TechNote: Building a Social Advocacy Campaign on the Convio Open Framework

A Session Presented by Steve Mook and Doug Fierro at Convio Summit 2010

Overview

The session "Building a Social Advocacy Campaign on the Convio Open Framework," presented at the Convio 2010 Summit conference in Baltimore, demonstrated how an end-to-end advocacy campaign might be built on the Convio Open platform, leveraging cloud computing infrastructure and open-source software to create a compelling user experience as part of a cohesive campaign.

This paper details the steps involved in creating that campaign and the demo site that was used in the presentation. In broad terms, the steps are as follows:

1. Create a Drupal Micro-Site in the Cloud

- Provision a Bitnami Drupal instance on Amazon EC2
- Access the micro-site server
- Configure the necessary modules

2. Configure the Convio Site

- Set up APIs
- Link to the micro-site
- Create the Action Alert and Custom Interactions

3. Build the Micro-Site Content

- Create Content Types
- Create Content

4. Code links to Social Networks

- Facebook
- Twitter

5. Code the Action Alert

- Personalize wih the Convio Drupal Module
- Take action with the Convio Advocacy API

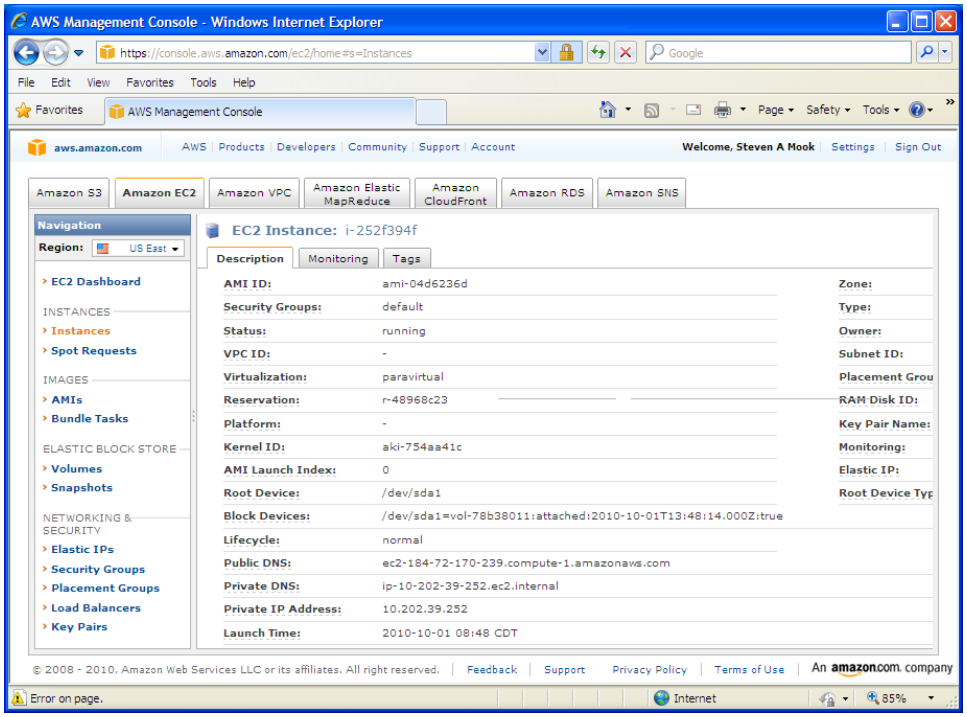
The goal of this TechNote is to pass on lessons learned in building the cloud site and integrating it with the Convio Open platform, and provide working example code that you can adapt and customize for your own purposes.

1. Create a Drupal Micro-Site in the Cloud

There are many hosting vendors and cloud computing packages to choose from. Our choice of Amazon Web Services running a Bitnami Drupal instance for this presentation was driven primarily by the need to make a choice. It should not be interpreted as an endorsement of a particular company or technology, and the methods discussed here are generally applicable to other computing platforms as well.

