| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/LoginContext.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/security/auth/login/FailedLoginException.html)   [**NEXT CLASS**](http://docs.google.com/javax/security/auth/login/LoginException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/security/auth/login/LoginContext.html)    [**NO FRAMES**](http://docs.google.com/LoginContext.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#26in1rg) |

## **javax.security.auth.login**

Class LoginContext

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.security.auth.login.LoginContext**

public class **LoginContext**extends [Object](http://docs.google.com/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.
3. 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](http://docs.google.com/java/security/Security.html), [AuthPermission](http://docs.google.com/javax/security/auth/AuthPermission.html), [Subject](http://docs.google.com/javax/security/auth/Subject.html), [CallbackHandler](http://docs.google.com/javax/security/auth/callback/CallbackHandler.html), [Configuration](http://docs.google.com/javax/security/auth/login/Configuration.html), [LoginModule](http://docs.google.com/javax/security/auth/spi/LoginModule.html)

| **Constructor Summary** | |
| --- | --- |
| [**LoginContext**](http://docs.google.com/javax/security/auth/login/LoginContext.html#LoginContext(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Instantiate a new LoginContext object with a name. |
| [**LoginContext**](http://docs.google.com/javax/security/auth/login/LoginContext.html#LoginContext(java.lang.String,%20javax.security.auth.callback.CallbackHandler))([String](http://docs.google.com/java/lang/String.html) name, [CallbackHandler](http://docs.google.com/javax/security/auth/callback/CallbackHandler.html) callbackHandler)            Instantiate a new LoginContext object with a name and a CallbackHandler object. |
| [**LoginContext**](http://docs.google.com/javax/security/auth/login/LoginContext.html#LoginContext(java.lang.String,%20javax.security.auth.Subject))([String](http://docs.google.com/java/lang/String.html) name, [Subject](http://docs.google.com/javax/security/auth/Subject.html) subject)            Instantiate a new LoginContext object with a name and a Subject object. |
| [**LoginContext**](http://docs.google.com/javax/security/auth/login/LoginContext.html#LoginContext(java.lang.String,%20javax.security.auth.Subject,%20javax.security.auth.callback.CallbackHandler))([String](http://docs.google.com/java/lang/String.html) name, [Subject](http://docs.google.com/javax/security/auth/Subject.html) subject, [CallbackHandler](http://docs.google.com/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**](http://docs.google.com/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](http://docs.google.com/java/lang/String.html) name, [Subject](http://docs.google.com/javax/security/auth/Subject.html) subject, [CallbackHandler](http://docs.google.com/javax/security/auth/callback/CallbackHandler.html) callbackHandler, [Configuration](http://docs.google.com/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** | |
| --- | --- |
| [Subject](http://docs.google.com/javax/security/auth/Subject.html) | [**getSubject**](http://docs.google.com/javax/security/auth/login/LoginContext.html#getSubject())()            Return the authenticated Subject. |
| void | [**login**](http://docs.google.com/javax/security/auth/login/LoginContext.html#login())()            Perform the authentication. |
| void | [**logout**](http://docs.google.com/javax/security/auth/login/LoginContext.html#logout())()            Logout the Subject. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Constructor Detail** |
| --- |

### LoginContext

public **LoginContext**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [LoginException](http://docs.google.com/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](http://docs.google.com/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](http://docs.google.com/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](http://docs.google.com/java/lang/String.html) name,  
 [Subject](http://docs.google.com/javax/security/auth/Subject.html) subject)  
 throws [LoginException](http://docs.google.com/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](http://docs.google.com/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](http://docs.google.com/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](http://docs.google.com/java/lang/String.html) name,  
 [CallbackHandler](http://docs.google.com/javax/security/auth/callback/CallbackHandler.html) callbackHandler)  
 throws [LoginException](http://docs.google.com/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](http://docs.google.com/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](http://docs.google.com/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](http://docs.google.com/java/lang/String.html) name,  
 [Subject](http://docs.google.com/javax/security/auth/Subject.html) subject,  
 [CallbackHandler](http://docs.google.com/javax/security/auth/callback/CallbackHandler.html) callbackHandler)  
 throws [LoginException](http://docs.google.com/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](http://docs.google.com/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](http://docs.google.com/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](http://docs.google.com/java/lang/String.html) name,  
 [Subject](http://docs.google.com/javax/security/auth/Subject.html) subject,  
 [CallbackHandler](http://docs.google.com/javax/security/auth/callback/CallbackHandler.html) callbackHandler,  
 [Configuration](http://docs.google.com/javax/security/auth/login/Configuration.html) config)  
 throws [LoginException](http://docs.google.com/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](http://docs.google.com/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](http://docs.google.com/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](http://docs.google.com/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](http://docs.google.com/javax/security/auth/login/LoginException.html) - if the authentication fails.

### logout

public void **logout**()  
 throws [LoginException](http://docs.google.com/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](http://docs.google.com/javax/security/auth/login/LoginException.html) - if the logout fails.

### getSubject

public [Subject](http://docs.google.com/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.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/LoginContext.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/security/auth/login/FailedLoginException.html)   [**NEXT CLASS**](http://docs.google.com/javax/security/auth/login/LoginException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/security/auth/login/LoginContext.html)    [**NO FRAMES**](http://docs.google.com/LoginContext.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#26in1rg) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).