DATABASE IMPLEMENTATION OF XACML
ROLE-BASED ACCESS CONTROL SPECIFICATION

A Project

Presented to the faculty of the Department of Computer Science
California State University, Sacramento

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE
in
COMPUTER SCIENCE

By
Travis Sorley

FALL
2013
DATABASE IMPLEMENTATION OF XACML

ROLE-BASED ACCESS CONTROL SPECIFICATION

A Project

By

Travis Sorley

Approved by:

__________________________, Committee Chair
Ying Jin, Ph.D.

__________________________, Second Reader
Jinsong Ouyang, Ph.D.

__________________________
Date
Student: Travis Sorley

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

__________________________, Graduate Coordinator
Nikrouz Faroughi, Ph.D.                                  ________________

Department of Computer Science

iv
This project demonstrated a possible solution to implementing a Role Based Access Control (RBAC) security model in a database using XACML as an abstract specification layer. By using this process, we can define the RBAC system separate from any one-enterprise solution and use a translator to do the work of constructing it.

This application consists of three main parts; the Application Engine, the XACML Parser and the SQL Generator. The Application Engine is used to control the different parts of the application and to interface with the user. The XACML parser is used to extract the relevant information from the input files and store that information in memory. The SQL Generator generates the necessary Transact-SQL code upon user command and executes it against the specified database. In this solution, we used Microsoft SQL Server 2012, which currently does not natively support the XACML specification but does support the creation of an RBAC security model. We also used JAVA to create the main application along with the JAVA provided XML Parser to traverse the XACML files. The result of running the application is a database implementation of an RBAC security model defined using the XACMLv3.0 RBAC v1.0 specification.
Committee Chair
Ying Jin, Ph D.

Date
DEDICATION

Dedicated to my family who helped support me financially and emotionally through my college career.
I would like to thank Dr. Ying Jin for her guidance and support to this project.
TABLE OF CONTENTS

Dedication ................................................................................................................... vii
Acknowledgments...................................................................................................... viii
List of Figures .............................................................................................................. xi

Chapter

1. INTRODUCTION .....................................................................................................1

2. BACKGROUND .......................................................................................................3
  2.1 Role Based Access Control (RBAC) ..............................................................3
  2.2 eXtensible Access Control Markup Language (XACML) ......................4
    2.2.1 XACML Policy Language.................................................................4
    2.2.2 XACML Implementation .................................................................6
  2.3 SQL Server / Transact SQL (T-SQL) .............................................................6

3. XACML POLICY DEFINITION FOR ROLE BASED ACCESS CONTROL......8
  3.1 Role Based Access Control (RBAC) XACML Policies. ..............................8
    3.1.1 Role PolicySet (RPS).......................................................................8
    3.1.2 Permission PolicySet (PPS).............................................................9
    3.1.3 Role Assignment Policy (RAP)......................................................10

4. IMPLEMENTING XACML POLICIES IN SQL SERVER ...............................13
  4.1 Extraction of XACML RBAC Policies.......................................................13
    4.1.1 Setup and Requirements...............................................................13
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1 System Role Hierarchy</td>
<td>14</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

In today’s world of digital security and cyber spying, resource security in the enterprise environment is becoming ever more important. Large companies often need to maintain numerous points of access for all of the resources they are trying to protect. The maintenance overhead for these authorization applications can introduce inconsistency in the security policies that they implement. It has been identified that there is a need for an abstraction layer above all of these applications that can be used to define the security policies used within the company as well as providing a single location for maintenance for those policies.

To accomplish this, the policies must be defined in a simple and easy to use language that can be adapted to any enterprise environment the user needs to secure. Once these policies are defined, they can be distributed to each of the authorizing agents for updates reducing the risk of inconsistency in security policies across the enterprise. One proposed solution to this problem is the eXtensible Access Control Markup Language (XACML).

The purpose of this project is to develop an application that accepts Role Based Access Control policies defined in files and implement these policies in a SQL Server database. The files that the application will accept are written using XACML and the application
itself will stand as an example of how existing security software can add support for the XACML specification.
Chapter 2

BACKGROUND

This chapter presents background of Role-based Access Control (RBAC) and the eXtensible Access Control Markup Language (XACML).

2.1 Role Based Access Control (RBAC)

When implementing a security model in an enterprise environment, it is desirable to have a security system that can support the hierarchical nature of a business. Role Based Access Control is a security specification developed to enable enterprises to grant or deny access based on these hierarchical roles. In general, there are three key features or properties a system must have to be considered RBAC compliant.

One of these features is that the system must have role assignments. A user must not be allowed to execute a transaction unless they have been granted access to a role in the system. Another feature is that the role must be active and have the proper authorization to execute the specified command [1]. Lastly, a user can only execute a transaction if the users’ role membership is authorized and there are no constraints on the user preventing the transaction [1]. In this way, roles are treated as collections of permissions and users only have the permission of the roles they belong to and roles they inherit from in the role hierarchy.
The benefit of using an RBAC system in an enterprise is in the management of security policies. In business, large amounts of employees can be moving within an organization at any one time and RBAC allows for fast response to changing access rights. If an employee changes job roles an administrator need only move them from their current role to one that suites their new position and all of the users rights will be updated. This allows for a much faster response to a constantly changing security landscape.