PROVISION A DRUPAL SITE ON AMAZON EC2

Before beginning, we established an Amazon AWS account, done by clicking **Create an AWS Account** on the Amazon Web Services home page, aws.amazon.com.

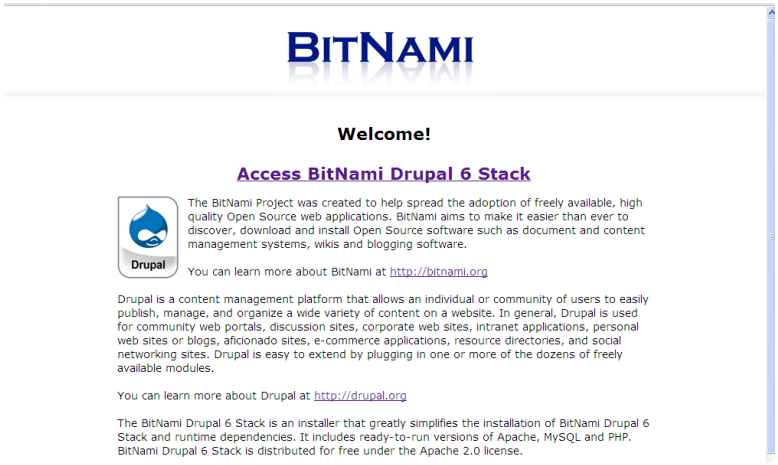


To provision a Bitnami Drupal web server instance on AWS:

1. Log in to the Amazon Management Console: aws.amazon.com/console and click the **Amazon EC2** tab.
2. Go to **Key Pairs** and create a key pair to use when accessing your instance via SCP. You will be prompted to download your private key for the site. Put it somewhere safe and accessible, as you will need it in order to obtain file system access to the site later.
3. Make sure ports 80 and 22 are open in the default group under **Security Groups**.
4. Go to the **AMIs** section, choose an image from the list, right-click and choose **Launch instance**.
5. The new instance will appear in the **Instances** section.
6. Obtain the public DNS for the site, for example "ec2-184-73-142-208.compute-1.amazonaws.com." You will use this URL to access the site via a browser. In Convio, you will add this domain to the list of trusted and allowed domains for JavaScript access. The three numbers after "ec2" are the IP address of the site; you will also white-list this address in the server API configuration of your Convio site.

RESET YOUR ADMIN LOGIN CREDENTIALS

Once your instance is up and running, you can browse to the public DNS URL for your site, which should display the welcome page:



Click **Access Bitnami Drupal 6 Stack** and log in to Drupal using the default administrator account:

- **User:** user
- **Password:** bitnami

Select **My account ->Administer** to change your admin user name, email and password.

CONFIGURE REMOTE CONSOLE ACCESS

To add modules and code to your virtual Drupal micro-site, you need remote file system access to the virtual server it runs on. When you created the EC2 instance, you associated it with a private key. We will use that key to access the server using a remote file system console.

Remote consoles **WinSCP** available from winscp.net/eng/index.php and **Putty** available from www.chiark.greenend.org.uk/~sgtatham/putty/download.html require a different key format than the one downloaded from Amazon. Use **PuTTYGen** from www.chiark.greenend.org.uk/~sgtatham/putty/download.html to convert it.

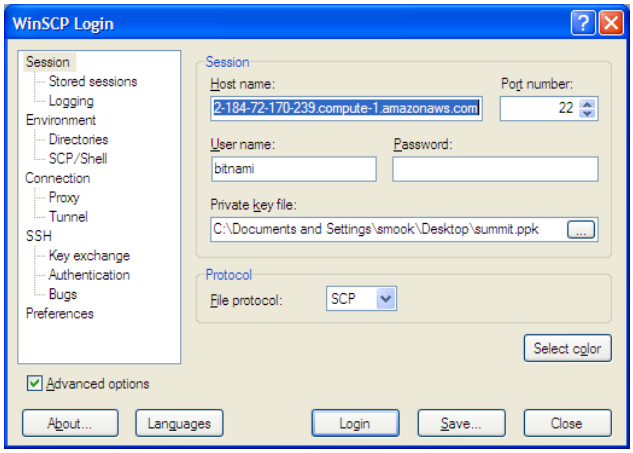
To convert the key file:

