ICT MyMedia Project
2008-215006

Social Networking Quickstart

17 June 2009

Public Document

1 Contents

2 Target Audience 4

3 Introduction 4

4 How To Implement 5

4.1 Integration with Facebook 5

4.2 OpenID Component 10

4.3 Virtual Profiles 11

Project acronym: MyMedia

Project full title: *Dynamic Personalization of Multimedia*

Work Package:	**2**

Document title:	**Social Networking Quickstart**
Version:	**N/A**
Official delivery date:	**N/A**
Actual publication date:	N/A
Type of document:	Open Source Software and Documentation
Nature:	Public

Authors:	Sergei Pavlov (MG)
Approved by:	N/A

2 Target Audience

This document tries to give a high level overview about the functionality of the social networking module, its concepts in brief and general information that the field trial implementers need to know prior to starting to use the part of the framework's API that is related to the social networking module.

See the MyMedia softare license, included at the root of the distribution, for terms of use.

3 Introduction

The core functionality provided by the social networking module is captured in the diagram below.

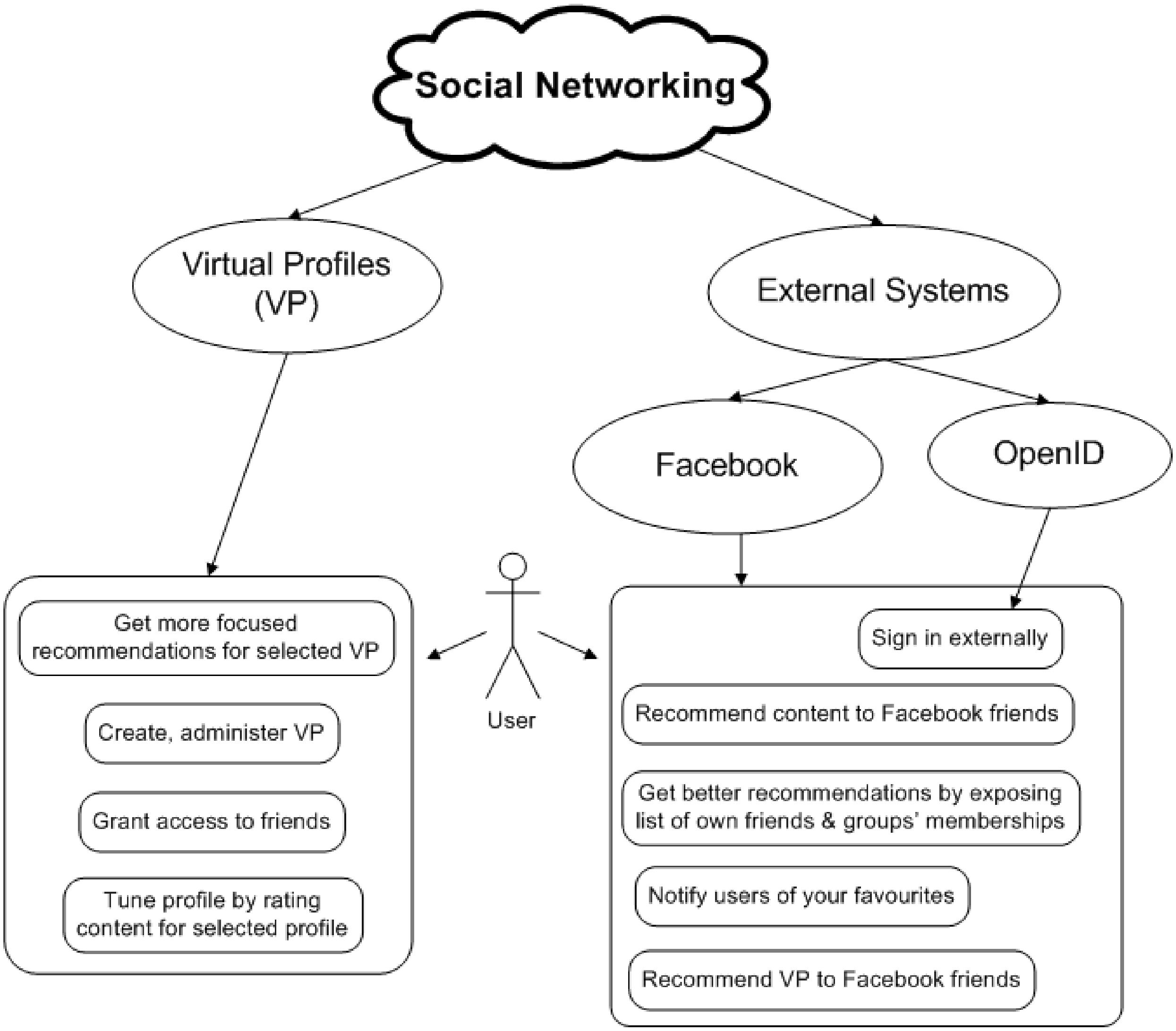

Social Networking
Virtual Profiles (VP)
External Systems
Facebook
OpenID
User
Get more focused recommendations for selected VP
Create, administer VP
Grant access to friends
Tune profile by rating content for selected profile
Sign in externally
Recommend content to Facebook friends
Get better recommendations by exposing list of own friends & groups' memberships
Notify users of your favourites
Recommend VP to Facebook friends

Illustration 1 Functionality provided by the social networking module

4 How To Implement

4.1 Integration with Facebook

4.1.1 Setting up a Facebook application

In order to connect to Facebook for authenticating a user, as well as any other operation (retrieving list of friends and groups, sending notifications to friends, personal news feed update, etc), an application account must be created on Facebook.

There the steps to do so below (You can have a look to other articles like *http://gathadams.com/2007/06/18/how-to-write-a-facebook-application-in-10-minutes/*

http://developers.facebook.com/get_started.php

as well).

1. Go to *http://www.facebook.com*
2. Create a user account (or sign in with an existing one)
