javax.security.jacc
PolicyConfiguration

Declaration

`public interface` **`PolicyConfiguration`**

Description

The methods of this interface are used by containers to create policy statements in a Policy provider. An object that implements the PolicyConfiguration interface provides the policy statement configuration interface for a corresponding policy context within the corresponding Policy provider.

The life cycle of a policy context is defined by three states; “open”, “inService”, and “deleted”. A policy context is in one of these three states.

A policy context in the “open” state is in the process of being configured, and may be operated on by any of the methods of the PolicyConfiguration interface. A policy context in the “open” state must not be assimilated at `Policy.refresh` into the policy statements used by the Policy provider in performing its access decisions. In order for the policy statements of a policy context to be assimilated by the associated provider, the policy context must be in the “inService” state. A policy context in the “open” state is transitioned to the “inService” state by calling the commit method.

A policy context in the “inService” state is available for assimilation into the policy statements being used to perform access decisions by the associated Policy provider. Providers assimilate policy contexts containing policy statements when the refresh method of the provider is called. When a provider’s refresh method is called, it must assimilate only those policy contexts whose state is “inService” and it must ensure that the policy statements put into service for each policy context are only those defined in the context at the time of the call to refresh. A policy context in the “inService” state is not available for additional configuration and may be returned to the “open” state by calling the getPolicyConfiguration method of the PolicyConfigurationFactory.

A policy context in the “deleted” state is neither available for configuration, nor is it available for assimilation into the Provider. A policy context whose state is “deleted” may be reclaimed for subsequent processing by calling the getPolicyConfiguration method of the associated PolicyConfigurationFactory. A “deleted” policy context is transitioned to the “open” state when it it returned as a result of a call to getPolicyConfiguration.

The following table captures the correspondence between the policy context life cycle and the methods of the PolicyConfiguration interface. The rightmost 3 columns of the table correspond to the PolicyConfiguration state identified at the head of the column. The values in the cells of these columns indicate the next state resulting from a call to the method identifed in the leftmost column of the corresponding row, or that calling the method is unsupported in the state represented by the column (in which case the state will remain unchanged).

Method	Current State to Next State		
	deleted	**open**	**inService**
addToExcludedPolicy	Unsupported Operation	open	Unsupported Operation
addToRole	Unsupported Operation	open	Unsupported Operation
addToUncheckedPolicy	Unsupported Operation	open	Unsupported Operation

commit	Unsupported Operation	inService	inService
delete	deleted	deleted	deleted
getContextID	deleted	open	inService
inService	deleted	open	inService
linkConfiguration	Unsupported Operation	open	Unsupported Operation
removeExcludedPolicy	Unsupported Operation	open	Unsupported Operation
removeRole	Unsupported Operation	open	Unsupported Operation
removeUncheckedPolicy	Unsupported Operation	open	Unsupported Operation

For a provider implementation to be compatible with multi-threaded environments, it may be necessary to synchronize the refresh method of the provider with the methods of its PolicyConfiguration interface and with the getPolicyConfiguration and inService methods of its PolicyConfigurationFactory.

See Also: `java.security.Permission`, `java.security.PermissionCollection`, `PolicyContextException`, `PolicyConfigurationFactory`

Member Summary

Methods

void	addToExcludedPolicy(java.security.Permission permission)
void	addToExcludedPolicy(java.security.PermissionCollection permissions)
void	addToRole(java.lang.String roleName, java.security.Permission permission)
void	addToRole(java.lang.String roleName, java.security.PermissionCollection permissions)
void	addToUncheckedPolicy(java.security.Permission permission)
void	addToUncheckedPolicy(java.security.PermissionCollection permissions)
void	commit()
void	delete()
java.lang.String	getContextID()
boolean	inService()
void	linkConfiguration(PolicyConfiguration link)
void	removeExcludedPolicy()
void	removeRole(java.lang.String roleName)
void	removeUncheckedPolicy()

Methods

addToExcludedPolicy(java.security.PermissionCollection permissions)

```
public void addToExcludedPolicy(java.security.PermissionCollection permissions)
            throws PolicyContextException
```

addToExcludedPolicy(`java.security.Permission` permission)

Used to add excluded policy statements to this PolicyConfiguration.

Parameters:

`permissions` - the collection of permissions to be added to the excluded policy statements. The collection may be either a homogenous or heterogenous collection.

Throws:

`java.lang.SecurityException` - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

`java.lang.UnsupportedOperationException` - if the state of the policy context whose interface is this PolicyConfiguration Object is "deleted" or "inService" when this method is called.

