

Introduction

In this article I discuss migrating SiteMinder policies to OpenAM. I have laid out an approach and framework to make the conversion possible using XML parsing, XML codifying and custom metadata that is necessary to resolve embedded repository externalities in the SiteMinder policy definition. I have also written a Java-based program that uses the methodology discussed in this article to parse through the SiteMinder policy and resolve each policy construct to an OpenAM equivalent policy object. The program codifies the XACML output step-by-step and on completion, one is able to import the XACML policy set into OpenAM via the policy editor.

There are going to be some complications in your situation such as when converting whitelisted resource URI – that SiteMinder does not ‘trap’ in the policy definition- you will need to know which URI are not in your SM policy extract and allowed by default, and then go about creating or adding the same URI in a new or ‘converted’ OpenAM policy. You would then need to whitelist those URI by explicitly allowing access in OpenAM.

I begin by describing the general layout of an XPS domain export at a high level, and how different policy constructs in a SiteMinder policy map over to OpenAM.

Survey of SiteMinder Policy Objects

While there is not a one-to-one mapping for every policy construct, there is a logical mapping that is useful to achieve a first pass on the conversion. Some of the complications are elaborated on below.

CA.SM::Domain -> OpenAM Policy Set Definition

The CA.SM::Domain object is the parent object to several domain-specific properties such as Mode, Name and UserDirectoriesLink references. Nested directly under the Domain object are CA.SM::Realm, CA.SM::Policy and CA.SM::Response objects.

CA.SM::Realm -> OpenAM Policy Resources

The Realm object includes properties that describe the AgentGroupLink and AuthScheme settings, session properties such as IdleTimeout, and among other things, the ResourceFilter that is the protected resource URI. Realm also contains a nested object, called the CA.SM::Rule with an associated XID that describes the Actions available for the ResourceFilter- such as GET, PUT and POST- and properties that describe regular expression matching, time restrictions, resource pattern and IsEnabled among other things.

CA.SM::Response -> OpenAM Response Attributes

Response object is a container for one or more name-value pairs, or 'response attributes' expected by the agent. These can be of type Web agent responses or RADIUS responses. This conversion article and the Java program are limited to handling only the Web agent responses currently.

The Response object has an associated XID, and includes properties that describe the authorization conditions when the response is applicable- such as AccessAccept and AccessChallenge- and properties for AgentTypeLink and Name.

CA.SM::ResponseAttr

Response also nests CA.SM::ResponseAttr objects, each with a unique XID, that each have properties for AgentTypeAttrLink- pointing to ReferenceObjects such as "WebAgent-HTTP-Header-Variable" indicating the means of transport for the response attributes- and also the Value of the response codified in the form of display-name=<%userattr="profile-attribute-name"%>. Here it is assumed that the value is sourced from a user attribute, which is mostly the case, although several types of ResponseAttrs are possible, including user attribute as already mentioned, DN Attribute, Static values, Session Variable, etc.

CA.SM::Policy -> OpenAM Policy Definition

Nested under the Domain object, and at peer level to the Realm and Response objects is the CA.SM::Policy object that serves to tie together protected resource URI to the corresponding user audience, rules including actions, response objects and optionally, IP address restrictions and time restrictions for those URI.

CA.SM::PolicyLink

The Policy object container has one or more CA.SM::PolicyLink objects that can be thought of as associations between the protected resources and the responses configured for each of them. The PolicyLink objects each have CA.SM::ResponseLink and CA.SM::RuleLink properties.

CA.SM::ResponseLink and CA.SM::RuleLink

The ResponseLink is a pointer to the Response object definition, and the RuleLink is a pointer to the Rule definition stored inside a CA.SM::Realm object. When invoked for a specific Realm, the policy enforces the Rule Actions as well as the Response attributes in case the policy evaluation is successful.

CA.SM::UserPolicy

The Policy object also contains one or more CA.SM::UserPolicy objects that can be thought of as

audience restriction conditions. The UserPolicy object has properties defining the FilterPath, which is a SQL query but one that could be used to locate a set of users this policy applies to. Another property is the UserDirectoryLink that is a pointer to the external repository that contains the user profile data, and specifically stores the XID of the ReferenceObject by the name CA.SM::UserDirectory.

