

Understanding XACML Policy Language - XACML (Extended Assertion Markup Language) - Part 2

By *amilaj.wso2.com*

Created 2011-12-04 03:02

Introduction

Let's first define a suitable use case for demonstration. Then we will define a sample XACML policy to satisfy authorization of defined use case. We will also discuss how you can use XACML testing tools which comes with ?WSO2 Identity Server?.

Finally we will walk through on how one can use ?WSO2 Identity Server? XACML UI to define new authorization policies.

Applies To

WSO2 Identity Server [1]	3.2.0
WSO2 Identity Server [1]	3.2.2
WSO2 Identity Server [1]	3.2.3

Contents

- [Use Case](#) [2]
- [Implementation](#) [3]
- [Using ?WSO2 Identity Server? XACML UI](#) [4]
- [Summary](#) [5]

Use Case

I have a web service called ?BankService?. ?BankService? has 2 operations.

- **deposit (int accountNumber, double amount)**
- **withdrew (int accountNumber, double amount)**

The deposit operation is allowed to ?managers? and ?executives?. But withdrew operation is only allowed to ?managers?. Thus the ?BankService? web service is deployed in ?wso2.org? domain. ?Manager? and ?Executive? is defined as roles in the system.

Implementation

Steps 1 - Let?s first try to understand the attributes and resources in above use case.

The use case says that above authorization is only applied to ?wso2.org? domain. So we can define that in policy?s target.

Resources in use case are;

- **BankService/deposit** ? Deposit operation
- **BankService/withdrew** ? Withdraw operation

Since we are always invoking a web service we can treat action as ?execute?.

Subjects are ?Manager? and ?Executive?.

Steps 2 - Define the XACML request.

We will build the XACML request based on data gathered in Step 1.

```

<Request>
  <Subject SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>manager</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>BankService/withdrew</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>execute</AttributeValue>
    </Attribute>
  </Action>
  <Environment>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:environment:environment-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>wso2.org</AttributeValue>
    </Attribute>
  </Environment>
</Request>

```

Figure 10

A sample XACML request which suits above use case is defined in Figure 10. The request in Figure 10 will arrive in to the system when a ?manager? tries to ?withdraw? money. In a complete setup there will be a component which will craft above request. In XACML terms we call that component, ?**Policy Enforcement Point** ?. In the case of ?[WSO2 Enterprise Service Bus](#) [6]?, you can use ?**Entitlement Mediator** ? to create XACML requests based on web service requests.

Steps 3 ? Define policy target.

According to use case description, the authorization logic should be evaluated only if request?s domain is identical to ?wso2.org ?. Therefore our policy should be evaluated only if request?s environment-id is equal to ?wso2.org?. We need to define our policy target to cater this.

```
<Target>
<Environments>
<Environment>
<EnvironmentMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">wso2.org</AttributeValue>
<EnvironmentAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:environment:environment-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</EnvironmentMatch>
</Environment>
</Environments>
</Target>
```

Figure 11

Figure 11 depicts the policy target, which we defined for the use case. As you can see we are comparing value ?wso2.org ? with the value in request. To retrieve XACML request value we are using an ?EnvironmentAttributeDesignator ?.

Steps 4 ? Define Rules

As per, use-case description, we can come up with 3 rules.

Rule 1 - If user is a **manager** or an **executive** allow access to resource ?**BankService/deposit** ?.

Rule 2 - If user is a **manager** only allow access to resource ?**BankService/withdrew** ?.

Rule 3 - If user is neither a **manager** nor an **executive** do not allow access to any of the resources.

We can specify rule 1 as following Boolean expression.

And (isResourceEqual(?**BankService/deposit** ?), Or(isManager(), isExecutive())

The XACML rule which depicts above Boolean expression is shown in Figure 12. Here we are using in-built functions in XACML engine. If the rule is evaluated to true we should ?permit? the operation.

```
<Rule Effect="Permit" RuleId="Rule_On_Deposit">
<Condition>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">BankService/deposit</
AttributeValue>
<ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Apply>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">manager</AttributeValue>
<SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Apply>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">executive</AttributeValue>
<SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Apply>
</Apply>
</Condition>
</Rule>
```

Figure 12

Equivalent Boolean expression for Rule 2 is as follows;

And (isResourceEqual(?**BankService/withdrew** ?), isManager())

Figure 13 depicts relevant rule for above Boolean expression.

```

<Rule Effect="Permit" RuleId="Rule_On_Withdraw">
<Condition>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">BankService/withdraw</
AttributeValue>
<ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Apply>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-is-in">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">manager</AttributeValue>
<SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Apply>
</Apply>
</Condition>
</Rule>

```

Figure 13

To implement Rule 3, we can use an empty deny rule and a rule combining algorithm. Since we set effect of other rules to be ?permit? we can set rule combining algorithm to ?permit-override? (urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides). Empty ?deny? rule would be as follows;

```
<Rule Effect="Deny" RuleId="Deny_Rule" />
```

Now we have all necessary elements to build the complete policy. You can download complete policy from [here](#) [7] .