2.2 eXtensible Access Control Markup Language (XACML)

2.2.1 XACML Policy Language

In an enterprise environment, security policies will encompass many different elements and include various points of enforcement. Likewise, different groups within the same company may manage their security policies separately. Enforcement of these security policies will also happen over various platforms spread out throughout the company. One way to manage these security policies is treat each system as an independent entity causing possible points of failure when updating entitlements. The XACML specification has been created to try to solve this issue by providing a central location for defining and maintaining security policies.

The guiding principal behind the creation of XACML is for it to provide a system in which an enterprise can define a security framework in one location and use that as the
root for all other platforms to be based off [2]. To accomplish this, OSASIS build XACML on the well establish and widely supported language Extensible Markup Language (XML). This provides platforms that already accept XML an easy path to adopting XACML.

In this manner, use of XML allows the XACML specification to unify the defining of Roles, Entities and Policies into one decision request. In the case of XACML RBAC, an Entity is identified with and Action or Actions on a Resource while the use of XML provides flexibility to support various enterprise environments [3]. The two areas where these are used are the Role Policy sets and the Permission Policy Sets.

In the Permission Policy Set, a role is assigned an action to a defined resource in the system. Using XACML a user can identify any given action with an <Action> tag and a resource using the <Resource> tag. When these tags are contained in a <Rule> block, the identified Policy Set will be given that permission to the resource.

The Role Policy set allows the users to assign a reference id to a subject in the environment, which the uses wishes to grant access to. XACML uses the tag <Target> to identify that subject in the system. By including the <PolicySetIdReference> tag, the user can now refer to that subject in any of the XACML policies and those rights will be granted to it.
2.2.2 XACML Implementation

In the article “MyABDAC: Compiling XACML Policies for Attribute-Based Database Access Control” [4], the authors performed XACML implementation by compiling XACML policies into Access Control Lists (ACLs) that are natively supported by the database system. The goal being to improve access decision times within the access engine.

Another project supporting XACML policies is Sun’s XACML Implementation [5]. In this project, a complete implementation of XACML is accomplished using JAVA and runs independently of the database management system. This allows the application to provide policy verification at the application layer.

2.3 SQL Server / Transact SQL (T-SQL)

To implement this security specification, this project will use Microsoft SQL Server 2012. SQL Server is a data warehousing solution developed by Microsoft to provide enterprise ready functionality across various platforms [6]. In conjunction with the development of SQL Server, Microsoft developed their own extension to the SQL language called Transact-SQL (T-SQL). T-SQL is designed to provide additional programming support beyond what SQL provides like the use of local variables, procedural programming and text/time manipulation. These additional features along
with modifications to the UPDATE and DELETE commands make T-SQL Turing complete [7].
As stated in the Introduction of this paper, the purpose of this project is to design an application that accepts RBAC policies defined using XACML. Examples of policy files are described below.

3.1 Role Based Access Control (RBAC) XACML Policies

The following policies used in the project are based on the XACML v3.0 RBAC Profile [2]: Role Policy Set, Permission Policy Set and Role Assignment.

3.1.1 Role PolicySet (RPS)

In the Role Policy Set file, we identify our system role and the id that we will use to identify it by. Here we have an example based on the project by Reyes [8] from the RolePolicy_Software_Engineer.xml file listed in Appendix F. Our system role software_engineer is identified as our <Target> and we assign it to the permission policy set id PPS:software_engineer:role.

```xml
<PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-core-v3-schema-wd-17.xsd"...>
```
PolicySetId="RPS:software_engineer:role"
Version="1.0"
PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <Target>
    <AnyOf>
      <AllOf>
        <Match
          MatchId="&function;anyURI-equal">
          <AttributeValue
            DataType="&xml;anyURI">
            &roles;software_engineer
          </AttributeValue>
          <AttributeDesignator
            MustBePresent="false"
            Category="&subject-category;access-subject"
            AttributeId="&role;"
            DataType="&xml;anyURI"/>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>

  <!-- Use permissions associated with the software_engineer role -->
  <PolicySetIdReference>
    PPS:software_engineer:role
  </PolicySetIdReference>
</PolicySet>

3.1.2 Permission PolicySet (PPS)

The second XACML file the application accepts is the Permission Policy Set. In this file, we can grant role permissions on a resource. In this example from Appendix C, we identify the resource as our requirement_doc table within the <Resource> block. After
that, we can then identify the desired rights, in this case SELECT, INSERT, DELETE, UPDATE within the <Action> block.

<!-- Permission to read and write table requirement_doc -->
<Rule
    RuleId="Permission:to:read and write:table:requirement_doc"
    Effect="Permit">
    <Description>
        A software engineer may read or write on table requirement_doc only in projects assigned to him
    </Description>
    <Target>
        <Resource>
            <ResourceMatch MatchId="&function;string-equal">
                <Attribute
                    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                    DataType="&xml;string">
                    <AttributeValue>
                        requirement_doc
                    </AttributeValue>
                </Attribute>
            </ResourceMatch>
        </Resource>
        <Action>
            <Attribute
                AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                DataType="&xml;string">
                <AttributeValue>
                    SELECT, INSERT, DELETE, UPDATE
                </AttributeValue>
            </Attribute>
        </Action>
    </Target>
</Rule>

3.1.3 Role Assignment Policy (RAP)

Finally, our application accepts the Role Assignment Policy file. In this example from Appendix G, we identify our subject Alice with the <AttributeDesignator> tag and the
resource she belongs to, project_manager. After the resource, we can then specify the action we want to take, enable role or disable role.

