you will see "Authentication failed:" followed by a reason for the failure.

For login troubleshooting suggestions, see [Troubleshooting](https://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/Troubleshooting.html).

# [MIT Kerberos Documentation](http://web.mit.edu/kerberos/www/krb5-latest/doc/index.html)

[CONTENTS](http://web.mit.edu/kerberos/www/krb5-latest/doc/index.html) | [PREVIOUS](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/install_appl_srv.html) | [NEXT](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html) | [INDEX](http://web.mit.edu/kerberos/www/krb5-latest/doc/genindex.html) | [SEARCH](http://web.mit.edu/kerberos/www/krb5-latest/doc/search.html) | [FEEDBACK](mailto:krb5-bugs@mit.edu?subject=Documentation__Configuration%20Files)

## Configuration Files

Kerberos uses configuration files to allow administrators to specify settings on a per-machine basis. [krb5.conf](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#krb5-conf-5) applies to all applications using the Kerboros library, on clients and servers. For KDC-specific applications, additional settings can be specified in [kdc.conf](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/kdc_conf.html" \l "kdc-conf-5); the two files are merged into a configuration profile used by applications accessing the KDC database directly. [kadm5.acl](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/kadm5_acl.html#kadm5-acl-5) is also only used on the KDC, it controls permissions for modifying the KDC database.

## Contents

* [krb5.conf](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html)
* [kdc.conf](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/kdc_conf.html)
* [kadm5.acl](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/kadm5_acl.html)

# **krb5.conf**

The krb5.conf file contains Kerberos configuration information, including the locations of KDCs and admin servers for the Kerberos realms of interest, defaults for the current realm and for Kerberos applications, and mappings of hostnames onto Kerberos realms. Normally, you should install your krb5.conf file in the directory /etc. You can override the default location by setting the environment variable **KRB5\_CONFIG**. Multiple colon-separated filenames may be specified in **KRB5\_CONFIG**; all files which are present will be read. Starting in release 1.14, directory names can also be specified in **KRB5\_CONFIG**; all files within the directory whose names consist solely of alphanumeric characters, dashes, or underscores will be read.

## Structure

The krb5.conf file is set up in the style of a Windows INI file. Lines beginning with ‘#’ or ‘;’ (possibly after initial whitespace) are ignored as comments. Sections are headed by the section name, in square brackets. Each section may contain zero or more relations, of the form:

foo = bar

or:

fubar = {

foo = bar

baz = quux

}

Placing a ‘\*’ after the closing bracket of a section name indicates that the section is final, meaning that if the same section appears within a later file specified in **KRB5\_CONFIG**, it will be ignored. A subsection can be marked as final by placing a ‘\*’ after either the tag name or the closing brace.

The krb5.conf file can include other files using either of the following directives at the beginning of a line:

include FILENAME

includedir DIRNAME

FILENAME or DIRNAME should be an absolute path. The named file or directory must exist and be readable. Including a directory includes all files within the directory whose names consist solely of alphanumeric characters, dashes, or underscores. Starting in release 1.15, files with names ending in “.conf” are also included, unless the name begins with “.”. Included profile files are syntactically independent of their parents, so each included file must begin with a section header. Starting in release 1.17, files are read in alphanumeric order; in previous releases, they may be read in any order.

The krb5.conf file can specify that configuration should be obtained from a loadable module, rather than the file itself, using the following directive at the beginning of a line before any section headers:

module MODULEPATH:RESIDUAL

MODULEPATH may be relative to the library path of the krb5 installation, or it may be an absolute path. RESIDUAL is provided to the module at initialization time. If krb5.conf uses a module directive, [kdc.conf](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/kdc_conf.html" \l "kdc-conf-5) should also use one if it exists.

## Sections

The krb5.conf file may contain the following sections:

|  |  |
| --- | --- |
| [[libdefaults]](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#libdefaults) | Settings used by the Kerberos V5 library |
| [[realms]](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#realms) | Realm-specific contact information and settings |
| [[domain\_realm]](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#domain-realm) | Maps server hostnames to Kerberos realms |
| [[capaths]](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#capaths) | Authentication paths for non-hierarchical cross-realm |
| [[appdefaults]](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#appdefaults) | Settings used by some Kerberos V5 applications |
| [[plugins]](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#plugins) | Controls plugin module registration |

Additionally, krb5.conf may include any of the relations described in [kdc.conf](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/kdc_conf.html" \l "kdc-conf-5), but it is not a recommended practice.

### **[libdefaults]**

The libdefaults section may contain any of the following relations:

**allow\_weak\_crypto**

If this flag is set to false, then weak encryption types (as noted in [Encryption types](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/kdc_conf.html#encryption-types) in [kdc.conf](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/kdc_conf.html" \l "kdc-conf-5)) will be filtered out of the lists **default\_tgs\_enctypes**, **default\_tkt\_enctypes**, and **permitted\_enctypes**. The default value for this tag is false.

**canonicalize**

If this flag is set to true, initial ticket requests to the KDC will request canonicalization of the client principal name, and answers with different client principals than the requested principal will be accepted. The default value is false.

**ccache\_type**

This parameter determines the format of credential cache types created by [kinit](http://web.mit.edu/kerberos/www/krb5-latest/doc/user/user_commands/kinit.html#kinit-1) or other programs. The default value is 4, which represents the most current format. Smaller values can be used for compatibility with very old implementations of Kerberos which interact with credential caches on the same host.

**clockskew**

Sets the maximum allowable amount of clockskew in seconds that the library will tolerate before assuming that a Kerberos message is invalid. The default value is 300 seconds, or five minutes.

The clockskew setting is also used when evaluating ticket start and expiration times. For example, tickets that have reached their expiration time can still be used (and renewed if they are renewable tickets) if they have been expired for a shorter duration than the **clockskew** setting.

**default\_ccache\_name**

This relation specifies the name of the default credential cache. The default is [DEFCCNAME](http://web.mit.edu/kerberos/www/krb5-latest/doc/mitK5defaults.html#paths). This relation is subject to parameter expansion (see below). New in release 1.11.

**default\_client\_keytab\_name**

This relation specifies the name of the default keytab for obtaining client credentials. The default is [DEFCKTNAME](http://web.mit.edu/kerberos/www/krb5-latest/doc/mitK5defaults.html#paths). This relation is subject to parameter expansion (see below). New in release 1.11.

**default\_keytab\_name**

This relation specifies the default keytab name to be used by application servers such as sshd. The default is [DEFKTNAME](http://web.mit.edu/kerberos/www/krb5-latest/doc/mitK5defaults.html#paths). This relation is subject to parameter expansion (see below).

**default\_rcache\_name**

This relation specifies the name of the default replay cache. The default is dfl:. This relation is subject to parameter expansion (see below). New in release 1.18.

**default\_realm**

Identifies the default Kerberos realm for the client. Set its value to your Kerberos realm. If this value is not set, then a realm must be specified with every Kerberos principal when invoking programs such as [kinit](http://web.mit.edu/kerberos/www/krb5-latest/doc/user/user_commands/kinit.html#kinit-1).

**default\_tgs\_enctypes**

Identifies the supported list of session key encryption types that the client should request when making a TGS-REQ, in order of preference from highest to lowest. The list may be delimited with commas or whitespace. See [Encryption types](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/kdc_conf.html#encryption-types) in [kdc.conf](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/kdc_conf.html" \l "kdc-conf-5) for a list of the accepted values for this tag. Starting in release 1.18, the default value is the value of **permitted\_enctypes**. For previous releases or if **permitted\_enctypes** is not set, the default value is aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 aes256-cts-hmac-sha384-192 aes128-cts-hmac-sha256-128 des3-cbc-sha1 arcfour-hmac-md5 camellia256-cts-cmac camellia128-cts-cmac.

Do not set this unless required for specific backward compatibility purposes; stale values of this setting can prevent clients from taking advantage of new stronger enctypes when the libraries are upgraded.

**default\_tkt\_enctypes**

Identifies the supported list of session key encryption types that the client should request when making an AS-REQ, in order of preference from highest to lowest. The format is the same as for default\_tgs\_enctypes. Starting in release 1.18, the default value is the value of **permitted\_enctypes**. For previous releases or if **permitted\_enctypes** is not set, the default value is aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 aes256-cts-hmac-sha384-192 aes128-cts-hmac-sha256-128 des3-cbc-sha1 arcfour-hmac-md5 camellia256-cts-cmac camellia128-cts-cmac.

Do not set this unless required for specific backward compatibility purposes; stale values of this setting can prevent clients from taking advantage of new stronger enctypes when the libraries are upgraded.

**dns\_canonicalize\_hostname**

Indicate whether name lookups will be used to canonicalize hostnames for use in service principal names. Setting this flag to false can improve security by reducing reliance on DNS, but means that short hostnames will not be canonicalized to fully-qualified hostnames. If this option is set to fallback (new in release 1.18), DNS canonicalization will only be performed the server hostname is not found with the original name when requesting credentials. The default value is true.

**dns\_lookup\_kdc**

Indicate whether DNS SRV records should be used to locate the KDCs and other servers for a realm, if they are not listed in the krb5.conf information for the realm. (Note that the admin\_server entry must be in the krb5.conf realm information in order to contact kadmind, because the DNS implementation for kadmin is incomplete.)

Enabling this option does open up a type of denial-of-service attack, if someone spoofs the DNS records and redirects you to another server. However, it’s no worse than a denial of service, because that fake KDC will be unable to decode anything you send it (besides the initial ticket request, which has no encrypted data), and anything the fake KDC sends will not be trusted without verification using some secret that it won’t know.

**dns\_uri\_lookup**

Indicate whether DNS URI records should be used to locate the KDCs and other servers for a realm, if they are not listed in the krb5.conf information for the realm. SRV records are used as a fallback if no URI records were found. The default value is true. New in release 1.15.

**enforce\_ok\_as\_delegate**

If this flag to true, GSSAPI credential delegation will be disabled when the ok-as-delegate flag is not set in the service ticket. If this flag is false, the ok-as-delegate ticket flag is only enforced when an application specifically requests enforcement. The default value is false.

**err\_fmt**

This relation allows for custom error message formatting. If a value is set, error messages will be formatted by substituting a normal error message for %M and an error code for %C in the value.

**extra\_addresses**

This allows a computer to use multiple local addresses, in order to allow Kerberos to work in a network that uses NATs while still using address-restricted tickets. The addresses should be in a comma-separated list. This option has no effect if **noaddresses** is true.

**forwardable**

If this flag is true, initial tickets will be forwardable by default, if allowed by the KDC. The default value is false.

**ignore\_acceptor\_hostname**

When accepting GSSAPI or krb5 security contexts for host-based service principals, ignore any hostname passed by the calling application, and allow clients to authenticate to any service principal in the keytab matching the service name and realm name (if given). This option can improve the administrative flexibility of server applications on multihomed hosts, but could compromise the security of virtual hosting environments. The default value is false. New in release 1.10.

**k5login\_authoritative**

If this flag is true, principals must be listed in a local user’s k5login file to be granted login access, if a [.k5login](http://web.mit.edu/kerberos/www/krb5-latest/doc/user/user_config/k5login.html#k5login-5) file exists. If this flag is false, a principal may still be granted login access through other mechanisms even if a k5login file exists but does not list the principal. The default value is true.

**k5login\_directory**

If set, the library will look for a local user’s k5login file within the named directory, with a filename corresponding to the local username. If not set, the library will look for k5login files in the user’s home directory, with the filename .k5login. For security reasons, .k5login files must be owned by the local user or by root.

**kcm\_mach\_service**

On macOS only, determines the name of the bootstrap service used to contact the KCM daemon for the KCM credential cache type. If the value is -, Mach RPC will not be used to contact the KCM daemon. The default value is org.h5l.kcm.

**kcm\_socket**

Determines the path to the Unix domain socket used to access the KCM daemon for the KCM credential cache type. If the value is -, Unix domain sockets will not be used to contact the KCM daemon. The default value is /var/run/.heim\_org.h5l.kcm-socket.

**kdc\_default\_options**

Default KDC options (Xored for multiple values) when requesting initial tickets. By default it is set to 0x00000010 (KDC\_OPT\_RENEWABLE\_OK).

**kdc\_timesync**

Accepted values for this relation are 1 or 0. If it is nonzero, client machines will compute the difference between their time and the time returned by the KDC in the timestamps in the tickets and use this value to correct for an inaccurate system clock when requesting service tickets or authenticating to services. This corrective factor is only used by the Kerberos library; it is not used to change the system clock. The default value is 1.