- At the command prompt type: `puttygen.exe`
- Go to **Conversions**, choose **Import** and select your key file, e.g. *mykeyfile.pem*
- Choose **Save private key** and save the converted file, e.g. *mykeyfile.ppk*

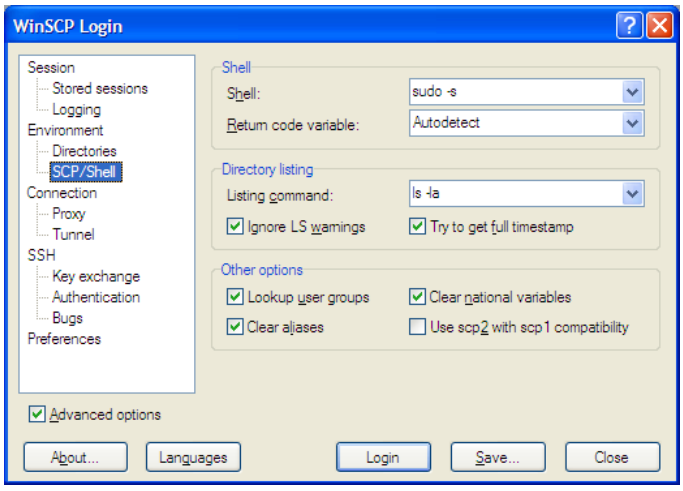
ACCESS THE EC2 INSTANCE VIA WINSCP

WinSCP and similar remote operating system consoles provide secure file system access to your virtual server. WinSCP lets you browse and update files on the remote system, sync them with files on your local machine, and run commands on the remote server. To use WinSCP to access the file system of our cloud site:

1. Launch WinSCP and click **New** to create a session.
2. For **Hostname** provide the DNS name of your server (e.g. ec2-184-73-142-208.compute-1.amazonaws.com).
3. **User name:** bitnami
4. For **Private key file** provide the path and file name of your private key file (e.g. *mykeyfile.ppk*).
5. Set **Protocol**: SCP.