<!-- project_manager role requirements rule -->
<Rule RuleId="project_manager:role:requirements" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="&function;string-equal">
          <AttributeValue
            DataType="&xml;string">
            Alice
          </AttributeValue>
          <AttributeDesignator
            MustBePresent="false"
            Category="&subject-category;access-subject"
            AttributeId="&subject;subject-id"
            DataType="&xml;string"/>
        </Match>
      </AllOf>
      <AllOf>
        <Match MatchId="&function;string-equal">
          <AttributeValue
            DataType="&xml;string">
            Bob
          </AttributeValue>
          <AttributeDesignator
            MustBePresent="false"
            Category="&subject-category;access-subject"
            AttributeId="&subject;subject-id"
            DataType="&xml;string"/>
        </Match>
      </AllOf>
      <AnyOf>
        <AnyOf>
          <Match MatchId="&function;anyURI-equal">
            <AttributeValue
              DataType="&xml;anyURI">
              &roles:project_manager
            </AttributeValue>
            <AttributeDesignator
              MustBePresent="false"/>
Category="&category:resource"
AttributeId="&role;"
DataType="&xml;anyURI"/>
</Match>
</AllOf>
</AnyOf>

<AllOf>
  <Match MatchId="&function;anyURI-equal">
    <AttributeValue
      DataType="&xml;anyURI">
      &actions;enableRole
    </AttributeValue>
    <AttributeDesignator
      MustBePresent="false"
      Category="&category:action"
      AttributeId="&action;action-id"
      DataType="&xml;anyURI"/>
  </Match>
</AllOf>
</AnyOf>
</Target>
</Rule>
Chapter 4

IMPLEMENTING XACML POLICIES IN SQL SERVER

This chapter presents the application implementation for processing XACML policy definitions.

4.1 Extraction of XACML RBAC Policies

4.1.1 Setup and Requirements

Tables:
The following tables are used in the project (from Case Study Z): code, design_doc, project_plan, requirement_doc, test_case_script and test_log.

Users:
The following users are used in the project (from Case Study Z):
XACMLParser (administrator, owner of tables), Ace, Alice, Bill, bob, Carol, Jack, James, Jane, Joe, Kate, Lily, Lucy, Nancy, Sue, Susan.
Roles:

![System Role Hierarchy](image)

Figure 1 System Role Hierarchy

For this project, the following roles were created: project_chief_manager, project_manager, software_developer, software_engineer, test_design_engineer, test_engineer.

4.1.2 Parsing XACML files

For the purposes of this project, Permission Policies and Role Policies are defined in separate files for each role.

When the user loads a file into the application, it is first passed to the java XML document builder library. This library provides XML navigation methods to assist in extracting the XACML policy definitions. The document object that is created is then used to extract the root element, which will be used to determine the file type.

```java
String permissionPolicyID = docEle.getAttribute("PolicySetId");
String policyID = docEle.getAttribute("PolicyId");
NodeList rules = docEle.getElementsByTagName("Rule");
NodeList match = docEle.getElementsByTagName("Match");
```
NodeList inheritedRules = docEle.getElementsByTagName("PolicySetIdReference");

Using this code, the file type is identified by either the existence of a Policy Set Id or Policy Id. Along with the policy Id’s, lists of all the rule blocks and match blocks for the document are also gathered. These lists are then used as parameters in their individual extraction methods.

If the file is a Permission Policy definition, each rule from the rule list is passed to the extractPermissionPolicy() method. This method is used to extract the resource and action, which along with the Policy Set Id, are added to the list in a Permission Policy object for use later in the application.

If the file is a Role Assignment Policy definition, each rule from the rule list is passed to the extractRoleAssignment() method. This method gathers a list of matches and for each match identifies the type, resource, action or access-subject. When all of the matches are processed for that rule, a record is added to a list in the role policy object for each subject listed in the block. See code example below:

```java
for(int i = 0; i<matches.getLength();i++)
{
    String val =
    getValue((Element)matches.item(i),"AttributeValue").replace("\n","" ).trim();
    String des =
    ((Element)((Element)matches.item(i)).getElementsByTagName("AttributeDesignator").item(0)).getAttribute("Category");
    switch(des){
    case "urn:oasis:names:tc:xacml:3.0:attribute-category:resource":
        resource = val;
        break;
```
case "urn:oasis:names:tc:xacml:3.0:attribute-category:action":
    action = val;
    break;

case "urn:oasis:names:tc:xacml:1.0:subject-category:access-subject":
    subjects[s]=val;
    s++;
    break;
}
}
rolePolicy.addRoleAssignments(subjects, resource, action);

If the file is determined to be a Role Policy definition, each rule from the rule list is passed to the extractRolePolicy() method. This method extracts the AttributeValue element from each rule and adds that to the role list as a map between the Policy Id used in the XACML definitions and the role Id used in the database.

When each file is processed, the bottom of the file is checked for any <PolicySetIdReference> tags. These tags are used in XACML as a way to include rights from other roles that the current role might belong to. If one of these tags is found, it is parsed and added to an inheritance list. This list is later used to construct the role hierarchical structure in the database.

4.2 Generation of Transact-SQL

After all of the files have been passed to the application, the user has the option to generate the SQL code and view it in the command window or run the code against the database. When the users selects either of these options, the inherited list, the
roleAssignment list, the permissionPolicy list and the rolePolicy list are passed to SQL builder.

In the SQL builder, each of the role policies are passed to the getRights() method. This method uses the rolePolicy list to match the role to a database role and inserts the matched record into a SQL GRANT string along with any permissions defined in the permissionPolicy list. This string is appended to the current SQL command and returned when all permission policies have been processed.