Steps 5 ? Upload policy to ?WSO2 Identity Server?

Save [policy](#) [7] to a file.

Start ?WSO2 Identity Server? and navigate to Entitlement -> Administration. Then select ?**Import New Entitlement Policy**? from the appearing screen.

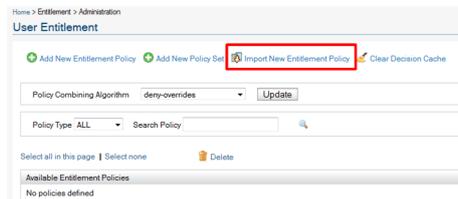


Figure 14

From next screen select ?**Import Entitlement Policy from FileSystem**? and browse and give the file location of the policy file. Then upload the policy.

Once policy is uploaded, it will be in **disable** state. **Therefore, you need to enable the policy as soon as you upload.**

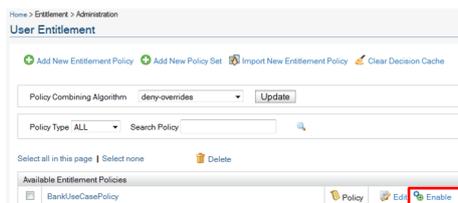


Figure 15

Step 6 ? Testing

To test XACML policy you can use the ?**TryIt**? functionality in ?WSO2 Identity Server?. Navigate to ?Entitlement -> TryIt?.

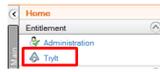


Figure 16

From the appearing screen you can create a simple XACML request by filling necessary details. But this does not have an option to specify ?Environment?. Therefore we will use ?Request Editor? to create an XACML request.

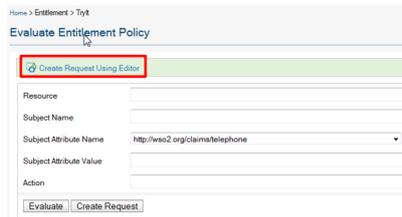


Figure 17

In the request editor you can create the request, evaluate and validate the policy accuracy.

For this use case we will use 4 requests to test.

- Test Deposit Rule ? For this you can use [request-deposit.xml](#) [8]. Copy and paste the content of request-deposit.xml into request editor and evaluate. The evaluation should **permit** the action.
- Test Withdrew Rule ? Use [request-withdrew.xml](#) [9]. This test is similar to deposit test. Here we are specifying only the manager. Request [request-withdrew-deney.xml](#) [10] should be **denied** as we are using ?executive? as the subject and executive is not allowed to withdraw money.
- Negative scenario ? Use [request-negative.xml](#) [11]. A cashier is trying to authorize. The output should be ?**deny** ?.
- Not Applicable ? Use [request-na.xml](#) [12]. Here we are specifying a different domain (test.org) as oppose to wso2.org. So the output should be ?**Not Applicable** ?.

Using ?WSO2 Identity Server? XACML UI

If you have a high level understanding about XACML policy components you can easily use XACML user interface to create new policies. Then you can overcome the hassle of dealing with tedious XML elements.

WSO2 Identity Server provides 2 types of UI?s to create XACML policies.

- Simple UI ? This is simple to use. You only need to focus on authorization attributes. You don?t need to have an in-depth knowledge about XACML policy elements. But functionality is somewhat limited. It is hard to define complex policies with this UI. But this works 90% of the time in web service?s world.
- Advance UI ? You need to have a sound knowledge on XACML to use this UI.

I will be limiting my discussion to ?**simple UI**? as it is mostly used in defining XACML policies and it satisfies the requirement, most of the time.

Defining new XACML policy using "Simple UI"

To define a policy, go to ?Identity Server? Home -> Administration -> ?Add New Entitlement Policy?. Then you will see a screen quite similar to Figure 18.

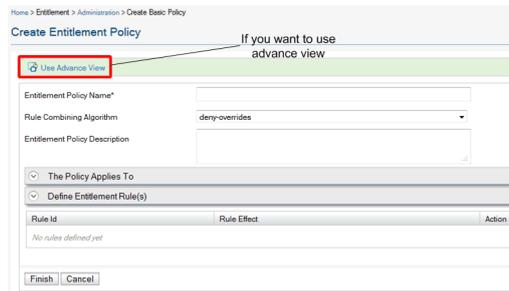


Figure 18

?The Policy Applies To? part defines the policy **Target** . Policy target is built based on the data entered in this section.

