This document describes UMA protocol implementation in details, so one can experience how UMA works.
Table of Contents

1 WHAT IS UMA? ............................................................................................................. 3
2 TERMINOLOGIES USED .............................................................................................. 4
   Authorizing user ........................................................................................................ 4
   Authorization manager .............................................................................................. 4
   Host ............................................................................................................................. 4
   Requester .................................................................................................................... 4
   Protected resource ..................................................................................................... 4
   Host access token ........................................................................................................ 4
   Requester access token .............................................................................................. 4
   Requesting party ......................................................................................................... 4
   Claim ............................................................................................................................ 4
   JSON ........................................................................................................................... 4
   XRD .............................................................................................................................. 4
3 WHAT WE HAVE DONE SO FAR? .................................................................................. 5
   3.1 THREE MAIN STEPS OF UMA ........................................................................... 5
      3.1.1 Step1: ............................................................................................................ 5
   AUTHORIZING USER INTRODUCES HOST TO AM .................................................. 5
      3.1.1.1 Host looks up AM metadata .................................................................... 5
      3.1.1.2 Host dynamically registers with AM ....................................................... 5
      3.1.1.3 Host obtains host access token ............................................................... 5
   3.1.2 Step2:................................................................................................................. 6
   REQUESTER GETS ACCESS TOKEN FROM AM ....................................................... 6
      3.1.3 Step3: ............................................................................................................. 6
   REQUESTER WIELD ACCESS TOKEN AT HOST TO GAIN ACCESS .......................... 6
      3.1.3.1 Requester attempts access ....................................................................... 6
      3.1.3.2 Host asks AM to validate requester access token ..................................... 6
      3.1.3.3 Valid response ......................................................................................... 6
      3.1.3.4 Error responses ...................................................................................... 6
   3.2 SCOPE& RESOURCE REGISTRATION ................................................................. 7
      3.2.1 Scope Registration ....................................................................................... 7
      3.2.2 Resource Registration ............................................................................... 7
   3.3 POLICY ADMINISTRATION ................................................................................. 8
   3.4 LOGIN VIA THIRD PARTY ................................................................................. 8
   3.5 SECURE END TO END COMMUNICATION ........................................................ 9
      3.5.1 Use of Https .................................................................................................. 9
   3.6 WEB INTERFACE .................................................................................................. 10
      3.6.1 Web Interface for Authorization Manager ................................................. 10
      3.6.2 Web Interface for Host .............................................................................. 11
      3.6.3 Web Interface for Client ............................................................................ 11
      3.6.4 Web Interface for all steps Involved ......................................................... 12
   3.7 ERROR RESPONSES ........................................................................................... 13
      3.7.1 Null or Empty Code .................................................................................... 13
      3.7.2 Ambiguous Policy ...................................................................................... 13
      3.7.3 Ambiguous Resource .................................................................................. 13
      3.7.4 Wrong Token .............................................................................................. 13
TECHNOLOGIES WE ARE USING ........................................................................13
4.1 JAVA ...........................................................................................................14
4.2 JSP .............................................................................................................14
4.3 SPRING 3.0 ...............................................................................................14
4.4 APACHE TOMCAT ......................................................................................14
4.5 IBATIS .........................................................................................................14
4.6 POSTGRESQL ...........................................................................................15
4.7 DREAMWEAVER .......................................................................................15
4.8 NAVICAT .....................................................................................................15

HOW OUR SYSTEM WORK? ............................................................................15
5.1 DEPLOYMENT DETAILS ..............................................................................15
5.2 FLOW OF THE SYSTEM .............................................................................15
5.3 SCENARIOS ...............................................................................................18
  5.3.1 UMA – Step 1 .......................................................................................18
  5.3.1.1 Scenario # 1 ....................................................................................18
  5.3.1.2 Scenario # 2 ....................................................................................33
  5.3.2 UMA – Step 2 & Step 3 ..........................................................................36
  5.3.2.1 Accessing resource if user is registered with AM and policy is assigned to it ........................................................................................................36
  5.3.2.2 Accessing resource via “Facebook” if this requester is in “Facebook Friends list” of resource owner and policy is assigned to it ..........44
  5.3.3 Resource Upload Scenario ...................................................................47
  5.3.4 Protecting Resource with AM .................................................................52
  5.3.5 Assigning Policies to Resources which are protected with AM .............56

WHAT’S NEXT? .................................................................................................61
6.1 TRUSTED CLAIM .......................................................................................61
6.2 JWT .............................................................................................................61
6.3 DIGITAL SIGNATURE .................................................................................61

HELP FOR DEVELOPERS ...............................................................................61

REFERENCES ..................................................................................................62
1 What is UMA?