`PolicyContextException` - if the implementation throws a checked exception that has not been accounted for by the addToExcludedPolicy method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

addToExcludedPolicy(`java.security.Permission` permission)

```
public void addToExcludedPolicy(java.security.Permission permission)
            throws PolicyContextException
```

Used to add a single excluded policy statement to this PolicyConfiguration.

Parameters:

`permission` - the permission to be added to the excluded policy statements.

Throws:

`java.lang.SecurityException` - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

`java.lang.UnsupportedOperationException` - if the state of the policy context whose interface is this PolicyConfiguration Object is "deleted" or "inService" when this method is called.

`PolicyContextException` - if the implementation throws a checked exception that has not been accounted for by the addToExcludedPolicy method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

addToRole(`java.lang.String` roleName, `java.security.PermissionCollection` permissions)

```
public void addToRole(java.lang.String roleName, java.security.PermissionCollection
            permissions)
            throws PolicyContextException
```

Used to add permissions to a named role in this PolicyConfiguration. If the named ~~Role~~ role does not exist in the PolicyConfiguration, it is created as a result of the call to this function.

It is the job of the Policy provider to ensure that all the permissions added to a role are granted to principals "mapped to the role".

Parameters:

`roleName` - the name of the Role to which the permissions are to be added.

`permissions` - the collection of permissions to be added to the role. The collection may be either a homogenous or heterogenous collection.

addToRole(java.lang.String roleName, java.security.Permission permission)

Throws:

java.lang.SecurityException - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

java.lang.UnsupportedOperationException - if the state of the policy context whose interface is this PolicyConfiguration Object is "deleted" or "inService" when this method is called.

PolicyContextException - if the implementation throws a checked exception that has not been accounted for by the addToRole method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

addToRole(java.lang.String roleName, java.security.Permission permission)

```
public void addToRole(java.lang.String roleName, java.security.Permission permission)
            throws PolicyContextException
```

Used to add a single permission to a named role in this PolicyConfiguration. If the named ~~Role~~ role does not exist in the PolicyConfiguration, it is created as a result of the call to this function.

It is the job of the Policy provider to ensure that all the permissions added to a role are granted to principals "mapped to the role".

Parameters:

roleName - the name of the Role to which the permission is to be added.

permission - the permission to be added to the role.

Throws:

java.lang.SecurityException - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

java.lang.UnsupportedOperationException - if the state of the policy context whose interface is this PolicyConfiguration Object is "deleted" or "inService" when this method is called.

PolicyContextException - if the implementation throws a checked exception that has not been accounted for by the addToRole method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

addToUncheckedPolicy(java.security.PermissionCollection permissions)

```
public void addToUncheckedPolicy(java.security.PermissionCollection permissions)
            throws PolicyContextException
```

Used to add unchecked policy statements to this PolicyConfiguration.

Parameters:

permissions - the collection of permissions to be added as unchecked policy statements. The collection may be either a homogenous or heterogenous collection.

Throws:

java.lang.SecurityException - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

java.lang.UnsupportedOperationException - if the state of the policy context whose interface is this PolicyConfiguration Object is "deleted" or "inService" when this method is called.

PolicyContextException - if the implementation throws a checked exception that has not been accounted for by the addToUncheckedPolicy method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

addToUncheckedPolicy(java.security.Permission permission)

```
public void addToUncheckedPolicy(java.security.Permission permission)
            throws PolicyContextException
```

Used to add a single unchecked policy statement to this PolicyConfiguration.

Parameters:

`permission` - the permission to be added to the unchecked policy statements.

Throws:

`java.lang.SecurityException` - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

`java.lang.UnsupportedOperationException` - if the state of the policy context whose interface is this PolicyConfiguration Object is "deleted" or "inService" when this method is called.

`PolicyContextException` - if the implementation throws a checked exception that has not been accounted for by the addToUncheckedPolicy method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

commit()

```
public void commit()
            throws PolicyContextException
```

This method is used to set to "inService" the state of the policy context whose interface is this PolicyConfiguration Object. Only those policy contexts whose state is "inService" will be included in the policy contexts processed by the Policy.refresh method. A policy context whose state is "inService" may be returned to the "open" state by calling the getPolicyConfiguration method of the PolicyConfiguration factory with the policy context identifier of the policy context.

When the state of a policy context is "inService", calling any method other than commit, delete, getContextID, or inService on its PolicyConfiguration Object will cause an UnsupportedOperationException to be thrown.

Throws:

`java.lang.SecurityException` - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

`java.lang.UnsupportedOperationException` - if the state of the policy context whose interface is this PolicyConfiguration Object is "deleted" when this method is called.

