NativeSPHandler

The `<Handler>` element is used to configure generic SP handlers that offer miscellaneous functionality that doesn't fall into the categories covered by other predefined elements.

Typically, if a handler implements a SSO protocol, issues SSO requests, deals with logout, or implements other capabilities baked into the SAML 2.0 specification, it will not use the generic element to configure itself.

Common Attributes

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

- `Location`(relative path)
    - Path used to invoke handler (when appended to the base `handlerURL`).

Metadata Generation Handler

Identified by `type="MetadataGenerator"`, generates sample metadata based on the SP configuration and a set of built-in assumptions. Can be supplemented through the use of a template, and supports signing.

Support for signing may leave your system vulnerable to trivial DOS attacks. If this is a concern, consider protecting the generation handler with some kind of authentication or firewall.

The purpose of this handler is **NOT** to supply other systems with production metadata but rather to assist with testing, and generation of metadata examples useful in understanding how to produce actual metadata. Mature deployments will often require metadata content that goes beyond what the handler can generate, and directly coupling metadata to a configuration makes certain configuration changes more likely to cause service disruptions.

Attributes

- `acl`(list of space-delimited IP addresses) (defaults to open access)
    - A set of requesting addresses to limit access to.

- `template`(local pathname)
    - Optional XML file containing an `<md:EntityDescriptor>` to use as a template for the generated metadata. Any content supplied will generally be maintained and supplemented by the generation process.

- `cacheDuration`(time in seconds)
    - Optional value to use in calculating the `cacheDuration` attribute to embed in the generated metadata. While optional, this value **SHOU LD** be used in any experimentation with this handler in real-time generation of metadata.

- `signing`(boolean) (defaults to false)
    - If true, metadata will be signed using the default credential (or the named credential if `keyname` is used as well).

- `keyName`(string)
    - Optional, used as input when selecting a credential to use for metadata signing.

- `signingAlg`(URI) (defaults to RSA with SHA-1)
    - The XML Signature signing algorithm to use for metadata signing.

- `digestAlg`(URI) (defaults to SHA-1)
    - The XML Signature digest algorithm to use for metadata signing.

- `mimeType`(MIME content type) (defaults to "application/samlmetadata+xml")
    - Optional override of MIME type of content returned by handler.

- `http`(boolean)
    - Forces or disables the generation of protocol endpoints using the http scheme, regardless of which scheme is used when accessing the handler.

- `https`(boolean)
    - Forces or disables the generation of protocol endpoints using the https scheme, regardless of which scheme is used when accessing the handler.

Version 2.2 and Above

- `validUntil`(time in seconds)
    - Optional value to use in calculating the `validUntil` attribute to embed in the generated metadata. The value is added to the current time. Note that prior to version 2.2, the `cacheDuration` setting was improperly being used for this purpose.

Version 2.4 and Above

- `salt`(string)
    - Optional salt to include in a hash with the entityID to produce the ID attribute placed into the metadata. Can be used to control the ID value in concert with configuration changes to signal to metadata consumers that the metadata has or hasn't been altered.

Child Elements

- `<EndpointBase>`(zero or more)
    - If provided, must contain a "base" handler URL to generate endpoints against. When used, these base locations override the default behavior, which generates endpoints based only on the handler URL configured for the request to the generator, which usually means only a single virtual host can be represented in the metadata.

Version 2.4 and Above:

- Various Metadata Content (zero or more)
    - A variety of metadata elements can be embedded directly within the handler configuration for inclusion in the generated metadata. This includes:
        - `<md:NameIDFormat>` (any number)
        - `<md:ContactPerson>` (any number)
        - `<md:RequestedAttribute>` (any number)
        - `<md:AttributeConsumingService>` (any number)
        - `<md:Organization>` (up to one)
        - `<mdui:UIInfo>` (up to one)
        - `<mdattr:EntityAttributes>` (up to one)

Runtime Parameters

Parameters are supplied using a standard query string appended to the request. They must be URL-encoded as usual.

- `entityID`(URI)
    - Identifies the requesting IdP, for the purposes of establishing settings that would influence the contents of the metadata, such as message signing flags and the credentials used. If omitted, the default settings are used.

Status Handler

Identified by `type="Status"`, reports on the operational status of the SP software and some configuration options. The information is returned in the form of a simple XML document, suitable for processing with a style sheet.

> ⚠ Some of the information returned might be considered confidential by some deployers, so limiting access may be warranted. On Version 2.4 and above, the handler now includes some basic operating system and platform information, which could be used to aid an attacker.

Attributes

- `acl`(list of space-delimited IP addresses) (defaults to open access)
    - A set of requesting addresses to limit access to. As of V2.5, use of CIDR notation and IPV6 addressses and ranges is supported.

