3.1.5.1 Translating EJB method-permission Elements

For each method element of each method-permission element, an EJBMethodPermission object translated from the method element must be added to the policy statements of the PolicyConfiguration object. The name of each such EJBMethodPermission object must be the ejb-name from the corresponding method element, and the actions must be established by translating the method element into a method specification according to the methodSpec syntax defined in the documentation of the EJBMethodPermission class. The actions translation must preserve the degree of specificity with respect to method-name, method-intf, and method-params inherent in the method element.

If the method-permission element contains the unchecked element, then the deployment tools must call the addToUncheckedPolicy method to add the permissions resulting from the translation to the PolicyConfiguration object. Alternatively, if the method-permission element contains one or more role-name elements, then the deployment tools must call the addToRole method to add the permissions resulting from the translation to the corresponding roles of the PolicyConfiguration object.

3.1.5.2 Translating the EJB exclude-list

An EJBMethodPermission object must be created for each method element occurring in the exclude-list element of the deployment descriptor. The name and actions of each EJBMethodPermission must be established as described in Section 3.1.5.1, “Translating EJB method-permission Elements.”

The deployment tools must use the addToExcludedPolicy method to add the EJBMethodPermission objects resulting from the translation of the exclude-list to the excluded policy statements of the PolicyConfiguration object.

3.1.5.3 Translating EJB security-role-ref Elements

For each security-role-ref element appearing in the deployment descriptor, a corresponding EJBRoleRefPermission must be created. The value of the ejb-name element within the element containining the security-role-ref element must be used as the name of the EJBRoleRefPermission. The actions used to construct the permission must be the value of the role-name (that is the reference), appearing in the security-role-ref. The deployment tools must call the addToRole method on the PolicyConfiguration object to add a policy statement corresponding to the EJBRoleRefPermission to the role identified in the `role-link` appearing in the `security-role-ref`.

Additional EJBRoleRefPermission objects must be added to the `PolicyConfiguration` as follows. For each element in the deployment descriptor for which the descriptor schema supports[11] inclusion of `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 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.

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[12] by calling the linkConfiguration method on the PolicyConfiguration objects of the provider.

After the `PolicyConfiguration` objects are linked, the `commit` method must be called on all the `PolicyConfiguration` objects to place them in service such that their policy statements will be assimilated by the corresponding Policy providers.

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

[11] EJB 3.0 supports incusion of `security-role-ref` elements in `entity` and `session` elements. Future versions could support inclusion in `message-driven`.

[12] 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.