AuthZForce - User and Programmers Guide

Release 4.4.1-FIWARE-R4

Cyril Dangerville, Thales Services

January 27, 2016

Contents

AuthZForce is the reference implementation of the Authorization PDP Generic Enabler (formerly called Access Control GE). Indeed, as mandated by the GE specification, this implementation provides an API to get authorization decisions based on authorization policies, and authorization requests from PEPs. The API follows the REST architecture style, and complies with XACML v3.0. XACML (eXtensible Access Control Markup Language) is a OASIS standard for authorization policy format and evaluation logic, as well as for the authorization decision request/response format. The PDP (Policy Decision Point) and the PEP (Policy Enforcement Point) terms are defined in the XACML standard. This GEri plays the role of a PDP.

To fulfill the XACML architecture, you may need a PEP (Policy Enforcement Point) to protect your application, which is not provided here. For REST APIs, we can use the PEP Proxy (Wilma) available in the FIWARE catalogue.

CHAPTER 1

AuthZForce - User and Programmers Guide

AuthZForce is the reference implementation of the Authorization PDP GE. In this regard, it provides an API to manage XACML-based access control policies and provide authorization decisions based on such policies and the context of a given access request. This guide explains how to use the API.

1.1 Background and Detail

This User and Programmers Guide relates to the reference implementation of the Authorization PDP GE which is part of FIWARE Security Architecture. Please find more information about this Generic Enabler in the following Open Specification.

1.2 User Guide

Since the Authorization PDP is a Generic Enabler which provides backend functionality to other applications (e.g. Generic Enablers or end user facing applications) and security administrators, we do not distinguish between the User and Programmers Guide. Please refer to the Programmers Guide section for more information.

1.3 Programmer Guide

AuthZForce provides the following APIs:

- PDP API (PDP = Policy Decision Point in the XACML terminology): provides an API for getting authorization decisions computed by a XACML-compliant access control engine;
- PAP API (PAP = Policy Administration Point in XACML terminology): provides API for managing XACML policies to be handled by the Authorization Service PDP.

The full API (RESTful) is described by a document written in the Web Application Description Language format (WADL) and associated XML schema files available in the source release of Github project 'rest-api-model', more specifically in file `src/main/resources/authz-api.wadl`.

XACML is the main international OASIS standard for access control language and request-response formats, that addresses most use cases of access control. AuthZForce supports the full core XACML 3.0 language; therefore it allows to enforce very generic and complex access control policies.

1.3.1 Attribute-Based Access Control

AuthZForce provides Attribute-Based Access Control. To understand what is meant by "attribute" in the context of access control, below is the list of categories of attributes identified by the XACML standard:

- Subject attributes: the subject is an actor (human, program, device, etc.) requesting access to a resource; attributes may be user ID, Organization, Role, Clearance, etc.
- Resource attributes: the resource is a passive entity (from the access control perspective) on which subject requests to act upon (e.g. data but also human, device, application, etc.); resource attributes may be resource ID, URL, classification, etc.
- Action attributes: the action is the action that the subject requests to perform on the resource (e.g. create, read, delete); attributes may be action ID, parameter A, parameter B, etc.
- Environment attributes: anything else, e.g. current time, CPU load of the PEP/PDP, global threat level, etc.

1.3.2 Domain Management API

The API allows AuthZForce application administrators or administration interfaces to create domains for the users, and remove domains once they are no longer used. This part of the API is described in the Installation and Administration guide. The API also allows users to update certain properties of the domain allocated to them:

- An externalId (optional) for the domain, which users/clients can modify and more easily use as reference, as opposed to the unique and read-only domain ID assigned by the API - once and for all - when the domain is created;
- Root policy reference (mandatory): a policy ID and version constraints expected to match one of the domain's policies and used as the root policy enforced by the domain's PDP. These policies are managed via the Policy Administration API described in the next section;
- A description of the domain (optional).

