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

## **javax.security.auth**

Class Policy

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

**Deprecated.** *as of JDK version 1.4 -- Replaced by java.security.Policy. java.security.Policy has a method:*

*public PermissionCollection getPermissions  
 (java.security.ProtectionDomain pd)*

*and ProtectionDomain has a constructor:*

*public ProtectionDomain  
 (CodeSource cs,  
 PermissionCollection permissions,  
 ClassLoader loader,  
 Principal[] principals)*

*These two APIs provide callers the means to query the Policy for Principal-based Permission entries.*

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

This is an abstract class for representing the system policy for Subject-based authorization. A subclass implementation of this class provides a means to specify a Subject-based access control Policy.

A Policy object can be queried for the set of Permissions granted to code running as a Principal in the following manner:

policy = Policy.getPolicy();  
 PermissionCollection perms = policy.getPermissions(subject,  
 codeSource);

The Policy object consults the local policy and returns and appropriate Permissions object with the Permissions granted to the Principals associated with the provided *subject*, and granted to the code specified by the provided *codeSource*.

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

Each entry in the Policy is represented as a ***grant*** entry. Each ***grant*** entry specifies a codebase, code signers, and Principals triplet, as well as the Permissions granted to that triplet.

grant CodeBase ["URL"], Signedby ["signers"],  
 Principal [Principal\_Class] "Principal\_Name" {  
 Permission Permission\_Class ["Target\_Name"]  
 [, "Permission\_Actions"]  
 [, signedBy "SignerName"];  
 };

The CodeBase and Signedby components of the triplet name/value pairs are optional. If they are not present, then any any codebase will match, and any signer (including unsigned code) will match. For Example,

grant CodeBase "foo.com", Signedby "foo",  
 Principal com.sun.security.auth.SolarisPrincipal "duke" {  
 permission java.io.FilePermission "/home/duke", "read, write";  
 };

This ***grant*** entry specifies that code from "foo.com", signed by "foo', and running as a SolarisPrincipal with the name, duke, has one Permission. This Permission permits the executing code to read and write files in the directory, "/home/duke".

To "run" as a particular Principal, code invokes the Subject.doAs(subject, ...) method. After invoking that method, the code runs as all the Principals associated with the specified Subject. Note that this Policy (and the Permissions granted in this Policy) only become effective after the call to Subject.doAs has occurred.

Multiple Principals may be listed within one ***grant*** entry. All the Principals in the grant entry must be associated with the Subject provided to Subject.doAs for that Subject to be granted the specified Permissions.

grant Principal com.sun.security.auth.SolarisPrincipal "duke",  
 Principal com.sun.security.auth.SolarisNumericUserPrincipal "0" {  
 permission java.io.FilePermission "/home/duke", "read, write";  
 permission java.net.SocketPermission "duke.com", "connect";  
 };

This entry grants any code running as both "duke" and "0" permission to read and write files in duke's home directory, as well as permission to make socket connections to "duke.com".

Note that non Principal-based grant entries are not permitted in this Policy. Therefore, grant entries such as:

grant CodeBase "foo.com", Signedby "foo" {  
 permission java.io.FilePermission "/tmp/scratch", "read, write";  
 };

are rejected. Such permission must be listed in the java.security.Policy.

The default Policy implementation can be changed by setting the value of the "auth.policy.provider" security property (in the Java security properties file) to the fully qualified name of the desired Policy implementation class. 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.

| **Constructor Summary** | |
| --- | --- |
| protected | [**Policy**](http://docs.google.com/javax/security/auth/Policy.html#Policy())()  **Deprecated.** Sole constructor. |

| **Method Summary** | |
| --- | --- |
| abstract  [PermissionCollection](http://docs.google.com/java/security/PermissionCollection.html) | [**getPermissions**](http://docs.google.com/javax/security/auth/Policy.html#getPermissions(javax.security.auth.Subject,%20java.security.CodeSource))([Subject](http://docs.google.com/javax/security/auth/Subject.html) subject, [CodeSource](http://docs.google.com/java/security/CodeSource.html) cs)  **Deprecated.** Retrieve the Permissions granted to the Principals associated with the specified CodeSource. |
| static [Policy](http://docs.google.com/javax/security/auth/Policy.html) | [**getPolicy**](http://docs.google.com/javax/security/auth/Policy.html#getPolicy())()  **Deprecated.** Returns the installed Policy object. |
| abstract  void | [**refresh**](http://docs.google.com/javax/security/auth/Policy.html#refresh())()  **Deprecated.** Refresh and reload the Policy. |
| static void | [**setPolicy**](http://docs.google.com/javax/security/auth/Policy.html#setPolicy(javax.security.auth.Policy))([Policy](http://docs.google.com/javax/security/auth/Policy.html) policy)  **Deprecated.** Sets the system-wide Policy object. |

| **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** |
| --- |

### Policy

protected **Policy**()

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

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

### getPolicy

public static [Policy](http://docs.google.com/javax/security/auth/Policy.html) **getPolicy**()

**Deprecated.** Returns the installed Policy object. This method first calls SecurityManager.checkPermission with the AuthPermission("getPolicy") permission to ensure the caller has permission to get the Policy object.

**Returns:**the installed Policy. The return value cannot be null. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread does not have permission to get the Policy object.**See Also:**[setPolicy(javax.security.auth.Policy)](http://docs.google.com/javax/security/auth/Policy.html#setPolicy(javax.security.auth.Policy))

### setPolicy

public static void **setPolicy**([Policy](http://docs.google.com/javax/security/auth/Policy.html) policy)

**Deprecated.** Sets the system-wide Policy object. This method first calls SecurityManager.checkPermission with the AuthPermission("setPolicy") permission to ensure the caller has permission to set the Policy.

**Parameters:**policy - the new system Policy object. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread does not have permission to set the Policy.**See Also:**[getPolicy()](http://docs.google.com/javax/security/auth/Policy.html#getPolicy())

### getPermissions

public abstract [PermissionCollection](http://docs.google.com/java/security/PermissionCollection.html) **getPermissions**([Subject](http://docs.google.com/javax/security/auth/Subject.html) subject,  
 [CodeSource](http://docs.google.com/java/security/CodeSource.html) cs)

**Deprecated.** Retrieve the Permissions granted to the Principals associated with the specified CodeSource.

**Parameters:**subject - the Subject whose associated Principals, in conjunction with the provided CodeSource, determines the Permissions returned by this method. This parameter may be null.

cs - the code specified by its CodeSource that determines, in conjunction with the provided Subject, the Permissions returned by this method. This parameter may be null. **Returns:**the Collection of Permissions granted to all the Subject and code specified in the provided *subject* and *cs* parameters.

### refresh

public abstract void **refresh**()

**Deprecated.** Refresh and reload the Policy.

This method causes this object to refresh/reload its current Policy. This is implementation-dependent. For example, if the Policy object is stored in a file, calling refresh will cause the file to be re-read.

**Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the caller does not have permission to refresh the Policy.

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

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

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

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