NativeSPAttributeResolver

The `<AttributeResolver>` element configures a plugin responsible for obtaining additional identity attributes about the logged in user following a SSO event.

During SSO, the IdP can supply attributes in a "push" fashion inside the SAML assertions it issues. These attributes are decoded with an attribute extractor and cached with the user's session. The purpose of a resolver plugin is to "pull" attributes from additional sources or to transform existing attributes in some way.

Common Attributes

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

Query AttributeResolver

Indicated by `type="Query"`, issues a SAML attribute query back to the IdP that issued a SSO assertion if no attributes are pushed. This is compatible with older Shibboleth behavior. Obviously, metadata for a compatible attribute authority must be available.

After execution, the resolver applies the attribute extractor and filter configured for the application before returning the resulting attributes.

Attributes

- `policyId` (string) (Version 2.2 and Above)
    - Optional identifier of a customized security policy to use.

- `subjectMatch` (boolean) (defaults to false) (Version 2.3.1 and Above)
    - If true, enforces SAML "strong matching" requirements on the subject of the resulting assertions. By default, the IdP is trusted to return an assertion about the queried subject without explicitly comparing the result.

- `exceptionId` (string) (Version 2.4 and Above)
    - Optional identifier of a special attribute to create in the event of a "transient" failure during the query. Errors are considered transient if they are caused by system outages or misconfiguration. If an IdP appears to support the query protocol, then transient errors include any failure to obtain a successful SAML response or a violation of security policy while processing the result. If such errors occur, the attribute will contain one or more URL-encoded exception messages, and the application should be aware that not all of the "usual" attributes it might receive are available.

Child Elements

- `<saml2:Attribute>`(zero or more)
    - Supplies a set of attribute and value filters to include in any SAML 2.0 queries.

- `<saml1:AttributeDesignator>`(zero or more)
    - Supplies a set of attribute designators to include in any SAML 1.x queries.

SimpleAggregation AttributeResolver (Version 2.2 and Above)

Indicated by `type="SimpleAggregation"`, supports a primitive form of aggregation of attributes from multiple sources of authority through the use of SAML 2.0 attribute queries using an identifier derived from the attributes obtained prior to this plugin executing.

This simple form of aggregation relies on a plain-text identifier (which could be pseudonymous but is still "seen" by the SP) as a "link" to accounts at other SAML attribute authorities. Furthermore, this plugin only supports the same forms of authentication usable when making primary attribute queries with the "Query" plugin type above. This includes mechanisms like client TLS, message signing, or some forms of HTTP authentication, but is not "presence-oriented"; that is, there is no proof supplied the user is "present" at the SP, so there are no constraints on when the SP could perform such queries.

Conceptually, you can think of this mechanism as equivalent to a networked set of LDAP or X.500 directories queried by DN, merely in SAML terms. It is, however, relatively easy to implement and support when there are batch processes in place for the exchange of identity data to establish the links. No user intervention is required, which is a plus for simplicity but a minus for privacy and user control.

After **each** query is performed, the resolver applies the attribute extractor and filter configured for the application before continuing with other queries and eventually returning the resulting attributes. Each filtering step will operate on only the attributes extracted as a result of a particular query, and the filter policies can be expressed in terms of the actual "issuer" of each set of attributes for fine-grained control.

SimpleAggregation Example

```
<!-- Uses eduPersonPrincipalName from IdP to query, and asks for eduPersonEntitlement. -->
<AttributeResolver type="SimpleAggregation" attributeId="eppn" format="urn:oid:1.3.6.1.4.1.5923.1.1.1.6">
    <Entity>https://ieee.org/idp/shibboleth</Entity>
    <EntityReference>External-Links</EntityReference>
    <saml2:Attribute xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" Name="urn:oid:
1.3.6.1.4.1.5923.1.1.1.7" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" FriendlyName="
eduPersonEntitlement"/>
</AttributeResolver>
```

Attributes

- `policyId` (string)
  - Optional reference to a Security Policy to apply to the resulting messages and SAML assertions obtained. If omitted, the application's default policy is used.
- `attributeId` (space-delimited list of strings)
  - Optional list of attribute IDs to search for a value to use as the identifier in the queries performed. The first attribute value found will be used as the value of the `<NameID>`. If this attribute is omitted, the original `<NameID>` supplied by the user's original IdP is simply copied directly.
- `format` (URI)
  - Optional value to use as the `Format` attribute in a "generated" `<NameID>` created from an arbitrary source attribute using the `attributeId` setting above. This value is ignored if the `attributeId` setting is omitted, or if the specified attribute used is a so-called "NameID-valued" attribute that resulted from a NameID-aware attribute decoder.
- `subjectMatch` (boolean) (defaults to false) (Version 2.3.1 and Above)
  - If true, enforces SAML "strong matching" requirements on the subject of the resulting assertions. By default, each AA is trusted to return an assertion about the queried subject without explicitly comparing the result.
- `exceptionId` (string) (Version 2.4 and Above)
  - Optional identifier of a special attribute to create in the event of a "transient" failure during the aggregation process. Errors are considered transient if they are caused by system outages or misconfiguration. If an IdP appears to support the query protocol, then transient errors include any failure to obtain a successful SAML response or a violation of security policy while processing the result. If such errors occur, the attribute will contain one or more URL-encoded exception messages, and the application should be aware that not all of the "usual" attributes it might receive are available.