References

Besides Domain objects, there are also reference objects at a peer level to Domain. The Reference object tree encapsulates all external information that the policy references.

CA.SM::UserDirectory

The user repository information is stored under the CA.SM::UserDirectory ReferenceObject that includes a NameSpace attribute- such as “ODBC:”- and Server name. As mentioned previously the UserPolicy object stores a reference to the UserDirectory object using a UserDirectoryLink.

The program relies on these user directory references to be resolved using a metadata container in OpenDJ.

CA.SM::AgentGroup

The AgentGroup object refers to the SiteMinder agent configurations that all protect the same resources. The attribute AgentTypeLink contains a value that is referenced from CA.SM::Response objects using the AgentTypeLink property tying the web agent to that particular response object. There could be one or more Agent Groups.

CA.SM::AgentTypeAttr

One or more of these Reference objects define in what form the agent expects the response objects to be returned. Two currently supported, and commonly used, values include “WebAgent-HTTP-Header-Variable” and “WebAgent-HTTP-Cookie-Variable”. These are available for use in accept or reject responses only in SiteMinder.

Some other Reference objects include CA.SM::AgentType, which describes the SM agent- basically a Web Agent. Another is the CA.SM::AuthScheme which includes details about the form of authentication such as Forms-based, Basic auth or API.

Methodology for Conversion to OpenAM

Converting CA.SM::Domain and CA.SM::Realm objects

Simply parse out these objects from the SiteMinder policy extract and create OpenAM policy set XACML definitions.

A code sample to establish the proper order and parity between the XACML elements is shown here:

```
Element target = doc.createElementNS("ns2", "ns2:Target");
Element anyOf_subject = getAnyOfSubjectTree(doc, xmlMap);
target.appendChild(anyOf_subject);

Element anyOf_resource = getAnyOfResourceTree(doc, xmlMap, resourceUri);
target.appendChild(anyOf_resource);

Element anyOf_application = getAnyOfApplicationTree(doc, xmlMap);
target.appendChild(anyOf_application);

Element anyOf_action = getAnyOfActionTree(doc, xmlMap, actions);
target.appendChild(anyOf_action);
```

A code sample for creating the Resource URI in XACML format is shown below:

```
private static Element getAnyOfResourceTree(Document doc, Map<String, String> xmlMap, List resourceUri) {
// many AnyOf elements to define subject, resources, actions, etc
Element anyOf = doc.createElementNS("ns2", "ns2:AnyOf");
for(String url : resourceUri) {
    Element allof = doc.createElementNS("ns2", "ns2:AllOf");
    Element Match = doc.createElementNS("ns2", "ns2:Match");
    Match.setAttribute("MatchId",
"urn:sun:opensso:entitlement:resource-match:application:"+xmlMap.get("PolicyId"));
    Element AttributeValue = doc.createElementNS("ns2", "ns2:AttributeValue");
    AttributeValue.setAttribute("DataType", "http://www.w3.org/2001/XMLSchema#string");
    AttributeValue.setTextContent("*/:*:*" + url);
    Match.appendChild(AttributeValue);
    Element AttributeDesignator = doc.createElementNS("ns2", "ns2:AttributeDesignator");
    AttributeDesignator.setAttribute("Category",
"urn:oasis:names:tc:xacml:3.0:attribute-category:resource");
    AttributeDesignator.setAttribute("AttributeId",
"urn:oasis:names:tc:xacml:1.0:resource:resource-id");
    AttributeDesignator.setAttribute("MustBePresent", "true");
    AttributeDesignator.setAttribute("DataType", "http://www.w3.org/2001/XMLSchema#string");
    Match.appendChild(AttributeDesignator);
    allof.appendChild(Match);
    anyOf.appendChild(allof);
}
return anyOf;
}
```

```
}
```

