The declarative authorization policy statements derived from the application or module deployment descriptor(s) must be translated to create instances of the corresponding `javax.security.jacc` Permission classes.

```
WebResourcePermission webPerm =
    new WebResourcePermission("/elephant","GET");
```

Methods of the `PolicyConfiguration` interface must be used with the permissions resulting from the translation to create policy statements within the `PolicyConfiguration` objects.

```
petPC.addToRole("customer",webPerm);
```

The `PolicyConfiguration` objects must be linked such that the same principal-to-role mapping will be applied to all the modules of the application.

```
petPC.linkConfiguration(petFoodPC);
```

The PolicyConfiguration objects must be placed in Service such that they will be assimilated into the Policy providers used by the containers to which the application has been deployed.

```
petPC.commit();
```

Independent of this specification, J2EE deployment tools must translate and complete the declarative policy statements appearing in deployment descriptors into a form suitable for securing applications on the platform. On versions of the Java EE platform that require support for authorization policy annotations, the deployment tools must combine policy annotations in Java code with policy statements appearing in deployment descriptors to yield complete representations of authorization policy suitable for securing applications on the platform. The rules for combining authorization policy annotations with declarative policy statements are described in the versions of the EJB, Servlet, and Java EE platform specifications that require support for the annotations. Independent of whether annotations factor in the translation, the resulting policy statements may differ in form from the policy statements appearing in the deployment descriptors. The policy translation defined by this subcontract is described assuming that the policy statement form used by a platform is identical to that used to express policy in the deployment descriptors. Where this is not the case, the output of the translation must be equivalent to the translation that would occur if policy was completely specified in the deployment descriptors and the translation had proceeded directly from the deployment descriptors to the Java SE policy forms defined by this subcontract. Two translations are equivalent if they produce corresponding collections of unchecked, excluded, and role permissions, and if all of the

permissions of each such collection are implied[1] by the permissions of the corresponding or excluded collection of the other translation. Translation equivalence is only required with respect to the permission types that are the subject of the translation.

3.1.1 Policy Contexts and Policy Context Identifiers

It must be possible to define separate authorization policy contexts corresponding to each deployed instance of a Java EE module. This per module scoping of policy context is necessary to provide for the independent administration of policy contexts corresponding to individual application modules (perhaps multiply deployed) within a common Policy provider.

Each policy context contains all of the policy statements (as defined by this specification) that effect access to the resources in one or more deployed modules. At policy configuration, a `PolicyConfiguration` object is created for each policy context, and populated with the policy statements (represented by permission objects) corresponding to the context. Each policy context has an associated policy context identifier.

In the Policy Decision and Enforcement Subcontract, access decisions are performed by checking permissions that identify resources by name and perhaps action. When a permission is checked, this specification requires identification of the authorization policy context in which the evaluation is to be performed (see Section 4.6, “Setting the Policy Context,” on page 50).

3.1.1.1 Policy Context Life Cycle

Figure 3.1 depicts the policy context life cycle as effected through the methods of the PolicyConfiguration interface. A policy context is in one of three states and all implementations of the PolicyConfiguration interface must implement the state semantics defined in this section.

- open

  A policy context in the open state must be available for configuration 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.

---

[1] For some permission types, such as the EJBMethodPermission, it will generally not be possible to use the implies method of the PermissionCollection to compute collection equivalence (because the implies method is unable to determine when a collection contains all the permissions implied by a wild carded form of the permission).