Policy based Medical Record System – User Guide

This document describes policy based OpenMRS system and it's new features. It helps users to deploy this new system and play around with it. Intended audience for this document are OpenMRS developers and security enthusiasts. In addition to that, this guide expects fair bit of OpenMRS knowledge and it's tools before playing around with this.

1. Configure System

Step 1 : Checkout Source Code
Checkout source code using following subversion command.

```
svn checkout http://xacmlauth.googlecode.com/svn/trunk/openmrs-policy-version openmrs
```

Step 2 : Build
Once source code is checkout in to your local machine, use following maven command to build the system.

```
mvn clean install -Dmaven.test.skip=true
```

Step 3 : Deploy & Installation
Deploy generated war file in to a web container such as Apache Tomcat. Login in to the system using web browser and it will redirects you in to normal OpenMRS installation URL as shown Figure 1.

Figure 1 : Default OpenMRS installation wizard

For this user guide recommend to install OpenMRS simple configurations with demo data. To test the installation, log in system as admin and observe the 'Manage Policies' section.

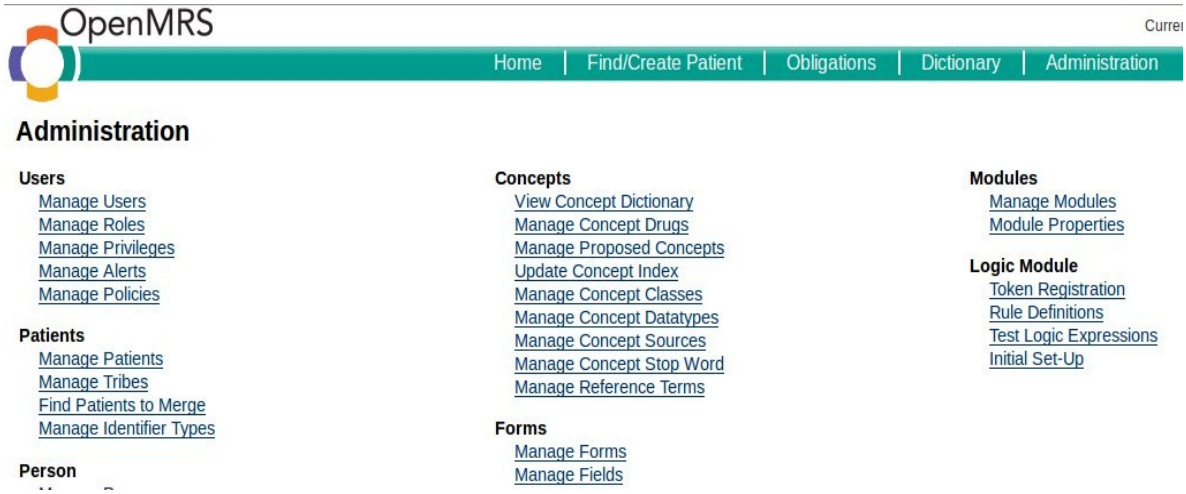

Figure 2 : Administration menu with Manage Policies

Thereafter, create a sample user called 'lasantha' with Clinician role (you can user whatever name or role you like ;)).

