IdPAddAttributeFilter

Define a New Attribute Filter

An attribute filter policy describes which attributes are sent to a service provider. The default attribute filter policy file is *IdP_HOME/conf/attribute-filter.xml*.

1. Define a Policy

An attribute filter policy is defined by the element `<AttributeFilterPolicy>` with the following attribute:

- **id** - a unique identifier for this policy

Example Attribute Filter Policy Definition

```
<AttributeFilterPolicy id="releaseToAnyone">
    <!-- Policy Requirement Rule would go here -->

    <!-- Attribute Rules would go here -->
</AttributeFilterPolicy>
```

2. Define a Policy Requirement Rule

Each attribute filter policy must contain one and only one policy requirement rule. This rule determines if the given attribute filter policy is active for a given request. If the policy requirement rule evaluates to true than the policy is active; if it evaluates to false the policy is not active.

A policy requirement rule is defined with the element `<PolicyRequirementRule xsi:type="MATCHING_RULE_TYPE">`. The matching rule type may be any matching rule type.

Example Attribute Filter Policy Definition with Policy Requirement

```
<AttributeFilterPolicy id="releaseToAnyone">
    <PolicyRequirementRule xsi:type="basic:ANY" />

    <!-- Attribute Rules would go here -->
</AttributeFilterPolicy>
```

3. Define Attribute Rules

Each attribute filter policy contains zero or more attribute rules. If an attribute filter policy is active, then the set of attribute rules determines which attributes the policy affects.

An attribute rule is defined with the element `<AttributeRule>` with the following required attribute:

- **attributeID** -attributeID - the case-sensitive ID, as assigned in the attribute resolver, of the attribute to which the rule applies

Example Attribute Filter Policy Definition with Attribute Rules

```
<AttributeFilterPolicy id="releaseToAnyone">
    <PolicyRequirementRule xsi:type="basic:ANY" />

    <AttributeRule attributeID="transientId">
        <!-- Permit/Deny Rules go here -->
    </AttributeRule>

    <AttributeRule attributeID="eduPersonAffiliation">
        <!-- Permit/Deny Rules go here -->
    </AttributeRule>
</AttributeFilterPolicy>
```

4. Define Permit/Deny Value Rules

Each attribute rule contains one and only one value rule. A permit value rule specifies which attribute values are permitted to be released. A deny value rule specifies which attribute values are not permitted to be released. A value is released if and only if it has been permitted and has not been denied. That is, a deny always takes precedence over a permit.

A permit value rule is defined with the element `<PermitValueRule xsi:type="MATCHING_RULE_TYPE">`. The matching rule type may be any matching rule type.

A deny value rule is defined with the element `<DenyValueRule xsi:type="MATCHING_RULE_TYPE">`. The matching rule type may be any matching rule type.

Example Attribute Filter Policy Definition with Attribute Rules

```
<AttributeFilterPolicy id="releaseToAnyone">
    <PolicyRequirementRule xsi:type="basic:ANY" />

    <AttributeRule attributeID="transientId">
        <PermitValueRule xsi:type="basic:ANY" />
    </AttributeRule>

    <AttributeRule attributeID="eduPersonAffiliation">
        <PermitValueRule xsi:type="basic:OR">
            <basic:Rule xsi:type="basic:AttributeValueString" value="faculty" ignoreCase="true"/>
            <basic:Rule xsi:type="basic:AttributeValueString" value="student" ignoreCase="true"/>
            <basic:Rule xsi:type="basic:AttributeValueString" value="staff" ignoreCase="true"/>
            <basic:Rule xsi:type="basic:AttributeValueString" value="alum" ignoreCase="true"/>
            <basic:Rule xsi:type="basic:AttributeValueString" value="member" ignoreCase="true"/>
            <basic:Rule xsi:type="basic:AttributeValueString" value="affiliate" ignoreCase="true"/>
            <basic:Rule xsi:type="basic:AttributeValueString" value="employee" ignoreCase="true"/>
            <basic:Rule xsi:type="basic:AttributeValueString" value="library-walk-in" ignoreCase="true"/>
        </PermitValueRule>
    </AttributeRule>
</AttributeFilterPolicy>
```

Matching Rules

The following matching rules are supported in a standard Shibboleth IdP installation and usable as the type for `<PolicyRequirementRule>`, `<PermitValueRule>`, and `<DenyValueRule>` elements .