**noaddresses**

If this flag is true, requests for initial tickets will not be made with address restrictions set, allowing the tickets to be used across NATs. The default value is true.

**permitted\_enctypes**

Identifies the encryption types that servers will permit for session keys and for ticket and authenticator encryption, ordered by preference from highest to lowest. Starting in release 1.18, this tag also acts as the default value for **default\_tgs\_enctypes** and **default\_tkt\_enctypes**. The default value for this tag is aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 aes256-cts-hmac-sha384-192 aes128-cts-hmac-sha256-128 des3-cbc-sha1 arcfour-hmac-md5 camellia256-cts-cmac camellia128-cts-cmac.

**plugin\_base\_dir**

If set, determines the base directory where krb5 plugins are located. The default value is the krb5/plugins subdirectory of the krb5 library directory. This relation is subject to parameter expansion (see below) in release 1.17 and later.

**preferred\_preauth\_types**

This allows you to set the preferred preauthentication types which the client will attempt before others which may be advertised by a KDC. The default value for this setting is “17, 16, 15, 14”, which forces libkrb5 to attempt to use PKINIT if it is supported.

**proxiable**

If this flag is true, initial tickets will be proxiable by default, if allowed by the KDC. The default value is false.

**qualify\_shortname**

If this string is set, it determines the domain suffix for single-component hostnames when DNS canonicalization is not used (either because **dns\_canonicalize\_hostname** is false or because forward canonicalization failed). The default value is the first search domain of the system’s DNS configuration. To disable qualification of shortnames, set this relation to the empty string with qualify\_shortname = "". (New in release 1.18.)

**rdns**

If this flag is true, reverse name lookup will be used in addition to forward name lookup to canonicalizing hostnames for use in service principal names. If **dns\_canonicalize\_hostname** is set to false, this flag has no effect. The default value is true.

**realm\_try\_domains**

Indicate whether a host’s domain components should be used to determine the Kerberos realm of the host. The value of this variable is an integer: -1 means not to search, 0 means to try the host’s domain itself, 1 means to also try the domain’s immediate parent, and so forth. The library’s usual mechanism for locating Kerberos realms is used to determine whether a domain is a valid realm, which may involve consulting DNS if **dns\_lookup\_kdc** is set. The default is not to search domain components.

**renew\_lifetime**

([Time duration](http://web.mit.edu/kerberos/www/krb5-latest/doc/basic/date_format.html#duration) string.) Sets the default renewable lifetime for initial ticket requests. The default value is 0.

**spake\_preauth\_groups**

A whitespace or comma-separated list of words which specifies the groups allowed for SPAKE preauthentication. The possible values are:

|  |  |
| --- | --- |
| edwards25519 | Edwards25519 curve ([**RFC 7748**](https://tools.ietf.org/html/rfc7748.html)) |
| P-256 | NIST P-256 curve ([**RFC 5480**](https://tools.ietf.org/html/rfc5480.html)) |
| P-384 | NIST P-384 curve ([**RFC 5480**](https://tools.ietf.org/html/rfc5480.html)) |
| P-521 | NIST P-521 curve ([**RFC 5480**](https://tools.ietf.org/html/rfc5480.html)) |

The default value for the client is edwards25519. The default value for the KDC is empty. New in release 1.17.

**ticket\_lifetime**

([Time duration](http://web.mit.edu/kerberos/www/krb5-latest/doc/basic/date_format.html#duration) string.) Sets the default lifetime for initial ticket requests. The default value is 1 day.

**udp\_preference\_limit**

When sending a message to the KDC, the library will try using TCP before UDP if the size of the message is above **udp\_preference\_limit**. If the message is smaller than **udp\_preference\_limit**, then UDP will be tried before TCP. Regardless of the size, both protocols will be tried if the first attempt fails.

**verify\_ap\_req\_nofail**

If this flag is true, then an attempt to verify initial credentials will fail if the client machine does not have a keytab. The default value is false.

**client\_aware\_channel\_bindings**

If this flag is true, then all application protocol authentication requests will be flagged to indicate that the application supports channel bindings when operating over a secure channel. The default value is false.

### **[realms]**

Each tag in the [realms] section of the file is the name of a Kerberos realm. The value of the tag is a subsection with relations that define the properties of that particular realm. For each realm, the following tags may be specified in the realm’s subsection:

**admin\_server**

Identifies the host where the administration server is running. Typically, this is the primary Kerberos server. This tag must be given a value in order to communicate with the [kadmind](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/admin_commands/kadmind.html" \l "kadmind-8) server for the realm.

**auth\_to\_local**

This tag allows you to set a general rule for mapping principal names to local user names. It will be used if there is not an explicit mapping for the principal name that is being translated. The possible values are:

**RULE:**exp

The local name will be formulated from exp.

The format for exp is **[**n**:**string**](**regexp**)s/**pattern**/**replacement**/g**. The integer n indicates how many components the target principal should have. If this matches, then a string will be formed from string, substituting the realm of the principal for $0 and the n’th component of the principal for $n (e.g., if the principal was johndoe/admin then [2:$2$1foo] would result in the string adminjohndoefoo). If this string matches regexp, then the s//[g] substitution command will be run over the string. The optional **g** will cause the substitution to be global over the string, instead of replacing only the first match in the string.

**DEFAULT**

The principal name will be used as the local user name. If the principal has more than one component or is not in the default realm, this rule is not applicable and the conversion will fail.

For example:

[realms]

ATHENA.MIT.EDU = {

auth\_to\_local = RULE:[2:$1](johndoe)s/^.\*$/guest/

auth\_to\_local = RULE:[2:$1;$2](^.\*;admin$)s/;admin$//

auth\_to\_local = RULE:[2:$2](^.\*;root)s/^.\*$/root/

auth\_to\_local = DEFAULT

}

would result in any principal without root or admin as the second component to be translated with the default rule. A principal with a second component of admin will become its first component. root will be used as the local name for any principal with a second component of root. The exception to these two rules are any principals johndoe/\*, which will always get the local name guest.

**auth\_to\_local\_names**

This subsection allows you to set explicit mappings from principal names to local user names. The tag is the mapping name, and the value is the corresponding local user name.

**default\_domain**

This tag specifies the domain used to expand hostnames when translating Kerberos 4 service principals to Kerberos 5 principals (for example, when converting rcmd.hostname to host/hostname.domain).

**disable\_encrypted\_timestamp**

If this flag is true, the client will not perform encrypted timestamp preauthentication if requested by the KDC. Setting this flag can help to prevent dictionary attacks by active attackers, if the realm’s KDCs support SPAKE preauthentication or if initial authentication always uses another mechanism or always uses FAST. This flag persists across client referrals during initial authentication. This flag does not prevent the KDC from offering encrypted timestamp. New in release 1.17.

**http\_anchors**

When KDCs and kpasswd servers are accessed through HTTPS proxies, this tag can be used to specify the location of the CA certificate which should be trusted to issue the certificate for a proxy server. If left unspecified, the system-wide default set of CA certificates is used.

The syntax for values is similar to that of values for the **pkinit\_anchors** tag:

**FILE:** filename

filename is assumed to be the name of an OpenSSL-style ca-bundle file.

**DIR:** dirname

dirname is assumed to be an directory which contains CA certificates. All files in the directory will be examined; if they contain certificates (in PEM format), they will be used.

**ENV:** envvar

envvar specifies the name of an environment variable which has been set to a value conforming to one of the previous values. For example, ENV:X509\_PROXY\_CA, where environment variable X509\_PROXY\_CA has been set to FILE:/tmp/my\_proxy.pem.

**kdc**

The name or address of a host running a KDC for that realm. An optional port number, separated from the hostname by a colon, may be included. If the name or address contains colons (for example, if it is an IPv6 address), enclose it in square brackets to distinguish the colon from a port separator. For your computer to be able to communicate with the KDC for each realm, this tag must be given a value in each realm subsection in the configuration file, or there must be DNS SRV records specifying the KDCs.

**kpasswd\_server**

Points to the server where all the password changes are performed. If there is no such entry, DNS will be queried (unless forbidden by **dns\_lookup\_kdc**). Finally, port 464 on the **admin\_server** host will be tried.

**master\_kdc**

The name for **primary\_kdc** prior to release 1.19. Its value is used as a fallback if **primary\_kdc** is not specified.

**primary\_kdc**

Identifies the primary KDC(s). Currently, this tag is used in only one case: If an attempt to get credentials fails because of an invalid password, the client software will attempt to contact the primary KDC, in case the user’s password has just been changed, and the updated database has not been propagated to the replica servers yet. New in release 1.19.

**v4\_instance\_convert**

This subsection allows the administrator to configure exceptions to the **default\_domain** mapping rule. It contains V4 instances (the tag name) which should be translated to some specific hostname (the tag value) as the second component in a Kerberos V5 principal name.

**v4\_realm**

This relation is used by the krb524 library routines when converting a V5 principal name to a V4 principal name. It is used when the V4 realm name and the V5 realm name are not the same, but still share the same principal names and passwords. The tag value is the Kerberos V4 realm name.

### **[domain\_realm]**

The [domain\_realm] section provides a translation from a domain name or hostname to a Kerberos realm name. The tag name can be a host name or domain name, where domain names are indicated by a prefix of a period (.). The value of the relation is the Kerberos realm name for that particular host or domain. A host name relation implicitly provides the corresponding domain name relation, unless an explicit domain name relation is provided. The Kerberos realm may be identified either in the [realms](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#realms) section or using DNS SRV records. Host names and domain names should be in lower case. For example:

[domain\_realm]

crash.mit.edu = TEST.ATHENA.MIT.EDU

.dev.mit.edu = TEST.ATHENA.MIT.EDU

mit.edu = ATHENA.MIT.EDU

maps the host with the name crash.mit.edu into the TEST.ATHENA.MIT.EDU realm. The second entry maps all hosts under the domain dev.mit.edu into the TEST.ATHENA.MIT.EDU realm, but not the host with the name dev.mit.edu. That host is matched by the third entry, which maps the host mit.edu and all hosts under the domain mit.edu that do not match a preceding rule into the realm ATHENA.MIT.EDU.

If no translation entry applies to a hostname used for a service principal for a service ticket request, the library will try to get a referral to the appropriate realm from the client realm’s KDC. If that does not succeed, the host’s realm is considered to be the hostname’s domain portion converted to uppercase, unless the **realm\_try\_domains** setting in [libdefaults] causes a different parent domain to be used.

### **[capaths]**

In order to perform direct (non-hierarchical) cross-realm authentication, configuration is needed to determine the authentication paths between realms.

A client will use this section to find the authentication path between its realm and the realm of the server. The server will use this section to verify the authentication path used by the client, by checking the transited field of the received ticket.

There is a tag for each participating client realm, and each tag has subtags for each of the server realms. The value of the subtags is an intermediate realm which may participate in the cross-realm authentication. The subtags may be repeated if there is more then one intermediate realm. A value of “.” means that the two realms share keys directly, and no intermediate realms should be allowed to participate.

Only those entries which will be needed on the client or the server need to be present. A client needs a tag for its local realm with subtags for all the realms of servers it will need to authenticate to. A server needs a tag for each realm of the clients it will serve, with a subtag of the server realm.

For example, ANL.GOV, PNL.GOV, and NERSC.GOV all wish to use the ES.NET realm as an intermediate realm. ANL has a sub realm of TEST.ANL.GOV which will authenticate with NERSC.GOV but not PNL.GOV. The [capaths] section for ANL.GOV systems would look like this:

[capaths]

ANL.GOV = {

TEST.ANL.GOV = .

PNL.GOV = ES.NET

NERSC.GOV = ES.NET

ES.NET = .

}

TEST.ANL.GOV = {

ANL.GOV = .

}

PNL.GOV = {

ANL.GOV = ES.NET

}

NERSC.GOV = {

ANL.GOV = ES.NET

}

ES.NET = {

ANL.GOV = .

}

The [capaths] section of the configuration file used on NERSC.GOV systems would look like this:

[capaths]

NERSC.GOV = {

ANL.GOV = ES.NET

TEST.ANL.GOV = ES.NET

TEST.ANL.GOV = ANL.GOV

PNL.GOV = ES.NET

ES.NET = .

}

ANL.GOV = {

NERSC.GOV = ES.NET

}

PNL.GOV = {

NERSC.GOV = ES.NET

}

ES.NET = {

NERSC.GOV = .