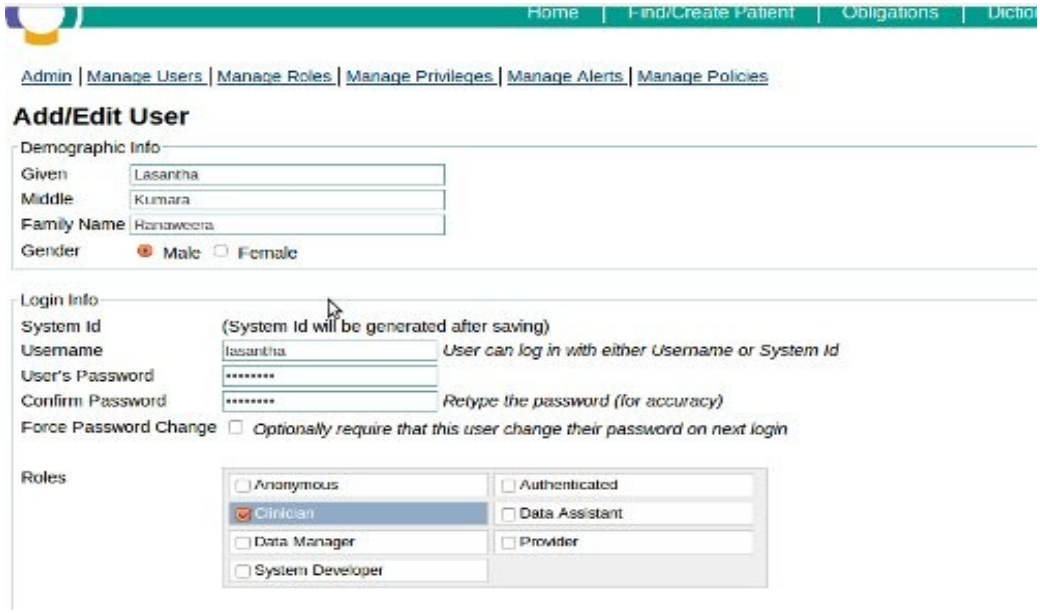

Figure 3 : Creating a sample user

2. Migration and Deploy Policies

When initial configurations are completed, administrator needs to setup system with XACML based policy. This security policy needs to contain all privileges assigned to the users/roles and special conditions to be applied. To help administrator to do this activity easily, 'Policy Migration' functionality is provided. It generates a base security policy with the user/roles and privileges information available in OpenMRS database.

Now we to create a working security policy for the system using 'Policy Migration' in the system. In this section you will see list of users in the system and their roles and virtual roles (important to note virtual role is an extension created with existing roles assigned to a particular user).

System supports policy generation in both Role and User Based generation methodologies, but we recommend to use Role Based policies.

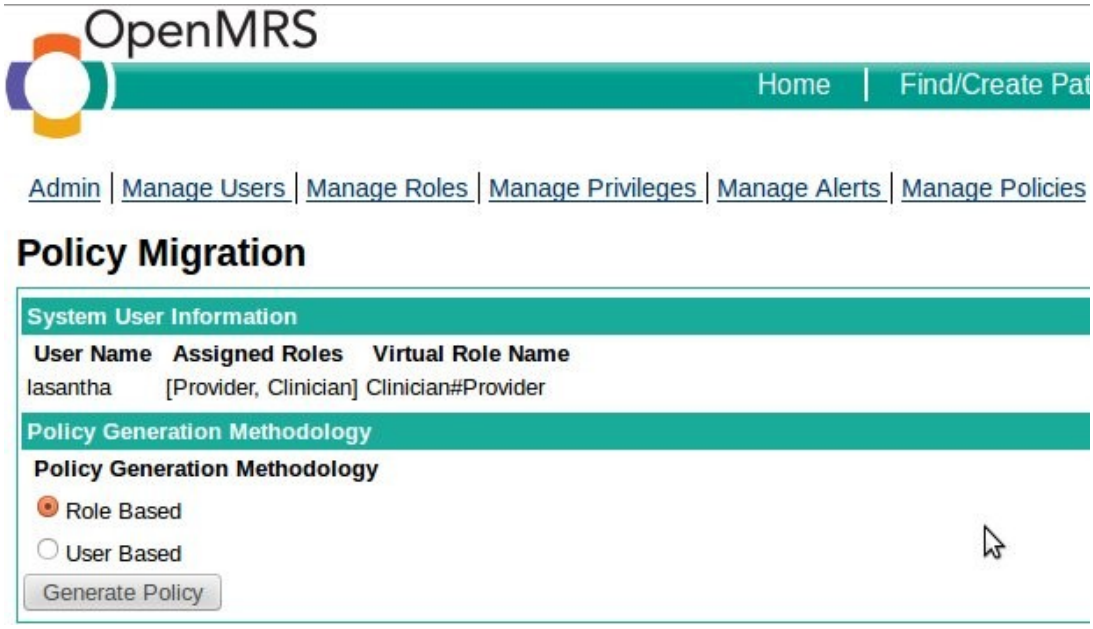

Figure 4 : Generating base security policy

Deploy generated policy in to the system using 'Add Policy' sub menu. Once deployed policies will be displayed as Figure 5.