Managing of sharing information in social networks is a difficult task for users and can violate the user's privacy. With respect to privacy, the User-Managed Access—a novel access management solution (consist of an architecture and access control delegation protocol based on OAuth 2.0. It is being developed at the kantara Initiative for ultimate contribution to the IETF) that enables the "Authorizing User" to control access to his/her own resources — identities, preferences, data, through an "Authorization Manager" (AM). Based on user's policies the AM may grant access to third parties that request for that information.

When a "Requester" attempts access to a protected resource owned and protected by the authorizing user stored at some "Host", he is told where to go — the location of the AM, in order to request an access token. When he gets to the AM, he might be asked to convey claims about the requesting party, e.g. a web user or corporation, which satisfy the user's policy. After obtaining the right access token the requester is then able to receive the information he originally asked for.
2 Terminologies used

**Authorizing user**: An UMA-defined variant of an OAuth resource owner; a web
user who configures an authorization manager with policies that control how it
makes access decisions when a requester attempts to access a protected resource
at a host [UMA core protocol]

**Authorization manager**: An UMA-defined variant of an OAuth authorization
server that carries out an authorizing user’s policies governing access to a
protected resource. [UMA core protocol]

**Host**: An UMA-defined variant of an OAuth resource server that enforces access
to the protected resources it hosts, as decided by an authorization manager. [UMA
core protocol]

**Requester**: An UMA-defined variant of an OAuth client that seeks access to a
protected resource. [UMA core protocol]

**Protected resource**: An access-restricted resource at a host. [UMA core
protocol]

**Host access token**: An access token representing the authorizing user’s consent
for a host to trust a particular authorization manager for access decisions about
resources hosted there. [UMA core protocol]

**Requester access token**: An access token representing the authorizing user’s
consent for a requester’s access to particular resources at a host. [UMA core
protocol]

**Requesting party**: A web user, or a corporation (or other legal person), that uses
a requester to seek access to a protected resource. [UMA core protocol]

**Claim**: A statement of the value or values of one or more identity attributes of a
requesting party. Claims are conveyed by a requester on behalf of a requesting
party to an authorization manager in an attempt to satisfy an authorizing user’s
policy. [UMA core protocol]

**JSON**: JavaScript Object Notation is a lightweight data-interchange format.

**XRD**: XRD document is a well-formed XML document as defined by [XML 1.0]
with a root element of <XRD>.
3 What we have done so far?

Three servers are developed namely Authorization manager, Host and Client. Client provides an interface which initiates the process of introducing Host at Authorization manager by Authorizing user.

3.1 Three main steps of UMA

3.1.1 Step 1: Authorizing user introduces host to AM

3.1.1.1 Host looks up AM metadata

Authorizing user initiates the process on Client interface and provides initialization endpoint and metadata endpoint. On the basis of initialization endpoint, authorizing user is redirected to Host. The metadata endpoint provided by authorizing user is used by Host to get metadata from Authorization manager. When user clicks on Introduce Host, control is passed to Host and Host gets metadata from AM. The retrieval of metadata is done at backend and is not visible to authorizing user. This metadata is retrieved in xml form and is saved as an xml file on Host server for future use.

3.1.1.2 Host dynamically registers with AM

When the control is passed to Host, it redirects to AM where AM will confirm that you really want to introduce this Host on AM. In case of YES, the host will be registered with AM if it is not already. In case of NO, the process is cancelled and control is given back to Client. In case of Registration, record is saved in both AM and Host’s database. In Host’s database we save that this host is registered with the named AM. In AM’s database we save that with this AM, the named Host is registered.