After all permission policies have been processed, the role assignments and role policy lists are passed to the getRoleAssignment() method. This method steps through the role assignment list and adds each user and the corresponding SQL role from the role policy list to a SQL ALTER ROLE command. This string is then appended to the current overall SQL command string.

The final step in the SQL builder is the processing of inherited rights. SQL Server supports Role Based Access Control functionality and allows a role to be added to another role as a member. The getAssignments() method accepts the inherited role list and the role policy list for mapping purposes, and inserts the values in a SQL command to execute the sp_addrolemember store procedure. When all assignments are processed, the final SQL command string is returned to the application core and either executed on the database or printed to the command window.

Using the files defined in Appendices A-G, the final SQL output is as follows:

GRANT SELECT ON [requirement_doc] TO [software_engineer];
GRANT INSERT ON [requirement_doc] TO [software_engineer];
GRANT DELETE ON [requirement_doc] TO [software_engineer];
GRANT UPDATE ON [requirement_doc] TO [software_engineer];
GRANT SELECT ON [code] TO [software_engineer];
GRANT SELECT ON [design_doc] TO [software_engineer];
GRANT SELECT ON [test_case_script] TO [software_engineer];
GRANT SELECT ON [test_log] TO [software_engineer];
GRANT SELECT ON [project_plan] TO [software_engineer];
GRANT INSERT ON [project_plan] TO [software_engineer];
GRANT DELETE ON [project_plan] TO [software_engineer];
GRANT UPDATE ON [project_plan] TO [software_engineer];
EXEC sp_addrolemember 'project_manager','project_chief_manager'
EXEC sp_addrolemember 'software_engineer','project_manager'
ALTER ROLE [project_chief_manager] ADD MEMBER [Bill]
ALTER ROLE [project_manager] ADD MEMBER [Alice]
ALTER ROLE [project_manager] ADD MEMBER [Bob]
ALTER ROLE [software_engineer] ADD MEMBER [Ace]
ALTER ROLE [software_engineer] ADD MEMBER [Lily]
ALTER ROLE [software_engineer] ADD MEMBER [James]
ALTER ROLE [software_developer] ADD MEMBER [Joe]
ALTER ROLE [software_developer] ADD MEMBER [Kate]
ALTER ROLE [software_developer] ADD MEMBER [Jane]
ALTER ROLE [test_design_engineer] ADD MEMBER [Susan]
ALTER ROLE [test_design_engineer] ADD MEMBER [Sue]
ALTER ROLE [test_design_engineer] ADD MEMBER [Nancy]
ALTER ROLE [test_engineer] ADD MEMBER [Carol]
ALTER ROLE [test_engineer] ADD MEMBER [Lucy]
ALTER ROLE [test_engineer] ADD MEMBER [Jack]
Chapter 5

SUMMARY AND FUTURE PROJECT

In this project, I have provided an example of application support for the XACML security policy definitions. In this example, creation of the security policies and management of policy assignments assumes the environment supports the security model being represented as well as prior creation of the roles and subjects. As a future project, a student could expand the application to detect and implement missing roles, subjects and resources.

In addition, XACML is not only a policy creation language but also supports queries. It provides a method by which a user could query the system to find if a given role or user has privileges on a resource in the system. Since XACML is an abstraction layer above the points of implementation, all of the necessary information to answer these queries is contained in the XACML files themselves as well as in the application once the files have been processed. It would be easy then to look into the application tables to determine what rights a given user has.
Appendix A

Permission Policy for Chief Manager

<!-- Include permissions associated with project_manager role -->

<PolicySetIdReference>PPS:project_manager:role</PolicySetIdReference>

</PolicySet>
Appendix B

Permission Policy Project Manager

<!DOCTYPE xacml[
<!ENTITY xml "http://www.w3.org/2001/XMLSchema#">
<!ENTITY rule-combine "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:">
<!ENTITY policy-combine "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:">
<!ENTITY function "urn:oasis:names:tc:xacml:1.0:function:">
<!ENTITY subject-category "urn:oasis:names:tc:xacml:1.0:subject-category:">
<!ENTITY subject "urn:oasis:names:tc:xacml:1.0:subject:">
<!ENTITY role "urn:oasis:names:tc:xacml:2.0:subject:role">
<!ENTITY roles "urn:example:role-values:">
<!ENTITY resource "urn:oasis:names:tc:xacml:1.0:resource:">
<!ENTITY action "urn:oasis:names:tc:xacml:1.0:action:">
<!ENTITY actions "urn:oasis:names:tc:xacml:2.0:actions:">
<!ENTITY environment "urn:oasis:names:tc:xacml:1.0:environment:">
<!ENTITY category "urn:oasis:names:tc:xacml:3.0:attribute-category:">
]

<PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
schema-wd-17.xsd"
PolicySetId="PPS:project_manager:role"
Version="1.0"
PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <Target/>

<!-- Include permissions associated with software_engineer role -->
<PolicySetIdReference>PPS:software_engineer:role</PolicySetIdReference>

<!-- Include permissions associated with software_developer role -->
<PolicySetIdReference>PPS:software_developer:role</PolicySetIdReference>

<!-- Include permissions associated with test_design_engineer role -->
<PolicySetIdReference>PPS:test_design_engineer:role</PolicySetIdReference>
<!-- Include permissions associated with test_engineer role -->
<PolicySetIdReference>PPS:test_engineer:role</PolicySetIdReference>

