This Section contains the basic information related to the three renowned access control models i.e. ABAC, FGAC and UCON. These models are used in the EACF framework for the demonstration and verification purpose.

- **Attribute Based Access Control Model (ABAC)**

  ABAC model is a logical access control methodology, where authorization to perform a set of operations is determined by evaluating the attributes. These attributes are associated with the Subject, Object, requested operations, and, in some cases, environment conditions against Policy, Rules, or Relationships. These attributes mainly allow the operations for a given set of attributes. ABAC is distinguishable from other logical access control models because it controls access to objects by evaluating rules against the attributes of the entities (Subject and Object) and the environment relevant to a request. An ABAC system can implement the existing RBAC policies and can support a migration from role-based to a more granular access control policy based on many different characteristics of the individual requestor. ABAC offers the flexibility to apply access control policy without the prior knowledge of the specific subject and for an unlimited number of subjects that might require access. One of the primary benefits of ABAC is its ability to accommodate external or new users, without modifying the existing rules or object attributes, as long as the subjects are assigned the attributes necessary for the access control decision. ABAC avoids the need for explicit authorizations to be directly assigned to individual subjects prior to a request to perform an operation on the object. A key feature of ABAC is the ability of the object owner to protect and share the object without any prior knowledge of the individual subjects.

- **Fine-Grained Access Control Model (FGAC)**

  FGAC model is an access control technique, which provides access restrictions at the granular level. FGAC model was developed to meet the diverse and dynamic security requirements of applications and their users by defining the constraints from highest to lowest level of authorization. High-level means few access restrictions (coarse-grained), and low-level refers to more fine-grained access control restrictions. FGAC process is generally carried out by having fine grained access control policies which define the instances that a user can access and then the user properties are defined in the application context. Depending on user properties and environment, FGAC makes its decision to give access to the resources. This model provides...
authorization based on the security level of Objects (Resource, Subject, Action). The security level defines the privileges or constraints that are imposed on the Object to restrict access of illegal or malicious entities. Constraints are applied on the Objects (Subject or Resource to be accessed, or Action to be performed) and also on the entities (Subject, Action, Resource, or Environment) that are participating in the access decision. The refinement layers of FGAC model may vary according to the organizational needs, where an administrator can enforce the levels on respective Objects. For instance, in an organization the access of a higher level Subject, say CEO, to any particular Resource has to be less restricted, than the access to a lower level Subject, say Manager. This objective can be achieved by putting more Rules in the Policy governing the access of Subject Manager, and less Rules in the Policy governing the access of Subject CEO. Moreover, the Resource can be a particular row, column or table as per requirement. FGAC helps to define discrete levels of privileges for critical organizational services and resources, especially when they are shared among users with distinct authorization levels. Consequently, there is an increasing demand for a standardized FGAC specification, which describes all the minute details and leaves no chance for any misconception or error. Database developers are focusing on this technique as well to make their databases secure in the Cloud infrastructure.

- Usage Control Model (UCON)

UCON model fulfills capabilities for continuous monitoring of usage session and are more suitable for dynamic and distributed computing environment, such as Cloud and Grid Computing. UCON model considers multiple factors in access decision, which makes it a better choice as compared to other traditional access control models. UCON consists of pre, ongoing and post models that perform policy evaluation before, during and after the access phase. UCON encompasses traditional access control, trust management, and digital rights management and goes beyond them in its definition and scope. UCON model mainly restricts the usage of digital objects and also supports efficient techniques to include traditional access control models. The traditional access control models encompass only the Authorization (A) rules to make access decision; whereas, the UCON model also incorporates the oBligations (B) and Conditions (C). UCON oBligations represent certain actions that need to be completed before granting permission to the requested resources. On the other hand, Conditions are the system oriented decision factors to further improve the accuracy of access decisions. The term usage control is a generalization of access control to cover authorizations, oBligations, conditions, and continuity (ongoing controls). Traditionally authorization decisions are only made at the time of request, but there is no support for ongoing controls in case of long-lived access or immediate revocation. UCON, on the other hand, provides an extension to regular access control by not only allowing decisions for who gets to access what, but also allowing the liberty of access revocation based on certain attributes, that may change during the course of usage. Additionally, UCON model provides another significant innovation known as attribute mutability, which deals with those mutable attributes that can affect the access decision during usage session.
APPENDIX-C

This Section contains the Primefaces UI Components that we have converted into custom components belonging to the Framework:

Learning Phase:

1. **Navigation Bar:** This component remains on the page, even as the lower portion of the page is updated when the user makes a selection from the menu. It is also common among the PAPs for all models, with a few changes.