Converting the CA.SM::Rule.Actions objects

Here the challenge is to convert the ResourceFilters into ResourceURI objects in the XACML definition, and then add the relevant actions to those resource URI as defined in the CA.SM::Realm object definition. A sample method that adds basic actions to a policy is shown here:

```
private static Element getAnyOfActionTree(Document doc, Map<String, String> xmlMap, List actions) {
    // many AnyOf elements to define subject, resources, actions, etc
    Element anyOf = doc.createElementNS("ns2", "ns2:AnyOf");
    for(String action : actions) {
        Element allOf = doc.createElementNS("ns2", "ns2:AllOf");
        Element Match = doc.createElementNS("ns2", "ns2:Match");
        Match.setAttribute("MatchId",
            "urn:sun:openso:entitlement:action-match:application:"+xmlMap.get("PolicyId"));
        Element AttributeValue = doc.createElementNS("ns2", "ns2:AttributeValue");
        AttributeValue.setAttribute("DataType", "http://www.w3.org/2001/XMLSchema#string");
        AttributeValue.setTextContent(action);
        Match.appendChild(AttributeValue);
        Element AttributeDesignator = doc.createElementNS("ns2", "ns2:AttributeDesignator");
        AttributeDesignator.setAttribute("Category",
            "urn:oasis:names:tc:xacml:3.0:attribute-category:action");
        AttributeDesignator.setAttribute("AttributeId",
            "urn:oasis:names:tc:xacml:1.0:action:action-id");
        AttributeDesignator.setAttribute("MustBePresent", "true");
        AttributeDesignator.setAttribute("DataType", "http://www.w3.org/2001/XMLSchema#string");
        Match.appendChild(AttributeDesignator);
        allOf.appendChild(Match);
        anyOf.appendChild(allOf);
    }
    return anyOf;
}
```

Converting Policy bindings for CA.SM::Realm, CA.SM::Response and CA.SM::UserPolicy objects

This portion is the most complex in the conversion because linkages have to be parsed out of the SiteMinder policy extract and rendered as separate OpenAM policies in XACML format. In order to convert a CA.SM::Policy object the program needs to follow the following algorithm at the very at least for EACH Policy object in the SiteMinder extract and in order:

1. Establish the linkages between the CA.SM::Realm objects and the CA.SM::Response objects using the PolicyLink binder object
2. Resolve the Response object, dynamic key-value pairs, by either directly reading the data from an external repository, or converting the references attributes to LDAP specific attributes
3. Resolve the Subject Restrictions manifest in the PolicyLink.UserPolicy by converting the SQL

Query to an LDAP search filter

4. Create one OpenAM policy under the previously established Policy Set per Realm-Response-UserPolicy linkage

These steps are necessary in order to preserve the linkage of specific resource URI being accessible over the policy decision and returning specific response attributes.

Parsing out the bindings between CA.SM::Realm and CA.SM::Response objects

Since not all CA.SM::Realm objects in the SiteMinder policy extract refer to all the CA.SM::Response objects it becomes necessary to collect those objects that are bound together and dispatch them into one OpenAM policy. This is because OpenAM ties together a group of protected resource URI to a group of subject conditions, environment conditions and response attributes.

Resolving the Response object into static or dynamic OpenAM Response Attributes

A design decision here could be that the response attributes will be pulled from a user repository such as OpenDJ, making the CA.SM::ResponseAttr.Value definitions very easy to resolve. After parsing the definitions that usually are of the form:

```
CL_Header=<%userattr="CL_Cookie_Value"%>|UID=<%userattr="DEF_PROFILE_ID"%>|State=<%userattr="ST"%>|City=<%userattr="ADDR_CITY"%>|AddressLine1=<%userattr="ADDR1"%>|AddressLine2=<%userattr="ADDR2"%>|Company=<%userattr="COMPANY"%>
```

One could construct the response objects one by one, with the display attribute (left of the “=” sign) as key and user profile attribute within quotes above, as the value. Simplistic string split and parsing techniques could be used to achieve this. The mapping of user profile attributes shown here could also be stored as key-value pairs in a metadata container in OpenDJ:

The screenshot shows the LDAP Browser/Editor v2.8.2 interface. The left pane displays a tree view of LDAP entries under the domain dc=forgerock,dc=com. The right pane shows the details of the selected entry, uid=Ref00003.