}

TEST.ANL.GOV = {

NERSC.GOV = ANL.GOV

NERSC.GOV = ES.NET

}

When a subtag is used more than once within a tag, clients will use the order of values to determine the path. The order of values is not important to servers.

### **[appdefaults]**

Each tag in the [appdefaults] section names a Kerberos V5 application or an option that is used by some Kerberos V5 application[s]. The value of the tag defines the default behaviors for that application.

For example:

[appdefaults]

telnet = {

ATHENA.MIT.EDU = {

option1 = false

}

}

telnet = {

option1 = true

option2 = true

}

ATHENA.MIT.EDU = {

option2 = false

}

option2 = true

The above four ways of specifying the value of an option are shown in order of decreasing precedence. In this example, if telnet is running in the realm EXAMPLE.COM, it should, by default, have option1 and option2 set to true. However, a telnet program in the realm ATHENA.MIT.EDU should have option1 set to false and option2 set to true. Any other programs in ATHENA.MIT.EDU should have option2 set to false by default. Any programs running in other realms should have option2 set to true.

The list of specifiable options for each application may be found in that application’s man pages. The application defaults specified here are overridden by those specified in the [realms](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#realms) section.

### **[plugins]**

* [pwqual](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#pwqual) interface
* [kadm5\_hook](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#kadm5-hook) interface
* [clpreauth](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#clpreauth) and [kdcpreauth](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html" \l "kdcpreauth) interfaces

Tags in the [plugins] section can be used to register dynamic plugin modules and to turn modules on and off. Not every krb5 pluggable interface uses the [plugins] section; the ones that do are documented here.

New in release 1.9.

Each pluggable interface corresponds to a subsection of [plugins]. All subsections support the same tags:

**disable**

This tag may have multiple values. If there are values for this tag, then the named modules will be disabled for the pluggable interface.

**enable\_only**

This tag may have multiple values. If there are values for this tag, then only the named modules will be enabled for the pluggable interface.

**module**

This tag may have multiple values. Each value is a string of the form modulename:pathname, which causes the shared object located at pathname to be registered as a dynamic module named modulename for the pluggable interface. If pathname is not an absolute path, it will be treated as relative to the **plugin\_base\_dir** value from [[libdefaults]](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/krb5_conf.html#libdefaults).

For pluggable interfaces where module order matters, modules registered with a **module** tag normally come first, in the order they are registered, followed by built-in modules in the order they are documented below. If **enable\_only** tags are used, then the order of those tags overrides the normal module order.

The following subsections are currently supported within the [plugins] section:

#### **ccselect interface**

The ccselect subsection controls modules for credential cache selection within a cache collection. In addition to any registered dynamic modules, the following built-in modules exist (and may be disabled with the disable tag):

**k5identity**

Uses a .k5identity file in the user’s home directory to select a client principal

**realm**

Uses the service realm to guess an appropriate cache from the collection

**hostname**

If the service principal is host-based, uses the service hostname to guess an appropriate cache from the collection

#### **pwqual interface**

The pwqual subsection controls modules for the password quality interface, which is used to reject weak passwords when passwords are changed. The following built-in modules exist for this interface:

**dict**

Checks against the realm dictionary file

**empty**

Rejects empty passwords

**hesiod**

Checks against user information stored in Hesiod (only if Kerberos was built with Hesiod support)

**princ**

Checks against components of the principal name

#### **kadm5\_hook interface**

The kadm5\_hook interface provides plugins with information on principal creation, modification, password changes and deletion. This interface can be used to write a plugin to synchronize MIT Kerberos with another database such as Active Directory. No plugins are built in for this interface.

#### **kadm5\_auth interface**

The kadm5\_auth section (introduced in release 1.16) controls modules for the kadmin authorization interface, which determines whether a client principal is allowed to perform a kadmin operation. The following built-in modules exist for this interface:

**acl**

This module reads the [kadm5.acl](http://web.mit.edu/kerberos/www/krb5-latest/doc/admin/conf_files/kadm5_acl.html#kadm5-acl-5) file, and authorizes operations which are allowed according to the rules in the file.

**self**

This module authorizes self-service operations including password changes, creation of new random keys, fetching the client’s principal record or string attributes, and fetching the policy record associated with the client principal.

#### **clpreauth and kdcpreauth interfaces**

The clpreauth and kdcpreauth interfaces allow plugin modules to provide client and KDC preauthentication mechanisms. The following built-in modules exist for these interfaces:

**pkinit**

This module implements the PKINIT preauthentication mechanism.

**encrypted\_challenge**

This module implements the encrypted challenge FAST factor.

**encrypted\_timestamp**

This module implements the encrypted timestamp mechanism.

#### **hostrealm interface**

The hostrealm section (introduced in release 1.12) controls modules for the host-to-realm interface, which affects the local mapping of hostnames to realm names and the choice of default realm. The following built-in modules exist for this interface:

**profile**

This module consults the [domain\_realm] section of the profile for authoritative host-to-realm mappings, and the **default\_realm** variable for the default realm.

**dns**

This module looks for DNS records for fallback host-to-realm mappings and the default realm. It only operates if the **dns\_lookup\_realm** variable is set to true.

**domain**

This module applies heuristics for fallback host-to-realm mappings. It implements the **realm\_try\_domains** variable, and uses the uppercased parent domain of the hostname if that does not produce a result.

#### **localauth interface**

The localauth section (introduced in release 1.12) controls modules for the local authorization interface, which affects the relationship between Kerberos principals and local system accounts. The following built-in modules exist for this interface:

**default**

This module implements the **DEFAULT** type for **auth\_to\_local** values.

**rule**

This module implements the **RULE** type for **auth\_to\_local** values.

**names**

This module looks for an **auth\_to\_local\_names** mapping for the principal name.

**auth\_to\_local**

This module processes **auth\_to\_local** values in the default realm’s section, and applies the default method if no **auth\_to\_local** values exist.

**k5login**

This module authorizes a principal to a local account according to the account’s [.k5login](http://web.mit.edu/kerberos/www/krb5-latest/doc/user/user_config/k5login.html#k5login-5) file.

**an2ln**

This module authorizes a principal to a local account if the principal name maps to the local account name.

#### **certauth interface**

The certauth section (introduced in release 1.16) controls modules for the certificate authorization interface, which determines whether a certificate is allowed to preauthenticate a user via PKINIT. The following built-in modules exist for this interface:

**pkinit\_san**

This module authorizes the certificate if it contains a PKINIT Subject Alternative Name for the requested client principal, or a Microsoft UPN SAN matching the principal if **pkinit\_allow\_upn** is set to true for the realm.

**pkinit\_eku**

This module rejects the certificate if it does not contain an Extended Key Usage attribute consistent with the **pkinit\_eku\_checking** value for the realm.

**dbmatch**

This module authorizes or rejects the certificate according to whether it matches the **pkinit\_cert\_match** string attribute on the client principal, if that attribute is present.

## PKINIT options

**Note**

The following are PKINIT-specific options. These values may be specified in [libdefaults] as global defaults, or within a realm-specific subsection of [libdefaults], or may be specified as realm-specific values in the [realms] section. A realm-specific value overrides, not adds to, a generic [libdefaults] specification. The search order is:

1. realm-specific subsection of [libdefaults]:
2. [libdefaults]
3. EXAMPLE.COM = {
4. pkinit\_anchors = FILE:/usr/local/example.com.crt
5. }
6. realm-specific value in the [realms] section:
7. [realms]
8. OTHERREALM.ORG = {
9. pkinit\_anchors = FILE:/usr/local/otherrealm.org.crt
10. }
11. generic value in the [libdefaults] section:
12. [libdefaults]
13. pkinit\_anchors = DIR:/usr/local/generic\_trusted\_cas/

### **Specifying PKINIT identity information**

The syntax for specifying Public Key identity, trust, and revocation information for PKINIT is as follows:

**FILE:**filename[**,**keyfilename]

This option has context-specific behavior.

In **pkinit\_identity** or **pkinit\_identities**, filename specifies the name of a PEM-format file containing the user’s certificate. If keyfilename is not specified, the user’s private key is expected to be in filename as well. Otherwise, keyfilename is the name of the file containing the private key.

In **pkinit\_anchors** or **pkinit\_pool**, filename is assumed to be the name of an OpenSSL-style ca-bundle file.

**DIR:**dirname

This option has context-specific behavior.

In **pkinit\_identity** or **pkinit\_identities**, dirname specifies a directory with files named \*.crt and \*.key where the first part of the file name is the same for matching pairs of certificate and private key files. When a file with a name ending with .crt is found, a matching file ending with .key is assumed to contain the private key. If no such file is found, then the certificate in the .crt is not used.

In **pkinit\_anchors** or **pkinit\_pool**, dirname is assumed to be an OpenSSL-style hashed CA directory where each CA cert is stored in a file named hash-of-ca-cert.#. This infrastructure is encouraged, but all files in the directory will be examined and if they contain certificates (in PEM format), they will be used.

In **pkinit\_revoke**, dirname is assumed to be an OpenSSL-style hashed CA directory where each revocation list is stored in a file named hash-of-ca-cert.r#. This infrastructure is encouraged, but all files in the directory will be examined and if they contain a revocation list (in PEM format), they will be used.

**PKCS12:**filename

filename is the name of a PKCS #12 format file, containing the user’s certificate and private key.

**PKCS11:**[**module\_name=**]modname[**:slotid=**slot-id][**:token=**token-label][**:certid=**cert-id][**:certlabel=**cert-label]

All keyword/values are optional. modname specifies the location of a library implementing PKCS #11. If a value is encountered with no keyword, it is assumed to be the modname. If no module-name is specified, the default is opensc-pkcs11.so. slotid= and/or token= may be specified to force the use of a particular smard card reader or token if there is more than one available. certid= and/or certlabel= may be specified to force the selection of a particular certificate on the device. See the **pkinit\_cert\_match** configuration option for more ways to select a particular certificate to use for PKINIT.

**ENV:**envvar

envvar specifies the name of an environment variable which has been set to a value conforming to one of the previous values. For example, ENV:X509\_PROXY, where environment variable X509\_PROXY has been set to FILE:/tmp/my\_proxy.pem.

### **PKINIT krb5.conf options**

**pkinit\_anchors**

Specifies the location of trusted anchor (root) certificates which the client trusts to sign KDC certificates. This option may be specified multiple times. These values from the config file are not used if the user specifies X509\_anchors on the command line.

**pkinit\_cert\_match**

Specifies matching rules that the client certificate must match before it is used to attempt PKINIT authentication. If a user has multiple certificates available (on a smart card, or via other media), there must be exactly one certificate chosen before attempting PKINIT authentication. This option may be specified multiple times. All the available certificates are checked against each rule in order until there is a match of exactly one certificate.

The Subject and Issuer comparison strings are the [**RFC 2253**](https://tools.ietf.org/html/rfc2253.html) string representations from the certificate Subject DN and Issuer DN values.

The syntax of the matching rules is:

[relation-operator]component-rule …

where:

relation-operator

can be either &&, meaning all component rules must match, or ||, meaning only one component rule must match. The default is &&.

component-rule

can be one of the following. Note that there is no punctuation or whitespace between component rules.

**<SUBJECT>**regular-expression

**<ISSUER>**regular-expression

**<SAN>**regular-expression

**<EKU>**extended-key-usage-list

**<KU>**key-usage-list

extended-key-usage-list is a comma-separated list of required Extended Key Usage values. All values in the list must be present in the certificate. Extended Key Usage values can be:

* pkinit
* msScLogin
* clientAuth
* emailProtection

key-usage-list is a comma-separated list of required Key Usage values. All values in the list must be present in the certificate. Key Usage values can be:

* digitalSignature
* keyEncipherment

Examples:

pkinit\_cert\_match = ||<SUBJECT>.\*DoE.\*<SAN>.\***@EXAMPLE**.COM

pkinit\_cert\_match = &&<EKU>msScLogin,clientAuth<ISSUER>.\*DoE.\*

pkinit\_cert\_match = <EKU>msScLogin,clientAuth<KU>digitalSignature

**pkinit\_eku\_checking**

This option specifies what Extended Key Usage value the KDC certificate presented to the client must contain. (Note that if the KDC certificate has the pkinit SubjectAlternativeName encoded as the Kerberos TGS name, EKU checking is not necessary since the issuing CA has certified this as a KDC certificate.) The values recognized in the krb5.conf file are:

**kpKDC**

This is the default value and specifies that the KDC must have the id-pkinit-KPKdc EKU as defined in [**RFC 4556**](https://tools.ietf.org/html/rfc4556.html).