2. **DataTables:** This component is ubiquitous in our GUI, where we customize it to handle *Select* and *Unselect* events, and include other features like facets and paginator. These tables show the corresponding database entries of the selected category.

3. **Context Menu:** The Context Menu can be seen on almost every page, which allows user to select a database entry and *Update* or *Delete* it.

4. **Attribute-AttributeValueDataTable Pair:** The X Attribute- X Attribute Value DataTable pair can be found on all pages under *System Learning*. Moreover, they can also be found in Create X Dialogs, where X can be Subject, Action, Resource or Environment.
5. **Name-Description Input Boxes:** All the Create X dialogs contain the Name and Description input boxes that can become a custom control.

6. **Add X Attribute:** This group of input controls can be found on every Add X Attribute dialog.

7. **Add X Attribute Value:** This group of input controls can be found on every Add X Attribute Value dialog.

**Policy Creation:**

8. **Add X in Target:** This set of DataTables is used in every Add X in Target Wizard. These are used to select an appropriate X Attribute Value for a Target.
9. **Rule Controls**: This set of Rule input controls are the necessary fields for any Rule.

10. **Create Rule Tables**: These are the tables that a user needs to select from to create a Rule, including Target and Condition.

11. **Policy Controls**: These are the necessary fields for any kind of Policy, according to any access control model.
12. **Create Policy Tables**: This pair of tables should be sufficient for creating Policies for most of the basic access control models, including Target and Rules for the user to select from.

13. **Policy Set Controls**: These are the necessary fields for any kind of Policy Set.