</PolicySet>
Appendix C

Permission Policy Software Engineer

<!DOCTYPE xacml[
<!ENTITY xml "http://www.w3.org/2001/XMLSchema#">
<!ENTITY rule-combine "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:">
<!ENTITY policy-combine "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:">
<!ENTITY function "urn:oasis:names:tc:xacml:1.0:function:">
<!ENTITY subject-category "urn:oasis:names:tc:xacml:1.0:subject-category:">
<!ENTITY subject "urn:oasis:names:tc:xacml:1.0:subject:">
<!ENTITY role "urn:oasis:names:tc:xacml:2.0:subject:role">
<!ENTITY roles "urn:example:role-values:">
<!ENTITY resource "urn:oasis:names:tc:xacml:1.0:resource:">
<!ENTITY action "urn:oasis:names:tc:xacml:1.0:action:">
<!ENTITY actions "urn:oasis:names:tc:xacml:2.0:actions:">
<!ENTITY environment "urn:oasis:names:tc:xacml:1.0:environment:">
<!ENTITY category "urn:oasis:names:tc:xacml:3.0:attribute-category:">
]

<PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
schema-wd-17.xsd"
xmlns:access_policy="company:domain/schemas/access_policy.xsd"
PolicySetId="PPS:software_engineer:role"
Version="1.0"
PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <Target/>

  <Policy PolicyId="Permissions:specifically:for:the:software_engineer:role"
Version="1.0"
RuleCombiningAlgId="&rule-combine;permit-overrides">
    <Target/>

    <!-- Permission to read and write table requirement_doc -->
  
  <Rule
RuleId="Permission:to:read and write:table:requirement_doc"
Effect="Permit">
  <Description>
    A software engineer may read or write on table requirement_doc
    only in projects assigned to him
  </Description>
  <Target>
    <Resource>
      <ResourceMatch MatchId="&function;string-equal">
        <Attribute
          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
          DataType="&xml;string">
          <AttributeValue>
            requirement_doc
          </AttributeValue>
        </Attribute>
      </ResourceMatch>
    </Resource>
    <Action>
      <Attribute
        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="&xml;string">
        <AttributeValue>
          SELECT, INSERT, DELETE, UPDATE
        </AttributeValue>
      </Attribute>
    </Action>
  </Target>
</Rule>

<!-- Permission to read table code-->
.Rule
RuleId="Permission:to:read:table:code"
Effect="Permit">
  <Target>
    <Resource>
      <ResourceMatch MatchId="&function;string-equal">
        <Attribute
          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
          DataType="&xml;string">
          <AttributeValue>
            code
          </AttributeValue>
        </Attribute>
      </ResourceMatch>
    </Resource>
  </Target>
</Rule>
<AttributeValue>
  code
</AttributeValue>
</Attribute>
</ResourceMatch>
</Resource>
<Action>
  <Attribute
    AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
    DataType="&xml;string">
    <AttributeValue>
      SELECT
    </AttributeValue>
  </Attribute>
</Action>
</Target>
</Rule>

<!-- Permission to read table design_doc-->
<Rule
  RuleId="Permission:to:read:table:design_doc"
  Effect="Permit">
  <Target>
    <Resource>
      <ResourceMatch MatchId="&function;string-equal">
        <Attribute
          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
          DataType="&xml;string">
          <AttributeValue>
            design_doc
          </AttributeValue>
        </Attribute>
      </ResourceMatch>
    </Resource>
  </Target>
</Rule>
<!-- Permission to read table test_case_script-->
<Rule
RuleId="Permission:to:read:table:test_case_script"
Effect="Permit">
<Target>
<Resource>
<ResourceMatch MatchId="&function;string-equal">
<Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="&xml;string">
<AttributeValue>
test_case_script
</AttributeValue>
</Attribute>
</ResourceMatch>
</Resource>
>Action
<Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="&xml;string">
<AttributeValue>
SELECT
</AttributeValue>
</Attribute>
</Action>
</Target>
</Rule>

<!-- Permission to read table test_log-->
<Rule
RuleId="Permission:to:read:table:test_log"
Effect="Permit">
<Target>
<Resource>
<ResourceMatch MatchId="&function;string-equal">
  <Attribute
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
    DataType="&xml;string">
    <AttributeValue>
      test_log
    </AttributeValue>
  </Attribute>
</ResourceMatch>
</Resource>
</Action>
</Target>
</Rule>

<!-- Permission to write and read table project_plan-->
<Rule
  RuleId="Permission:to:read:table:project_plan"
  Effect="Permit">
  <Target>
    <Resource>
      <ResourceMatch MatchId="&function;string-equal">
        <Attribute
          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
          DataType="&xml;string">
          <AttributeValue>
            project_plan
          </AttributeValue>
        </Attribute>
      </ResourceMatch>
    </Resource>
    <Action>
      <Attribute
        AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="&xml;string">
        <AttributeValue>
          SELECT
        </AttributeValue>
      </Attribute>
    </Action>
  </Target>
</Rule>
<AttributeValue>
  SELECT, INSERT, DELETE, UPDATE
</AttributeValue>
</Attribute>
</Action>
</Target>
</Rule>
</Policy>
</PolicySet>
Appendix D