**kpServerAuth**

If **kpServerAuth** is specified, a KDC certificate with the id-kp-serverAuth EKU will be accepted. This key usage value is used in most commercially issued server certificates.

**none**

If **none** is specified, then the KDC certificate will not be checked to verify it has an acceptable EKU. The use of this option is not recommended.

**pkinit\_dh\_min\_bits**

Specifies the size of the Diffie-Hellman key the client will attempt to use. The acceptable values are 1024, 2048, and 4096. The default is 2048.

**pkinit\_identities**

Specifies the location(s) to be used to find the user’s X.509 identity information. If this option is specified multiple times, each value is attempted in order until certificates are found. Note that these values are not used if the user specifies **X509\_user\_identity** on the command line.

**pkinit\_kdc\_hostname**

The presence of this option indicates that the client is willing to accept a KDC certificate with a dNSName SAN (Subject Alternative Name) rather than requiring the id-pkinit-san as defined in [**RFC 4556**](https://tools.ietf.org/html/rfc4556.html). This option may be specified multiple times. Its value should contain the acceptable hostname for the KDC (as contained in its certificate).

**pkinit\_pool**

Specifies the location of intermediate certificates which may be used by the client to complete the trust chain between a KDC certificate and a trusted anchor. This option may be specified multiple times.

**pkinit\_require\_crl\_checking**

The default certificate verification process will always check the available revocation information to see if a certificate has been revoked. If a match is found for the certificate in a CRL, verification fails. If the certificate being verified is not listed in a CRL, or there is no CRL present for its issuing CA, and **pkinit\_require\_crl\_checking** is false, then verification succeeds.

However, if **pkinit\_require\_crl\_checking** is true and there is no CRL information available for the issuing CA, then verification fails.

**pkinit\_require\_crl\_checking** should be set to true if the policy is such that up-to-date CRLs must be present for every CA.

**pkinit\_revoke**

Specifies the location of Certificate Revocation List (CRL) information to be used by the client when verifying the validity of the KDC certificate presented. This option may be specified multiple times.

## Parameter expansion

Starting with release 1.11, several variables, such as **default\_keytab\_name**, allow parameters to be expanded. Valid parameters are:

|  |  |
| --- | --- |
| %{TEMP} | Temporary directory |
| %{uid} | Unix real UID or Windows SID |
| %{euid} | Unix effective user ID or Windows SID |
| %{USERID} | Same as %{uid} |
| %{null} | Empty string |
| %{LIBDIR} | Installation library directory |
| %{BINDIR} | Installation binary directory |
| %{SBINDIR} | Installation admin binary directory |
| %{username} | (Unix) Username of effective user ID |
| %{APPDATA} | (Windows) Roaming application data for current user |
| %{COMMON\_APPDATA} | (Windows) Application data for all users |
| %{LOCAL\_APPDATA} | (Windows) Local application data for current user |
| %{SYSTEM} | (Windows) Windows system folder |
| %{WINDOWS} | (Windows) Windows folder |
| %{USERCONFIG} | (Windows) Per-user MIT krb5 config file directory |
| %{COMMONCONFIG} | (Windows) Common MIT krb5 config file directory |

## Sample krb5.conf file

Here is an example of a generic krb5.conf file:

[libdefaults]

default\_realm = ATHENA.MIT.EDU

dns\_lookup\_kdc = true

dns\_lookup\_realm = false

[realms]

ATHENA.MIT.EDU = {

kdc = kerberos.mit.edu

kdc = kerberos-1.mit.edu

kdc = kerberos-2.mit.edu

admin\_server = kerberos.mit.edu

primary\_kdc = kerberos.mit.edu

}

EXAMPLE.COM = {

kdc = kerberos.example.com

kdc = kerberos-1.example.com

admin\_server = kerberos.example.com

}

[domain\_realm]

mit.edu = ATHENA.MIT.EDU

[capaths]

ATHENA.MIT.EDU = {

EXAMPLE.COM = .

}

EXAMPLE.COM = {

ATHENA.MIT.EDU = .

}

## FILES

/etc/krb5.conf

javax.security.auth.login

# Class LoginContext

* [java.lang.Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)
  + javax.security.auth.login.LoginContext

public class **LoginContext**

extends [Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

The LoginContext class describes the basic methods used to authenticate Subjects and provides a way to develop an application independent of the underlying authentication technology. A Configuration specifies the authentication technology, or LoginModule, to be used with a particular application. Different LoginModules can be plugged in under an application without requiring any modifications to the application itself.

In addition to supporting *pluggable* authentication, this class also supports the notion of *stacked* authentication. Applications may be configured to use more than one LoginModule. For example, one could configure both a Kerberos LoginModule and a smart card LoginModule under an application.

A typical caller instantiates a LoginContext with a *name* and a CallbackHandler. LoginContext uses the *name* as the index into a Configuration to determine which LoginModules should be used, and which ones must succeed in order for the overall authentication to succeed. The CallbackHandler is passed to the underlying LoginModules so they may communicate and interact with users (prompting for a username and password via a graphical user interface, for example).

Once the caller has instantiated a LoginContext, it invokes the login method to authenticate a Subject. The login method invokes the configured modules to perform their respective types of authentication (username/password, smart card pin verification, etc.). Note that the LoginModules will not attempt authentication retries nor introduce delays if the authentication fails. Such tasks belong to the LoginContext caller.

If the login method returns without throwing an exception, then the overall authentication succeeded. The caller can then retrieve the newly authenticated Subject by invoking the getSubject method. Principals and Credentials associated with the Subject may be retrieved by invoking the Subject's respective getPrincipals, getPublicCredentials, and getPrivateCredentials methods.

To logout the Subject, the caller calls the logout method. As with the login method, this logout method invokes the logout method for the configured modules.

A LoginContext should not be used to authenticate more than one Subject. A separate LoginContext should be used to authenticate each different Subject.

The following documentation applies to all LoginContext constructors:

* 1. Subject
     + If the constructor has a Subject input parameter, the LoginContext uses the caller-specified Subject object.
     + If the caller specifies a null Subject and a null value is permitted, the LoginContext instantiates a new Subject.
     + If the constructor does **not** have a Subject input parameter, the LoginContext instantiates a new Subject.
  2. Configuration
     + If the constructor has a Configuration input parameter and the caller specifies a non-null Configuration, the LoginContext uses the caller-specified Configuration.

If the constructor does **not** have a Configuration input parameter, or if the caller specifies a null Configuration object, the constructor uses the following call to get the installed Configuration:

config = Configuration.getConfiguration();

For both cases, the *name* argument given to the constructor is passed to the Configuration.getAppConfigurationEntry method. If the Configuration has no entries for the specified *name*, then the LoginContext calls getAppConfigurationEntry with the name, "*other*" (the default entry name). If there is no entry for "*other*", then a LoginException is thrown.

* + - When LoginContext uses the installed Configuration, the caller requires the createLoginContext.name and possibly createLoginContext.other AuthPermissions. Furthermore, the LoginContext will invoke configured modules from within an AccessController.doPrivileged call so that modules that perform security-sensitive tasks (such as connecting to remote hosts, and updating the Subject) will require the respective permissions, but the callers of the LoginContext will not require those permissions.
    - When LoginContext uses a caller-specified Configuration, the caller does not require any createLoginContext AuthPermission. The LoginContext saves the AccessControlContext for the caller, and invokes the configured modules from within an AccessController.doPrivileged call constrained by that context. This means the caller context (stored when the LoginContext was created) must have sufficient permissions to perform any security-sensitive tasks that the modules may perform.
  1. CallbackHandler
     + If the constructor has a CallbackHandler input parameter, the LoginContext uses the caller-specified CallbackHandler object.
     + If the constructor does **not** have a CallbackHandler input parameter, or if the caller specifies a null CallbackHandler object (and a null value is permitted), the LoginContext queries the *auth.login.defaultCallbackHandler* security property for the fully qualified class name of a default handler implementation. If the security property is not set, then the underlying modules will not have a CallbackHandler for use in communicating with users. The caller thus assumes that the configured modules have alternative means for authenticating the user.
     + When the LoginContext uses the installed Configuration (instead of a caller-specified Configuration, see above), then this LoginContext must wrap any caller-specified or default CallbackHandler implementation in a new CallbackHandler implementation whose handle method implementation invokes the specified CallbackHandler's handle method in a java.security.AccessController.doPrivileged call constrained by the caller's current AccessControlContext.

Note that Security Properties (such as auth.login.defaultCallbackHandler) can be set programmatically via the java.security.Security class, or statically in the Java security properties file located in the file named <JAVA\_HOME>/lib/security/java.security. <JAVA\_HOME> refers to the value of the java.home system property, and specifies the directory where the JRE is installed.

**See Also:**

[Security](https://docs.oracle.com/javase/7/docs/api/java/security/Security.html), [AuthPermission](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/AuthPermission.html), [Subject](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html), [CallbackHandler](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/CallbackHandler.html), [Configuration](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html), [LoginModule](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/spi/LoginModule.html)

## Constructor Summary

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**LoginContext**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginContext.html#LoginContext(java.lang.String))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) name)  Instantiate a new LoginContext object with a name. |
| [**LoginContext**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginContext.html#LoginContext(java.lang.String,%20javax.security.auth.callback.CallbackHandler))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) name, **[CallbackHandler](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/CallbackHandler.html" \o "interface in javax.security.auth.callback)** callbackHandler)  Instantiate a new LoginContext object with a name and a CallbackHandler object. |
| [**LoginContext**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginContext.html#LoginContext(java.lang.String,%20javax.security.auth.Subject))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) name, [**Subject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) subject)  Instantiate a new LoginContext object with a name and a Subject object. |
| [**LoginContext**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginContext.html#LoginContext(java.lang.String,%20javax.security.auth.Subject,%20javax.security.auth.callback.CallbackHandler))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) name, [**Subject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) subject, [**CallbackHandler**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/CallbackHandler.html) callbackHandler)  Instantiate a new LoginContext object with a name, a Subject to be authenticated, and a CallbackHandler object. |
| [**LoginContext**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginContext.html#LoginContext(java.lang.String,%20javax.security.auth.Subject,%20javax.security.auth.callback.CallbackHandler,%20javax.security.auth.login.Configuration))([**String**](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) name, [**Subject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) subject, [**CallbackHandler**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/CallbackHandler.html) callbackHandler, [**Configuration**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) config)  Instantiate a new LoginContext object with a name, a Subject to be authenticated, a CallbackHandler object, and a login Configuration. |

## Method Summary

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| [**Subject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) | [**getSubject**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginContext.html#getSubject())()  Return the authenticated Subject. |
| void | [**login**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginContext.html#login())()  Perform the authentication. |
| void | [**logout**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginContext.html#logout())()  Logout the Subject. |

## Methods inherited from class java.lang.[Object](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

[clone](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#clone()), [equals](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals(java.lang.Object)), [finalize](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#finalize()), [getClass](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#getClass()), [hashCode](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#hashCode()), [notify](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#notify()), [notifyAll](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#notifyAll()), [toString](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#toString()), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait()), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait(long)), [wait](https://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#wait(long,%20int))

## Constructor Detail

#### LoginContext

* + - public LoginContext([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html" \o "class in java.lang) name)

throws [LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html)

Instantiate a new LoginContext object with a name.

**Parameters:**

name - the name used as the index into the Configuration.

**Throws:**

[LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html) - if the caller-specified name does not appear in the Configuration and there is no Configuration entry for "*other*", or if the *auth.login.defaultCallbackHandler* security property was set, but the implementation class could not be loaded.

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if a SecurityManager is set and the caller does not have AuthPermission("createLoginContext.*name*"), or if a configuration entry for *name* does not exist and the caller does not additionally have AuthPermission("createLoginContext.other")

#### LoginContext

* + - public LoginContext([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html" \o "class in java.lang) name,
    - [Subject](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) subject)

throws [LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html)

Instantiate a new LoginContext object with a name and a Subject object.

**Parameters:**

name - the name used as the index into the Configuration.

subject - the Subject to authenticate.