3.1.1.3 Host obtains host access token

After confirmation of authorizing user that he/she wants to introduce the Host on the given AM, if the Host is not registered then it is registered first otherwise the OAuth 2.0 steps are followed for getting host access token. In this process, firstly host is authenticated and then token is issued to it. Host gets access token from AM and saves it in a database against the AM which issued this token. This token is also saved in AM’s database.
3.1.2 Step2:

**Requester gets access token from AM**

Resources page is available to requester. When requester clicks on a certain resource, an interface is provided to requester by Host in order to communicate with AM and access token from AM. Requester provides its credentials/details to AM. Based on the policy defined by Authorizing user, an access token is issued to the requester. Currently we are using very simple policy which just checks the existence of the requester in AM’s database and offering access token for one time access.

3.1.3 Step3:

**Requester yields access token at host to gain access**

3.1.3.1 Requester attempts access

When requester gets access token from AM, it then presents it to Host to get access to the desired resource.

3.1.3.2 Host asks AM to validate requester access token

When the Host gets access token from requester, it will validate it from AM. For validation, Host needs to send this requester access token along with its own access token to AM.

3.1.3.3 Valid response

AM validates both access tokens if they are existing in database of AM against the given Host and Requester. Currently we are just checking the existence of both tokens in the AM’s database against the given requester and Host. We will improve it later to check also the scope and policy of the tokens. If both the tokens are validated and they are OK, then a response is sent to Host by AM that requester can be given access otherwise an erroneous response is sent to Host and the requester will not be allowed to access the desired resource. In case of valid response from AM to Host, the requester is given access to the desired resource as defined by the scope of the token.

3.1.3.4 Error responses

Different errors can take place at different times so there are different responses which are sent to Host based on error nature. For example, the requester token is not valid, AM sends the error message to Host that this requester is not allowed to access the resource and the Host displays it to the requester.
3.2 **Scope & Resource Registration**

### 3.2.1 Scope Registration

Scopes are basically simple strings which only have meaning to the Host and are completely transparent to the Authorization Manager. For instance a scope can be all photos of a Host being marked as "for family only" or all photos of a certain resolution.

Scopes are registered with an AM by the Host and can be revoked again at any time.

Resources are bundled to scopes only on the Host side. The Authorization Manager never knows in advance which resource (and thus URI pointing to one) falls under which scope unless a request is made to a resource in which case the Host informs the Authorization Manager about the scopes it falls under. The Authorization Manager can then make a decision based on the scope information if access is granted or not.

In order to do so, resource owners attach policies to scopes. This can be as simple as being asked on every request or as complex as fulfilling a chain of claims.

### 3.2.2 Resource Registration

To provide a method for a host to register, update, check, and delete resource-related information at an authorization manager (AM) so that the user’s resources can be put under scoped protection.

Resource registration will provide the following functionality.

The host will be able to register information about resources to be protected so that the AM can apply authorization constraints to them according to the user's wishes.

The host will be able to determine (unilaterally or as instructed by the user) the universe of resources belonging to this user that are to be protected by the AM, and will assign a resource identifier to sets of these resources. Such a set might include a status update API endpoint (applying to the user's entire update stream), or an individual status update, or a photo album, or all photos with a particular user-assigned tag, or even the set of "all resources managed by this user at this host".
The host will also be able to determine the universe of actions that is possible for requesting parties to perform on each resource set. Such actions might involve "viewing", "adding", "printing", or whatever other actions the application supports for that resource set. Typically each action covers a broad list of methods in the host’s published API.

Once the host has registered resource sets and their descriptive information with the AM, the AM (under the authorizing user’s instructions) is able to map particular authorization constraints to a particular set of resources and a particular set of actions and to take part in the process of limiting (scoping) access to resource sets.

Host will be free to offer the option to protect any subset of the user's resources using different AMs or other means entirely, or to protect some resources and not others.

### 3.3 Policy Administration

On AM side we have a section for policy administration. We have a page on Host side which contains all protected resources. When resource is uploaded and protected with AM, it is not displayed on protected resources page until policies are assigned to this resource. When policy is assigned to the resource on AM side, a message is sent to Host regarding this and then the resource can be displayed on protected resources page.

In policy administration, policies are assigned to resource. Policies can be assigned to requesters, groups of requesters, facebook friends etc. If user assigns policy to a facebook friend, his/her friends are fetched from facebook and policy is assigned to them.