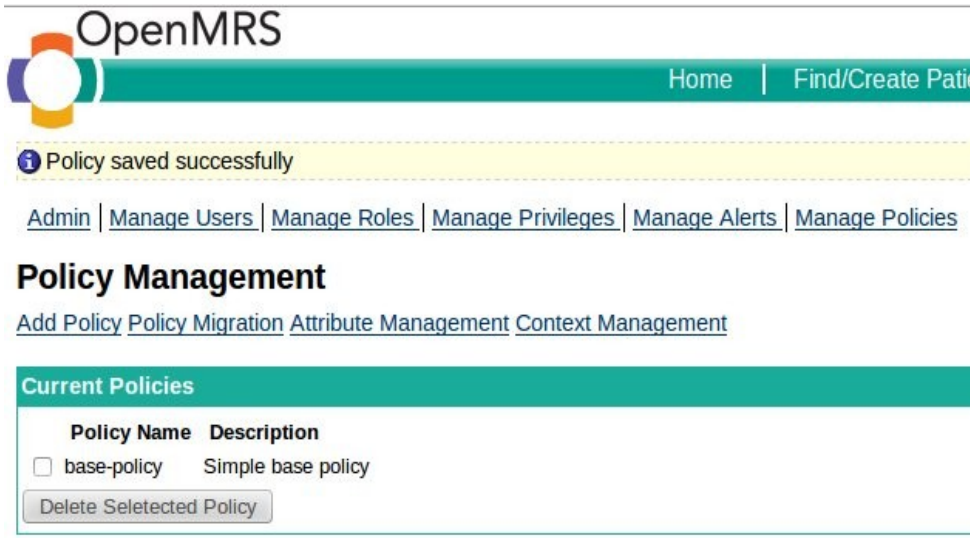

Figure 5 : Deployed Security Policy Information

To test system with new authorization framework, log in to system with user created in last part of configurations section. Then carry out 'Find Patient' → 'Patient Search' while observing your web server logs. Web server logs will display XACML request and response passed in between PEP and PDP of the system.

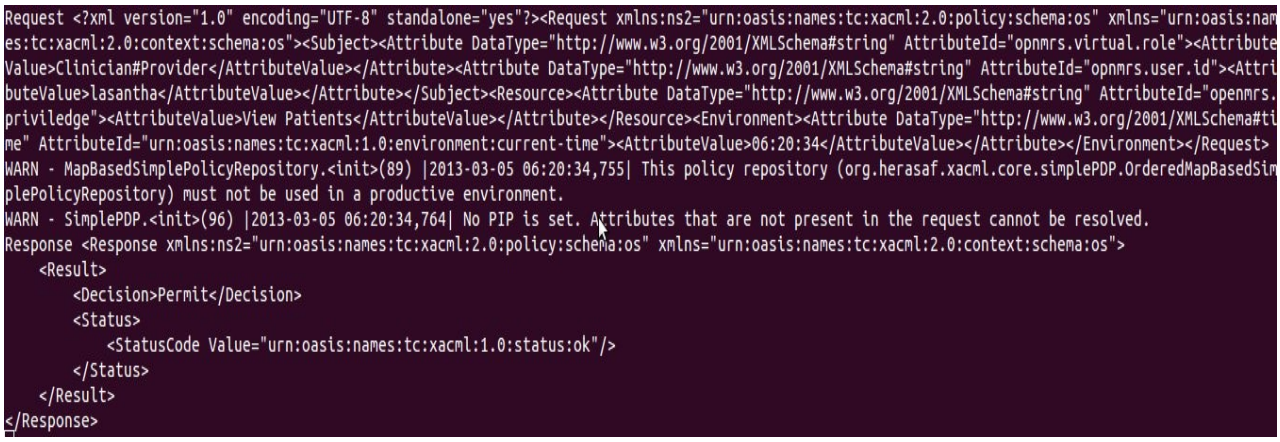

Figure 6 : Web server log displays XACML request and response

3. Environment Attribute Based Policies

Now we going to change base security policy to an environment attribute based condition. XACML support list of environment based attributes in their policies and implementation needs to be pass those information in to PDP for authorization. With this implementation of PEP, system supports system time as an attribute, and we going to add constraint in to 'Find Patient' functionality based on that attribute. So we going to add constraint for Clinician to allow access between 'Find Patient' service 8 a.m to 4 p.m only.