Role Policy Chief Manager

```xml
<!--DOCTYPE xacml[
<!ENTITY xml "http://www.w3.org/2001/XMLSchema#">
<!ENTITY rule-combine "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:">
<!ENTITY policy-combine "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:">
<!ENTITY function "urn:oasis:names:tc:xacml:1.0:function:">
<!ENTITY subject-category "urn:oasis:names:tc:xacml:1.0:subject-category:">
<!ENTITY subject "urn:oasis:names:tc:xacml:1.0:subject:">
<!ENTITY role "urn:oasis:names:tc:xacml:2.0:subject:role">
<!ENTITY roles "urn:example:role-values:">
<!ENTITY resource "urn:oasis:names:tc:xacml:1.0:resource:">
<!ENTITY action "urn:oasis:names:tc:xacml:1.0:action:">
<!ENTITY actions "urn:oasis:names:tc:xacml:2.0:actions:">
<!ENTITY environment "urn:oasis:names:tc:xacml:1.0:environment:">
<!ENTITY category "urn:oasis:names:tc:xacml:3.0:attribute-category:">
]>

<PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-core-v3-schema-wd-17.xsd"
  PolicySetId="RPS:project_chief_manager:role"
  Version="1.0"
  PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="&function;anyURI-equal">
          <AttributeValue
            DataType="&xml;anyURI">
            &roles;project_chief_manager
          </AttributeValue>
          <AttributeDesignator
            MustBePresent="false"
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
</PolicySet>
```
Category="&subject-category;access-subject"
AttributeId="&role;"
DataType="&xml;anyURI"/>
</Match>
</AllOf>
</AnyOf>
</Target>

<!-- Use permissions associated with the project_chief_manager role -->
<PolicySetIdReference>
  PPS:project_chief_manager:role
</PolicySetIdReference>
</PolicySet>
Appendix E

Role Policy Project Manager

```xml
<PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-core-v3-schema-wd-17.xsd"
    PolicySetId="RPS:project_manager:role"
    Version="1.0"
    PolicyCombiningAlgId="&policy-combine;permit-overrides">
    <Target>
        <AnyOf>
            <AllOf>
                <Match MatchId="&function;anyURI-equal">
                    <AttributeValue
                        DataType="&xml;anyURI">
                        &roles;project_manager
                    </AttributeValue>
                    <AttributeDesignator
                        MustBePresent="false"
                        Category="&subject-category;access-subject"
                </AllOf>
            </AnyOf>
        </AnyOf>
    </Target>
</PolicySet>
```

<Target>
  <AnyOf>
    <Match>
      <AttributeId>"&role;"
      <DataType>"&xml;anyURI"/></Match>
    </AnyOf>
  </AnyOf>
</Target>

<!-- Use permissions associated with the project_manager role -->
<PolicySetIdReference>
  PPS:project_manager:role
</PolicySetIdReference>
</PolicySet>
Appendix F

Role Policy Software Engineer

```xml
<!DOCTYPE xacml[
<!ENTITY xml "http://www.w3.org/2001/XMLSchema#">
<!ENTITY rule-combine "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:">
<!ENTITY policy-combine "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:">
<!ENTITY function "urn:oasis:names:tc:xacml:1.0:function:">
<!ENTITY subject-category "urn:oasis:names:tc:xacml:1.0:subject-category:">
<!ENTITY subject "urn:oasis:names:tc:xacml:1.0:subject:">
<!ENTITY role "urn:oasis:names:tc:xacml:2.0:subject:role">
<!ENTITY roles "urn:example:role-values:"/>
<!ENTITY resource "urn:oasis:names:tc:xacml:1.0:resource:">
<!ENTITY action "urn:oasis:names:tc:xacml:1.0:action:">
<!ENTITY actions "urn:oasis:names:tc:xacml:2.0:actions:">
<!ENTITY environment "urn:oasis:names:tc:xacml:1.0:environment:">
<!ENTITY category "urn:oasis:names:tc:xacml:3.0:attribute-category:">
]>

<PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-core-v3-schema-wd-17.xsd"
  PolicySetId="RPS:software_engineer:role"
  Version="1.0"
  PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <Target>
    <AnyOf>
      <AllOf>
        <Match
          MatchId="&function;anyURI-equal">
          <AttributeValue
            DataType="&xml;anyURI">
            &roles;software_engineer
          </AttributeValue>
          <AttributeDesignator
            MustBePresent="false"
            Category="&subject-category;access-subject"
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
</PolicySet>
```
AttributeId="&role;"
    DataType="&xml:anYURI"/>
</Match>
</AllOf>
</AnyOf>
</Target>

<!-- Use permissions associated with the software_engineer role -->
<PolicySetIdReference>
   PPS:software_engineer:role
</PolicySetIdReference>
</PolicySet>
Appendix G

Role Assignment

<!DOCTYPE xacml[
<!ENTITY xml "http://www.w3.org/2001/XMLSchema#">
<!ENTITY rule-combine "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:">
<!ENTITY policy-combine "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:">
<!ENTITY function "urn:oasis:names:tc:xacml:1.0:function:">
<!ENTITY subject-category "urn:oasis:names:tc:xacml:1.0:subject-category:">
<!ENTITY subject "urn:oasis:names:tc:xacml:1.0:subject:">
<!ENTITY role "urn:oasis:names:tc:xacml:2.0:subject:role">  
<!ENTITY roles "urn:example:role-values:">
<!ENTITY resource "urn:oasis:names:tc:xacml:1.0:resource:">
<!ENTITY action "urn:oasis:names:tc:xacml:1.0:action:">
<!ENTITY actions "urn:oasis:names:tc:xacml:2.0:actions:">
<!ENTITY environment "urn:oasis:names:tc:xacml:1.0:environment:">
<!ENTITY category "urn:oasis:names:tc:xacml:3.0:attribute-category:">
]>

