Integrating Drupal and iMIS: ALAconnect Case Study

December 24th, 2008

by Abhijeet Chavan, Ki Kim, Jenny Levine, and Chris Steins

Table of Contents:

1. Introduction

Over the last year, some users of the Drupal content management system (CMS) have expressed interest in integrating Drupal with iMIS, a membership and financial management system used by non-profits, membership organizations, and educational institutions. We have successfully implemented a custom solution for an ongoing project that links Drupal to iMIS. The solution includes a custom module for integrating Drupal & iMIS. We plan to release this module to the Drupal community once we have completed testing, made the module more suitable for general use, and prepared detailed documentation. Since the module is not likely to be ready until Spring 2009, we felt that the Drupal and iMIS communities might benefit from a summary of our approach and progress to date.

2. Background

The American Library Association (ALA) is the world's oldest and largest library association. Based in the United States, the organization promotes libraries and library education internationally. The American Library Association (ALA) and Urban Insight (UI), a technology consulting firm, began working together in March, 2008, to develop an online community and professional networking website for ALA's 65,000 members. The online community website, called *ALA Connect*, is being built using the Drupal web content management system (Version 5.x) using MySQL and running on a GNU/Linux operating system. In this case study, we will refer to the online community website as *ALA Connect/Drupal*. The main website for ALA associated with the iMIS system is referred to as *ALA/iMIS*.

Integration of Drupal with iMIS for the ALA Connect/Drupal website has two primary goals:

1. **Single Sign-on:** In order to enable members to navigate between the main ALA/iMIS website and the ALAconnect/Drupal website, the project requires *single sign-on* (SSO) for the ALAconnect/Drupal website. This would enable members to authenticate to the ALA Connect/Drupal website using credentials stored in ALA's primary membership system, which is operated using iMIS.
2. **Data Exchange:** ALA Connect/Drupal must have information about ALA members in order to assign members permissions to various community groups, and display member profile information. To collect information about ALA members, the ALA Connect/Drupal website must regularly, and in an automated fashion, retrieve selected membership data from ALA/iMIS.

3. Technical Architecture

In order to provide SSO, we built a custom Drupal module that works hand-in-hand with a custom update engine we developed to handle the data exchange from iMIS to Drupal. The custom module and update engine together form what we call the *Authentication and Update Subsystem* (AUS). The architecture of the AUS consists of two parts:

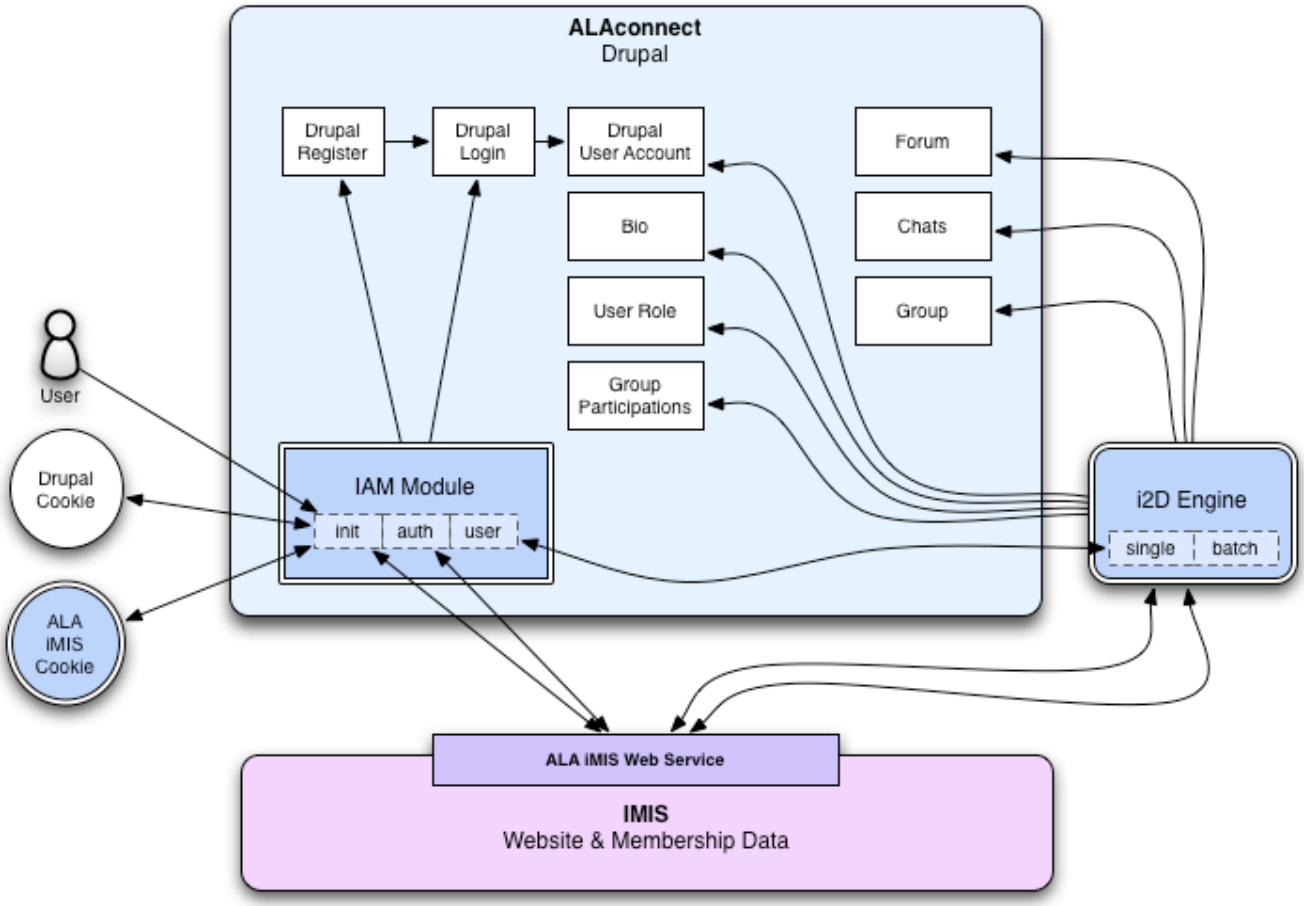
- **IAM:** The *Integrated Authentication and Single Sign-on Module* (IAM) is a custom module we developed that modifies Drupal's default behavior for handling authentication and basic user settings. The module integrates authentication between Drupal and iMIS and provides the single sign-on feature required by the project. IAM enables an ALA member to authenticate and gain access to ALA Connect/Drupal using their membership username and password stored in ALA/iMIS.

