NativeSPRequestMapper

The `<RequestMapper>` element configures the component used by the SP to map incoming requests to the set of configuration options that should be applied. You can think of it as a portable equivalent of the Apache `<Location>` feature, which associates Apache directives with specific URLs. Like that feature, the request mapper operates based on URL, not on the physical path of files.

Another critical role of the request mapper is to map each request to the appropriate Shibboleth application, thereby connecting the resource-oriented settings to the more general behavior defined per-application.

For a general overview with examples, see the request mapper HOWTO.

While Apache-based deployments can rely entirely on Apache functionality for this purpose, IIS provides no such capability. The request mapper allows the SP to insulate itself from the differences between servers.

On Version 2.4 and above, omitting this element will result in a "Native" plugin with an empty/default configuration. This empty configuration maps all requests to default application settings and adds no other settings, unless overridden by web server-specific options.

Common Attributes

- `type`(string)
    - Plugin type name.

Native Request Mapper

Identified by `type="Native"`, integrates native web server content configuration features with the portable syntax supported by the XML request mapper (see below).

For most deployments, this is the type to use. It is a hybrid that allows you to combine Apache commands in `.htaccess` files with XML-based configuration. The native commands override any XML-based attributes. For servers without native commands (IIS), this type is equivalent to the XML request mapper type below.

The Native request mapper's XML "portion" is a reloadable resource, which means that the XML content can be supplied inline, in a local file, or a remote file, and can be monitored for changes and reloaded on the fly. The root of the XML instance **MUST** be a `<RequestMap>` element.

```
<RequestMapper type="Native">
    <RequestMap applicationId="default"/>
</RequestMapper>
```

Attributes

Inherits attributes supported by reloadable resources.

Child Elements

- `<RequestMap>`
    - Root element of configuration, can be supplied inline as a child element, or as the root of a reloadable resource in an external file.

XML Request Mapper

Identified by `type="XML"`, provides a portable XML syntax for configuring settings based on mappings assigned to URL host, path, and query string information. It does not permit settings to be defined in the web server's configuration.

The only reason for using this type in favor of the "Native" type above would be to prevent developers with access to content from using `.htaccess` commands. Since Apache itself can prevent this already, it's largely superfluous, but there might be very small efficiency gains from using it in such cases.

The XML request mapper is a reloadable resource, which means that the XML content can be supplied inline, in a local file, or a remote file, and can be monitored for changes and reloaded on the fly. The root of the XML instance **MUST** be a `<RequestMap>` element.

```
<RequestMapper type="XML">
    <RequestMap applicationId="default"/>
</RequestMapper>
```

Attributes

Inherits attributes supported by reloadable resources.

Child Elements

- `<RequestMap>`
  - Root element of configuration, can be supplied inline as a child element, or as the root of a reloadable resource in an external file.

Properties

The general function of the request mapper is to associate "properties" with a request. These properties are named and typed and provide the basic interface between the configuration of the software and the software itself. Obviously similar to XML attributes, these properties may in fact be expressed with XML, but do not have to be.

The set of supported properties is as follows:

- `authType` (string)
  - Analagous to Apache's `AuthType` command, just set this to "shibboleth" unless you want to bypass any processing by the SP.
- `applicationId` (string)
  - Overrides the application associated with the resource by matching the `id` attribute in an `<ApplicationOveride>` element.
- `requireSession` (boolean) (defaults to false)
  - Master trigger that will require an authenticated session. If none exists, the SP will try to automatically establish one using the default SessionInitiator.
- `requireSessionWith` (string)
  - Same as `requireSession`, but uses the SessionInitiator with the specified `id` attribute instead of the default.
- `exportAssertion` (boolean) (defaults to false)
  - If true, special attributes are exported to provide applications with access to the underlying SAML assertions that are cached with the user's session. See the NativeSPAssertionExport topic.
- `redirectToSSL` (integer)
  - A port to redirect non-SSL GET or HEAD requests to. Other HTTP methods like POST will result in an error. Used to automate the blocking of non-SSL requests in a portable way. Most servers can do this anyway, but some like IIS won't enforce the rule until it's too late to prevent problems with the Shibboleth filter.
- `entityID` (URI)
  - The name of a specific IdP to use when automatically requesting authentication because a session does not exist. Allows for resource-based selection of an IdP to use, and overrides the `entityID` attribute of a SessionInitiator.
- `isPassive` (boolean) (defaults to false)
  - Sets the value of the `IsPassive` attribute of any SAML 2.0 AuthnRequest messages issued automatically as a result of accessing the resource. Has no effect for other SSO protocols. Overrides the `isPassive` attribute of a SessionInitiator. Also have a look at the page in order to see what isPassive can be used for.
- `forceAuthn` (boolean) (defaults to false)
  - Sets the value of the `ForceAuthn` attribute of any SAML 2.0 AuthnRequest messages issued automatically as a result of accessing the resource. This asks for forced reauthentication by the IdP (bypassing SSO). Has no effect for other SSO protocols. Overrides the `forceAuthn` attribute of a SessionInitiator.
- `authnContextClassRef` (URI)
  - Specifies a SAML 2.0 AuthnContext class reference to request in any SAML 2.0 AuthnRequest messages issued automatically as a result of accessing the resource. Has no effect for other SSO protocols. Overrides the `authnContextClassRef` attribute of a SessionInitiator. As of V2.5, this can be a whitespace-delimited list of classes to request.
- `authnContextComparison` ("exact", "better", "minimum", "maximum")
  - Specifies the SAML 2.0 AuthnContext comparison operator to use in any SAML 2.0 AuthnRequest messages issued automatically as a result of accessing the resource. Has no effect for other SSO protocols. Overrides the `authnContextComparison` attribute of a SessionInitiator.
- `redirectErrors` (absolute URL, relative URL supported in V2.5+)
  - Location to redirect to when errors occur, instead of using a generated HTML template. Particularly necessary when using passive SSO. Overrides the `redirectErrors` attribute of the `<Errors>` element.
- `sessionError` (local pathname)
  - Error template to use for general processing errors. Overrides the `sessionError` attribute of the `<Errors>` element.
- `metadataError` (local pathname)
  - Error template to use for metadata-related errors. Overrides the `metadataError` attribute of the `<Errors>` element.
- `accessError` (local pathname)
  - Error template to use for authorization failures. Overrides the `accessError` attribute of the `<Errors>` element.
- `sslError` (local pathname)
  - Error template to use for blocking non-SSL requests that could not be redirected. Overrides the `sslError` attribute of the `<Errors>` element.
- `REMOTE_ADDR` (string) (Version 2.2 and Above)