`PolicyContextException` - if the implementation throws a checked exception that has not been accounted for by the commit method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

delete()

```
public void delete()
            throws PolicyContextException
```

Causes all policy statements to be deleted from this PolicyConfiguration and sets its internal state such that calling any method, other than delete, getContextID, or inService on the PolicyConfiguration will be rejected and cause an UnsupportedOperationException to be thrown.

This operation has no affect on any linked PolicyConfigurations other than removing any links involving the deleted PolicyConfiguration.

Throws:

`java.lang.SecurityException` - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

`PolicyContextException` - if the implementation throws a checked exception that has not been accounted for by the delete method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

getContextID()

```
public java.lang.String getContextID()
            throws PolicyContextException
```

This method returns this object's policy context identifier.

Returns: this object's policy context identifier.

Throws:

`java.lang.SecurityException` - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

`PolicyContextException` - if the implementation throws a checked exception that has not been accounted for by the getContextID method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

inService()

```
public boolean inService()
            throws PolicyContextException
```

This method is used to determine if the policy context whose interface is this PolicyConfiguration Object is in the "inService" state.

Returns: true if the state of the associated policy context is "inService"; false otherwise.

Throws:

`java.lang.SecurityException` - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

`PolicyContextException` - if the implementation throws a checked exception that has not been accounted for by the inService method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

linkConfiguration(PolicyConfiguration link)

```
public void linkConfiguration(PolicyConfiguration link)
            throws PolicyContextException
```

Creates a relationship between this configuration and another such that they share the same principal-to-role mappings. PolicyConfigurations are linked to apply a common principal-to-role mapping to multiple seperately manageable PolicyConfigurations, as is required when an application is composed of multiple modules.

Note that the policy statements which comprise a role, or comprise the excluded or unchecked policy collections in a PolicyConfiguration are unaffected by the configuration being linked to another.

Parameters:

`link` - a reference to a different PolicyConfiguration than this PolicyConfiguration.

The relationship formed by this method is symetric, transitive and idempotent. If the argument PolicyConfiguration does not have a different Policy context identifier than this PolicyConfiguration no relationship is formed, and an exception, as described below, is thrown.

Throws:

`java.lang.SecurityException` - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

`java.lang.UnsupportedOperationException` - if the state of the policy context whose interface is this PolicyConfiguration Object is "deleted" or "inService" when this method is called.

`java.lang.IllegalArgumentException` - if called with an argument PolicyConfiguration whose Policy context is equivalent to that of this PolicyConfiguration.

`PolicyContextException` - if the implementation throws a checked exception that has not been accounted for by the linkConfiguration method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

removeExcludedPolicy()

```
public void removeExcludedPolicy()
            throws PolicyContextException
```

Used to remove any excluded policy statements from this PolicyConfiguration. This method has no effect on the links between this PolicyConfiguration and others.

Throws:

`java.lang.SecurityException` - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

`java.lang.UnsupportedOperationException` - if the state of the policy context whose interface is this PolicyConfiguration Object is "deleted" or "inService" when this method is called.

`PolicyContextException` - if the implementation throws a checked exception that has not been accounted for by the removeExcludedPolicy method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

removeRole(`java.lang.String` roleName)

```
public void removeRole(java.lang.String roleName)
            throws PolicyContextException
```

Used to remove a role and all its permissions from this PolicyConfiguration. This method has no effect on the links between this PolicyConfiguration and others.

Parameters:

`roleName` - the name of the ~~Role~~ role to remove from this PolicyConfiguration. If the value of the roleName parameter is "*" and no role with name "*" exists in this PolicyConfiguration, then all roles must be removed from this PolicyConfiguration.

Throws:

`java.lang.SecurityException` - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

`java.lang.UnsupportedOperationException` - if the state of the policy context whose interface is this PolicyConfiguration Object is "deleted" or "inService" when this method is called.

PolicyContextException - if the implementation throws a checked exception that has not been accounted for by the removeRole method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

removeUncheckedPolicy()

```
public void removeUncheckedPolicy()
            throws PolicyContextException
```

Used to remove any unchecked policy statements from this PolicyConfiguration. This method has no effect on the links between this PolicyConfiguration and others.

Throws:

java.lang.SecurityException - if called by an AccessControlContext that has not been granted the "setPolicy" SecurityPermission.

java.lang.UnsupportedOperationException - if the state of the policy context whose interface is this PolicyConfiguration Object is "deleted" or "inService" when this method is called.

PolicyContextException - if the implementation throws a checked exception that has not been accounted for by the removeUncheckedPolicy method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.