### 3.4 Login via third party

At both AM and Host, login process can be performed through third party like facebook, google, yahoo etc. Currently we are only handling facebook but it will also cover other IDP in future. User can also login to the system using his/her credential of this system.
3.5 Secure end to end Communication

3.5.1 Use of HTTPS

HTTPS (HTTP over SSL or HTTP Secure) is the use of Secure Socket Layer (SSL) or Transport Layer Security (TLS) as a sub layer under regular HTTP application layering. HTTPS encrypts and decrypts user page requests as well as the pages that are returned by the Web server. The use of HTTPS protects against eavesdropping and man-in-the-middle attacks. HTTPS was developed by Netscape.

HTTPS and SSL support the use of X.509 digital certificates from the server so that, if necessary, a user can authenticate the sender. Unless a different port is specified, HTTPS uses port 443 instead of HTTP port 80 in its interactions with the lower layer, TCP/IP. In our System we changed the ports according to our requirements. As for communication with https server (in code), the digital certificate of the server must be obtained from any well know certificate Issuer Company otherwise it will not work. In our system, we developed two programs, one for certificates generation.
and the other is used for adding these certificates to trusted zone for the local computer. For the real deployment of our system we should get these digital certificates from the Issuer Company so that it can be accepted by every system and browser. All our three servers are upgraded and they are now running on Https server.

3.6 Web Interface

3.6.1 Web Interface for Authorization Manager

In UMA protocol Authorization manager is a separate application server. A lot of operations are performed on this server indirectly by Host server but some of the functionalities must be handled directly by Authorization manager for which we designed web interface for this server. With the use of this direct interface to AM, the authorizing user will be able to define policies, operate as admin for his/her account (accept/reject requests), see list of all registered host with the AM, see list of all requesters registered with AM and much more. This server is currently running on Https.

![Figure 1.3](image-url)
3.6.2 Web Interface for Host

In UMA protocol Host is a separate web application server. On Host we provided different interfaces to requester e.g. protected resources page, upload resource page, protect resource with AM page etc.

![Web Interface for Host](image)

Figure 1.4

3.6.3 Web Interface for Client

Client is also a separate web application. As before any communication between Host and AM, they should know each other or we can say that Host must be introduced at AM. This application provides the starting point of Host introduction at AM and then the whole process is continued step by step through graphical interface between Host and AM.
3.6.4 Web Interface for all steps Involved

During the process of introducing Host at AM by Authorizing user and getting access to resource by requester requires certain steps. In some steps the user must respond (for these steps interface is mandatory) but in some steps user response is not important and doesn’t matter so there is no need for showing that to Authorizing user. For making the system understandable and for those who are not familiar with UMA, we provided interface for each major step, so that viewers can understand what’s going on in the system. Later on during real deployment, we will remove the interface of unimportant steps and will only leave those, in which user response is necessary. For example when the user confirms that he/she wants to introduce this Host at AM, then directly the last step in which access token is issued to the host will be shown to the authorizing user. All the intermediate steps of registration, authorization etc will take place at backend. Currently we are showing all these steps to user through an interface.
3.7 Error Responses

3.7.1 Null or Empty Code
In case of Null or Empty code the following response is sent.
“HTTP/1.1 400 Bad - Request No Token”

3.7.2 Ambiguous Policy
In case of ambiguous action on resource, the following response is sent.
“HTTP/1.1 403 Forbidden - ambiguous-policy”

3.7.3 Ambiguous Resource
In case of accessing an undefined resource, the following response is sent.
“HTTP/1.1 403 Forbidden - ambiguous-resource”

3.7.4 Wrong Token
In case of presenting wrong token to AM, the following response is sent.
“HTTP/1.1 401 Unauthorized - No such token exists”

3.7.5 Insufficient Access Rights
In case of accessing a resource for which user doesn’t have enough privileges, the following response is sent.
"HTTP/1.1 403 Forbidden - Insufficient access rights"

4 Technologies we are using

We are using the following technologies in the implementation of UMA.
4.1 **Java**
Due to many distinct qualities of Java we are using Java as our development language.

4.2 **JSP**
For frontend interfaces, .jsp pages are used.

4.3 **Spring 3.0**
In our application we are following MVC architecture, for which we selected Spring 3.0.