Runtime Parameters

Parameters are supplied using a standard query string appended to the request. They must be URL-encoded as usual.

- `target`(URI)
    - Identifies a resource location hosted by the SP web site. Instead of returning generic status information, the handler will return a summary of the content settings in effect for requests for the supplied resource.

Session Handler

Identified by `type="Session"`, reports on a specific user session to aid in debugging.

Much of the information returned might be considered confidential, so limiting access may be warranted, although in practice only a given user should have access to information associated with his own session.

Attributes

- `acl`(list of space-delimited IP addresses) (defaults to open access)
  - A set of requesting addresses to limit access to. As of V2.5, use of CIDR notation and IPV6 addressses and ranges is supported.
- `showAttributeValues`(boolean) (defaults to false)
  - If true, the session summary will include actual attribute values.
- `contentType` (string) (defaults to "text/html") (Version 2.5 and Above)
  - Selects the content type of the response, affecting the output format. The default, as in older versions, is a simple HTML page. For programmatic use, the value "application/json" is supported, causing the result to be a JSON struct.

Discovery Feed Handler (Version 2.4 and Above)

Identified by `type="DiscoveryFeed"`, produces a combined JSON feed of IdP information from all `<MetadataProvider>` sources that support the feature.

While producing this information through the SP is not ideal from an efficiency point of view, doing so helps ensure accurate information is supplied and addresses synchronization issues across the metadata sources as they internally refresh themselves. Deployers needing additional performance may choose to manually access the handler to produce a static file for delivery, updating it on some schedule.

Note

You might also consider to set the option `legacyOrgNames`="true" on the MetadataProviders in order to show an Identity Provider's OrganizationDisplayName if no MDUI DisplayName is present. This option is transitional however and may be removed in future versions.

Attributes

- `cacheToClient`(boolean) (defaults to false)
  - If true, the feed includes cache directives intended to support client-side caching of the information. Many clients are overly aggressive about caching, so this feature is disabled by default.
- `cacheToDisk`(boolean) (defaults to true)
  - If true, discovery feeds are cached to disk by the `shibd` process, and read back from disk while producing the feed from the web server. This is much more efficient, but requires shared file access. If a `shibd` process is used remotely, this can be disabled to supply the feed through the remoting layer.
- `dir`(relative or absolute path)
  - If set and `cacheToDisk` is true, specifies the directory to store feed files. If omitted, feeds will be written to the package-specific "run" directory, usually **/var/run/shibboleth** or **/opt/shibboleth-sp/var/run/shibboleth**.

Attribute Checker Handler (Version 2.5 and Above)

Identified by `type="AttributeChecker"`, validates a user's session against a list of required attributes (and optionally values) and either returns the user to complete the login process or displays an error template. The template is in the same form described by the NativeSPErrors topic, and also has access to the user's session, such that attributes in the session can be used via `<shibmlp attrID />` tags.

This handler is designed to complement the `sessionHook` setting by leveraging the hook to check for required attributes.

The attributes to check for can be specified in one of two ways:

- a list of attribute IDs via the `attributes` setting (see below)
- by embedding a valid access control policy inside the element

The latter option allows arbitrary checking of the session against boolean combinations of attributes and values. For example, instead of requiring that all of a set of attributes be present, an `<OR>` can be used.

Typical Example

```
<Handler type="AttributeChecker" Location="/AttrChecker" template="attrChecker.html"
    attributes="eppn displayName" flushSession="true"/>
```

Extended Syntax

```
<Handler type="AttributeChecker" Location="/AttrChecker" template="attrChecker.html"
        flushSession="true">
    <AND>
        <Rule require="eppn">jdoe@example.edu</Rule>
        <Rule require="displayName"/>
    </AND>
</Handler>
```

Attributes

- `template`(local pathname)
  - Required attribute specifying the path to an error template to use in the event that checking fails.
- `flushSession`(boolean) (defaults to "false")
  - If true, the user's session is forcibly removed if the session fails the check.
- `attributes`(whitespace-delimited list of attribute IDs)
  - Specifies a list of attributes to look for. If the session does not contain at least one value for each attribute designated, the session "fails" the check.

Example on how to use AttributeChecker

One example approach how to use the Attribute Checker Handler to mitigate the case where an IdP released too few attributes to an SP is shown in the eduGAIN Wiki on the page How to configure Shibboleth SP attribute checker. Following the instructions there, a Shibboleth SP will show a helpful error message and provide the user with an easy way (2 clicks) to inform his IdP administrator regarding the attribute release problem. Also, the approach described on the wiki page makes use of a tracking cookie to log (locally or remotely) cases where users ended up on the error page.