- **i2D:** After a user has been authenticated via the IAM module, the update engine we built, called the *iMIS to Drupal Retrieval and Conversion Logic Engine* (i2D), checks with iMIS and ensures that ALA Connect/Drupal has the most current data for the member who has just signed on. If needed, i2D transfers selected member data from iMIS to ALA Connect/Drupal.

The i2D engine operates in two different modes. In *single mode*, i2D retrieves an individual user's data upon authentication. i2D also runs in a *batch mode,* sequentially updating hundreds or thousands of member records during a regularly-scheduled membership update process. The i2D data exchange logic is similar in i2D single and batch modes. (There are a few important differences but they are not relevant to this case study.)

AUS depends upon and uses a custom iMIS web service that provides authentication and membership data to IAM and i2D. The ALA/iMIS web service was not part of the development work performed by Urban Insight. IAM and i2D use the SOAP ptrotocol to communicate with the ALA/iMIS web service. It provides several methods, some of which return member profiles, member/group participations, an updated member list for the last 24 hours, and also verifies member login information among many functions.

The following diagram is a visual representation of the interaction among the different components of AUS.



4. Solution

4.1 Integrated Authentication and Single Sign-on (IAM)

Drupal's modular architecture makes it possible to extend and add features to the CMS by creating custom modules. Custom modules may include *hook functions* -- PHP functions that Drupal calls to implement new features. IAM, the custom Drupal module we developed performs three major tasks, each of which is accomplished by three corresponding hook functions:

- Single Sign-on
- Remote Authentication

- Update User Information

4.1.1 Single Sign-on

IAM's first task is to implement single sign-on by allowing the user to be authenticated if IAM finds a valid cookie created by the ALA/iMIS website, without having to go through Drupal's regular log-in process. This task is handled by the `hook_init()` hook function within the IAM module. This hook is invoked the very first time during Drupal *bootstrap*, even before the authentication process kicks in. Following is the pseudo-code of the `iam_init()` hook function, stripped of all trivial steps to show the core functionality:

```
<?php
  function iam_init() {
    if (Cookie exists and imis confirms user session is valid) {
      if (User account exists in Drupal) {
        Make the user authenticated.
      }
    }
  }
?>
```

If the user had visited the main ALA/iMIS website before coming to the ALA Connect/Drupal site, the user's browser would have a cookie that is intended to be read by the ALA sister sites. The cookie contains a *session ID*. IAM checks the cookie's content against the iMIS web service. An iMIS web service method would hold the session information while the user is active in the domain. By sending the session ID to the web service, `iam_init()` determines if the session ID is valid. When this step successfully authenticates the user, the user would be logged in automatically and see a website welcome message.

4.1.2 Remote Authentication

The second task the IAM module performs is to process remote authentication. It implements two related hooks: `hook_info()` and `hook_auth()`. (Note that these hooks apply only to Drupal 5, and they are dropped from Drupal 6. In Drupal 6, developers are expected to use `hook_form_alter()` instead for the same effect. More useful information about migrating from Drupal 5 to 6 regarding this issue can be found at www.norio.be/blog/2008/08/migrating-drupal-6-hookauth-and-hookinfo.)