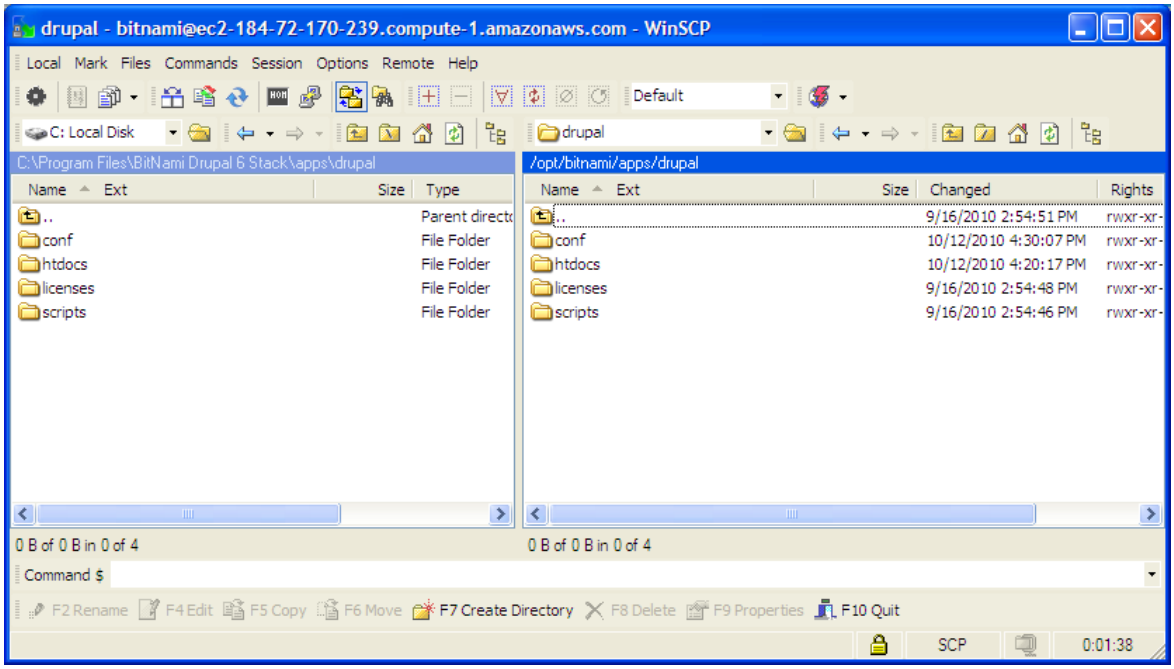


6. In the left menu, under **Environment**, choose **SCP/Shell**.
7. Set **Shell:** sudo –s



Click **Save** to save your session configuration.

To connect, click **Login**. If login is successful, the console will open, displaying the local file system on the left and the remote file system on the right.



INSTALL DRUPAL MODULES

The Drupal stack is installed under **/opt/bitnami/apps/drupal**. This is where we will apply modifications and customizations to the Drupal site.

- Create a directory called **modules** in **/opt/bitnami/apps/drupal/htdocs/sites/all**
- Copy any needed modules from your local instance of Drupal to your remote EC2 server.

For the purposes of the demo, we installed the Content Creation Kit (CCK), Organic Groups (OG), and Views modules, all of which were downloaded from drupal.org/download.

We also installed the prototype Convio module referenced on the Convio Open Downloads page open.convio.com/downloads/convio-api-integration.html

IMPORTANT NOTE

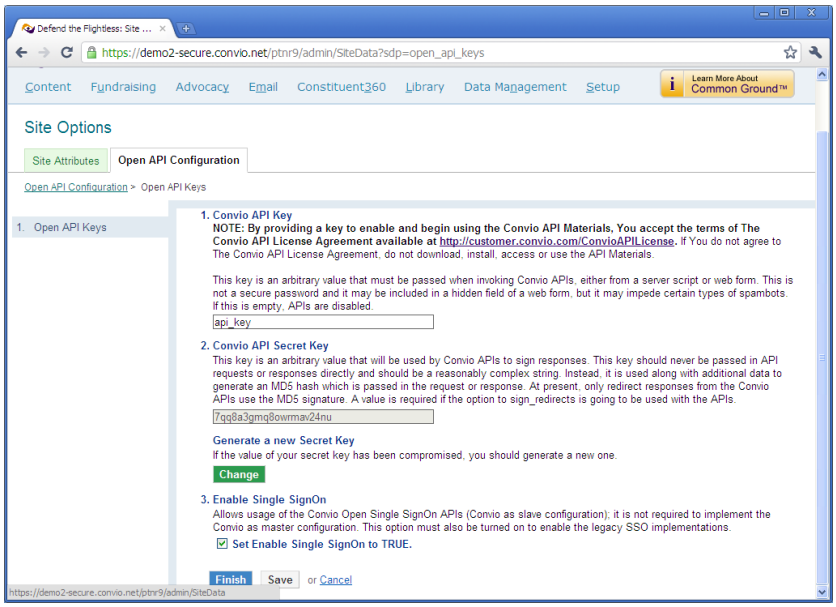
- DO NOT ENABLE the Convio module before creating a constituent on your Convio site with the same username/password as your Drupal admin.

2. Configure the Convio Site

In addition to creating and configuring the instance for your Drupal micro-site, you also need to configure your Convio site for API access. If your site is already configured for API access you only need to add your Drupal micro-site's IP address and domains to the appropriate white lists, create a constituent record for your Drupal site administrator, and copy the site's JavaScript API client file to the Drupal micro-site.

SET UP CONVIO FOR API ACCESS

Log in to Convio as a site administrator and go to **Setup/Site Options/Open API Configuration**.



Configure an **API Key** and **Secret Key,** and **Enable Single Sign On**.