This condition needs to be placed on your generated policy in 'View Patients' privilege section of Clinician.

```
<xacml:Rule Effect="Permit" RuleId="Clinician#Provider_7">
    <xacml:Target>
        <xacml:Resources>
            <xacml:Resource>
                <xacml:ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <xacml:AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">View Patients</xacml:AttributeValue>
                    <xacml:ResourceAttributeDesignator AttributeId="openmrs.priviledge" DataType="http://www.w3.org/2001/XMLSchema#string"/>
                </xacml:ResourceMatch>
            </xacml:Resource>
        </xacml:Resources>
    </xacml:Target>
     <xacml:Condition>
                <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:time-in-range">
                        <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
                                <xacml:EnvironmentAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#time"
teId="urn:oasis:names:tc:xacml:1.0:environment:current-time" />
                        </xacml:Apply>
                        <xacml:AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">08:00:00</xacml:AttributeValue>
                        <xacml:AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">16:00:00</xacml:AttributeValue>
                </xacml:Apply>
        </xacml:Condition>
 </xacml:Rule>
```

Figure 7 : Time based condition with View Patient privilege for Clinician

After modifications, remove existing policy and deploy new modified policy. You can test effect of this policy based on your system time. Once time expired, system will not allow Clinician user to access 'Find Patient' service.

4. Usage attribute based Policies

This part of implementation supports policy attributes, and those attributes can be configured in 'Policy Attribute Management' section in assigned for admin user. Once admin user defined attributes, they are included in to the PEP request and passed PDP for authorization. This allows 'admin' to add more attributes and conditions in to the security policy based on their requirement.

In this implementation, an attribute can be either static or dynamic. Static attributes not change their values though out their life cycle. Dynamic attributes increased their 'Initial Value' by one with each successful usage of system service (in future we will support more than increasing). To demonstrate this feature we going to define a new dynamic attribute and limit 'Find Patient' for Clinician. So with this constraint Clinician is allow to access 'Find Patient' feature n times.

First add read-count dynamic attribute in to the system with given details with 'Add Attribute' sub menu in 'Attribute Management' as shown in Figure 8.