**Throws:**

[LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html) - if the caller-specified name does not appear in the Configuration and there is no Configuration entry for "*other*", if the caller-specified subject is null, or if the *auth.login.defaultCallbackHandler* security property was set, but the implementation class could not be loaded.

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if a SecurityManager is set and the caller does not have AuthPermission("createLoginContext.*name*"), or if a configuration entry for *name* does not exist and the caller does not additionally have AuthPermission("createLoginContext.other")

#### LoginContext

* + - public LoginContext([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html" \o "class in java.lang) name,
    - [CallbackHandler](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/CallbackHandler.html) callbackHandler)

throws [LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html)

Instantiate a new LoginContext object with a name and a CallbackHandler object.

**Parameters:**

name - the name used as the index into the Configuration.

callbackHandler - the CallbackHandler object used by LoginModules to communicate with the user.

**Throws:**

[LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html) - if the caller-specified name does not appear in the Configuration and there is no Configuration entry for "*other*", or if the caller-specified callbackHandler is null.

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if a SecurityManager is set and the caller does not have AuthPermission("createLoginContext.*name*"), or if a configuration entry for *name* does not exist and the caller does not additionally have AuthPermission("createLoginContext.other")

#### LoginContext

* + - public LoginContext([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html" \o "class in java.lang) name,
    - [Subject](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) subject,
    - [CallbackHandler](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/CallbackHandler.html) callbackHandler)

throws [LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html)

Instantiate a new LoginContext object with a name, a Subject to be authenticated, and a CallbackHandler object.

**Parameters:**

name - the name used as the index into the Configuration.

subject - the Subject to authenticate.

callbackHandler - the CallbackHandler object used by LoginModules to communicate with the user.

**Throws:**

[LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html) - if the caller-specified name does not appear in the Configuration and there is no Configuration entry for "*other*", or if the caller-specified subject is null, or if the caller-specified callbackHandler is null.

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if a SecurityManager is set and the caller does not have AuthPermission("createLoginContext.*name*"), or if a configuration entry for *name* does not exist and the caller does not additionally have AuthPermission("createLoginContext.other")

#### LoginContext

* + - public LoginContext([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html" \o "class in java.lang) name,
    - [Subject](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) subject,
    - [CallbackHandler](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/CallbackHandler.html) callbackHandler,
    - [Configuration](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/Configuration.html) config)

throws [LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html)

Instantiate a new LoginContext object with a name, a Subject to be authenticated, a CallbackHandler object, and a login Configuration.

**Parameters:**

name - the name used as the index into the caller-specified Configuration.

subject - the Subject to authenticate, or null.

callbackHandler - the CallbackHandler object used by LoginModules to communicate with the user, or null.

config - the Configuration that lists the login modules to be called to perform the authentication, or null.

**Throws:**

[LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html) - if the caller-specified name does not appear in the Configuration and there is no Configuration entry for "*other*".

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - if a SecurityManager is set, *config* is null, and either the caller does not have AuthPermission("createLoginContext.*name*"), or if a configuration entry for *name* does not exist and the caller does not additionally have AuthPermission("createLoginContext.other")

**Since:**

1.5

### Method Detail

#### login

* + - public void login()

throws [LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html)

Perform the authentication.

This method invokes the login method for each LoginModule configured for the *name* specified to the LoginContext constructor, as determined by the login Configuration. Each LoginModule then performs its respective type of authentication (username/password, smart card pin verification, etc.).

This method completes a 2-phase authentication process by calling each configured LoginModule's commit method if the overall authentication succeeded (the relevant REQUIRED, REQUISITE, SUFFICIENT, and OPTIONAL LoginModules succeeded), or by calling each configured LoginModule's abort method if the overall authentication failed. If authentication succeeded, each successful LoginModule's commit method associates the relevant Principals and Credentials with the Subject. If authentication failed, each LoginModule's abort method removes/destroys any previously stored state.

If the commit phase of the authentication process fails, then the overall authentication fails and this method invokes the abort method for each configured LoginModule.

If the abort phase fails for any reason, then this method propagates the original exception thrown either during the login phase or the commit phase. In either case, the overall authentication fails.

In the case where multiple LoginModules fail, this method propagates the exception raised by the first LoginModule which failed.

Note that if this method enters the abort phase (either the login or commit phase failed), this method invokes all LoginModules configured for the application regardless of their respective Configuration flag parameters. Essentially this means that Requisite and Sufficient semantics are ignored during the abort phase. This guarantees that proper cleanup and state restoration can take place.

**Throws:**

[LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html) - if the authentication fails.

#### logout

* + - public void logout()

throws [LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html)

Logout the Subject.

This method invokes the logout method for each LoginModule configured for this LoginContext. Each LoginModule performs its respective logout procedure which may include removing/destroying Principal and Credential information from the Subject and state cleanup.

Note that this method invokes all LoginModules configured for the application regardless of their respective Configuration flag parameters. Essentially this means that Requisite and Sufficient semantics are ignored for this method. This guarantees that proper cleanup and state restoration can take place.

**Throws:**

[LoginException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/login/LoginException.html) - if the logout fails.

#### getSubject

public [Subject](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/Subject.html) getSubject()

Return the authenticated Subject.

**Returns:**

the authenticated Subject. If the caller specified a Subject to this LoginContext's constructor, this method returns the caller-specified Subject. If a Subject was not specified and authentication succeeds, this method returns the Subject instantiated and used for authentication by this LoginContext. If a Subject was not specified, and authentication fails or has not been attempted, this method returns null.

javax.security.auth.callback

# Interface CallbackHandler

public interface **CallbackHandler**

An application implements a CallbackHandler and passes it to underlying security services so that they may interact with the application to retrieve specific authentication data, such as usernames and passwords, or to display certain information, such as error and warning messages.

CallbackHandlers are implemented in an application-dependent fashion. For example, implementations for an application with a graphical user interface (GUI) may pop up windows to prompt for requested information or to display error messages. An implementation may also choose to obtain requested information from an alternate source without asking the end user.

Underlying security services make requests for different types of information by passing individual Callbacks to the CallbackHandler. The CallbackHandler implementation decides how to retrieve and display information depending on the Callbacks passed to it. For example, if the underlying service needs a username and password to authenticate a user, it uses a NameCallback and PasswordCallback. The CallbackHandler can then choose to prompt for a username and password serially, or to prompt for both in a single window.

A default CallbackHandler class implementation may be specified in the *auth.login.defaultCallbackHandler* security property. The security property can be set in the Java security properties file located in the file named <JAVA\_HOME>/lib/security/java.security. <JAVA\_HOME> refers to the value of the java.home system property, and specifies the directory where the JRE is installed.

If the security property is set to the fully qualified name of a CallbackHandler implementation class, then a LoginContext will load the specified CallbackHandler and pass it to the underlying LoginModules. The LoginContext only loads the default handler if it was not provided one.

All default handler implementations must provide a public zero-argument constructor.

## Method Summary

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| void | [**handle**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/CallbackHandler.html#handle(javax.security.auth.callback.Callback[]))([**Callback**](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/Callback.html)[] callbacks)  Retrieve or display the information requested in the provided Callbacks. |

## Method Detail

### handle

* + - void handle([Callback](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/Callback.html)[] callbacks)
    - throws [IOException](https://docs.oracle.com/javase/7/docs/api/java/io/IOException.html),

[UnsupportedCallbackException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/UnsupportedCallbackException.html)

Retrieve or display the information requested in the provided Callbacks.

The handle method implementation checks the instance(s) of the Callback object(s) passed in to retrieve or display the requested information. The following example is provided to help demonstrate what an handle method implementation might look like. This example code is for guidance only. Many details, including proper error handling, are left out for simplicity.

public void handle(Callback[] callbacks)

throws IOException, UnsupportedCallbackException {

for (int i = 0; i < callbacks.length; i++) {

if (callbacks[i] instanceof TextOutputCallback) {

// display the message according to the specified type

TextOutputCallback toc = (TextOutputCallback)callbacks[i];

switch (toc.getMessageType()) {

case TextOutputCallback.INFORMATION:

System.out.println(toc.getMessage());

break;

case TextOutputCallback.ERROR:

System.out.println("ERROR: " + toc.getMessage());

break;

case TextOutputCallback.WARNING:

System.out.println("WARNING: " + toc.getMessage());

break;

default:

throw new IOException("Unsupported message type: " +

toc.getMessageType());

}

} else if (callbacks[i] instanceof NameCallback) {

// prompt the user for a username

NameCallback nc = (NameCallback)callbacks[i];

// ignore the provided defaultName

System.err.print(nc.getPrompt());

System.err.flush();

nc.setName((new BufferedReader

(new InputStreamReader(System.in))).readLine());

} else if (callbacks[i] instanceof PasswordCallback) {

// prompt the user for sensitive information

PasswordCallback pc = (PasswordCallback)callbacks[i];

System.err.print(pc.getPrompt());

System.err.flush();

pc.setPassword(readPassword(System.in));

} else {

throw new UnsupportedCallbackException

(callbacks[i], "Unrecognized Callback");

}

}

}

// Reads user password from given input stream.

private char[] readPassword(InputStream in) throws IOException {

// insert code to read a user password from the input stream

}

**Parameters:**

callbacks - an array of Callback objects provided by an underlying security service which contains the information requested to be retrieved or displayed.

**Throws:**

[IOException](https://docs.oracle.com/javase/7/docs/api/java/io/IOException.html) - if an input or output error occurs.

[UnsupportedCallbackException](https://docs.oracle.com/javase/7/docs/api/javax/security/auth/callback/UnsupportedCallbackException.html) - if the implementation of this method does not support one or more of the Callbacks specified in the callbacks parameter.

### Understanding Callbacks with Java - Inversion of control

# CALLBACK PATTERN IN JAVA ENVIRONMENT

Hi there! today i wanna share something with you, that it is very common and widely used in javascript for example. I'm speaking of callbacks. Do you know how and when this "pattern" is used? Do you really understand it in a java context (environment)? Well i was asking me also some of those questions and that's the reason i started to learn more about it. The ideia behind it is the**inversion of control**(abbreviated IoC). This paradigma describes the way frameworks work. It is also known as the "**Hollywood principle - Don't call me, we will call you**"

## SIMPLIFIED CALLBACK PATTERN IN JAVA JUST TO UNDERSTAND IT. CONCRETE EXAMPLE FOLLOWS BELLOW.

[Text

Description automatically generated](http://2.bp.blogspot.com/-hI_FAaoQdA0/UHnjxgPSNsI/AAAAAAAAAxc/QbrGPHuZyGY/s1600/callback1.png)

interface CallBack {

    void methodToCallBack();

}

class CallBackImpl implements CallBack {

    public void methodToCallBack() {

        System.out.println("I've been called back");

    }

}

class Caller {

    public void register(CallBack callback) {

        callback.methodToCallBack();

    }

    public static void main(String[] args) {

        Caller caller = new Caller();

        CallBack callBack = new CallBackImpl();

        caller.register(callBack);

    }

}

Ok you may be asking you, when this usefull or may be asking you what's the difference between calling directly callback.methodToCallBack() right?  
  
**ANSWER:** well, this example just shows to you how to construct such a callBack function when working in a java environment. Certainlly it doesn't make it any sense to use it that way. Let's get a little deeper into a concrete useful example now.  
  
The idea behind it is the "**INVERSION OF CONTROL**". Let's take a timer as a realistic example. Let's supose that you know, that a specific timer supports callback functions every hour. Exactly it means, that every hour, the timer will call your registed call method function.

## CONCTRETE EXAMPLE:

Let's say we wanna update the time of a website every hour. Here is the UML of the following example:

[Diagram

Description automatically generated](http://1.bp.blogspot.com/-i6V43ccN100/UHnjolpG_AI/AAAAAAAAAxU/R3uF09_n6Zk/s1600/callback2.png)

## CALLBACK INTERFACE

Let's define first the callback interface:

[A screenshot of a computer

Description automatically generated with medium confidence](http://1.bp.blogspot.com/-fmaLxS4A2iw/UHnkSzpoIgI/AAAAAAAAAxk/CQDD8lpwnco/s1600/callback3.png)

import java.util.ArrayList;

import java.util.List;

// For example: Let's assume that this interface is offered from your OS to be implemented

interface TimeUpdaterCallBack {

    void updateTime(long time);

}

// this is your implementation.