You may retrieve the current domain properties as follows:

- Method: GET
- Path: /domains/{domainId}/properties
- **Headers:**
  - Accept: application/xml; charset=UTF-8

For example, this request updates the externalId and the root policy reference to some policy 'PolicyABC' that must exist in the domain (added via the PAP API mentioned later) as a prerequisite:

```
GET /domains/iMnxv7sDEeWFwqVFFMDLTQ/properties
HTTP/1.1
Accept: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns4:domainProperties
  xmlns:ns4="http://authzforce.github.io/rest-api-model/xmlns/authz/4">
  <rootPolicyRef Version="1.0">PolicyABC</rootPolicyRef>
</ns4:domainProperties>
```

You may update the domain properties as follows:

- Method: PUT
- Path: /domains/{domainId}/properties

- Headers:
  - Content-Type: application/xml; charset=UTF-8
  - Accept: application/xml; charset=UTF-8
- Body: new properties.

For example, this request sets/updates the externalId to *my-domain-123* and the root policy reference to some policy *PolicyABC* (in version *2.1*) that must exist in the domain (added via the PAP API mentioned later) as a prerequisite:

```
PUT /domains/iMnxv7sDEeWFwqVFFMDLTQ/properties
HTTP/1.1
Accept: application/xml; charset=UTF-8
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns4:domainProperties
  xmlns:ns4="http://authzforce.github.io/rest-api-model/xmlns/authz/4"
  externalId="my-domain-123">
  <rootPolicyRef Version="2.1">PolicyDEF</rootPolicyRef>
</ns4:domainProperties>
```

Note that the *Version* attribute is optional here. If omitted, the latest version available is used. The response is the new properties.

As a result, the policy now enforced by the domain's Policy Decision Point (see the PDP API in the last section of this document) is *PolicyABC* (in version *2.1*) and the domain's external ID *my-domain-123* points to the domain *iMnxv7sDEeWFwqVFFMDLTQ*. Clients may only rely on the externalId under their control to recover the API-defined domain ID, before they begin to use other API operations that require the API-defined domain ID. Indeed, clients may request the API-defined ID corresponding to a given externalId as follows:

```
GET /domains?externalId=my-domain-123

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:resources
    xmlns:ns2="http://authzforce.github.io/rest-api-model/xmlns/authz/4"
    xmlns:ns3="http://www.w3.org/2005/Atom">
    <ns3:link rel="item" href="iMnxv7sDEeWFwqVFFMDLTQ" title="iMnxv7sDEeWFwqVFFMDLTQ"/>
</ns2:resources>
```

1.3.3 Policy Administration API

The PAP is used by policy administrators to manage the policy repository from which the PDP loads the enforced policies. The PAP supports multi-tenancy in the form of generic administration domains that are separate from each other. Each policy administator (except the Superadmin) is in fact a domain administrator, insofar as he is allowed to manage the policy for one or more specific domains. Domains are typically used to support isolation of tenants (one domain per tenant).

Adding Policies

The PAP provides a RESTful API for adding policies to a specific domain.HTTP requests to this API must be formatted as follows:

- Method: POST
- Path: /domains/{domainId}/pap/policies
- **Headers:**

  - Content-Type: application/xml; charset=UTF-8
  - Accept: application/xml; charset=UTF-8
- Body: XACML PolicySet as defined in the XACML 3.0 schema.

Example of request given below:

```
POST /domains/iMnxv7sDEeWFwqVFFMDLTQ/pap/policies
HTTP/1.1
Accept: application/xml; charset=UTF-8
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" PolicySetId="P1"
 Version="1.0"
 PolicyCombiningAlgId="urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit">
 <Description>Sample PolicySet</Description>
 <Target />
 <Policy PolicyId="MissionManagementApp" Version="1.0"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit">
  <Description>Policy for MissionManagementApp</Description>
  <Target>
   <AnyOf>
    <AllOf>
     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#string">MissionManagementApp</AttributeValue>
       <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
        DataType="http://www.w3.org /2001/XMLSchema#string"
        MustBePresent="true" />
      </Match>
     </AllOf>
    </AnyOf>
   </Target>
  <Rule RuleId="MissionManager_role_can_manage_team" Effect="Permit">
   <Description>Only MissionManager role authorized to manage the mission team</Description>
   <Target>
    <AnyOf>
     <AllOf>
      <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#string">Team</AttributeValue>
       <AttributeDesignator
        Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
        AttributeId="urn:thales:xacml:2.0:resource:sub-resource-id"
        DataType="http://www.w3.org/2001/XMLSchema#string"
        MustBePresent="true" />
      </Match>
     </AllOf>
    </AnyOf>
    <AnyOf>
     <AllOf>
      <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#string">manage</AttributeValue>
       <AttributeDesignator
        Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
```

```
        DataType="http://www.w3.org/2001/XMLSchema#string"
        MustBePresent="true" />
      </Match>
     </AllOf>
    </AnyOf>
   </Target>
   <Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
     <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
      <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#string">MissionManager</AttributeValue>
      <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
       DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="false"
       Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" />
    </Apply>
   </Condition>
  </Rule>
 </Policy>
</PolicySet>
```