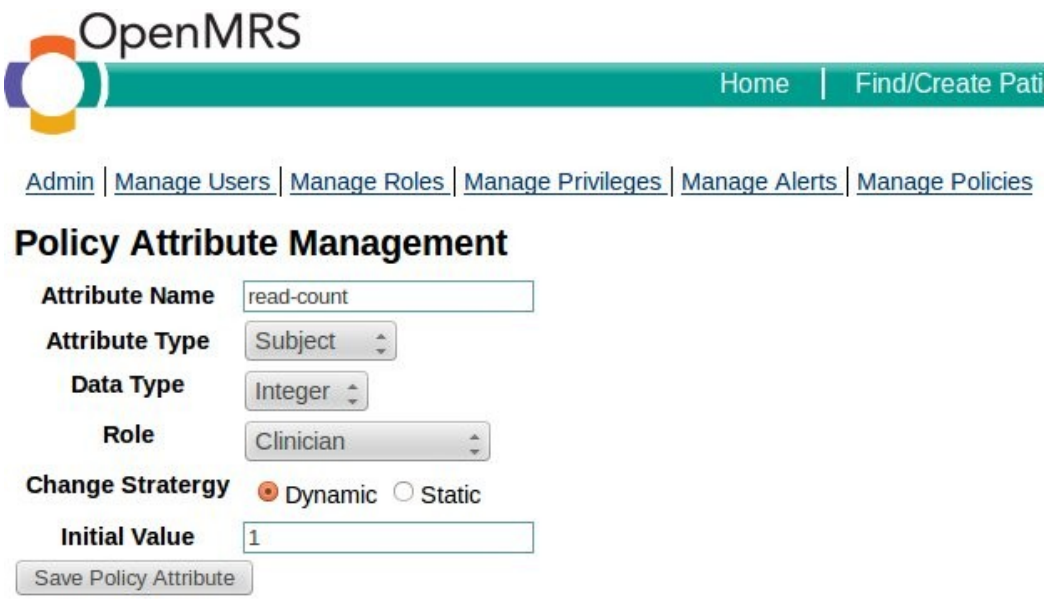

Figure 8 : Defining read-count dynamic policy attribute

Next important section here is incorporate this attribute information with your security policy. This information needs to added to the default policy as depicted in Figure 9. After adding this Condition in to 'View Patient' privilege, admin user needs to deploy new security policy in to the system.

```
<xacml:Rule Effect="Permit" RuleId="Clinician#Provider_6">
    <xacml:Target>
        <xacml:Resources>
            <xacml:Resource>
                <xacml:ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <xacml:AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">View Patients</xacml:AttributeValue>
                    <xacml:ResourceAttributeDesignator AttributeId="openmrs.priviledge" DataType="http://www.w3.org/2001/XMLSchema#string"/>
                </xacml:ResourceMatch>
            </xacml:Resource>
        </xacml:Resources>
    </xacml:Target>
    <xacml:Condition>
        <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-than">
                <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only">
                        <xacml:SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#integer" AttributeId="read-count" />
                </xacml:Apply>
                <xacml:AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">10</xacml:AttributeValue>
        </xacml:Apply>
    </xacml:Condition>
</xacml:Rule>
```

Figure 9 : Extended policy with read-count variable

Once user has exceed it's maximum limit system will display normal error message as given in Figure 10.

Figure 10 : User allows only n times to access service

Note: That each search in 'Find Patient' section increase read_count variable value by two. So if you want to limit it to 5 times per service, value specified in policy needs to be setup as 10.

5. Obligation based policies

Another important feature supports with XACML policies are Obligations. To demonstrate this feature, our default implementation support 'Token Based' obligations in PERMIT conditions. In concrete terms, this feature can only be used when system permits privileges assigned with particular user and expect additional information before using that service.

In demonstration, we going to setup special token Clinician role before using 'Find Patient' service. Admin can define this obligation, 'Context Managemet' definition section in 'Policy Management' as below. With this Obligation definitions admin user setup a secret code 'value' section and this value needs to be shared between valid users in the system.