- ANY - Always evaluates to true
- AND - Evaluates to true if all contained rules are true
- OR - Evaluated to true if any contained rule is true
- NOT - Evaluates to true if the contained rule evaluates to false
- AttributeRequesterString - Evaluates to true if the attribute requester's entity ID matches a given string
- AttributeIssuerString - Evaluates to true if the attribute issuer's entity ID matches a given string
- PrincipalNameString - Evaluates to true if the user's principal name matches a given string
- AuthenticationMethodString - Evaluates to true if the method used to authenticate the user matches a given string
- AttributeValueString - Evaluates to true if the value of a given attribute matches a given string
- AttributeScopeString - Evaluates to the true if the scope of a value of a given attribute matches a given string
- AttributeRequesterRegex - Evaluates to true if the attribute requester's entity ID matches a given regular expression
- AttributeIssuerRegex - Evaluates to true if the attribute issuer's entity ID matches a given regular expression

- PrincipalNameRegex - Evaluates to true if the user's principal name matches a given regular expression
- AuthenticationMethodRegex - Evaluates to true if the method used to authenticate the user matches a given regular expression
- AttributeValueRegex - Evaluates to true if the value of a given attribute matches a given regular expression
- AttributeScopeRegex - Evaluates to the true if the scope of a value of a given attribute matches a given regular expression
- Script - Evaluates a scriptlet to determine if the rule evaluates to true
- AttributeRequesterInEntityGroup - Evaluates to true if the attribute requester is defined within a given entity group in SAML metadata
- AttributeIssuerInEntityGroup - Evaluates to true if the attribute issuer is defined within a given entity group in SAML metadata
- AttributeIssuerNameIDFormatExactMatch - Evaluates to true if the metadata that a given name identifier format is supported by the attribute issuer.
- AttributeRequesterNameIDFormatExactMatch - Evaluates to true if the metadata that a given name identifier format is supported by the attribute requester.
- AttributeIssuerEntityAttributeExactMatch - Evaluates to true if an entity attribute of the issuer exactly matches a given value.
- AttributeRequesterEntityAttributeExactMatch - Evaluates to true if an entity attribute of the requester exactly matches a given value.
- AttributeIssuerEntityAttributeRegexMatch - Evaluates to true if an entity attribute of the issuer matches a given regular expression.
- AttributeRequesterEntityAttributeRegexMatch - Evaluates to true if an entity attribute of the requester matches a given regular expression.
- AttributeInMetadata - (V2.4.0+) Evaluates to true if a given attribute is found in the SP's metadata as a requested attribute.

Loading Multiple Policy Group Files

Some installations may wish to define attribute filter policies in multiple files, for administrative and maintenance purposes. The IdP is capable of loading multiple policy files. To enable this, within the *IDP_HOME/conf/service.xml* file, locate the service `AttributeFilterEngine`. Add one additional `<ConfigurationResource>` element, with an appropriate resource type for each additional file you wish the IdP to load.

Each filter policy group must have a unique ID. If you create a copy of an existing filter file, as the seed for your new file, be sure to change the `id` attribute on the `<AttributeFilterPolicyGroup>` element.

Example Filter Configuration Loading Three Files

```
<Service id="shibboleth.AttributeFilterEngine"
         xmlns="urn:mace:shibboleth:2.0:services"
         xsi:type="afp:ShibbolethAttributeFilteringEngine">
    <ConfigurationResource file="/opt/shibboleth-idp/conf/attribute-filter.xml" xsi:type="resource:
FilesystemResource" />
    <ConfigurationResource file="/opt/shibboleth-idp/conf/group-attribute-filter.xml" xsi:type="resource:
FilesystemResource" />
    <ConfigurationResource file="/opt/shibboleth-idp/conf/user-attribute-filter.xml" xsi:type="resource:
FilesystemResource" />
</Service>
```

Examples

Additional examples are also available. These provide more complete examples and are contributed by users of the software.

Example 1	Releases the controlled values for eduPersonAffiliation to anyone.
Example 2	Releases email address to a specific service provider
Example 2	Denies the release of personal information if FERPA suppression is enabled.