The HTTP response status is 200 with a link to manage the new policy, if the request was successfull. The link is made of the policy ID and version separated by '/'.

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:link xmlns:ns3="http://www.w3.org/2005/Atom"
  rel="item" href="P1/1.0" title="Policy 'P1' v1.0"/>
```

Getting Policies and Policy Versions

Once added to the domain as shown previously, you can get the policy by its ID as follows:

- Method: GET
- Path: /domains/{domainId}/pap/policies/{policyId}
- **Headers:**
  - Accept: application/xml; charset=UTF-8

For example:

```
GET /domains/iMnxv7sDEeWFwqVFFMDLTQ/pap/policies/P1
HTTP/1.1
Accept: application/xml; charset=UTF-8
```

The response is the list of links to the versions of the policy available in the domain:

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:resources
  xmlns:ns2="http://authzforce.github.io/rest-api-model/xmlns/authz/4"
  xmlns:ns3="http://www.w3.org/2005/Atom">
    <ns3:link rel="item" href="1.0"/>
```

```
    <ns3:link rel="item" href="1.1"/>
    <ns3:link rel="item" href="2.0"/>
    <ns3:link rel="item" href="2.1"/>
    <ns3:link rel="item" href="2.2"/>
    ...
</ns2:resources>
```

Therefore, you may get a specific version of the policy as follows:

- Method: GET
- Path: /domains/{domainId}/pap/policies/{policyId}/{version}
- **Headers:**
  - Accept: application/xml; charset=UTF-8

For example:

```
GET /domains/iMnxv7sDEeWFwqVFFMDLTQ/pap/policies/P1/1.0
HTTP/1.1
Accept: application/xml; charset=UTF-8
```

The response is the policy document (XACML PolicySet) in this version.

Last but not least, you may get all policies in the domain as follows:

- Method: GET
- Path: /domains/{domainId}/pap/policies
- Headers:
  - Accept: application/xml; charset=UTF-8

For example:

```
GET /domains/iMnxv7sDEeWFwqVFFMDLTQ/pap/policies
HTTP/1.1
Accept: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:resources
  xmlns:ns2="http://authzforce.github.io/rest-api-model/xmlns/authz/4"
  xmlns:ns3="http://www.w3.org/2005/Atom">
    <ns3:link rel="item" href="root"/>
    <ns3:link rel="item" href="P1"/>
    <ns3:link rel="item" href="P2"/>
    ...
</ns2:resources>
```

Removing Policies and Policy Versions

You may remove a policy version from the domain as follows:

- Method: DELETE
- Path: /domains/{domainId}/pap/policies/{policyId}/{version}
- **Headers:**
  - Accept: application/xml; charset=UTF-8

For example:

```
DELETE /domains/iMnxv7sDEeWFwqVFFMDLTQ/pap/policies/P1/1.0
HTTP/1.1
Accept: application/xml; charset=UTF-8
```

The response is the removed policy document (XACML PolicySet) in this version.

You may remove all versions of a policy from the domain as follows:

- Method: DELETE
- Path: /domains/{domainId}/pap/policies/{policyId}
- **Headers:**
  - Accept: application/xml; charset=UTF-8

For example:

```
DELETE /domains/iMnxv7sDEeWFwqVFFMDLTQ/pap/policies/P1
HTTP/1.1
Accept: application/xml; charset=UTF-8
```

The response is the list of links to all the removed versions of the policy, similar to the the GET request on the same URL.

