

for mapping the target name or address information of an HTTP request to the appropriate hostname.

To satisfy this specification, an application server must establish servlet policy context identifiers sufficient to differentiate all instances of a web application deployed on the logical host or on any other logical host that may share the same policy statement repository. One way to satisfy this requirement is to compose policy context identifiers by concatenating the hostname with the context path (as defined in the Servlet specification) identifying the web application at the host.

When an application is composed of multiple web modules, a separate policy context must be defined per module. This is necessary to ensure that url-pattern based and servlet name based policy statements configured for one module do not interfere with those configured for another.

In Servlet containers that support the programmatic registration and security configuration of servlets (e.g., Servlet 3.0 compatible Servlet containers), the policy contexts assigned to web applications and web modules must be distinct from those to which any EJB³ components are assigned.

3.1.3 Translating Servlet Deployment Descriptors

A reference to a `PolicyConfiguration` object must be obtained by calling the `getPolicyConfiguration` method on the `PolicyConfigurationFactory` implementation class of the provider configured into the container. The policy context identifier used in the call to the `getPolicyConfiguration` method must be a `String` composed as described in Section 3.1.2, “Servlet Policy Context Identifiers,” on page 23. The `security-constraint` and `security-role-ref` elements in the deployment descriptor must be translated into permissions and added to the `PolicyConfiguration` object as defined in the following sections. Before the translation is performed, all policy statements must have been removed⁴ from the policy context associated with the returned `PolicyConfiguration`.

3.1.3.1 Programmatic Servlet Registrations

In Servlet containers that support the programmatic registration and security configuration of servlets (e.g., Servlet 3.0 compatible Servlet containers), the

³ See Section 3.1.4, “EJB Policy Context Identifiers” for further clarification.

⁴ This can be achieved by passing `true` as the second parameter in the call to `getPolicyConfiguration`, or by calling `delete` on the `PolicyConfiguration` before calling `getPolicyConfiguration` to transition it to the open state.

Servlet policy translation defined by this subcontract is described assuming that all such programmatic registration and security configuration has completed (for the servlet module corresponding to the policy context) before the translation is performed and that the resulting security related configuration has been represented in its equivalent form within the deployment descriptors on which the translation is performed. Where this is not the case, the result of the translation must be equivalent, as described previously, to the translation that would occur if it was the case. The mapping to equivalent deployment descriptor representation of security related configuration corresponding to programmatically registered servlets is defined in the Servlet 3.0 specification.

If the results of a prior translation are invalidated by subsequent programmatic registration and security configuration (as might occur if an initial translation is performed before the programmatic effects), the translation must be repeated. Before the translation is repeated, a reference must be obtained to the `PolicyConfiguration` object in the *open* state, and its policy statements must be removed. If the `PolicyConfiguration` has already been linked to other committed policy contexts, then it may be necessary or preferable (in order to satisfy the linking requirements defined in Section 3.1.6, “Deploying an Application or Module”) to obtain the reference and remove the policy statements while preserving the linkages established for the context by the prior translation. Policy statements may be removed while preserving linkages by calling the `removeUncheckedPolicy`, `removeExcludedPolicy`, and `removeRole` methods on the open `PolicyConfiguration` object.

3.1.3.2 Translating security-constraint Elements

The paragraphs of this section describe the translation of security-constraints into `WebResourcePermission` and `WebUserDataPermission` objects constructed using qualified URL pattern names. In the exceptional case, as defined in “Qualified URL Pattern Names”, where a pattern is made irrelevant by a qualifying pattern, the permission instantiations that would result from the translation of the pattern, as described below, must not be performed. Otherwise, the translation of URL patterns in security constraints must yield an equivalent translation to the translation that would result from following the instructions in the remainder of this section.

A `WebResourcePermission` and a `WebUserDataPermission`⁵ object must be added to the excluded policy statements for each distinct `url-pattern`

⁵ The `WebUserDataPermission` objects allow a container to determine when to reject a request before redirection if it would ultimately be rejected as the result of an excluding `auth-constraint`.