`security-role-ref` elements, an EJBRoleRefPermission must be added to each `security-role` of the application whose name does not appear as the `role-name` in a `security-role-ref` within the element. The name of each such EJBRoleRefPermission must be the value of the `ejb-name` element within the element in which the `security-role-ref` elements could otherwise occur. The actions (that is, reference) of each such EJBRoleRefPermission must be the corresponding (non-appearing) `role-name`. The resulting permissions must be added[13] to the corresponding roles by calling the `addToRole` method on the `PolicyConfiguration` object.

3.1.6 Deploying an Application or Module

The application server's deployment tools must translate the declarative authorization policy appearing in the application or module deployment descriptor(s) into policy statements within the Policy providers used by the containers to which the components of the application or module are being deployed. In Servlet 3.0 containers, the policy statements resulting from the deployment and initialization of a web module, must represent the effects of any programmatic registration and security configuration of servlets that occurred during the initialization of the module.

When a module is deployed, its policy context must be linked to all the other policy contexts with which it must share the same principal-to-role mapping. When an application is deployed, every policy context of the application must be linked to every other policy context of the application with which it shares a common Policy provider. Policy contexts are linked[14] by calling the linkConfiguration method on the PolicyConfiguration objects of the provider.

After the translation and linking has occurred (note that they may occur in either order) for a policy context, the `commit` method must be called on the corresponding `PolicyConfiguration` object to place it in service such that its policy statements will be assimilated by the corresponding Policy providers.

---

13. For example, if an application declares roles {R1, R2, R3} and defines a session EJB named "shoppingCart" that contains one `security-role-ref` element with `role-name` R1, then an additional EJBRoleRefPermission must be added to each of the roles R2 and R3. The name of both permissions must be "shoppingCart", and the actions value of the permission added to role R2 must be "R2", and the actions value of the permission added to role R3 must be "R3".
14. Policy context linking is transitive and symmetric, and this specification should not be interpreted as requiring that linkConfiguration be called on every combination of policy contexts that must share the same principal-to-role mapping, or that all contexts must be linked before any can be committed.

These three operations, translate, link and commit, must be performed for all of the policy contexts of the application.

Once the translation, linking, and committing has occurred, a call must be made to `Policy.refresh` on the Policy provider used by each of the containers to which the application or module is being deployed. The calls to `Policy.refresh` must occur before the containers will accept requests for the deployed resources. If a module corresponding to a policy context may have inter-module, initialization-time, dependencies that must be satisfied before the translation of the policy context of the dependent module can be completed[15], the `commit` of the depended upon modules must occur before the initialization of the dependent module, and the calls to `Policy.refresh` described above must additionally occur after the processing of the depended upon modules and before the initialization of the dependent module.

The policy context identifiers corresponding to the deployed application or module must be recorded in the application server so that they can be used by containers to establish the policy context as required by Section 4.6, "Setting the Policy Context" of the Policy Decision and Enforcement Subcontract, and such that the Deployer may subsequently remove or modify the corresponding policy contexts as a result of the undeployment or redeployment of the application.

3.1.7 Undeploying an Application or Module

To ensure that there is not a period during undeployment when the removal of policy statements on application components renders what were protected components unprotected, the application server must stop dispatching requests for the application's components before undeploying an application or module.

To undeploy an application or module, the deployment tools must indicate at all the Policy providers to which policy contexts of the application or module have been deployed that the policy contexts associated with the application or module that have been configured in the provider are to be removed from service. A deployment tool indicates that a policy context is to be removed from service either by calling getPolicyConfiguration with the identifier of the policy context on the provider's PolicyConfigurationFactory or by calling delete on the corresponding PolicyConfiguration object. If the getPolicyConfiguration method is used, the value `true` should be passed as the second argument to cause the corresponding policy statements to be deleted from the context. After the policy

---

[15] Such as having a Servlet 3.0 ServletContextListener configured that could programmatically register a servlet and configure its security constraints and that could also perform a local invocation of an EJB in another module of the application.