4.4 **Apache Tomcat**
Because of the ease of use and user friendliness we are using Tomcat 7.0.8 as our deployment server.

4.5 **iBatis**
We are using this plug-in for automatic generation of queries. We don’t need to write queries manually as it generates all the commonly used queries for us.
4.6 **PostgreSQL**
For data storage we are using PostgreSQL database.

4.7 **Dreamweaver**
For design of interfaces we are using Dreamweaver.

4.8 **Navicat**
For interface to database we are using Navicat.

5 **How our System work?**

5.1 **Deployment Details**
As we have three applications namely Client, Host and Authorization Manager. All these three applications are deployed on different servers and are using Https. Client is deployed on Tomcat 7.0.8 at port 8080, Host at port 8090 and Authorization Manager at 8070.

5.2 **Flow of the System**
We have three major steps which are explained by the following figures.
Figure 1.7

**1st OAuth - Step 1. Obtaining end-user authorization**

- User authenticates & decides on access request
- [Redirection of end-user user's agent to end-user authz. endpoint] Client Identifier & Redirect URI
- Metadata document
- Learn about User Authorization URL (OAuth end-user authz. endpoint)
- Construct host-meta url
- Provision of AM location

**1st OAuth - Step 2. Obtaining an Access Token (HOST ACCESS TOKEN)**

- Validate Authorization Code
- Access Token (w/ optional Refresh Token)
- Optional registration of scopes
- Client Credentials & Authorization Code & Redirect URI

HOST (as Client for 1st OAuth)

AUTHORIZING USER (End-user at browser)

AUTHORIZATION MANAGER (as Authorization Server)

1st OAuth: Web server client profile
UMA - Step 2: Requester gets access token from AM

HOST (as Resource Server for 2nd OAuth)

AUTHORIZATION MANAGER (as Authorization Server for 2nd OAuth)

REQUESTER (as Client for 2nd OAuth)

Access request

Unauthorized, Location of AM

2nd OAuth - Step 2. Obtaining an Access Token (REQUESTER ACCESS TOKEN)

Request for Access Token (Client credentials, optional scope)

Loop Evaluate against applicable policy

Alt

Access Token (w/ optional Refresh Token)

Unsuccessful access token response

Claims-required document

Submit claims

Figure 1.8
5.3 **Scenarios**

5.3.1 **UMA – Step 1**

In this step authorizing user Introduces Host at Authorization manager.

5.3.1.1 **Scenario #1**

When Host is not registered at Authorization manager and Authorizing user wants to Introduce Host at Authorization Manager.
Figure 2.0
Database with empty tables

Figure 2.1
No Host registered with Authorization Manager

*Figure 2.2*
Introduce Host at given URL

**Figure 2.3**

**UMA Client Application**

Required UMA parameters:
- Host Initialization URL: http://localhost:8080/UMA_Host/uma/initialize
- Host Meta data URL: http://localhost:8070/UMA_AM/uma/retrieve_metadata
Introduction Confirmation at AM

UMA Authorization Manager

Confirmation

Do you want to introduce host to this Authorization Manager

[YES] [NO]

Figure 2.4
End URIs Data

We need endpoint for certain functionality in order to access it. An xml file is transferred to Host by AM containing all the endpoints of AM which is saved to database for future use. The endpoint for this file is provided by the Authorizing user in the start of the process as metadata URI.

Figure 2.5
Host Registration at AM

**Figure 2.6**

Host Registration

Required UMA Registration parameters:

- **Application Name:** HOST INTRODUCTION AT AUTH2 MANAGER
- **Application URL:** http://localhost:8090/UMA_Host/
- **Application Description:** Demo Application of the OAuth 2.0 Protocol
- **OAuth Registration Endpoint:** http://localhost:8070/UMA_AM/urno/register

[Image of the host registration interface]
Database with new entry

Figure 2.7
Host(s) registered at Authorization Manager

![Registered Hosts](image)

**Figure 2.8**

<table>
<thead>
<tr>
<th>Host Id</th>
<th>Host Name</th>
<th>Host Secret</th>
</tr>
</thead>
<tbody>
<tr>
<td>host_id</td>
<td>HOST INTRODUCTION AT AUTHZ MANAGER</td>
<td>host_secret</td>
</tr>
</tbody>
</table>
Getting Authorization Code from AM