<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-core-v3-schema-wd-17.xsd"
PolicyId="Role:Assignment:Policy"
Version="1.0"
RuleCombiningAlgId="&rule-combine;permit-overrides">
  <Target/>

<!-- project_chief_manager role requirements rule -->
<Rule RuleId="project_chief_manager:role:requirements" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="&function;string-equal">
          <AttributeValue
            DataType="&xml;string">
            Bill
          </AttributeValue>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
</Rule>

<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-core-v3-schema-wd-17.xsd"
PolicyId="Role:Assignment:Policy"
Version="1.0"
RuleCombiningAlgId="&rule-combine;permit-overrides">
  <Target/>

<!-- project_chief_manager role requirements rule -->
<Rule RuleId="project_chief_manager:role:requirements" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="&function;string-equal">
          <AttributeValue
            DataType="&xml;string">
            Bill
          </AttributeValue>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
</Rule>

<AttributeDesignator
    MustBePresent="false"
    Category="&subject-category;access-subject"
    AttributeId="&subject;subject-id"
    DataType="&xml;string"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeDesignator MustBePresent="false" Category="&category;resource"
    AttributeId="&role;"
    DataType="&xml;anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeDesignator MustBePresent="false" Category="&category;action"
    AttributeId="&action;action-id"
    DataType="&xml;anyURI"/>
</Match>
</AllOf>
</AnyOf>
</Target>
<Rule RuleId="project_manager:role:requirements" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="&function;string-equal">
          <AttributeValue
            DataType="&xml;string">
            Alice
          </AttributeValue>
          <AttributeDesignator
            MustBePresent="false"
            Category="&subject-category;access-subject"
            AttributeId="&subject;subject-id"
            DataType="&xml;string"/>
        </Match>
      </AllOf>
      <AnyOf>
        <AllOf>
          <Match MatchId="&function;string-equal">
            <AttributeValue
              DataType="&xml;string">
              Bob
            </AttributeValue>
            <AttributeDesignator
              MustBePresent="false"
              Category="&subject-category;access-subject"
              AttributeId="&subject;subject-id"
              DataType="&xml;string"/>
          </Match>
        </AllOf>
      </AnyOf>
    </AnyOf>
    <AllOf>
      <Match MatchId="&function;anyURI-equal">
        <AttributeValue
          DataType="&xml;anyURI">
          &roles;project_manager
        </AttributeValue>
        <AttributeDesignator
          MustBePresent="false"
          Category="&subject-category;access-subject"
          AttributeId="&subject;subject-id"
          DataType="&xml;anyURI"/>
      </Match>
    </AllOf>
  </Target>
</Rule>
<Rule RuleId="software_engineer:role:requirements" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="&function;string-equal">
          <AttributeValue
            DataType="&xml;string">
            Ace
          </AttributeValue>
          <AttributeDesignator
            MustBePresent="false"
            Category="&subject-category;access-subject"
            AttributeId="&subject;subject-id"
            DataType="&xml;string"/>
        </Match>
      </AllOf>
      <AnyOf>
        <AllOf>
          <Match MatchId="&function;anyURI-equal">
            <AttributeValue
              DataType="&xml;anyURI">
              &actions;enableRole
            </AttributeValue>
            <AttributeDesignator
              MustBePresent="false"
              Category="&category;action"
              AttributeId="&action;action-id"
              DataType="&xml;anyURI"/>
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
  </Rule>

<!-- software_engineer role requirements rule -->

<AllOf>
  <Match MatchId="&function;string-equal">
    <AttributeValue
      DataType="&xml;string">
      Lily
  </AttributeValue>
  <AttributeDesignator
    MustBePresent="false"
    Category="&subject-category;access-subject"
    AttributeId="&subject;subject-id"
    DataType="&xml;string"/>
  </Match>
</AllOf>

<AllOf>
  <Match MatchId="&function;string-equal">
    <AttributeValue
      DataType="&xml;string">
      James
  </AttributeValue>
  <AttributeDesignator
    MustBePresent="false"
    Category="&subject-category;access-subject"
    AttributeId="&subject;subject-id"
    DataType="&xml;string"/>
  </Match>
</AllOf>

<AnyOf>
  <AllOf>
    <Match MatchId="&function;anyURI-equal">
      <AttributeValue
        DataType="&xml;anyURI">
        &roles;software_engineer
    </AttributeValue>
    <AttributeDesignator
      MustBePresent="false"
      Category="&category;resource"
      AttributeId="&role;"
<Rule RuleId="software_developer:role:requirements" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="/function:string-equal">
          <AttributeValue
            DataType="/xml:string">
            Joe
          </AttributeValue>
          <AttributeDesignator
            MustBePresent="false"
            Category="/subject-category:access-subject"
            AttributeId="/subject:subject-id"
            DataType="/xml:string"/>
        </Match>
      </AllOf>
    </AllOf>
  </Target>
</Rule>

<!-- software_developer role requirements rule -->
<Match MatchId="&function;string-equal">
  <AttributeValue
    DataType="&xml;string">
    Kate
  </AttributeValue>
  <AttributeDesignator
    MustBePresent="false"
    Category="&subject-category;access-subject"
    AttributeId="&subject;subject-id"
    DataType="&xml;string"/>
</Match>
</AllOf>