External Authentication Handler (Version 2.5 and Above)

Identified by `type="ExternalAuth"`, implements a loopback REST interface for creating user sessions based on external authentication logic. Complete documentation on use of the handler is available in the NativeSPBackDoor topic.

This handler **SHOULD NOT** be exposed to any untrusted network interfaces and addresses or you will create a security exposure in your system. It allows any trusted caller to create user sessions based on arbitrary submitted information.

Attributes

- `acl`(list of space-delimited IPv4 or IPv6 CIDR ranges or explicit addresses) (defaults to 127.0.0.1 and ::1)
  - A set of requesting addresses to limit access to.

Attribute Resolver Handler (Version 2.6 and Above)

Identified by `type="AttributeResolver"`, implements a loopback query-based protocol to invoke the SP's AttributeResolver machinery in a manner similar to the resolvertest utility/example, and provides JSON-based output. In comparison to the resolvertest binary, this plugin is a lot faster because it does not have to load the whole configuration (including metadata download etc.) and it can be queried via a web request.

To use this plugin, the **plugins.so** shared library must be loaded via the `<OutOfProcess>` element's `<Library>` element. Also the **plugins-lite.so** shared library must be loaded via the `<InProcess>` element's.

This handler **SHOULD NOT** be exposed to any untrusted network interfaces and addresses or you will potentially expose user information to an untrusted requester. There is no security implemented in addition to the acl (access control list) option on this interface and it is designed for local use only.

Configuration Example

```
<!-- Add as first element within root element <SPConfig> of shibboleth2.xml -->
<OutOfProcess>
    <Extensions>
        <Library path="plugins.so" fatal="true"/>
    </Extensions>
</OutOfProcess>

<InProcess>
    <Extensions>
        <Library path="plugins-lite.so" fatal="true"/>
    </Extensions>
</InProcess>

[...]

<!-- Add at end of <Sessions> element -->
<Handler type="AttributeResolver" Location="/AttributeResolver"
         acl="127.0.0.1 ::1" />
```

Attributes

- `acl`(list of space-delimited IP addresses or CIDR ranges) (defaults to localhost-only access)
    - A set of requesting addresses to limit access to.

The following parameters may be supplied either in fixed form inside the XML, or as query string parameters:

- `nameId` (required)
    - the SAML NameIdentifier/NameID value to supply in any queries issued
- `format`
    - the SAML Nameidentifier/NameID Format to use
- `nameQualifier`
    - the SAML NameIdentifier/NameID NameQualifier to set (defaults to entityID parameter if that's supplied)
- `spNameQualifier`
    - the SAML NameID SPNameQualifier to set (defaults to SP's own identifier)
- `protocol`
    - protocol support constant used during any metadata lookups, or abbreviated values of "SAML2.0", "SAML1.1", or "SAML1.0", defaults to "SAML2.0"
- `entityID`
    - the "issuing" entity for the purposes of looking up SAML metadata for input to the resolution process
- `encoding`
    - one of "JSON" or "JSON/CGI" (default is "JSON")

The resolution process behaves as though an assertion containing a subject identifier (e.g. a persistentID) was received from the entity identified by the various parameters, and then performs a call to the resolver equivalent to what would be performed if no attributes were initially received. Suitably manipulated, this makes it possible to generate arbitrary attribute queries to systems for which metadata is available. One use case is to retrieve user attributes from a user's Identity Provider without the user's involvement, provided the SP has for example the users persistentID Name ID.

> Once the AttributeResolver handler is configured, it can for example be queried (e.g. from localhost) with:
> ```
> $ curl -k 'https://localhost/Shibboleth.sso/AttributeResolver?format=urn:oasis:names:tc:SAML:2.0:nameid-format:
> persistent&entityID=https%3A%2F%2Fyour.idp.example.org%2Fidp%2Fshibboleth&nameId=123456789PfvsH8k4gvHoeq6QtM='
> ```

This will return a JSON data structure like:

```
{
    "displayName" : [

        "Lukas Hämmerle"
    ],
    "mail" : [

        "lukas.haemmerle@switch.ch"
    ],
    "schacHomeOrganization" : [

        "switch.ch"
    ],
    "persistent-id" : [

        "https://your.idp.example.org/idp/shibboleth!https://test.sp.example.org/shibboleth!123456789PfvsH8k4gv
Hoeq6QtM="
    ],
    "idp" : [

        "https://your.idp.example.org/idp/shibboleth"
    ]

}
```

The output is currently limited to JSON, and is either dumped in a structure containing an array field named for each attribute, with each value serialized to its own own array slot, or is encoded in a way that combines multiple values into delimited strings identical to what would appear in server variables or headers. The latter is enabled by setting the `encoding` parameter to "JSON/CGI".