3. Go to *http://www.facebook.com/developers/apps.php* and click 'Allow' if asked to provide access to user account's data
4. Click 'Set Up New Application'
5. Follow the registration steps, specifying the name of the application which *should be* field trial specific (e.g. BestDvdWorld *Recommender*). This name will be visible to users (e.g. when they are asked by Facebook to grant the access to their data to the application).
6. Facebook will generate 2 important settings for the application:
    a. API Key (corresponds to MyMedia. FacebookIntegrator setting 'API_KEY')
    b. Secret (corresponds to MyMedia. FacebookIntegrator setting 'APP_SECRET')

    These settings must be stored in the configuration (configuration file 'FacebookIntegrator.mmconf.xml') according to the setting names mentioned above for each of them.
7. The setting 'Callback URL' (on Facebook) should be set to some publicly accessible URL controlled by the field trial application (the data sent to this URL from Facebook would need to be passed to the MyMedia framework in certain cases). This URL must be saved in the

MyMedia’s configuration for ‘FacebookIntegrator’ module also, corresponding to the setting name ‘AUTHENTICATION_CALLBACK_DEFAULT_URI’.

8. Complete the registration of the application, uploading icon, logo, specifying descriptions for the users. This will make the application be available to its developers.

9. After editing quite a number of settings, and obtaining more than 5 users, the application can be submitted to be made available to every Facebook user. In order to obtain 5 users, 5 different Facebook users need to add the application. The adding is done by accessing the following URL

   http://www.facebook.com/add.php?api_key=<API_KEY>

   (where <*API_KEY*> is the key generated by Facebook for the application.)

   with a WEB browser by registered Facebook users having authenticated sessions (first go to facebook.com, authenticate, and then paste the URL to the WEB browser & add the application).

10. Finally, after all descriptions are entered, the application should be submitted to the Facebook ‘Application Directory’ (from the ‘My Applications’ page). Once the application is approved by the Facebook reviewers, normal Facebook users will be able to find, add and use the application (so, the application should be tested thoroughly, before it’s submitted).