Attribute	Value
uid	Ref00003
description	ldap://.compute.amazonaws.com:1389
objectClass	person
objectClass	organizationalPerson
objectClass	inetOrgPerson
objectClass	top
givenName	
sn	cn=directory manager
cn	userid=uid,last_name=sn,Email_Add_NM=mail,DEF_PROFILE_ID=uid,CL_COOKIE_VALUE=devicePrintProfiles

Here, the connection information is stored as an LDAP url in description, admin username in sn and password (hidden) in givenName. The mapping of the user attributes is stored in cn. Using this information the program resolves the key-value pairs from the source repository to that of the target

repository schema and the resulting key-value pairs are codified as OpenAM Response objects into XACML, as shown:

```
urn:sun:opensso:entitlement:json-resource-attribute:com.sun.identity.entitlement.UserAttributes:<attr-name>
```

Another example of metadata that helps in resolving SiteMinder attributes into LDAP attributes is to store the following string in the metadata entry for an external repository references in the SiteMinder extract, such as REF00010:

```
LANG_CODE=physicalDeliveryOfficeName,ST=givenName, FAX_CITY_CD=1,  
TELE_EXTN_CD_BUS= telephoneNumber, EMP_ID= employeeNumber,  
ADDR= postalAddress, ADDR_NM_BUS= registeredAddress, TELE_PHONE= telephoneNumber
```

Resolve the Subject Restrictions manifest in the PolicyLink.UserPolicy

The program converts subject conditions derived from CA.SM::UserPolicy objects such as this one:

```
▼<Object Class="CA.SM::UserPolicy" Xid="CA.SM::UserPolicy  
06-28T08:31:15" UpdatedBy="SMSTUB" UpdateMethod="Interna  
▼<Property Name="CA.SM::UserPolicy.FilterClass">  
  <StringValue>Query</StringValue>  
  </Property>  
▼<Property Name="CA.SM::UserPolicy.FilterPath">  
  <StringValue>select A FROM B</StringValue>  
  </Property>  
▼<Property Name="CA.SM::UserPolicy.PolicyFlags">  
  <NumberValue>0</NumberValue>  
  </Property>  
▼<Property Name="CA.SM::UserPolicy.PolicyResolution">  
  <NumberValue>6</NumberValue>  
  </Property>  
▼<Property Name="CA.SM::UserPolicy.UserDirectoryLink">  
  ▼<LinkValue>  
    <XREF>Ref00009</XREF>  
  </LinkValue>  
  </Property>  
</Object>
```

It uses the same framework presented above, except this time the program creates an LDAP search filter from the SQL Query, to ensure the policy is only fired when the subject condition is satisfied. The user attribute 'A' is resolved to an attribute in an LDAP repository, or any external repository as long as connection information is included in the metadata.

The program also provides support for **directly** retrieving user profile attributes – either for Subject Restrictions presented above in the form of LDAP filters, or for dynamically creating the Response key-value pairs in XACML. The metadata can have connection information stored as shown above that

could be used to achieve this external lookup.

Automated OpenAM Policy Construction

Policy Set and Policy Definitions


The SiteMinder policy extract contains multiple “Rule to Response bindings per Realm” and these linkages are held in a container under the Policy object called PolicyLink. This means that for a set of resource URI, there is a defined set of actions, held under Rule objects, and a defined set of subject restrictions, environment restrictions and response attributes, held under Response objects. Each such linkage translates to one OpenAM policy.

The program examines all Realm objects and collects all resource URI from this Realm object and adds the Rule XIDs, also contained in the Realm object, as a key in a Rule-2-Realm Map. Rules XIDs and Response XIDs are also extracted from PolicyLink objects and stored together in a Rule-2-Response Map. At this level we have one OpenAMPolicy object mapped to a set of protected resources, resource actions, subject conditions and a response attributes collection. Because there could be many PolicyLink objects in this Policy container each pointing to different realms and responses, it is necessary to preserve the binding between rules and responses.

Presently, the program makes one assumption- that a SiteMinder admin has captured all policy bindings for a set of resources, actions, subject restrictions and response attributes in a single Policy binding, which seems to be the best practice anyway.

