# javax.security.jacc
# PolicyConfigurationFactory

```
java.lang.Object
  |
  +--javax.security.jacc.PolicyConfigurationFactory
```

### Declaration

```
public abstract class PolicyConfigurationFactory
```

### Description

Abstract factory and finder class for obtaining the instance of the class that implements the PolicyConfigurationFactory of a provider. The factory will be used to instantiate PolicyConfiguration objects that will be used by the deployment tools of the container to create and manage policy contexts within the Policy Provider.

Implementation classes must have a public no argument constructor that may be used to create an operational instance of the factory implementation class.

**See Also:** `java.security.Permission`, `PolicyConfiguration`, `PolicyContextException`

| Member Summary | |
|---|---|
| **Constructors** | |
| | `PolicyConfigurationFactory()` |
| **Methods** | |
| abstract `PolicyConfiguration` | `getPolicyConfiguration(java.lang.String contextID, boolean remove)` |
| static `PolicyConfigurationFactory` | `getPolicyConfigurationFactory()` |
| abstract boolean | `inService(java.lang.String contextID)` |

| Inherited Member Summary |
|---|
| **Methods inherited from class `Object`** |
| `clone()`, `equals(Object)`, `finalize()`, `getClass()`, `hashCode()`, `notify()`, `notifyAll()`, `toString()`, `wait()`, `wait(long)`, `wait(long, int)` |

## Constructors

### PolicyConfigurationFactory()

```
public PolicyConfigurationFactory()
```

# Methods

### getPolicyConfiguration(java.lang.String contextID, boolean remove)

```
public abstract PolicyConfiguration getPolicyConfiguration(java.lang.String contextID,
            boolean remove)
            throws PolicyContextException
```

This method is used to obtain an instance of the provider specific class that implements the PolicyConfiguration interface that corresponds to the identified policy context within the provider. The methods of the PolicyConfiguration interface are used to define the policy statements of the identified policy context.

If at the time of the call, the identified policy context does not exist in the provider, then the policy context will be created in the provider and the Object that implements the context's PolicyConfiguration Interface will be returned. If the state of the identified context is "deleted" or "inService" it will be transitioned to the "open" state as a result of the call. The states in the lifecycle of a policy context are defined by the PolicyConfiguration interface.

For a given value of policy context identifier, this method must always return the same instance of PolicyConfiguration and there must be at most one actual instance of a PolicyConfiguration with a given policy context identifier (during a process context).

To preserve the invariant that there be at most one PolicyConfiguration object for a given policy context, it may be necessary for this method to be thread safe.

**Parameters:**

`contextID` - A String identifying the policy context whose PolicyConfiguration interface is to be returned. The value passed to this parameter must not be null.

`remove` - A boolean value that establishes whether or not the policy statements and linkages of an existing policy context are to be removed before its PolicyConfiguration object is returned. If the value passed to this parameter is true, the policy statements and linkages of an existing policy context will be removed. If the value is false, they will not be removed.

**Returns:** an Object that implements the PolicyConfiguration Interface matched to the Policy provider and corresponding to the identified policy context.

**Throws:**

`java.lang.SecurityException` - when 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 getPolicyConfiguration method signature. The exception thrown by the implementation class will be encapsulated (during construction) in the thrown PolicyContextException.

### getPolicyConfigurationFactory()

```
public static PolicyConfigurationFactory getPolicyConfigurationFactory()
            throws ClassNotFoundException, PolicyContextException
```

This static method uses a system property to find and instantiate (via a public constructor) a provider specific factory implementation class. The name of the provider specific factory implementation class is obtained from the value of the system property,

`javax.security.jacc.PolicyConfigurationFactory.provider`.

**Returns:** the singleton instance of the provider specific PolicyConfigurationFactory implementation class.

**Throws:**

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

`java.lang.ClassNotFoundException` - when the class named by the system property could not be found including because the value of the system property has not be set.

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

### inService(`java.lang.String` contextID)

```
public abstract boolean inService(java.lang.String contextID)
            throws PolicyContextException
```

This method determines if the identified policy context exists with state "inService" in the Policy provider associated with the factory.

**Parameters:**

`contextID` - A string identifying a policy context

**Returns:** true if the identified policy context exists within the provider and its state is "inService", false otherwise.

**Throws:**

`java.lang.SecurityException` - when 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.