![Host](image_url)

**UMA Host**

**Host Authorization**

- Required UMA parameters:
  - Requested Access Scope
  - End-User Authorization URL: [http://localhost:8070/UMA_A#umyouauthorize](http://localhost:8070/UMA_A#umyouauthorize)
  - Token Endpoint: [http://localhost:8070/UMA_A#umyoutoken](http://localhost:8070/UMA_A#umyoutoken)
  - Client ID: `host_id`
  - Client Secret: `host_secret`
  - Redirect URI: [http://localhost:8090/UMA_H#umaredirect](http://localhost:8090/UMA_H#umaredirect)

![Get Authorization Button](image_url)

---

**Figure 2.9**

---

Copyright © Fraunhofer-AISEC. All rights reserved.
Getting Access token from AM

**Figure 3.0**
Host Access token\n
---

**Figure 3.1**
Host access token saved at Authorization Manager

Figure 3.2
Access token saved at Host

Figure 3.3
5.3.1.2 Scenario # 2

In this Scenario authorization code will be issued to Host without asking for registration, because Host is already registered at Authorization server.

Getting Authorization Code from AM

![Diagram of Host authorization process]

**Figure 3.4**
Getting Access token from AM

**Figure 3.5**
Host Access token

*Figure 3.6*
5.3.2  UMA – Step 2 & Step 3

In these steps Requester (i.e. web browser) wants to access resource located on Host, Requester is asked to get permission (i.e. access token) from Authorization Manager.

5.3.2.1  Accessing resource if user is registered with AM and policy is assigned to it.

In this case access token is provided on the right user code.

Host Protected Resources Page

![Host Protected Resources Page](image)

*Figure 3.7*
Getting Requester access token

**Figure 3.8**

### UMA Host

**Getting Requestor Access Token**

Required OAuth parameters:

- **Token Endpoint**: `http://localhost:8070/UMA/AM/uma/requestor_token`
- **Client ID**: `srneebkhan@gmail.com`

[Button: Get Token]
Validating access token

![Diagram of UMA Host and validating token process](image)

**Figure 3.9**
Requested Resource

**UMA Host**

**Your Resource:**

![Figure 4.0](image)

*Figure 4.0*
In case of wrong credentials

Figure 4.1
Result of wrong credentials (Unauthorized User)

Figure 4.2
In case of wrong requester access token

Figure 4.3
Result of wrong requester access token

```
HTTP/1.1 401 Unauthorized
No such REQUESTER token exists
...
```

"We are Sorry. You can not access resource"

Figure 4.5
5.3.2.2 Accessing resource via “Facebook” if this requester is in “Facebook Friends list” of resource owner and policy is assigned to it

In this case the requester provides his/her code just like the previous case. System checks this code and if the user is facebook user, then he/she is redirected to facebook login page. When the user is verified by both facebook and system, an authorization code is then given to the requester.

Getting Requester access token

Figure 4.6
Figure 4.7
Requester Access token issued after verification

![HOST authentication interface with UMA parameters]

**Figure 4.8**
5.3.3 **Resource Upload Scenario**

This scenario will explain how a resource is uploaded on Host.

Login page

![Login page](image)

*Figure 4.9*
Welcome page is displayed after logging in to application

Figure 5.0
After clicking on “Upload Resource” on welcome page, upload resource page is displayed.

*Figure 5.1*
After selecting resource and filling in the fields

Figure 5.2
After uploading resource

![Image of a screen showing a resource upload interface with a message "Resource successfully uploaded"]

*Figure 5.3*
5.3.4 Protecting Resource with AM

Protecting resource after uploading it

Figure 5.4
Here we select the resource we want to protect and give location of AM

Figure 5.5
After specifying AM location and clicking on Protect Resource button, we are redirected to login page of AM where we can login with both “Account on AM” and “Facebook”.

Figure 5.6
After logging in, this page is displayed and the resource is protected with AM but policies are not yet assigned. Using the below page, requester can go back to “Upload Resource page” and “Assign Policy page”.