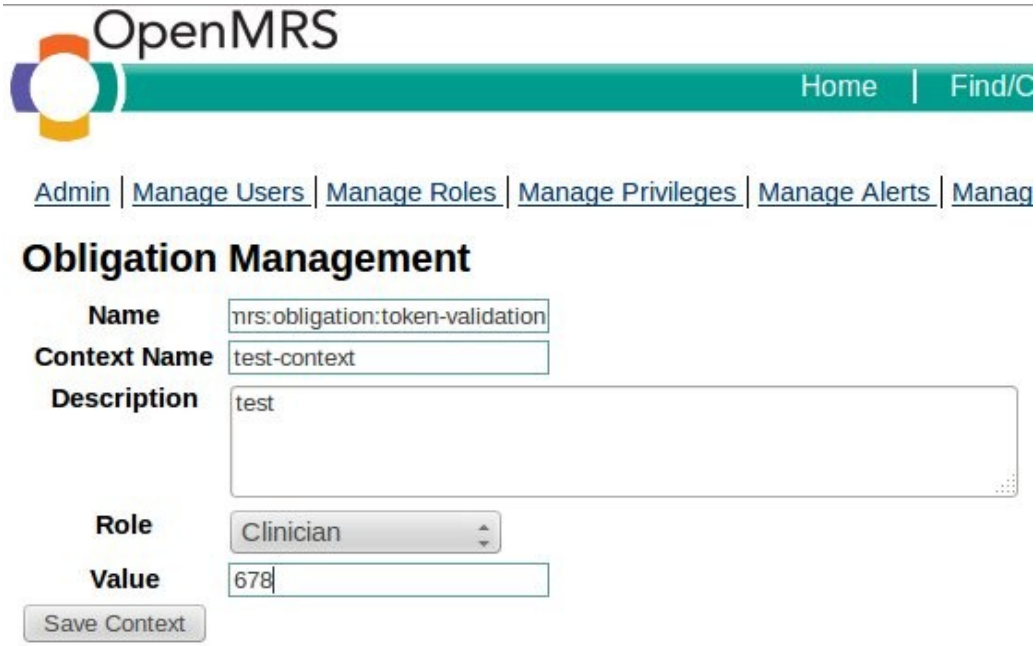

Figure 11 : Set up obligation context with token value

To incorporate this information in to our default policy, admin user needs to add them in to the 'Policy' level of Clinician role as shown in Figure 12. Once this information is included in to the security policy, this new policy needs to be deploy as system policy.

```
</xacml:Policy>
<xacml:Obligations>
    <xacml:Obligation FulfillOn="Permit" ObligationId="urn:oasis:names:tc:xacml:2.0:openmrs:obligation:token-validation">
    </xacml:Obligation>
</xacml:Obligations>
```

Figure 12 : Obligation added with Clinician Policy

Once user receive this secret token, they needs to enter that value using 'Obligations → User Obligation Management' section. If they setup those values correctly, then only those users allow to access necessary service.

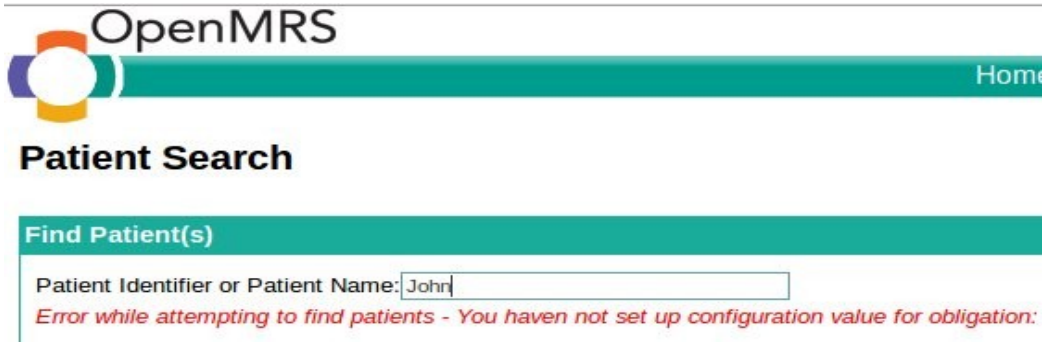

Figure 13 : User not allow to access service without setting up token

User can assign token values from Obligations menu as given in figure 14.

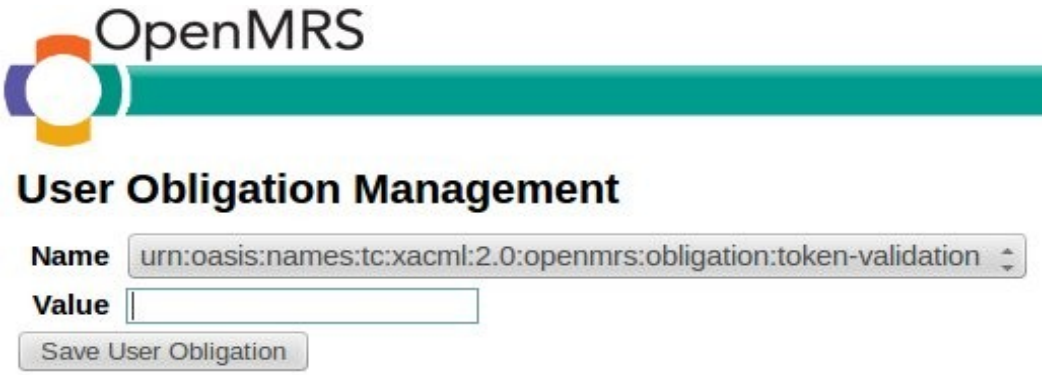

Figure 14 : Clinician user configure their obligation value

Note : When you add Obligation in to the Policy level, it apply in to all Rules defined for a particular role. If it is needed only for one privilege, then extensive Policy modification is needed.

6. TODO List

Fixing all test cases.
Add more features and data type support in to attribute management section.
Add more Obligation types and implement them in generic manner.
Better error reporting if possible.