Naming

The program uses the name of the Domain in the SiteMinder extract as the name of the PolicySet. The individual policies that result out of the linkages defined above are assigned a unique identifier as shown below:



POLICY SET

AG_200581_ABC

Delete

Help

Policy Set AG_200581_ABC

























Policies

Details

+ Add a Policy

Configure this policy set by adding, editing, or removing authorization policies.

Filter...

NAME	STATUS	
 AG_200581_ABC-06f44409-b846-4b8e-870a-fb217e3151e8	 Active	 
 AG_200581_ABC-51871258-fc80-440f-9e87-ac448b5edcf5	 Active	 
 AG_200581_ABC-659d795b-1703-4f66-af0f-7b5de2fb5a4a	 Active	 
 AG_200581_ABC-8b497444-6ddd-47da-8454-8355fe47cd1b	 Active	 
 AG_200581_ABC-de0a5369-aa16-4128-b89f-312802fd2c1a	 Active	 
 AG_200581_ABC-ea82d39a-e3d8-4dbc-84d4-4753a24e26a7	 Active	 

Each policy combines a set of protected resources that have specific audience restrictions and response attributes bound to them. Specific actions such as OnAuthAccept and OnAccessReject can be accommodated into OpenAM as well. When an OnAuthAccept action in SiteMinder – used to redirect a user to a landing page for example- is read, OpenAM can return as a custom AdviceExpression to the PEP:

```
{ "advices": { "type": [ "advice" ] } }
```

In SiteMinder, one could use a CA.SM::Rule.TimeRestriction object to enforce time-based authorization and generated OnAccessReject events. When migrating over OnAccessReject events, the CA.SM::Rule object could be parsed looking for those conditions, such as TimeRestriction, and equivalent conditions in OpenAM could be built, such as:

AG_200581_ABC-51871258-fc80-440f-9e87-ac448b5edcf5

Resources

Actions

Subjects

Environments

Response attributes



Specify the environment conditions to which the policy applies.

All of...



Type

LDAP Filter Condition

LDAP Filter

(&(objectclass=inetorgperson)(uid=*))



Type

Time (day, date, time, and timezone)

Start Time

11:30

End Time

12:00

Start Day

fri

End Day

sun

Time Zone

GMT




+ Add an Environment Condition

+ Add a Logical Operator

OpenAM Policy Subjects

The program converts every SiteMinder policy with an assumption that only Authenticated Users have access to it:



Delete

Help

POLICY | Active

AG_200581_ABC-06f44409-b846-4b8e-870a-fb217e3151e8

Resources

Actions

Subjects

Environments

Response attributes

Specify the subject conditions to which the policy applies.

All of...


Type

Authenticated Users

+ Add a Subject Condition

+ Add a Logical Operator

However, if the program finds additional subject restrictions such as LDAP filter audience restrictions, those are applied instead:



Delete

Help

POLICY | Active

AG_200581_ABC-51871258-fc80-440f-9e87-ac448b5edcf5

Resources

Actions

Subjects

Environments

Response attributes

Specify the environment conditions to which the policy applies.

All of...

Type

LDAP Filter Condition

LDAP Filter
(&(objectclass=inetorgperson)(uid=*))

+ Add an Environment Condition


+ Add a Logical Operator

Subject Restrictions

The program resolves subject restrictions such as group memberships in the SiteMinder extract below:

```
▼<Object Class="CA.SM::UserPolicy" Xid="CA.SM::UserPolicy@1f-406aac75-9398-4
28T15:28:29" UpdatedBy="siteminder" UpdateMethod="GUI">
  ▼<Property Name="CA.SM::UserPolicy.PolicyFlags">
    <NumberValue>0</NumberValue>
  </Property>
  ▼<Property Name="CA.SM::UserPolicy.FilterPath">
    <StringValue>cn=policyRule,ou=groups,dc=forgerock,dc=com</StringValue>
  </Property>
  ▼<Property Name="CA.SM::UserPolicy.UserDirectoryLink">
    ▼<LinkValue>
      <XREF>Ref00006</XREF>
    </LinkValue>
  </Property>
  ▼<Property Name="CA.SM::UserPolicy.FilterClass">
    <StringValue>groupOfUniqueNames</StringValue>
  </Property>
  ▼<Property Name="CA.SM::UserPolicy.PolicyResolution">
    <NumberValue>2</NumberValue>
  </Property>
</Object>
```

The group DN's are resolved into an LDAP group first, and then into the format needed for XACML packaging. The resulting set of subject restrictions in an OpenAM policy are shown here:



Delete

Help

POLICY | Active

AG_200581_ABC-51871258-fc80-440f-9e87-ac448b5edcf5

Resources

Actions

Subjects

Environments

Response attributes

Specify the subject conditions to which the policy applies.