// for example: You want to update your website time every hour

class WebSiteTimeUpdaterCallBack implements TimeUpdaterCallBack {

    @Override

    public void updateTime(long time) {

        // print the updated time anywhere in your website's example

        System.out.println(time);

    }

}

## THE SYSTEMTIMER THAT SUPPORTS CALLBACK FUNCTIONS IN OUR EXAMPLE:

[Text

Description automatically generated](http://2.bp.blogspot.com/-Seeod0CS5Do/UHnk2SrABXI/AAAAAAAAAxs/029OtSZQRKo/s1600/callback4png.png)

// This is the SystemTimer implemented by your Operating System (OS)

// You don't know how this timer was implemented. This example just

// show to you how it could looks like. How you could implement a

// callback by yourself if you want to.

class SystemTimer {

    List<TimeUpdaterCallBack> callbacks = new ArrayList<TimeUpdaterCallBack>();

    public void registerCallBackForUpdatesEveryHour(TimeUpdaterCallBack timerCallBack) {

        callbacks.add(timerCallBack);

    }

    // ... This SystemTimer may have more logic here we don't know ...

    // At some point of the implementaion of this SystemTimer (you don't know)

    // this method will be called and every registered timerCallBack

    // will be called. Every registered timerCallBack may have a totally

    // different implementation of the method updateTime() and my be

    // used in different ways by different clients.

    public void oneHourHasBeenExprired() {

        for (TimeUpdaterCallBack timerCallBack : callbacks) {

            timerCallBack.updateTime(System.currentTimeMillis());

        }

    }

}

## AND FINALLY OUR WEBSITETIMEUPDATER WHICH IS OUR CLIENT IN THIS FICTIVE AND SIMPLE EXAMPLE:

[Text

Description automatically generated](http://2.bp.blogspot.com/-_6AkG1vV0n8/UHnlV4Eb4HI/AAAAAAAAAx0/WDkwzeekeWE/s1600/callback5.png)

// This is our client. It will be used in our WebSite example. It shall update

// the website's time every hour.

class WebSiteTimeUpdater {

    public static void main(String[] args) {

        SystemTimer SystemTimer = new SystemTimer();

        TimeUpdaterCallBack webSiteCallBackUpdater = new WebSiteTimeUpdaterCallBack();

        SystemTimer.registerCallBackForUpdatesEveryHour(webSiteCallBackUpdater);

    }

}

# Anatomy of a Synchronizer

* [State](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#state)
* [Access Condition](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#accesscondition)
* [State Changes](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#statechanges)
* [Notification Strategy](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#notificationstrategy)
* [Test and Set Method](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#testandset)
* [Set Method](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#set)

|  |  |
| --- | --- |
| A picture containing text, person, person, wearing  Description automatically generated | Jakob Jenkov Last update: 2014-10-01 |

Even if many synchronizers (locks, semaphores, blocking queue etc.) are different in function, they are often not that different in their internal design. In other words, they consist of the same (or similar) basic parts internally. Knowing these basic parts can be a great help when designing synchronizers. It is these parts this text looks closer at.

**Note:** The content of this text is a part result of a M.Sc. student project at the IT University of Copenhagen in the spring 2004 by Jakob Jenkov, Toke Johansen and Lars Bjørn. During this project we asked Doug Lea if he knew of similar work. Interestingly he had come up with similar conclusions independently of this project during the development of the Java 5 concurrency utilities. Doug Lea's work, I believe, is described in the book [**"Java Concurrency in Practice"**](http://www.amazon.com/Java-Concurrency-Practice-Brian-Goetz/dp/0321349601/ref=pd_bbs_sr_1?ie=UTF8&s=books&qid=1215418711&sr=8-1). This book also contains a chapter with the title "Anatomy of a Synchronizer" with content similar to this text, though not exactly the same.

The purpose of most (if not all) synchronizers is to guard some area of the code (critical section) from concurrent access by threads. To do this the following parts are often needed in a synchronizer:

1. [**State**](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#state)
2. [**Access Condition**](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#accesscondition)
3. [**State Changes**](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#statechanges)
4. [**Notification Strategy**](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#notificationstrategy)
5. [**Test and Set Method**](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#testandset)
6. [**Set Method**](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#set)

Not all synchronizers have all of these parts, and those that have may not have them exactly as they are described here. Usually you can find one or more of these parts, though.

## State

The state of a synchronizer is used by the access condition to determine if a thread can be granted access. In a [**Lock**](http://tutorials.jenkov.com/java-concurrency/locks.html) the state is kept in a boolean saying whether the Lock is locked or not. In a [**Bounded Semaphore**](http://tutorials.jenkov.com/java-concurrency/semaphores.html#bounded) the internal state is kept in a counter (int) and an upper bound (int) which state the current number of "takes" and the maximum number of "takes". In a [**Blocking Queue**](http://tutorials.jenkov.com/java-concurrency/blocking-queues.html) the state is kept in the List of elements in the queue and the maximum queue size (int) member (if any).

Here are two code snippets from both Lock and a BoundedSemaphore. The state code is marked in bold.

public class Lock{

**//state is kept here**

**private boolean isLocked = false;**

public synchronized void lock()

throws InterruptedException{

while(isLocked){

wait();

}

isLocked = true;

}

...

}

public class BoundedSemaphore {

**//state is kept here**

**private int signals = 0;**

**private int bound = 0;**

public BoundedSemaphore(int upperBound){

this.bound = upperBound;

}

public synchronized void take() throws InterruptedException{

while(this.signals == bound) wait();

this.signal++;

this.notify();

}

...

}

## Access Condition

The access conditions is what determines if a thread calling a test-and-set-state method can be allowed to set the state or not. The access condition is typically based on the [**state**](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#state) of the synchronizer. The access condition is typically checked in a while loop to guard against [**Spurious Wakeups**](http://tutorials.jenkov.com/java-concurrency/thread-signaling.html#spuriouswakeups). When the access condition is evaluated it is either true or false.

In a [**Lock**](http://tutorials.jenkov.com/java-concurrency/locks.html) the access condition simply checks the value of the isLocked member variable. In a [**Bounded Semaphore**](http://tutorials.jenkov.com/java-concurrency/semaphores.html#bounded) there are actually two access conditions depending on whether you are trying to "take" or "release" the semaphore. If a thread tries to take the semaphore the signals variable is checked against the upper bound. If a thread tries to release the semaphore the signals variable is checked against 0.

Here are two code snippets of a Lock and a BoundedSemaphore with the access condition marked in bold. Notice how the conditions is always checked inside a while loop.

public class Lock{

private boolean isLocked = false;

public synchronized void lock()

throws InterruptedException{

**//access condition**

while(**isLocked**){

wait();

}

isLocked = true;

}

...

}

public class BoundedSemaphore {

private int signals = 0;

private int bound = 0;

public BoundedSemaphore(int upperBound){

this.bound = upperBound;

}

public synchronized void take() throws InterruptedException{

**//access condition**

while(**this.signals == bound**) wait();

this.signals++;

this.notify();

}

public synchronized void release() throws InterruptedException{

**//access condition**

while(**this.signals == 0**) wait();

this.signals--;

this.notify();

}

}

## State Changes

Once a thread gains access to the critical section it has to change the state of the synchronizer to (possibly) block other threads from entering it. In other words, the state needs to reflect the fact that a thread is now executing inside the critical section. This should affect the access conditions of other threads attempting to gain access.

In a [**Lock**](http://tutorials.jenkov.com/java-concurrency/locks.html) the state change is the code setting isLocked = true. In a semaphore it is either the code signals-- or signals++;

Here are two code snippets with the state change code marked in bold:

public class Lock{

private boolean isLocked = false;

public synchronized void lock()

throws InterruptedException{

while(isLocked){

wait();

}

**//state change**

**isLocked = true;**

}

public synchronized void unlock(){

**//state change**

**isLocked = false;**

notify();

}

}

public class BoundedSemaphore {

private int signals = 0;

private int bound = 0;

public BoundedSemaphore(int upperBound){

this.bound = upperBound;

}

public synchronized void take() throws InterruptedException{

while(this.signals == bound) wait();

**//state change**

**this.signals++;**

this.notify();

}

public synchronized void release() throws InterruptedException{

while(this.signals == 0) wait();

**//state change**

**this.signals--;**

this.notify();

}

}

## Notification Strategy

Once a thread has changed the state of a synchronizer it may sometimes need to notify other waiting threads about the state change. Perhaps this state change might turn the access condition true for other threads.

Notification Strategies typically fall into three categories.

1. Notify all waiting threads.
2. Notify 1 random of N waiting threads.
3. Notify 1 specific of N waiting thread.

Notifying all waiting threads is pretty easy. All waiting threads call wait() on the same object. Once a thread want to notify the waiting threads it calls notifyAll() on the object the waiting threads called wait() on.

Notifying a single random waiting thread is also pretty easy. Just have the notifying thread call notify() on the object the waiting threads have called wait() on. Calling notify makes no guarantee about which of the waiting threads will be notified. Hence the term "random waiting thread".

Sometimes you may need to notify a specific rather than a random waiting thread. For instance if you need to guarantee that waiting threads are notified in a specific order, be it the order they called the synchronizer in, or some prioritized order. To achive this each waiting thread must call wait() on its own, separate object. When the notifying thread wants to notify a specific waiting thread it will call notify() on the object this specific thread has called wait() on. An example of this can be found in the text [**Starvation and Fairness**](http://tutorials.jenkov.com/java-concurrency/starvation-and-fairness.html).

Below is a code snippet with the notification strategy (notify 1 random waiting thread) marked in bold:

public class Lock{

private boolean isLocked = false;

public synchronized void lock()

throws InterruptedException{

while(isLocked){

**//wait strategy - related to notification strategy**

**wait();**

}

isLocked = true;

}

public synchronized void unlock(){

isLocked = false;

**notify(); //notification strategy**

}

}

## Test and Set Method

Synchronizer most often have two types of methods of which test-and-set is the first type ([**set**](http://tutorials.jenkov.com/java-concurrency/anatomy-of-a-synchronizer.html#set) is the other). Test-and-set means that the thread calling this method **tests** the internal state of the synchronizer against the access condition. If the condition is met the thread **sets** the internal state of the synchronizer to reflect that the thread has gained access.

The state transition usually results in the access condition turning false for other threads trying to gain access, but may not always do so. For instance, in a [**Read - Write Lock**](http://tutorials.jenkov.com/java-concurrency/read-write-locks.html) a thread gaining read access will update the state of the read-write lock to reflect this, but other threads requesting read access will also be granted access as long as no threads has requested write access.

It is imperative that the test-and-set operations are executed atomically meaning no other threads are allowed to execute in the test-and-set method in between the test and the setting of the state.

The program flow of a test-and-set method is usually something along the lines of:

1. Set state before test if necessary
2. Test state against access condition
3. If access condition is not met, wait
4. If access condition is met, set state, and notify waiting threads if necessary

The lockWrite() method of a **[ReadWriteLock](http://tutorials.jenkov.com/java-concurrency/read-write-locks.html)** class shown below is an example of a test-and-set method. Threads calling lockWrite() first sets the state before the test (writeRequests++). Then it tests the internal state against the access condition in the canGrantWriteAccess() method. If the test succeeds the internal state is set again before the method is exited. Notice that this method does not notify waiting threads.

public class ReadWriteLock{

private Map<Thread, Integer> readingThreads =

new HashMap<Thread, Integer>();

private int writeAccesses = 0;

private int writeRequests = 0;

private Thread writingThread = null;

...

**public synchronized void lockWrite() throws InterruptedException{**

**writeRequests++;**

**Thread callingThread = Thread.currentThread();**

**while(! canGrantWriteAccess(callingThread)){**

**wait();**

**}**

**writeRequests--;**

**writeAccesses++;**

**writingThread = callingThread;**

**}**

...

}

The BoundedSemaphore class shown below has two test-and-set methods: take() and  
release(). Both methods test and sets the internal state.

public class BoundedSemaphore {

private int signals = 0;

private int bound = 0;

public BoundedSemaphore(int upperBound){

this.bound = upperBound;

}

**public synchronized void take() throws InterruptedException{**

**while(this.signals == bound) wait();**

**this.signals++;**

**this.notify();**

**}**

**public synchronized void release() throws InterruptedException{**

**while(this.signals == 0) wait();**

**this.signals--;**

**this.notify();**

**}**

}

## Set Method

The set method is the second type of method that synchronizers often contain. The set method just sets the internal state of the synchronizer without testing it first. A typical example of a set method is the unlock() method of a Lock class. A thread holding the lock can always unlock it without having to test if the Lock is unlocked.

