If an application server or JRE employs a custom SecurityManager, the necessary reliance on Policy object may be accomplished by ensuring that the custom SecurityManager relies on the appropriate (as defined above) Policy object for all of the policy decisions defined by this contract.

4.12 Optimization of Permission Evaluations

Containers may employ the following optimizations (based on reuse) when the result obtained by repeating the evaluation will not differ from the previous result or when the time since the previous evaluation is less than the container's threshold for being affected by policy changes:

- Containers may reuse an authorization result obtained from a previous equivalent permission evaluation.
- Containers may reuse an authorization result obtained for an unauthenticated caller (i.e. a caller with no principals) performed as defined in Section 4.8, "Checking the Caller for a Permission" to grant, independent of caller identity, any permission implied by the unauthenticated result.

This specification does not prescribe how a container determines when a repeated evaluation will return the same result. That said, one way that containers could make this determination is if they are, and can determine if they will be, notified of policy changes and if they can establish that their policy provider does not employ additional context (such as could be acquired by calling a `PolicyContextHandler`) in its policy evaluations.

Common practice for containers to receive such notification could be for them to register to the `"java.security.Policy.supportsReuse"` key a `PolicyContextHandler` and for the container to determine if its provider will notify it of policy changes by making a test call to the provider's `refresh` method. Only a provider that is compatible with the optimizations described above (including because it does not employ additional context in its policy evaluations) may deliver notice of policy changes by activating this handler when its `refresh` method is called.