The purpose of `hook_info()` is to declare custom authentication scheme information, which is required of an authentication module. Here is the pseudo-code for `hook_auth()`:

```
<?php
  function iam_auth($username, $password, $server) {
    if (iMIS web service method verifies log-in info collected from user input) {
      if (User exists in Drupal) {
        Create user object to be used throughout Drupal loading
        Save a cookie for sister sites in the domain to use for single sign-on
      }
      return true
    }
    else {
      return false
    }
  }
?>
```

We devised a scheme to authenticate remotely by passing user credentials to iMIS. The user's password is not saved in Drupal. In fact, even the username in Drupal is not used for authentication. Instead, an iMIS ID is used to identify users between Drupal and iMIS. To hold the iMIS ID, we added a new field, *"imisid"*, to Drupal's "`users`" database table. (Modifying a Drupal table is not ideal for various reasons. We could have stored the iMIS ID by creating a separate table to store this information. However, the iMIS ID was an essential part of this scheme and in the interest of quickly implementing a solution that worked, we decided to modify the "`users`" table. Before releasing the module to the Drupal community, we intend to rebuild this module so no modifications to Drupal tables are required.)

Note that Drupal's default behavior is that if `hook_auth()` returns true while user account does not exist in Drupal, Drupal will create an account for the user automatically. We'd also like to mention that, since Drupal username is not used for remote authentication at all, we had to modify a couple of lines in user module to ensure that iMIS ID was used to load a user object instead of username. We didn't want to use the iMIS ID for the Drupal username because it is numeric, and we didn't want to expose member numbers on the open web.

4.1.3 Update User Information

The last task of IAM is to *update user's profile and group participation information*. Two iMIS web service methods are called for the task and each call to the web service could take one or two seconds. This means if the module tries to call several web services upon user log in, it could take 4 or more seconds until user information is updated. Waiting several seconds to log in would not have been desirable. So we decided not to update the user profile at the time of log-in, but rather to leave the task to our second module, i2D, in a separate process.

4.2. iMIS to Drupal Retrieval and Conversion Logic Engine (i2D)

i2D module is not a Drupal module, but a stand alone script that can run independently. However, it employs Drupal's *bootstrap* process to have access to all the facilities and functions Drupal provides. We decided to use Drupal's *bootstrap* to avoid having to write a large amount of code from simple helper functions to database access functions.

i2D runs in two modes, single and batch. Two modes share the majority of the logic with slightly different purposes.

Triggered by *cron*, the scheduler service for Unix-like operating systems, an i2D batch mode runs every night when the site load is low. It calls the ALA/iMIS web service, which returns a list of user ID's for profiles that have changed during a specified date range. i2D batch normally sends parameters for the start and end dates. Start date is the timestamp of the last successful batch run, and end date is the current timestamp. Since i2D batch runs daily, the specified date range requested by i2D is usually for the last day. When start date is omitted or set to a far past date, iMIS would return profiles for almost all users.

i2D single mode, again activated by cron, runs every minute. This mode works in conjunction with the IAM module. The IAM module, upon successful authentication, inserts an entry into a table to be used by i2D single mode, which processes users in the table if it finds the table is not empty. This way, when a user logs in, the profile update is not performed right away in order to reduce the waiting time for log-in. Instead, it is executed by the i2D single mode that runs no more than a minute later. Therefore, it is possible that a user may not see the updated profile information within the first minute after log-in. We felt this to be an acceptable compromise in the interest of letting users log on quickly without having to wait for the profile to be updated.

Following is the pseudo-code for the main logic of i2D:

```
<?php

 if (Process is already running) Exit

 if (In batch mode) {
   Request to web service for list of "changed" users since last successful run
   Append the received list to sync list
 }

 loop {
   if (Process ran too long) Time out and break out of loop;

   while (There is a member to process in the sync list) {
     Request to web service for current member's profile
     Update user profile

     Request to web service for current user's group participation

     foreach (group in group participation) {
       if (group does not exist) Create group
       Subscribe current user to the group
       Set the user to be the group's admin if he has approriate role in the group
     }

     Delete any old group participations that were not in the participation just received
   }
 } until (sync list is empty)
?>
```

Note that "`sync list`" table is a different table from the one used by i2D single mode. The "sync list" table can have thousands of rows, which can be an overhead for single mode to work with, because it runs every minute and should have very little overhead possible. The table used by single mode keeps entries for a minute or so and empties every minute as soon as single mode has done its job.

i2D single mode and batch mode are run by different cron jobs and are considered separate processes. Even though there could be only one process for each mode at a given time, single mode and batch mode could run at the same time with no problem.

5. Conclusion

The ALA Connect/Drupal project is in the early alpha testing phase. During this period, AUS has been able to reliably update membership data from iMIS to Drupal. Users are able to log in without noticeable delays. We are now working on making AUS more suitable for general use and refining aspects of it to make them more in-line with Drupal best practices for custom modules. We plan to release the custom module to the Drupal community in Spring 2009. We welcome feedback from those considering integrating Drupal and iMIS.

About the Authors:

- Abhijeet Chavan is the Chief Technology Officer of Urban Insight, Inc.
- Ki Kim is a Senior Developer at Urban Insight, Inc.
- Jenny Levine is the Internet Development Specialist & Strategy Guide for the American Library Association.
- Chris Steins is the Chief Executive Officer of Urban Insight, Inc.

For more information about this project please see http://www.urbaninsight.com/articles/drupalimis/