4.1.2 Facebook Integrator’s Supported Features

At the moment of writing Facebook Integrator supports 5 “features”. The “features” are operations that are registered via configuration (framework’s social networking module reads in the “FacebookIntegratorFeatures.mmconf.xml” file on start up and registers the list of supported features according to the configuration). Nevertheless, it is up to the client application to verify whether some feature is enabled or not, e.g. prior to the attempt to execute the operation.

Optionally, the field trial application could ask users to choose which Facebook related features they want to enable, and which ones to disable. The framework provides corresponding methods to retrieve and query the list of features enabled by each user.

The list of the supported features described in more details follows.

- ExternalAuthentication

  This feature allows users to sign in to the field trial application via Facebook., which means users do not need to go through the registration process and/or remember their user name and password at the field trial application site. If a user is authenticated with Facebok already (has Facebook cookies in his WEB browser and an active session at Facebook site), signing in to the field trial application could be the matter of 2 or even just 1 mouse clicks. All other features require Facebook authentication to be done.

Brief technical explanation: when user decides to sign in via Facebook, his WEB browser would be redirected to Facebook's site to the URL like the following:

> *http://www.facebook.com/login.php?v=1.0&api_key=e29ebe49ca454d6708019ad3b841234f&next=login-fall-back-page.php*

The user would then need to sign in to Facebook with his existing Facebook account, and explicitly allow MyMedia client application to get the access to his data. After that the WEB browser would be redirected to the specified callback page, specifying an '*auth_token*' parameter. This token is then used to retrieve another parameter from Facebook called '*session_key*' with a direct HTTP GET request on the background (both values will be needed for possible further communication to Facebook). Sample request follows:

http://api.facebook.com/restserver.php?v=1.0&method=facebook.auth.getSession&auth_token=f76b003515cf83143f57990284721234&api_key=e29ebe49ca454d6708019ad3b841234f&call_id=1236604069812,5&sig=82580615eeda56a237d8d9d6b4df0e90

A sample response from Facebook follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<auth_getSession_response xmlns="http://api.facebook.com/1.0/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://api.facebook.com/1.0/
http://api.facebook.com/1.0/facebook.xsd">