All of...

Type	User Subjects	Group Subjects	
Users & Groups		policyRule	
Users & Groups		TestA	

+ Add a Subject Condition

+ Add a Logical Operator

Response Attributes

Responses can be of several different types in SiteMinder. Examples include complex key-value combinations such as:

```
CL_Header=<%userattr="<%userattr="CL_Cookie_Value"%>|UID=<%userattr="DEF_PROFILE_ID"%>|State=<%userattr="S
T"%>
```

or, could be specified as a redirect link meant to redirect the user to:

```
www.example.com/requestaccess.jsp
```

or, a static value such as:

```
uid=administrator.
```

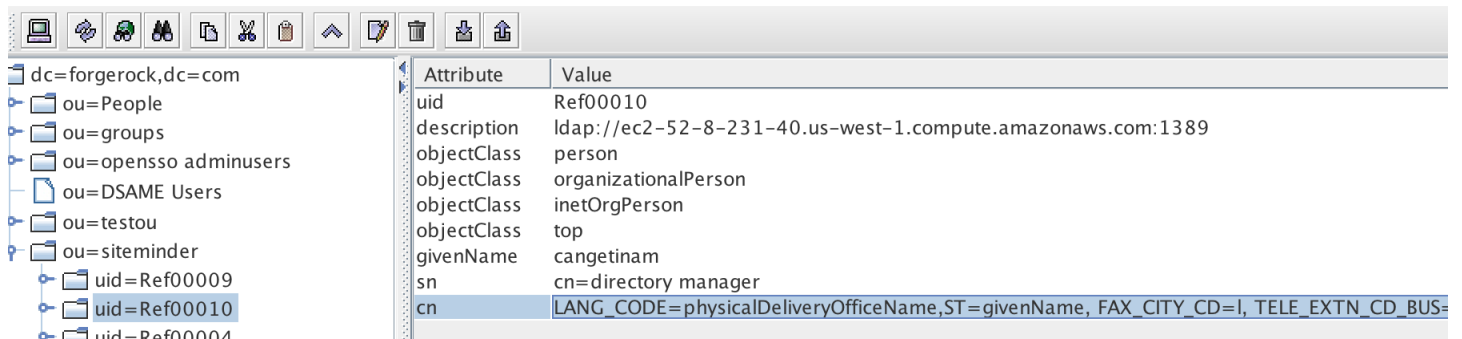
Currently, the program does not support redirect links on OnAccessReject or conditions under OnAuthAccept. Some of these are simply redundant in the OpenAM policy framework, while others require setting up specific agent profile configurations.

An example of dynamically resolving response attributes from the CA.SM::ResponseAttr.Value object is

shown here. Suppose the following response definition was encountered:

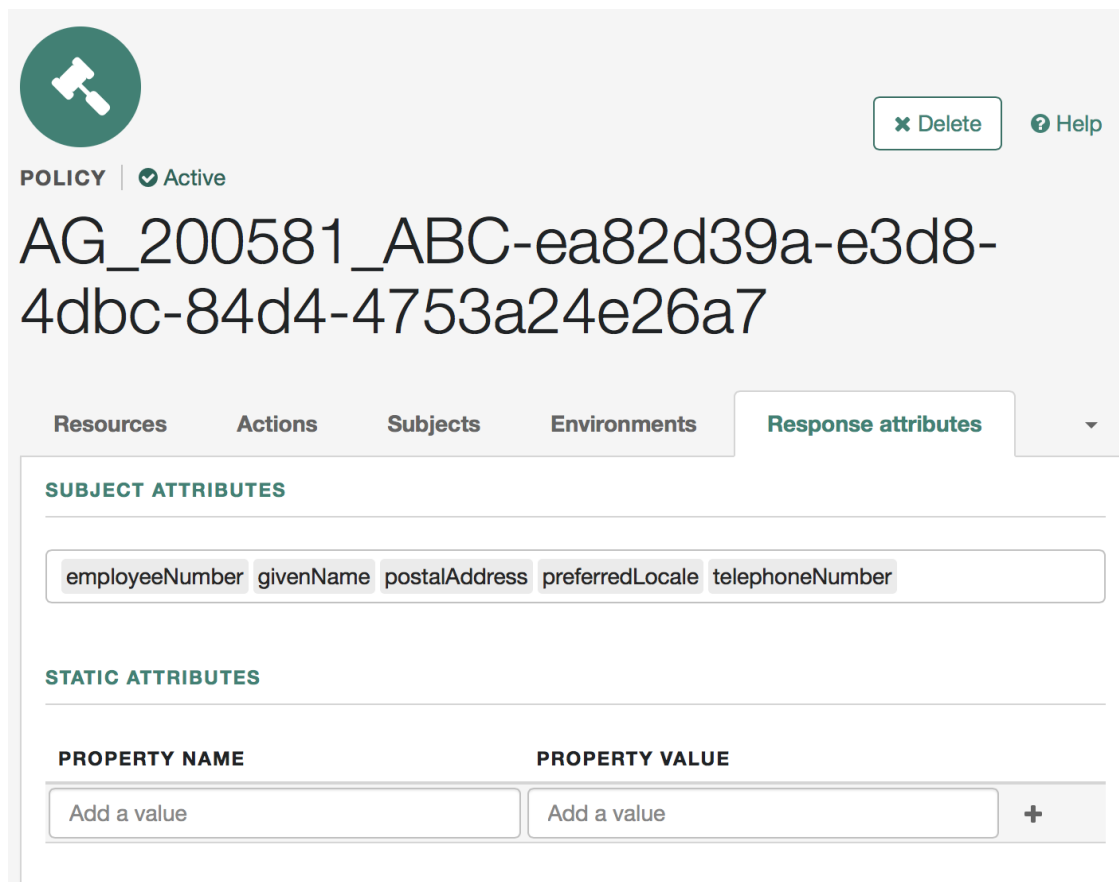
```
EMP_ID=<%userattr="EMP_ID"%>
|State=<%userattr="ST"%>|City=<%userattr="ADDR_CITY_BUS"%>|AddressLine1=<%userattr="ADDR_NM_BUS"%>|
AddressLine2=<%userattr="ADDR_NM2_BUS"%>|CompanyName=<%userattr="COMPANY_NM"%>
|PostalCode=<%userattr="ZIP_POSTAL_CD_BUS"%>|TelephoneCityCode=<%userattr="TELE_CITY_CD_BUS"%>
|TelephoneNumber=<%userattr="TELE_PHONE_NR_BUS"%>|TelephoneExtension=<%userattr="TELE_EXTN_CD_BUS"%>
```



The following metadata container is used to resolve the LDAP attributes (assume that the source attributes in the definition above, are resolved into LDAP attributes below):



Attribute	Value
uid	Ref00010
description	ldap://ec2-52-8-231-40.us-west-1.compute.amazonaws.com:1389
objectClass	person
objectClass	organizationalPerson
objectClass	inetOrgPerson
objectClass	top
givenName	cangetinam
sn	cn=directory manager
cn	LANG_CODE=physicalDeliveryOfficeName,ST=givenName, FAX_CITY_CD=I, TELE_EXTN_CD_BUS=

The resulting set of response attributes packaged into the policy is:



 **POLICY** |  Active ✕ Delete 🔗 Help

AG_200581_ABC-ea82d39a-e3d8-4dbc-84d4-4753a24e26a7

Resources Actions Subjects Environments **Response attributes**

SUBJECT ATTRIBUTES

employeeNumber givenName postalAddress preferredLocale telephoneNumber

STATIC ATTRIBUTES

PROPERTY NAME	PROPERTY VALUE
<input type="text" value="Add a value"/>	<input type="text" value="Add a value"/>