    - Optional name of an HTTP request header to use for the IP address of the client. Used to divert this lookup from the REMOTE_ADDR variable to a header set by a proxy, such as "X-Forwarded-For". If you rely on this feature, you'd better ensure that the header can't be spoofed by a client.

- `target` (URL) (Version 2.4 and Above)
    - Allows the resources to return to after SSO to be "locked" to a specific value, even when running as a result of active protection of other resources. In other words, this value overrides the actual resource location when SSO redirection is automatic, including initial access and after a timeout.

- `encoding` (string) (Version 2.4 and Above)
    - Controls the encoding of attribute values exported to headers or environment variables. If omitted, the default is to encode the data as UTF8. The only supported value is "URL", which applies URL-encoding to the UTF8 data before export.

- `NameIDFormat` (URI) (Version 2.4 and Above)
    - NameIDPolicy Format attribute to use in authentication request. Overrides the `NameIDFormat` attribute of a SessionInitiator.
- `SPNameQualifier` (string) (Version 2.4 and Above)
    - NameQualifier to use in authentication request. For instance, entityID of an EntityDescriptor with an AffiliationDescriptor.

- `exportStdVars` (boolean) (defaults to true) (Version 2.5 and Above)
    - If true, causes the SP to export a built-in set of standard variables based on the users's session. This set includes "Shib-Identity-Provider", "Shib-Authentication-Instant", "Shib-Authentication-Method", "Shib-AuthnContext-Class", "Shib-AuthnContext-Decl", and "Shib-Session-Index". A future version of the SP may remove these built-ins in favor of explicit configuration using the NativeSPAttributeExtractor of `type="Assertion"`.

- `exportCookie` (boolean) (defaults to false) (Version 2.5 and Above)
    - If true, causes the SP to export a variable called "Shib-Cookie-Name" with the algorithmically-generated portion of the implementation-specific cookies used by the SP to maintain sessions with users and track other state. Applications that want to unilaterally dispose of SP state and session information can delete any cookie whose name contains the value of this variable.

- `discoveryURL` (URL) (Version 2.5 and Above)
    - Overrides the default location used by "discovery" session initiators. Advanced option that can be used to direct the user to different discovery interfaces based on the resource accessed.

- `discoveryPolicy` (string) (Version 2.5 and Above)
    - Used as input to some discovery protocols that take parameters modifying discovery behavior. In the case of the `type="SAMLDS"` SessionInitiator, this is passed as a `policy` parameter value.

- `requireLogoutWith` (URL) (Version 2.5 and Above)
    - Used in conjunction with passive protection of a resource, this property will automate a redirect to the URL specified (usually the SP's logout handler) and then a return to the location being accessed, passing control to it. Assuming SP logout proceeds successfully, this will invoke that mechanism and pass control to this resource with the SP session disposed of, enabling application logout to proceed.

- `exportDuplicateValues` (boolean) (defaults to true) (Version 2.6 and Above)
    - If set to false, the export of attribute values to variables or headers will filter out duplicate values. This occurs per-header, so accounts for aliasing and multiply-sourced headers, but also adds some overhead to the processing of every request for larger attribute sets.