Re-usable Policies (e.g. for Hierarchical RBAC)

The PAP API supports policies that have references to other policies existing in the domain. This allows to include/reuse a given policy from multiple policies, or multiple parts of the same policy, by means of XACML <PolicySetIdReference>s. One major application of this is Hierarchical RBAC. You can refer to the ''Core and hierarchical role based access control (RBAC) profile of XACML v3.0" specification for how to achieve Hierarchical RBAC with <PolicySetIdReference>s.

For example, I want to define a role *Employee* and a role *Manager* derived from *Employee*. In other words, permissions of an *Employee* are included in the permissions of a *Manager*.

In order to create this role hierarchy, we first add the Employee's *Permission PolicySet*:

```
POST /domains/iMnxv7sDEeWFwqVFFMDLTQ/pap/policies
HTTP/1.1
Accept: application/xml; charset=UTF-8
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<PolicySet PolicySetId="PPS:Employee" Version="1.0"
 PolicyCombiningAlgId="urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit">
 <Description>Permissions specific to the Employee role</Description>
 <Target />
 <Policy PolicyId="PP:Employee" Version="1.0"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit">
  <Target />
  <Rule RuleId="Permission_to_create_issue_ticket" Effect="Permit">
   <Target>
    <AnyOf>
     <AllOf>
      <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">https://acme.com/tickets</AttributeValue>
       <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
```

```
        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
        DataType="http://www.w3.org/2001/XMLSchema#string"
        MustBePresent="true" />
      </Match>
     </AllOf>
    </AnyOf>
    <AnyOf>
     <AllOf>
      <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">POST</AttributeValue>
       <AttributeDesignator
        Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
      </Match>
     </AllOf>
    </AnyOf>
   </Target>
  </Rule>
 </Policy>
</PolicySet>
```

Then we add the role-based hierarchical policy defining the Employee role and the Manager role, both with a reference (<PolicySetIdReference>) to the Employee's *Permission PolicySet* added previously; except the Manager role one policy more, so more permissions:

```
POST /domains/iMnxv7sDEeWFwqVFFMDLTQ/pap/policies
HTTP/1.1
Accept: application/xml; charset=UTF-8
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 PolicySetId="rbac:policyset" Version="1.0"
 PolicyCombiningAlgId="urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit">
 <Description>Root PolicySet</Description>
 <Target />
 <PolicySet PolicySetId="RPS:Employee" Version="1.0"
  PolicyCombiningAlgId="urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit">
  <Description>Employee Role PolicySet</Description>
  <Target>
   <AnyOf>
    <AllOf>
     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#string">Employee</AttributeValue>
      <AttributeDesignator
       Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
       AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
       DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
     </Match>
    </AllOf>
   </AnyOf>
  </Target>
  <PolicySetIdReference>PPS:Employee</PolicySetIdReference>
 </PolicySet>
 <PolicySet PolicySetId="RPS:Manager" Version="1.0"
```

```
  PolicyCombiningAlgId="urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit">
  <Description>Manager Role PolicySet</Description>
  <Target>
   <AnyOf>
    <AllOf>
     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Manager</AttributeValue>
      <AttributeDesignator
       Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
       AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
       DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
     </Match>
    </AllOf>
   </AnyOf>
  </Target>
  <Policy PolicyId="PP1:Manager" Version="1.0"
   RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit">
   <Description>Permissions specific to Manager Role</Description>
   <Target />
   <Rule RuleId="Permission_to_create_new_project" Effect="Permit">
    <Target>
     <AnyOf>
      <AllOf>
       <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">https://acme.com/projects</AttributeValue>
        <AttributeDesignator
         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
         AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
         DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
       </Match>
      </AllOf>
     </AnyOf>
     <AnyOf>
      <AllOf>
       <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">POST</AttributeValue>
        <AttributeDesignator
         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
         AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
         DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
       </Match>
      </AllOf>
     </AnyOf>
    </Target>
   </Rule>
  </Policy>
  <!-- This role is senior to the Employee role,
    therefore includes the Employee role Permission PolicySet -->
  <PolicySetIdReference>PPS:Employee</PolicySetIdReference>
 </PolicySet>
</PolicySet>
```

You may add more policies for more roles as you wish. Once you are satisfied with your role hierarchy, you may apply your new RBAC policy by updating the domain's root policy reference (this may not be necessary if you reused the same root policy ID as before, in which case your policy is already active by now):

```
PUT /domains/iMnxv7sDEeWFwqVFFMDLTQ/properties
HTTP/1.1
Accept: application/xml; charset=UTF-8
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns4:domainProperties xmlns:ns4="http://authzforce.github.io/rest-api-model/xmlns/authz/4">
  <rootPolicyRef>rbac:policyset</rootPolicyRef>
</ns4:domainProperties>
```

The policy is now enforced by the PDP as described in the next section.