14. **Create Policy Set Tables**: This pair of tables should be sufficient for creating Policy Sets for most of the access control models, including Target, existing Policy Sets and Policies for the user to select from.
This Section contains the source code of the XACML Policy generated using PAP which allows Read Permission to the Administration on the Academic Community.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
  PolicyId="readAdministrationAcademicCommunity"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-oversides">
  <Description>Policy for allowing read Permission to the Administration on the Academic Community.</Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatchMatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">Administrator</AttributeValue>
          <SubjectAttributeDesignatorAttributeId="urn:oasis:names:tc:xacml:1.0:subject:group-one"
            DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" />
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatchMatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">Academic</AttributeValue>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
</Policy>
```
This section of Appendix-D contains the steps needed to perform in order to exhibit the validation of test policy on the DSpace.
1. Log in to the **DSpace Repository** using the **Administrator credentials** created at the time of installation.

2. Click on the **Communities & Collections** on the right hand side of the page after successful login.
3. Click on the **AcademicCommunity**.

4. Given below is the message that will be pop-up according to the applicable policy which in this case allows **Read** access to the **Administrator** on the **Academic** community.

5. If the response from the policy decision point of “Permit” like in this case, then the following screen will be shown, which is the proof that DSpace is working correctly with modified authorization module and generated XAMCL policy.
APPENDIX-E

This Section contains the Deployment and Component diagram of the EACF framework.

Deployment Diagram:
Component Diagram
APPENDIX-F

Given below is one of the test cases to test the working of policy generation phase.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>ID: TC_17</th>
<th>Date: 15.09.2014</th>
<th>Test Runner: XACMLPolicyGenerationUtilTestCase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test purpose</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The purpose of this test case is to validate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the effectiveness of the functions in</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XACMLPolicyGenerationUtilTestCase class that</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>takes Policy-related parameters from the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>database and generates a XACML Policy</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Pre-Condition                                |           |                   |                                              |
| PolicyCreationInsertionTestSuite must be     |           |                   |                                              |
| run once before running this test case.      |           |                   |                                              |

| Unit Tests                                    |           |                   |                                              |
| Name                                           | Description                                                                 | Status |
| setRuleTypesTest_17_01_01                      | Test the setRuleType Function to check if it returns valid Set of JBossRuleType | Success |
| setRuleTypesTest_17_01_02                      | Test the setRuleType Function to check NullPointerException, as input Rule cannot be null | Success |
| setTargetTypeTest_17_02_01                     | Test the setTargetType Function to check if it returns valid Set of JBossTargetType | Success |
| setTargetTypeTest_17_02_02                     | Test the setTargetType Function to check NullPointerException, as input Target cannot be null | Success |

<table>
<thead>
<tr>
<th>Overall Result</th>
<th>Tests</th>
<th>Errors</th>
<th>Failures</th>
<th>Skipped</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>KTH</td>
</tr>
</tbody>
</table>

Comments: It has been observed that all the tests are successful without any failure.
Test case given below belongs to the test suite created for population of tables associated with System Learning phase.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>InsertActionTestCase</th>
<th>ID: TS_01_01</th>
<th>Date: 01.08.2014</th>
<th>Test Runner: SystemLearningInsertionTestSuite</th>
</tr>
</thead>
</table>

**Test purpose**
This Test Case is to validate the Insertion of Actions and their corresponding Attributes and Attributes Values for a University Scenario which includes Students, Faculty and Staff members.

**Pre-Condition**
MySQL Database should be properly configured before running the test case.

<table>
<thead>
<tr>
<th>Unit Tests</th>
<th>Name</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>InsertStudentActions</td>
<td>To test whether or not the Student Action is properly populated in the database</td>
<td>Success</td>
</tr>
<tr>
<td></td>
<td>InsertFacultyActions</td>
<td>To test the insertion of Faculty Action in the database</td>
<td>Success</td>
</tr>
<tr>
<td></td>
<td>InsertStaffActions</td>
<td>To test the population the Staff Action in the database</td>
<td>Success</td>
</tr>
</tbody>
</table>

**Overall Result**

<table>
<thead>
<tr>
<th>Tests</th>
<th>Errors</th>
<th>Failures</th>
<th>Skipped</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>KTH</td>
</tr>
</tbody>
</table>

**Comments**
Runs collectively with the SystemLearningInsertionTestSuite

**APPENDIX-G**

Following are the listings containing XACML-SAML based Authorization Decision Query request and responses.

**Listing # 1:**

\begin{lstlisting}[language=XML, caption= XACML-SAML Authorization Decision Query, linewidth=0.5cm, basicstyle=\small]
<xacml-samlp:XACMLAuthzDecisionQuery
  IssueInstant="2015-02-27T06:44:57.766Z"...>
  <xacml-context:Request>
  <xacml-context:Subject>

```xml
  ...
```
<xacml-context:Attribute.../>
<xacml-context:AttributeValue>
"Administrator"
</xacml-context:AttributeValue>
</xacml-context:Attribute>
</xacml-context:Subject>
<xacml-context:Resource>
<xacml-context:Attribute.../>
<xacml-context:AttributeValue>
"Academic Community"
</xacml-context:AttributeValue>
</xacml-context:Attribute>
</xacml-context:Resource>
<xacml-context:Action>
<xacml-context:Attribute.../>
<xacml-context:AttributeValue>
"Read"
</xacml-context:AttributeValue>
</xacml-context:Attribute>
</xacml-context:Action>
</xacml-context:Request>
<saml:Evidence>
<saml:Assertion>
<saml:Issuer>
Client
</saml:Issuer>
Listing # 2:

\begin{lstlisting}[language=XML, caption= XACML-SAML Authorization Decision Query, linewidth=0.5cm, basicstyle=\small]
<xacml-samlp:XACMLAuthzDecisionQuery
IssueInstant=
"2015-02-27T06:44:57.766Z"
...

<xacml-context:Request>
<xacml-context:Subject>
<xacml-context:Attribute...>
<xacml-context:AttributeValue>
"Administrator"
</xacml-context:AttributeValue>
</xacml-context:Attribute>
</xacml-context:Subject>
<xacml-context:Resource>
</xacml-context:Resource>
\end{lstlisting}
"Academic Community"

"Read"

"2015-02-27T06:40:03.766Z"
</xacml-samlp:XACMLAuthzDecisionQuery>
\end{lstlisting}