Child Elements

- `<Entity>` (zero or more)
  - The value of the element is an `entityID` to query against. Metadata for a SAML 2.0 Attribute Authority role must be available.
- `<EntityReference>` {zero or more)
  - The value of the element is the ID of an attribute available for the user. Each of the attribute's serialized values is interpreted as an `entityID`, as per the `<Entity>` element above.

Note that the `<Entity>` and `<EntityReference>` elements can be supplied in any order, and are processed in the order they appear, with a query attempt per `entityID` obtained.

- `<saml2:Attribute>` (zero or more)
  - Supplies a set of attribute and value filters to include in any queries.
- `<MetadataProvider>` (optional)
  - Supplies a dedicated MetadataProvider to use in place of the application-defined source.
- `<TrustEngine>` (optional)
  - Supplies a dedicated TrustEngine to use in place of the application-defined engine.

Version 2.5 and Above

- `<AttributeExtractor>` (optional)
  - Supplies a dedicated AttributeExtractor to use in place of the application-defined extractor.
- `<AttributeFilter>` (optional)
  - Supplies a dedicated AttributeFilter to use in place of the application-defined filter.

Transform AttributeResolver (Version 2.5 and Above)

Indicated by `type="Transform"`, applies one or more regular expressions to an input attribute, either replacing its values, or generating new attributes.

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

The transforms to apply are specified using `<Regex>` child elements. Each element specifies a transform rule to apply to each value of the designated input attribute(s).

A `match` attribute specifies a regular expression to match against the input value(s). It can contain parentheses to specify capturing groups. The content of the element is a replacement expression that can contain group placeholders to match the capturing groups (e.g., $1, $2, etc.). If the expression does not match, the value is passed through unchanged.

A `dest` attribute, if present, specifies the ID of a new attribute to create that will contain the transformed values. If not present, the transformed values replace the original attribute's values "in-place", if and only if the original attribute was a "simple" string-valued attribute. Other attribute types with more complex values cannot be transformed in-place.

Finally, the `caseSensitive` attribute specifies whether the containng `<Regex>` element's `match` expression is to be interpreted case-sensitively. It is a boolean that defaults to "true".

Transform Example

```
<AttributeResolver type="Transform" source="displayName">
    <Regex match="^(.+) (.+)$" dest="givenName">$1</Regex>
    <Regex match="^(.+) (.+)$" dest="sn">$2</Regex>
    <Regex match="^(.+) (.+)$">$2, $1</Regex>
</AttributeResolver>
```

Note that the example above is overly simplistic, and assumes the source values are in a particular format.

Attributes

- `source` (string)
  - Identifies the attribute ID of the input attribute(s) to process. All attributes with an ID that matches the value will be evaluated.

Child Elements

- `<Regex>` (one or more)
  - Specifies a transform rule to apply to each value of the designated input attribute(s). See above for details on the element syntax and function.

Template AttributeResolver (Version 2.5 and Above)

Indicated by `type="Template"`, plugs values from one or more existing attributes into a template string that can combine the original attributes into a new attribute.

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

The template syntax consists of a string containing tokens of the form *$id* where "id" is the attribute ID whose value should be plugged into the string. The attributes to combine are specified in a `sources` XML attribute (see below). The new attribute is named with a `dest` XML attribute.

The first attribute object found with a particular ID is used, and all of the attributes supplied must contain the same number of values, or the plugin is not applied. It works best with simple combinations of single-valued attributes.

Template Example

```
<AttributeResolver type="Template" sources="givenName sn" dest="displayName">
    <Template>$givenName $sn</Template>
</AttributeResolver>
```

Attributes

- `sources` (space-delimited list of strings)
  - Identifies the attribute IDs of the input attribute(s) to process.
- `dest` (string)
  - Used as the ID of the attribute created by the plugin.

Child Elements

- `<Template>` (exactly one)
  - The content of the element is the template string to apply in constructing the new attribute's value(s).

UpperCase AttributeResolver (Version 2.5 and Above)

Indicated by `type="UpperCase"`, converts the values of an attribute into upper case, either replacing its values, or generating a new attribute.

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

Attributes

- `source`(string)
  - Identifies the attribute ID of the input attribute(s) to process. All attributes with an ID that matches the value will be evaluated.
- `dest`(string)
  - Used as the ID of the attribute created by the plugin. If not set, the original attribute's values will be converted in-place, if and only if the original attribute was a "simple" string-valued attribute. Other attribute types with more complex values cannot be transformed in-place.

LowerCase AttributeResolver (Version 2.5 and Above)

Indicated by `type="LowerCase"`, converts the values of an attribute into lower case, either replacing its values, or generating a new attribute.

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

Attributes

- `source`(string)
  - Identifies the attribute ID of the input attribute(s) to process. All attributes with an ID that matches the value will be evaluated.
- `dest`(string)
  - Used as the ID of the attribute created by the plugin. If not set, the original attribute's values will be converted in-place, if and only if the original attribute was a "simple" string-valued attribute. Other attribute types with more complex values cannot be transformed in-place.

Chaining AttributeResolver

Indicated by `type="Chaining"`, executes multiple resolvers in sequence. Each resolver can see and use the output of the previous.

With V2.4 and above, this is implied by any configuration with multiple `<AttributeResolver>` elements, so is no longer explicitly needed.

Child Elements

- `<AttributeResolver>`(one or more)
  - Embedded attribute resolver plugins to chain together.