  <session_key>480c732aa8640c484ff9b7fb-732021234</session_key>

  <uid>732021234</uid>

  <expires>0</expires>

</auth_getSession_response>
```

> Further requests can be made to retrieve e.g. user's first name and last name.

After retrieving user's Facebook ID, a client application can check if it is linked to some existing user account, and if not, e.g. create a new local account.

Once user is authenticated with Facebook, the rest of the features can be used (invisibly for user) to send or receive information to / from Facebook.

- GetUsersFriends

This feature might be useful when (Facebook-authenticated) user wants to recommend certain content item to his Facebook friends. This request might be slightly "heavy" in terms of the amount of information sent through the network (from Facebook to MyMedia Framework and from the Framework to the field trial client application), depending on the number of friends user has (in case of 100 friends, the amount of the information sent is around 16 Kbytes).

The feature allows retrieving first name, last name and Facebook user ID of each friend of the Facebook-authenticated user. The Facebook IDs of the users can then be used to send messages (typically, recommendations) to specific friends.

- SendMessageToAUser

Having Facebook user IDs of an authenticated user's friends, allows sending them messages (called "notifications" in Facebook's terminology). According to the Facebook API:

> *Notifications sent to the notifications page for non-application users are subject to spam control.*
>
> *[http://wiki.developers.facebook.com/index.php/Notifications.send]*

- UpdateUsersNewsFeed

The Facebook integration component allows for updating user's Facebook news mini feed (textual list of user's latest activities that is visible to his friends) for the users that have explicitly added the application. To add the application, users should be redirected to

> *http://www.facebook.com/add.php?api_key=<API_KEY*>

and explicitly agree to the terms and conditions.

Facebook sets a number of restrictions on this operation (format, length, spam control). For more detailed information about the this feature see Facebook API:

> *http://wiki.developers.facebook.com/index.php/Feed.registerTemplateBundle*

A field trial application might use this feature (preferably occasionally) to publish e.g. user's purchases. User should *probably* be aware of this activity taking place (subject to the privacy policy).

Technically, in order to send the news feed update, a "template bundle" needs to be registered at the Facebook site first. It is a template where only predefined fields can be replaced. A sample template bundle:

> *{*actor*} liked {*item_name*} at &lt;a href='http://www.mymediaproject.org/'&gt;MyMedia&lt;/a&gt;*

The template must start with the '{*actor*}' string (without quotes). The Facebook integration component allows publishing short stories with only 1 dynamic element (replacing the '*{*item_name*}*' place holder) at the moment.

The number of registered templates per application is limited (currently, 100). The component provides methods to manage the registered template bundles. In order to *change* the text message of the template bundle that publishes the story about user's interest in some content item, the configuration property "*news_feed_text_user_liked_an_item*" stored in the "FacebookIntegrator.mmconf.xml" file should be changed, AND

the programmatically generated property called

news_feed_text_user_liked_an_item_template_bundle_id

should be deleted (or set to an empty value "") from the configuration file '*FacebookIntegrator.mmconf.**local**.xml*'.

For more detailed information about the interaction to Facebook see the official Facebook API:

http://wiki.developers.facebook.com/index.php/API

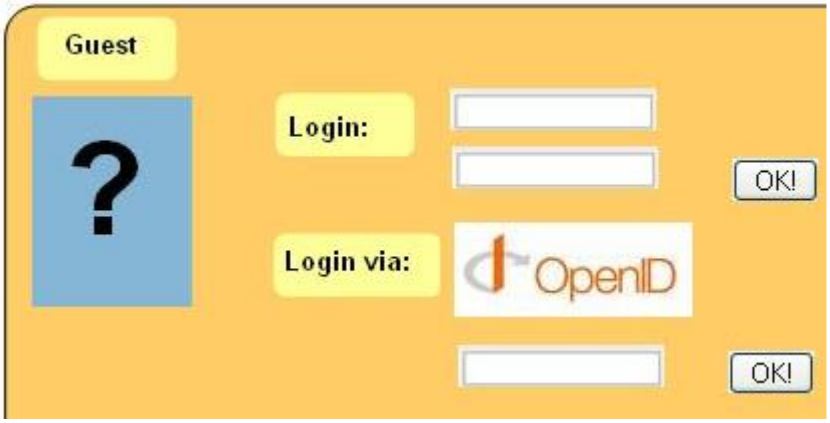
4.1.3 Restrictions

Facebook allows to store / cache user's data at external systems for 24 hours(that means that e.g. names of Facebook groups and groups' membership information must be deleted within 24 hours after retrieval). However, there is some information that can be stored indefinitely (importantly, Facebook user ids & group ids). More information about this restriction here:

http://wiki.developers.facebook.com/index.php/Storable_Information

4.2 OpenID Component

OpenID integration component provides an easy way to authenticate user with his OpenID account. The field trial application could allow users to sign in to the system with their OpenID by explicitly asking them to enter their OpenID, e.g.



There are 3 configuration files related to the OpenID integration component:

- OpenId.mmconf.xml : contains basic data; most likely no need to change anything.
- OpenIdIntegratorFeatures.mmconf.xml : information about the component to be registered in the framework correctly. Unless the list of the available authentication systems is truly dynamic, no need to change anything (The file just registers ‘ExternalAuthentication’ feature in the framework. So that the framework could be queried for the available authentication systems, if needed).
- **OpenIdIntegrator.mmconf.xml** : contains following settings:
    - ExternalSystemName : must coincide with the name of the OpenId integration component (“*OpenId*”).