**Figure 5.7**
5.3.5 Assigning Policies to Resources which are protected with AM

Index page of AM

User-centric Web-Resource Management System

Managing of sharing information in social networks is a difficult task for users and can violate the user’s privacy. The UMA protocol enables the “Authorizing User” to control access to his/her own resources, i.e., identities, preferences, data, through an “Authorization Manager” (AM). Based on user’s policies the AM may grant access to third parties that request for these information. When a “Requester” attempts access to a protected resource owned and protected by the authorizing user stored at some “Host”, he is told where to go (the location of the AM) in order to request an access token. When he gets to the AM, he might be asked to convey claims about the requesting party, e.g., a web user or corporation, that satisfy the user’s policy. After obtaining the right access token the requester is then able to receive the information he originally asked for.

Figure 5.8
Login page of AM where user can login using both “Account on AM” and Facebook.

Figure 5.9
After logging in to AM, user is redirected to home page of AM.
After clicking on “Policy Administration” on home page, user will be redirected to this page. On this page user can assign policies to different users, groups. User can also assign policies to Facebook Friends by using the checkbox present at the end. If user checks this checkbox, friends of user will be fetched from facebook and the specified policy will be assigned to them.

Figure 6.1
After assigning policies to users and groups for a specific resource.

Figure 6.2
6 What's next?

6.1 Trusted Claim

A statement of the value or values of one or more identity attributes of a requesting party. Claims are conveyed by a requester on behalf of a requesting party to an authorization manager in an attempt to satisfy an authorizing user's policy.

6.2 JWT

JSON Web Token (JWT) is a compact token format intended for space constrained environments such as HTTP Authorization headers and URI query parameters. JWTs encode claims to be transmitted as a JSON object that is base64url encoded and digitally signed. The JWT signature mechanisms are independent of the type of content being signed, allowing arbitrary content to be signed. Encryption for JWTs is described in a separate companion specification.

6.3 Digital Signature

A digital signature or digital signature scheme is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, and that it was not altered in transit. Digital signatures are commonly used for software distribution, financial transactions, and in other cases where it is important to detect forgery or tampering.

7 Help for Developers

- This application is for Demo purpose which demonstrates how UMA protocol works. It consist of three servers named Client, AM and Host.

- For better understanding of the protocol, it is better to deploy these servers on different IP addresses. As these servers communicate with each other so IP addresses and ports of other servers must be known to every server. Constants which should be changed according to the IP addresses and ports are located in de.fraunhofer.aisec.leeloo.common package.

- We are using spring (3.0) framework. In spring the communication is done with the application through controllers. The controllers (also called Endpoints) which are accessed from outside application are located in
de.fraunhofer.aisec.leeloo.endpoints package. All Controllers which are for internal access are located in de.fraunhofer.aisec.leeloo.controllers package.

- We are using PostgreSQL database server for both UMA_Authz_Server and UMA_Resource_Server. Data related to database configuration is in database.properties file which is located in src folder.

- For communication between database and application we are using ibatus plugin. Configuration file of Ibatus named as sqlMap-config.xml is located in src folder.

- To run UMA_Resource_Server, a folder named uma_files is required in C directory otherwise change the value of UMA_ENDPOINTS_XML_FILE_NAME in Common class which is located in de.fraunhofer.aisec.leeloo.common package to any location.

8 References

http://kantarainitiative.org/confluence/display/uma/UMA+1.0+Core+Protocol

[2] IETF OAuth 2.0 protocol draft  
http://oauth.net/2/

[3] UMA Resource Registration  
http://kantarainitiative.org/confluence/display/uma/UMA+Resource+Registration

[4] WG - User Managed Access  
http://kantarainitiative.org/confluence/display/uma/Home

[5] OAuth leeloo – Vo.1  
http://smartjisc.wordpress.com/2010/09/30/oauth-leeloo-v0-1-released/#

[6] JSON-based format for expressing claims  
http://kantarainitiative.org/confluence/display/uma/Claims+2.0

http://mrtopf.clprojects.net/uma/draft-uma-scope-registration.html

[8] UMA Scoped Access Proposal draft-uma-scoped-access
JSON Web Token (JWT) Draft -01
http://self-issued.info/?category_name=claims

Extensible Resource Descriptor (XRD) Version 1.0
http://docs.oasis-open.org/xri/xrd/v1.0/xrd-1.0.html

Web Host Metadata

Dynamic Client Registration Protocol