<AllOf>
  <Match MatchId="&function;string-equal">
    <AttributeValue
      DataType="&xml;string">
      Jane
    </AttributeValue>
    <AttributeDesignator
      MustBePresent="false"
      Category="&subject-category;access-subject"
      AttributeId="&subject;subject-id"
      DataType="&xml;string"/>
  </Match>
</AllOf>
</AnyOf>

<AnyOf>
  <AllOf>
    <Match MatchId="&function;anyURI-equal">
      <AttributeValue
        DataType="&xml;anyURI">
        &roles;software_developer
      </AttributeValue>
      <AttributeDesignator
        MustBePresent="false"
        Category="&category;resource"
        AttributeId="&role;"
        DataType="&xml;anyURI"/>
    </Match>
  </AllOf>
</AnyOf>
</AllOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
  <AttributeValue
  DataType="&xml;anyURI">
    &actions;enableRole
  </AttributeValue>
  <AttributeDesignator
    MustBePresent="false"
    Category="&category;action"
    AttributeId="&action;action-id"
    DataType="&xml;anyURI"/>
</Match>
</AllOf>
</AnyOf>
</Target>
</Rule>

<!-- test_design_engineer role requirements rule -->
<Rule RuleId="test_design_engineer:role:requirements" Effect="Permit">
<Target>
<AnyOf>
<AllOf>
<Match MatchId="&function;string-equal">
  <AttributeValue
  DataType="&xml;string">
    Susan
  </AttributeValue>
  <AttributeDesignator
    MustBePresent="false"
    Category="&subject-category;access-subject"
    AttributeId="&subject;subject-id"
    DataType="&xml;string"/>
</Match>
</AllOf>
</AnyOf>
<AllOf>
<Match MatchId="&function;string-equal">
  <AttributeValue
  DataType="&xml;string">
    Sue
  </AttributeValue>
</Match>
</AllOf>
</Target>
</Rule>
<AttributeDesignator MustBePresent="false"
Category="&subject-category;access-subject"
AttributeId="&subject;subject-id"
DataType="&xml;string"/>
</Match>
</AllOf>
<AllOf>
<Match MatchId="&function;string-equal">
<AttributeValue
DataType="&xml;string">
Nancy
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&subject-category;access-subject"
AttributeId="&subject;subject-id"
DataType="&xml;string"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
<AttributeValue
DataType="&xml:anyURI">
&roles;test_design_engineer
</AttributeValue>
<AttributeDesignator MustBePresent="false"
Category="&category;resource"
AttributeId="&role;"
DataType="&xml:any URI"/>
<Rule RuleId="test_engineer:role:requirements" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="&function;string-equal">
          <AttributeValue
            DataType="&xml;string">
            Carol
          </AttributeValue>
          <AttributeDesignator
            MustBePresent="false"
            Category="&subject-category;access-subject"
            AttributeId="&subject;subject-id"
            DataType="&xml;string"/>
        </Match>
      </AllOf>
      <AllOf>
        <Match MatchId="&function;string-equal">
          <AttributeValue
            DataType="&xml;string">
            Lucy
          </AttributeValue>
          <AttributeDesignator
            MustBePresent="false"
            Category="&subject-category;access-subject"
            AttributeId="&subject;subject-id"
            DataType="&xml;string"/>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
</Rule>
Category="&subject-category;access-subject"
AttributeId="&subject;subject-id"
DataType="&xml;string"/>
</Match>
</AllOf>
<AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;string-equal">
  <AttributeValue
    DataType="&xml;string">
    Jack
  </AttributeValue>
  <AttributeDesignator
    MustBePresent="false"
    Category="&subject-category;access-subject"
    AttributeId="&subject;subject-id"
    DataType="&xml;string"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
  <AttributeValue
    DataType="&xml;anyURI">
    &roles;test_engineer
  </AttributeValue>
  <AttributeDesignator
    MustBePresent="false"
    Category="&category;resource"
    AttributeId="&role;"
    DataType="&xml;anyURI"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="&function;anyURI-equal">
  <AttributeValue
    DataType="&xml;anyURI">
    &actions;enableRole
  </AttributeValue>
  <AttributeDesignator
    MustBePresent="false"
    Category="&category;resource"
    AttributeId="&role;"
    DataType="&xml;anyURI"/>
</Match>
</AllOf>
</AnyOf>
</AnyOf>
</AttributeValue>
<AttributeDesignator
  MustBePresent="false"
  Category="&category;action"
  AttributeId="&action;action-id"
  DataType="&xml;anyURI"/>
</Match>
</AllOf>
</AnyOf>
</Target>
</Rule>

</Policy>
Bibliography

[1] NIST. “Role Based Access Control”. [online]

Committee Specification 01, 10 August 2010. [online] http://docs.oasis-
open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf [Accessed: October 9,
2013]

[3] OASIS. "XACML v3.0 Core and Hierarchical Role Based Access Control
(RBAC) Profile" Version 1.0. Committee Draft 03, 11 March 2010. [online]
[Accessed: October 9, 2013]

Compiling XACML Policies for Attribute-Based Database Access Control,"
Proceedings of the first ACM conference on Data and application security and
privacy(CODASPY'11), New York, NY, USA, 2011, pages 97-108.


us/sqlserver/product-info/overview-capabilities.aspx [Accessed: October 30,
2013]

[8] Reyes Jose. "Incorporating Attributes Into Role-Based Access Control", Master Project Report, Computer Science Department, California State University, Sacramento. Fall 2012