The program flow of a set method is usually along the lines of:

1. Set internal state
2. Notify waiting threads

Here is an example unlock() method:

public class Lock{

private boolean isLocked = false;

**public synchronized void unlock(){**

**isLocked = false;**

**notify();**

**}**

}

# **How to use**

# set

# **method**

# **in**

# [**org.apache.hadoop.conf.Configuration**](https://www.tabnine.com/code/java/classes/org.apache.hadoop.conf.Configuration)

## Best [Java](https://www.tabnine.com/code/java) code snippets using [org.apache.hadoop.conf](https://www.tabnine.com/code/java/packages/org.apache.hadoop.conf" \o "org.apache.hadoop.conf).[Configuration](https://www.tabnine.com/code/java/classes/org.apache.hadoop.conf.Configuration).set(Showing top 20 results out of 8,802)

### **Refine search**

* [Configuration.<init>](https://www.tabnine.com/code/query/org.apache.hadoop.conf@Configuration@set+org.apache.hadoop.conf@Configuration@%3Cinit%3E)
* [Configuration.get](https://www.tabnine.com/code/query/org.apache.hadoop.conf@Configuration@set+org.apache.hadoop.conf@Configuration@get)
* [Test.<init>](https://www.tabnine.com/code/query/org.apache.hadoop.conf@Configuration@set+org.junit@Test@%3Cinit%3E)
* [Path.<init>](https://www.tabnine.com/code/query/org.apache.hadoop.conf@Configuration@set+org.apache.hadoop.fs@Path@%3Cinit%3E)
* [Configuration.setInt](https://www.tabnine.com/code/query/org.apache.hadoop.conf@Configuration@set+org.apache.hadoop.conf@Configuration@setInt)
* [Assert.assertEquals](https://www.tabnine.com/code/query/org.apache.hadoop.conf@Configuration@set+org.junit@Assert@assertEquals)
* [Assert.assertTrue](https://www.tabnine.com/code/query/org.apache.hadoop.conf@Configuration@set+org.junit@Assert@assertTrue)
* [Path.toString](https://www.tabnine.com/code/query/org.apache.hadoop.conf@Configuration@set+org.apache.hadoop.fs@Path@toString)
* [Configuration.setBoolean](https://www.tabnine.com/code/query/org.apache.hadoop.conf@Configuration@set+org.apache.hadoop.conf@Configuration@setBoolean)
* **Common ways to obtainConfiguration**

**private void myMethod () {**

**Configuration c =**

* **Codota Iconnew Configuration()**
* **Codota Icon**Mapper.Context context;**context.getConfiguration()**
* **Codota Icon**Reducer.Context context;**context.getConfiguration()**
* [*Smart code suggestions by Tabnine*](https://www.tabnine.com/?utm_source=search-web)

**}**

[**Get smart completions for your Java IDE**Add Tabnine to your IDE (free)](https://www.tabnine.com/?utm_source=search-web)

[*origin:***apache/flink**](https://github.com/apache/flink/tree/master/flink-filesystems/flink-fs-hadoop-shaded/src/main/java/org/apache/hadoop/conf/Configuration.java#L1301)

#### **Configuration.setIfUnset(...)**

*/\*\**

*\* Sets a property if it is currently unset.*

*\* @param name the property name*

*\* @param value the new value*

*\*/*

**public** **synchronized** **void** setIfUnset(String name, String value) {

**if** ([get](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/get)(name) == null) {

[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(name, value);

}

}

[*origin:***prestodb/presto**](https://github.com/prestodb/presto/tree/master/presto-hive/src/main/java/com/facebook/presto/hive/HdfsConfigurationUpdater.java#L110)

#### **HdfsConfigurationUpdater.updateConfiguration(...)**

**public** **void** updateConfiguration(Configuration config)

{

copy(resourcesConfiguration, config);

*// this is to prevent dfs client from doing reverse DNS lookups to determine whether nodes are rack local*

config.setClass(NET\_TOPOLOGY\_NODE\_SWITCH\_MAPPING\_IMPL\_KEY, NoOpDNSToSwitchMapping.**class**, DNSToSwitchMapping.**class**);

**if** (socksProxy != null) {

config.setClass(HADOOP\_RPC\_SOCKET\_FACTORY\_CLASS\_DEFAULT\_KEY, SocksSocketFactory.**class**, SocketFactory.**class**);

config.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(HADOOP\_SOCKS\_SERVER\_KEY, socksProxy.toString());

}

**if** (domainSocketPath != null) {

config.setStrings(DFS\_DOMAIN\_SOCKET\_PATH\_KEY, domainSocketPath);

}

*// only enable short circuit reads if domain socket path is properly configured*

**if** (!config.[get](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/get)(DFS\_DOMAIN\_SOCKET\_PATH\_KEY, "").trim().isEmpty()) {

config.setBooleanIfUnset(DFS\_CLIENT\_READ\_SHORTCIRCUIT\_KEY, **true**);

}

config.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)(DFS\_CLIENT\_SOCKET\_TIMEOUT\_KEY, toIntExact(dfsTimeout.toMillis()));

config.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)(IPC\_PING\_INTERVAL\_KEY, toIntExact(ipcPingInterval.toMillis()));

config.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)(IPC\_CLIENT\_CONNECT\_TIMEOUT\_KEY, toIntExact(dfsConnectTimeout.toMillis()));

config.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)(IPC\_CLIENT\_CONNECT\_MAX\_RETRIES\_KEY, dfsConnectMaxRetries);

**if** (isHdfsWireEncryptionEnabled) {

config.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(HADOOP\_RPC\_PROTECTION, "privacy");

config.[setBoolean](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setBoolean)("dfs.encrypt.data.transfer", **true**);

}

config.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)("fs.cache.max-size", fileSystemMaxCacheSize);

config.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)(LineRecordReader.MAX\_LINE\_LENGTH, textMaxLineLength);

configureCompression(config, compressionCodec);

s3ConfigurationUpdater.updateConfiguration(config);

}

[*origin:***azkaban/azkaban**](https://www.tabnine.com/web/assistant/code/rs/5c788937df79be0001ea88e9#L52)

#### **HadoopModule.createHadoopConfiguration()**

@Inject

@Provides

@Singleton

**public** Configuration createHadoopConfiguration() {

**final** String hadoopConfDirPath = requireNonNull(**this**.props.get(HADOOP\_CONF\_DIR\_PATH));

**final** File hadoopConfDir = **new** File(requireNonNull(hadoopConfDirPath));

checkArgument(hadoopConfDir.exists() && hadoopConfDir.isDirectory());

**final** Configuration hadoopConf = **new** [Configuration](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/%3Cinit%3E)(**false**);

hadoopConf.addResource(**new** [org.apache.hadoop.fs.Path](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/%3Cinit%3E)(hadoopConfDirPath, "core-site.xml"));

hadoopConf.addResource(**new** [org.apache.hadoop.fs.Path](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/%3Cinit%3E)(hadoopConfDirPath, "hdfs-site.xml"));

hadoopConf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)("fs.hdfs.impl", org.apache.hadoop.hdfs.DistributedFileSystem.**class**.getName());

**return** hadoopConf;

}

[*origin:***apache/hive**](https://github.com/apache/hive/tree/master/kafka-handler/src/test/org/apache/hadoop/hive/kafka/KafkaUtilsTest.java#L42)

#### **KafkaUtilsTest.testConsumerProperties()**

@[Test](https://www.tabnine.com/code/java/methods/org.junit.Test/%3Cinit%3E) **public** **void** testConsumerProperties() {

Configuration configuration = **new** [Configuration](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/%3Cinit%3E)();

configuration.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)("kafka.bootstrap.servers", "localhost:9090");

configuration.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)("kafka.consumer.fetch.max.wait.ms", "40");

configuration.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)("kafka.consumer.my.new.wait.ms", "400");

Properties properties = KafkaUtils.consumerProperties(configuration);

Assert.[assertEquals](https://www.tabnine.com/code/java/methods/org.junit.Assert/assertEquals)("localhost:9090", properties.getProperty("bootstrap.servers"));

Assert.[assertEquals](https://www.tabnine.com/code/java/methods/org.junit.Assert/assertEquals)("40", properties.getProperty("fetch.max.wait.ms"));

Assert.[assertEquals](https://www.tabnine.com/code/java/methods/org.junit.Assert/assertEquals)("400", properties.getProperty("my.new.wait.ms"));

}

[*origin:***apache/hbase**](https://github.com/apache/hbase/tree/master/hbase-common/src/test/java/org/apache/hadoop/hbase/zookeeper/TestZKConfig.java#L54)

#### **TestZKConfig.testGetZooKeeperClusterKey()**

@[Test](https://www.tabnine.com/code/java/methods/org.junit.Test/%3Cinit%3E)

**public** **void** testGetZooKeeperClusterKey() {

Configuration conf = HBaseConfiguration.create();

conf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(HConstants.ZOOKEEPER\_QUORUM, "\tlocalhost\n");

conf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(HConstants.ZOOKEEPER\_CLIENT\_PORT, "3333");

conf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(HConstants.ZOOKEEPER\_ZNODE\_PARENT, "hbase");

String clusterKey = ZKConfig.getZooKeeperClusterKey(conf, "test");

[assertTrue](https://www.tabnine.com/code/java/methods/org.junit.Assert/assertTrue)(!clusterKey.contains("\t") && !clusterKey.contains("\n"));

[assertEquals](https://www.tabnine.com/code/java/methods/org.junit.Assert/assertEquals)("localhost:3333:hbase,test", clusterKey);

}

[*origin:***apache/hive**](https://github.com/apache/hive/tree/master/standalone-metastore/metastore-server/src/test/java/org/apache/hadoop/hive/metastore/utils/TestHdfsUtils.java#L326)

#### **TestHdfsUtils.testSetFullFileStatusFailInheritPermsRecursive()**

*/\*\**

*\* Tests that HdfsUtils#setFullFileStatus*

*\* does not thrown an exception when setting permissions and with recursion.*

*\*/*

@[Test](https://www.tabnine.com/code/java/methods/org.junit.Test/%3Cinit%3E)

**public** **void** testSetFullFileStatusFailInheritPermsRecursive() **throws** Exception {

Configuration conf = **new** [Configuration](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/%3Cinit%3E)();

conf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)("dfs.namenode.acls.enabled", "false");

Path fakeTarget = **new** [Path](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/%3Cinit%3E)("fakePath");

HdfsUtils.HadoopFileStatus mockHadoopFileStatus = mock(HdfsUtils.HadoopFileStatus.**class**);

FileStatus mockSourceStatus = mock(FileStatus.**class**);

FsShell mockFsShell = mock(FsShell.**class**);

when(mockSourceStatus.getPermission()).thenReturn(**new** FsPermission((**short**) 777));

when(mockHadoopFileStatus.getFileStatus()).thenReturn(mockSourceStatus);

doThrow(RuntimeException.**class**).when(mockFsShell).run(any(String[].**class**));

HdfsUtils.setFullFileStatus(conf, mockHadoopFileStatus, "", mock(FileSystem.**class**), fakeTarget,

**true**, mockFsShell);

verify(mockFsShell).run(**new** String[]{"-chmod", "-R", any(String.**class**), fakeTarget.[toString](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/toString)()});

}

}

[*origin:***apache/hbase**](https://github.com/apache/hbase/tree/master/hbase-server/src/test/java/org/apache/hadoop/hbase/client/TestFromClientSideNoCodec.java#L113)

#### **TestFromClientSideNoCodec.testNoCodec()**

@[Test](https://www.tabnine.com/code/java/methods/org.junit.Test/%3Cinit%3E)

**public** **void** testNoCodec() {

Configuration c = **new** [Configuration](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/%3Cinit%3E)();

c.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)("hbase.client.default.rpc.codec", "");

String codec = AbstractRpcClient.getDefaultCodec(c);

[assertTrue](https://www.tabnine.com/code/java/methods/org.junit.Assert/assertTrue)(codec == null || codec.length() == 0);

}

}

[*origin:***apache/hbase**](https://github.com/apache/hbase/tree/master/hbase-server/src/test/java/org/apache/hadoop/hbase/regionserver/TestStripeStoreEngine.java#L57)

#### **TestStripeStoreEngine.testCreateBasedOnConfig()**

@[Test](https://www.tabnine.com/code/java/methods/org.junit.Test/%3Cinit%3E)