1.3.4 Policy Decision API

The PDP API returns an authorization decision based on the currently enforced policy, access control attributes provided in the request and possibly other attributes resolved by the PDP itself. The Authorization decision is typically Permit or Deny. The PDP is able to resolve extra attributes not provided directly in the request, such as the current date/time (environment attribute).

The PDP provides an HTTP RESTful API for requesting authorization decisions. The HTTP request must be formatted as follows:

- Method: POST
- Path: /domains/{domainId}/pdp
- **Headers:**
  - Content-Type: application/xml; charset=UTF-8
  - Accept: application/xml; charset=UTF-8
- Body: XACML Request as defined in the XACML 3.0 schema.

The HTTP response body is a XACML Response as defined in the XACML 3.0 schema.

Example of request given below:

```
POST /domains/iMnxv7sDEeWFwqVFFMDLTQ/pdp
HTTP/1.1
Host: 127.0.0.1:8080
Accept: application/xml; charset=UTF-8
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/xml; charset=UTF-8
Content-Length: 954

<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<Request xmlns='urn:oasis:names:tc:xacml:3.0:core:schema:wd-17'
 CombinedDecision="false"
 ReturnPolicyIdList="false">
 <Attributes
  Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
  <Attribute AttributeId='urn:oasis:names:tc:xacml:1.0:subject:subject-id'
   IncludeInResult="false">
   <AttributeValue
    DataType='http://www.w3.org/2001/XMLSchema#string'>joe</AttributeValue>
  </Attribute>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
   IncludeInResult="false">
```

```
   <AttributeValue
    DataType='http://www.w3.org/2001/XMLSchema#string'>Manager</AttributeValue>
  </Attribute>
 </Attributes>
 <Attributes
  Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
  <Attribute AttributeId='urn:oasis:names:tc:xacml:1.0:resource:resource-id'
   IncludeInResult="false">
   <AttributeValue
    DataType='http://www.w3.org/2001/XMLSchema#string'>MissionManagementApp</AttributeValue>
  </Attribute>
  <Attribute
   AttributeId='urn:thales:xacml:2.0:resource:sub-resource-id'
   IncludeInResult="false">
   <AttributeValue DataType='http://www.w3.org/2001/XMLSchema#string'>Team</AttributeValue>
  </Attribute>
 </Attributes>
 <Attributes
  Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
  <Attribute AttributeId='urn:oasis:names:tc:xacml:1.0:action:action-id'
   IncludeInResult="false">
   <AttributeValue
    DataType='http://www.w3.org/2001/XMLSchema#string'>manage</AttributeValue>
  </Attribute>
 </Attributes>
 <Attributes
  Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment" />
</Request>
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
   <Result>
       <Decision>Permit</Decision>
       <Status>
           <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok" />
       </Status>
   </Result>
</Response>
```

1.3.5 Integration with the IdM GE (e.g. for OAuth)

The easy way to integrate with IdM is to delegate the integration to the PEP up-front, i.e. we assume the PEP got all the required IdM-related info and forwards it to the Authorization PDP in the XACML request; the PEP Proxy by UPM can provide such a feature.

1.3.6 Software Libraries for clients of AuthZForce or other Authorization PDP GEis

The full API (RESTful) is described by a document written in the Web Application Description Language format (WADL) and associated XML schema files available in the source release of Github project 'rest-api-model', more specifically in file `src/main/resources/authz-api.wadl`. Therefore, you can use any WADL-supporting

REST framework for clients; for instance in Java: Jersey, Apache CXF. From that, you can use WADL-to-code generators to generate your client code. For example in Java, 'wadl2java' tools allow to generate code for JAX-RS compatible frameworks such as Apache CXF and Jersey. Actually, we can provide a CXF-based Java library created with this tool to facilitate the development of clients.