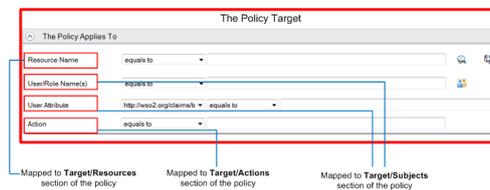


Figure 19

Figure 19 depicts how each attribute is mapped to Policy?s **Target** element.

You can add new rules to policy in section ?Define Entitlement Rule(s)?.

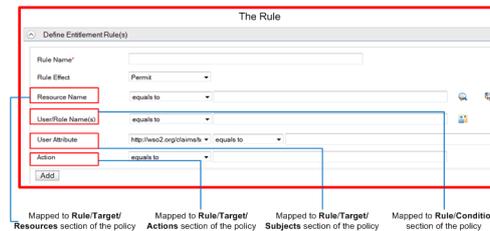


Figure 20

As you can see in Figure 20, all attributes except ?**User/Role Name(s)**? are mapped to an attribute in **Target** element of the rule. The ?**User/Role Name(s)**? is placed inside a **condition** within the rule. Condition is built so that XACML request?s subject id is matched with the policy.

Other fields are quite self explanatory. Therefore I will not go detail into those sections.

Even though simple UI in Identity Server 3.2.0 does not provide an option to define **enviorments**, it's supported in latest verisons. But as you may have noticed, simple UI can only define more general form of policies. Therefore it is not user friendly when we want to define more complex policies.

Summary

In summary this article highlights following concepts.

- After identifying authorization attributes and logic we can implement XACML policy.
- ?WSO2 Identity Server? provides tools to validate and verify XACML policies before they go on production.
- ?WSO2 Identity Server? also provides 2 forms of UIs to define XACML policies easily.
- ?Simple UI? is easy to use and can define more general form of policies. But ?Simple UI? has limitations when it comes to more complex policies.
- Advance UI can be used when we need more complex logic within policies.

[Read the Part 1.](#)^[13]

Attachment	Size
policy.xml ^[7]	2.95 KB
request-deposit.xml ^[8]	1.27 KB
request-na.xml ^[12]	1.27 KB
request-negative.xml ^[11]	1.27 KB
request-withdrew.xml ^[9]	1.27 KB
request-withdrew-deney.xml ^[10]	1.27 KB

[Articles Intermediate Security](#)

© 2012 WSO2 Inc.

footer

- [Home](#)
- [Products](#)
- [Platforms](#)
- [Cloud](#)
- [Solutions](#)
- [OEM](#)
- [Events](#)
- [Contact](#)

• © 2013 WSO2 Inc

- [Legal](#)
- [Privacy](#)

• [Report a problem with this page](#)

```
var rw_ext_id=""; var pkBaseURL = (("https:" == document.location.protocol)
?"https://connect.wso2.com/wso2/" : "http://connect.wso2.com/wso2/");
document.write(unescape("%3Cscript src='" + pkBaseURL + "std/resource/script/rwts.js'
type='text/javascript'%3E%3C/script%3E")); rw_log(pkBaseURL, 4220);
```

Source URL: <http://wso2.org/library/articles/2011/10/understanding-xacml-policy-language-xacml-extended-assertion-markup-language-part-2>

Links:

- [1] <http://wso2.org/library/identity-server>
- [2] <http://wso2.org/library/articles/2011/10/understanding-xacml-policy-language-xacml-extended-assertion-markup-language-part-2#usecase>
- [3] <http://wso2.org/library/articles/2011/10/understanding-xacml-policy-language-xacml-extended-assertion-markup-language-part-2#implement>
- [4] <http://wso2.org/library/articles/2011/10/understanding-xacml-policy-language-xacml-extended-assertion-markup-language-part-2#xacml-ui>
- [5] <http://wso2.org/library/articles/2011/10/understanding-xacml-policy-language-xacml-extended-assertion-markup-language-part-2#summary>
- [6] <http://wso2.com/products/enterprise-service-bus/>
- [7] http://wso2.org/files/policy_27.xml
- [8] <http://wso2.org/files/request-deposit.xml>
- [9] <http://wso2.org/files/request-withdrew.xml>
- [10] <http://wso2.org/files/request-withdrew-deny.xml>
- [11] <http://wso2.org/files/request-negative.xml>
- [12] <http://wso2.org/files/request-na.xml>
- [13] <http://wso2.org/library/articles/2011/10/understanding-xacml-policy-language-xacml-extended-assertion-markup-languge-part-1/>