**public** **void** testCreateBasedOnConfig() **throws** Exception {

Configuration conf = HBaseConfiguration.create();

conf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(StoreEngine.STORE\_ENGINE\_CLASS\_KEY, TestStoreEngine.**class**.getName());

StripeStoreEngine se = createEngine(conf);

[assertTrue](https://www.tabnine.com/code/java/methods/org.junit.Assert/assertTrue)(se.getCompactionPolicy() **instanceof** StripeCompactionPolicy);

}

[*origin:***apache/hbase**](https://github.com/apache/hbase/tree/master/hbase-server/src/test/java/org/apache/hadoop/hbase/io/hfile/TestCacheConfig.java#L284)

#### **TestCacheConfig.testFileBucketCacheConfig()**

@[Test](https://www.tabnine.com/code/java/methods/org.junit.Test/%3Cinit%3E)

**public** **void** testFileBucketCacheConfig() **throws** IOException {

HBaseTestingUtility htu = **new** HBaseTestingUtility(**this**.conf);

**try** {

Path p = **new** [Path](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/%3Cinit%3E)(htu.getDataTestDir(), "bc.txt");

FileSystem fs = FileSystem.get(**this**.conf);

fs.create(p).close();

**this**.conf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(HConstants.BUCKET\_CACHE\_IOENGINE\_KEY, "file:" + p);

doBucketCacheConfigTest();

} **finally** {

htu.cleanupTestDir();

}

}

[*origin:***prestodb/presto**](https://github.com/prestodb/presto/tree/master/presto-hive/src/test/java/com/facebook/presto/hive/s3/TestPrestoS3FileSystem.java#L425)

#### **TestPrestoS3FileSystem.testCustomCredentialsClassCannotBeFound()**

@Test(expectedExceptions = RuntimeException.**class**, expectedExceptionsMessageRegExp = "Error creating an instance of .\*")

**public** **void** testCustomCredentialsClassCannotBeFound()

**throws** Exception

{

Configuration config = **new** [Configuration](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/%3Cinit%3E)();

config.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(S3\_USE\_INSTANCE\_CREDENTIALS, "false");

config.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(S3\_CREDENTIALS\_PROVIDER, "com.example.DoesNotExist");

**try** (PrestoS3FileSystem fs = **new** PrestoS3FileSystem()) {

fs.initialize(**new** URI("s3n://test-bucket/"), config);

}

}

[*origin:***apache/hive**](https://github.com/apache/hive/tree/master/hcatalog/core/src/main/java/org/apache/hive/hcatalog/mapreduce/FileOutputFormatContainer.java#L245)

#### **FileOutputFormatContainer.setWorkOutputPath(...)**

**static** **void** setWorkOutputPath(TaskAttemptContext context) **throws** IOException {

String outputPath = context.getConfiguration().[get](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/get)("mapred.output.dir");

*//we need to do this to get the task path and set it for mapred implementation*

*//since it can't be done automatically because of mapreduce->mapred abstraction*

**if** (outputPath != null)

context.getConfiguration().[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)("mapred.work.output.dir",

**new** FileOutputCommitter(**new** [Path](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/%3Cinit%3E)(outputPath), context).getWorkPath().[toString](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/toString)());

}

}

[*origin:***prestodb/presto**](https://github.com/prestodb/presto/tree/master/presto-hive/src/test/java/com/facebook/presto/hive/s3/TestPrestoS3FileSystem.java#L206)

#### **TestPrestoS3FileSystem.testGetMetadataRetryCounter()**

@SuppressWarnings({"OverlyStrongTypeCast", "ConstantConditions"})

@Test

**public** **void** testGetMetadataRetryCounter()

{

**int** maxRetries = 2;

**try** (PrestoS3FileSystem fs = **new** PrestoS3FileSystem()) {

MockAmazonS3 s3 = **new** MockAmazonS3();

s3.setGetObjectMetadataHttpCode(SC\_INTERNAL\_SERVER\_ERROR);

Configuration configuration = **new** [Configuration](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/%3Cinit%3E)();

configuration.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(S3\_MAX\_BACKOFF\_TIME, "1ms");

configuration.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(S3\_MAX\_RETRY\_TIME, "5s");

configuration.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)(S3\_MAX\_CLIENT\_RETRIES, maxRetries);

fs.initialize(**new** URI("s3n://test-bucket/"), configuration);

fs.setS3Client(s3);

fs.getS3ObjectMetadata(**new** [Path](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/%3Cinit%3E)("s3n://test-bucket/test"));

}

**catch** (Throwable expected) {

assertInstanceOf(expected, AmazonS3Exception.**class**);

assertEquals(((AmazonS3Exception) expected).getStatusCode(), SC\_INTERNAL\_SERVER\_ERROR);

assertEquals(PrestoS3FileSystem.getFileSystemStats().getGetMetadataRetries().getTotalCount(), maxRetries);

}

}

[*origin:***apache/hbase**](https://github.com/apache/hbase/tree/master/hbase-common/src/test/java/org/apache/hadoop/hbase/HBaseCommonTestingUtility.java#L157)

#### **HBaseCommonTestingUtility.createSubDir(...)**

**protected** **void** createSubDir(String propertyName, Path parent, String subDirName) {

Path newPath = **new** [Path](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/%3Cinit%3E)(parent, subDirName);

File newDir = **new** File(newPath.[toString](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/toString)()).getAbsoluteFile();

**if** (deleteOnExit()) newDir.deleteOnExit();

conf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(propertyName, newDir.getAbsolutePath());

}

[*origin:***apache/incubator-gobblin**](https://www.tabnine.com/web/assistant/code/rs/5c76b9e049efcb00015363a2#L392)

#### **MRJobLauncher.serializeJobState(...)**

@VisibleForTesting

**static** **void** serializeJobState(FileSystem fs, Path mrJobDir, Configuration conf, JobState jobState, Job job)

**throws** IOException {

Path jobStateFilePath = **new** [Path](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/%3Cinit%3E)(mrJobDir, JOB\_STATE\_FILE\_NAME);

*// Write the job state with an empty task set (work units are read by the mapper from a different file)*

**try** (DataOutputStream dataOutputStream = **new** DataOutputStream(fs.create(jobStateFilePath))) {

jobState.write(dataOutputStream, **false**,

conf.getBoolean(SERIALIZE\_PREVIOUS\_WORKUNIT\_STATES\_KEY, DEFAULT\_SERIALIZE\_PREVIOUS\_WORKUNIT\_STATES));

}

job.getConfiguration().[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(ConfigurationKeys.JOB\_STATE\_FILE\_PATH\_KEY, jobStateFilePath.[toString](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/toString)());

DistributedCache.addCacheFile(jobStateFilePath.toUri(), job.getConfiguration());

job.getConfiguration().[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(ConfigurationKeys.JOB\_STATE\_DISTRIBUTED\_CACHE\_NAME, jobStateFilePath.getName());

}

[*origin:***apache/incubator-druid**](https://github.com/apache/incubator-druid/tree/master/indexing-hadoop/src/main/java/org/apache/druid/indexer/path/StaticPathSpec.java#L126)

#### **StaticPathSpec.addInputPath(...)**

**private** **static** **void** addInputPath(Job job, Iterable<String> pathStrings, Class<? **extends** InputFormat> inputFormatClass)

{

Configuration conf = job.getConfiguration();

StringBuilder inputFormats = **new** StringBuilder(

StringUtils.nullToEmptyNonDruidDataString(conf.[get](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/get)(MultipleInputs.DIR\_FORMATS))

);

String[] paths = Iterables.toArray(pathStrings, String.**class**);

**for** (**int** i = 0; i < paths.length - 1; i++) {

**if** (inputFormats.length() > 0) {

inputFormats.append(',');

}

inputFormats.append(paths[i]).append(';').append(inputFormatClass.getName());

}

**if** (inputFormats.length() > 0) {

conf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(MultipleInputs.DIR\_FORMATS, inputFormats.toString());

}

*// add last one separately for possible initialization in MultipleInputs*

MultipleInputs.addInputPath(job, **new** [Path](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/%3Cinit%3E)(paths[paths.length - 1]), inputFormatClass);

}

[*origin:***apache/hbase**](https://github.com/apache/hbase/tree/master/hbase-server/src/test/java/org/apache/hadoop/hbase/master/procedure/TestServerCrashProcedure.java#L74)

#### **TestServerCrashProcedure.setupConf(...)**

**private** **void** setupConf(Configuration conf) {

conf.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)(MasterProcedureConstants.MASTER\_PROCEDURE\_THREADS, 1);

conf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)("hbase.balancer.tablesOnMaster", "none");

conf.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)(HConstants.HBASE\_CLIENT\_RETRIES\_NUMBER, 3);

conf.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)(HConstants.HBASE\_CLIENT\_SERVERSIDE\_RETRIES\_MULTIPLIER, 3);

conf.[setBoolean](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setBoolean)("hbase.split.writer.creation.bounded", **true**);

conf.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)("hbase.regionserver.hlog.splitlog.writer.threads", 8);

LOG.info("WAL splitting coordinated by zk? {}", splitWALCoordinatedByZK);

conf.[setBoolean](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setBoolean)(HConstants.HBASE\_SPLIT\_WAL\_COORDINATED\_BY\_ZK, splitWALCoordinatedByZK);

}

[*origin:***apache/flink**](https://github.com/apache/flink/tree/master/flink-yarn/src/test/java/org/apache/flink/yarn/YarnFileStageTest.java#L101)

#### **YarnFileStageTest.initConfig()**

@Before

**public** **void** initConfig() {

hadoopConfig = **new** [org.apache.hadoop.conf.Configuration](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/%3Cinit%3E)();

hadoopConfig.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(org.apache.hadoop.fs.FileSystem.FS\_DEFAULT\_NAME\_KEY, hdfsRootPath.[toString](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/toString)());

}

[*origin:***apache/hbase**](https://github.com/apache/hbase/tree/master/hbase-mapreduce/src/test/java/org/apache/hadoop/hbase/snapshot/TestExportSnapshotNoCluster.java#L56)

#### **TestExportSnapshotNoCluster.setUpBaseConf(...)**

**public** **static** **void** setUpBaseConf(Configuration conf) {

conf.[setBoolean](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setBoolean)(SnapshotManager.HBASE\_SNAPSHOT\_ENABLED, **true**);

conf.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)("hbase.regionserver.msginterval", 100);

conf.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)("hbase.client.pause", 250);

conf.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)(HConstants.HBASE\_CLIENT\_RETRIES\_NUMBER, 6);

conf.[setBoolean](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setBoolean)("hbase.master.enabletable.roundrobin", **true**);

conf.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)("mapreduce.map.maxattempts", 10);

conf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)(HConstants.HBASE\_DIR, testDir.[toString](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/toString)());

}

[*origin:***apache/hive**](https://github.com/apache/hive/tree/master/ql/src/java/org/apache/hadoop/hive/llap/LlapCacheAwareFs.java#L63)

#### **LlapCacheAwareFs.registerFile(...)**

**public** **static** Path registerFile(DataCache cache, Path path, Object fileKey,

TreeMap<Long, Long> index, Configuration conf, String tag) **throws** IOException {

**long** splitId = currentSplitId.incrementAndGet();

CacheAwareInputStream stream = **new** CacheAwareInputStream(

cache, conf, index, path, fileKey, -1, tag);

**if** (files.putIfAbsent(splitId, stream) != null) {

**throw** **new** IOException("Record already exists for " + splitId);

}

conf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)("fs." + LlapCacheAwareFs.SCHEME + ".impl", LlapCacheAwareFs.**class**.getCanonicalName());

**return** **new** [Path](https://www.tabnine.com/code/java/methods/org.apache.hadoop.fs.Path/%3Cinit%3E)(SCHEME + "://" + SCHEME + "/" + splitId);

}

[*origin:***apache/hbase**](https://github.com/apache/hbase/tree/master/hbase-common/src/test/java/org/apache/hadoop/hbase/TestCompoundConfiguration.java#L45)

#### **TestCompoundConfiguration.setUp()**

@Before

**public** **void** setUp() **throws** Exception {

baseConf = **new** [Configuration](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/%3Cinit%3E)();

baseConf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)("A", "1");

baseConf.[setInt](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/setInt)("B", 2);

baseConf.[**set**](https://www.tabnine.com/code/java/methods/org.apache.hadoop.conf.Configuration/set)("C", "3");

baseConfSize = baseConf.size();

}