    - IsExtensionsUsageDisabled : defines if OpenID provider will be asked for user’s extra data (user’s names, language, address, etc) or will verify the claimed identity only. Values: *true* / *false*.
    - AUTHENTICATION_CALLBACK_DEFAULT_URI : The URL controlled by the field trial application where OpenID providers will reply with user’s data & validation details (The details of these kind of fallbacks would need to be passed to the component). Sample value:

      http://www.buy-media-content.com/mymedia/openid_fallback.jsp

This value will be used by the integration component in case its operations are called without specifying the callback URL.

- AUTHENTICATION_CALLBACK_DEFAULT_TRUSTED_HOST_URI : part of the 'AUTHENTICATION_CALLBACK_DEFAULT_URI' setting, defining the callback's trusted host. Sample value:

  http://www.buy-media-content.com/mymedia/

The OpenID integration component needs to be able to perform outgoing HTTP-requests to the OpenID provider's server. This is required for discovery of the OpenID provider's entry points, determination of the supported protocol version (and other protocol implementation details, e.g. communication's encryption protocol) and the validation of the user's claimed identity.

The OpenID integration component provides 2 methods/services allowing to:

1) generate the URL where user's browser should be redirected so that user would authenticate with the OpenID provider.
2) parse OpenID provider's fallback request to extract (and immediately validate by performing some HTTP-based communication on the background) user's claimed identity.

4.3 Virtual Profiles

Virtual Profile is a dynamic collection of subsets of preferences (mainly ratings) that is used for classification or sharing purposes. Modifications to a Virtual Profile are allowed at any time according to the user rights: new preference items can be added, existing ones modified or discarded.

The framework exposes methods to perform basic management operations with virtual profiles (such as registering new virtual profile, modifying and retrieving its data, managing access rights granted to users, handling invitations to use a virtual profile, etc).

The main purpose of virtual profiles' concept is the ability to receive recommendations for selected virtual profile, which are optimized according to the feedback collected for that virtual profile.

Metadata of a virtual profile

Field	Description
VirtualProfileId	Database-unique ID of a virtual profile(VP)
Name	Database-unique VP’s name
Description	User entered description (without any restrictions)
IsPublic	Flag, determining whether the VP is accessible(usage right only) to everyone, or is private (requiring explicitly granted permissions to use it)
IsModificationAllowed	Flag, determining whether feedback can be submitted / contributed to the virtual profile by anyone, or requires explicitly granted permissions for that.
CreatorUserId	Id of the local user who has created the VP. The creator always has all access rights on the VP.
IsDisabled	Flag, determining whether VP is deleted. No VPs are ever permanently deleted from the DB.
UrlToBigLogo	URL to a large VP’s logo that can be displayed on VP’s main page.
UrlToSmallLogo	URL to a small logo that can be used on non-main page of the virtual profile next to the name of the VP.
Created	VP’s creation date and time
Modified	VP’s last modification’s date and time

4.3.1 Access rights

There are 3 types of access rights on virtual profiles, allowing to:

1) Get recommendations for the selected VP

   When the flag ‘VirtualProfile.IsPublic’ is set to true, any user is able to “use” the virtual profile, receiving recommendations on behalf of it.

2) Submit feedback for the selected VP

   When the flag ‘VirtualProfile.IsModificationAllowed’ is set to true, any user is able to submit feedback (typically, the ratings) to the virtual profile, even if receiving recommendations is not available to the user. This option might be useful to collect a lot of feedback for the given topic of the VP.

3) Administrate the selected VP

This option allows the user that has this right granted to modify any user-entered VP related data (like name, description, ratings (existing ones can be deleted), access rights).

The creator of a virtual profile always has all access rights on it. S/He can grant any access right(s) to local or external users. If an access right is granted to an external user, this external user must authenticate via the external system (currently, just Facebook) that has been used when granting the access right. This external authentication must be performed at least one time (once local user's account is linked to the external one, the granted access right will be available to the user when authenticating locally as well).

4.3.2 Invitations to use a virtual profile

Users can invite their friends to use specific virtual profile. The framework provides functionality to register, retrieve invitations and mark them as read. When registering an invitation, the framework does not perform any other actions (like sending a message via Facebook). However, this step (sending the Facebook 'notification') can be performed separately, if desired. The framework does not provide the functionality to send out emails at the moment.