Copy the value of your **API Secret Key** value into a file and save it as **secret_key.txt**. We will copy this file to **/opt/bitnami/apps/drupal/htdocs/assets/** on the Drupal micro-site later when we configure the Convio Drupal module.

Configure an **API Administrator** account and add the site IP address (embedded in the public DNS) to the Server API **White List**.

Add your external site's domain to the **JavaScript/Flash Configuration** white lists of **Allowed** and **Trusted** sites.

CREATE A CONVIO LOGIN FOR YOUR DRUPAL ADMINISTRATOR

Go to **Constituent360/Constituents** to register your Drupal site administrator on the Convio site using the same user name and password you used for the Drupal site.

CREATE CUSTOM INTERACTIONS

In Convio, go to **Setup/Interaction Management**

Create custom interactions you want to track from the remote site, e.g. **Facebook Like**, **Twitter Tweet, Message Post**.

OBTAIN THE CONVIO CROSS-DOMAIN JAVASCRIPT CLIENT

Using your browser, navigate to a URL made up of your Convio web site domain plus **/api/api_client.html** (for example: http://www.my-npo.org/api/api_client.html).

Save a copy of **api_client.html** locally.

Use the remote console (WinSCP) to copy this file to the **/opt/bitnami/apps/drupal/htdocs** directory on your Drupal app server. This is necessary to allow the Convio cross-domain JavaScript library to allow AJAX calls between your Drupal and Convio domains.

Make a directory under **/opt/bitnami/apps/drupal/htdocs** called **assets**, and copy the **api_client.html** file here, along with the **API Secret Key** file **secret_key.txt** you saved in the earlier step. We will use these to configure the Convio Drupal module.

JAVASCRIPT UPDATE

- If your site used Convio APIs prior to the Fall 2010 release, first clear and then re-set your **API Key** in order to use JavaScript with:
  - Advocacy
  - AddressBook
  - Content
  - Groups
- Clearing and re-setting the API Key causes the JavaScript API files to be re-built from the product templates.

3. Build the Micro-Site Content

We configured the demonstration micro-site using Organic Groups to let new users register and post quotes that other users could "Like" or "Tweet" This involved the following basic steps:

- Log in to your Drupal site using the administrator account.
- Select **Administer->Modules** to enable installed modules (we're enabling **CCK**, **OG**, and **Views** modules).
- Go to **Administer->Site Configuration->Site Information** to change boilerplate site titles and information
- Create **Group_node** and **Group_post** content types using the Organic Groups module, as described in the Organic Groups Documentation. These allow us to create groups, and allow group members to post content to the group's message board. We will track activity in these groups by adding JavaScript hooks that log Convio custom interactions into the page theme templates for the site and the group node.

We also bootstrapped the Convio Drupal module installed earlier, to connect our micro-site with our Convio site via single-sign-on, and to give us access in PHP to constituent profile information.

BOOTSTRAP THE CONVIO MODULE

Before you do this, make sure that you have a constituent on your Convio site with the same username and password as your Drupal site administrator. Failure to do so may land you deep in the rough (says the voice of sad experience).

Logged in as the site admin on your Drupal site, Go to **Administer->Site building->Modules** and enable the **Convio** module.

Browse to **http:yourdomain/drupal/admin/convio/configure**:

Set up the Convio module as follows:

1. In **API Service URI** add the secure URI of your Convio site's API servlet root.
2. Under **API Public Key** add the value of the **API Key** from your Convio Open API Configuration (the value returned in the **api_key** parameter when calling the APIs).

3. Under API Secret Key File add the path and file name of the Secret Key file you copied to the Drupal site while setting up Convio Open API access: **/opt/bitnami/apps/drupal/htdocs/assets/secret_key.txt**
4. Under **API User** add and **API Password** add the user name of the Server **API Administrator** you configured when setting up Convio Open API access. Note this should not be the same as your Drupal site admin user.
5. Click **Save** to finish configuring the Convio Drupal module.

Registered user on your Convio site should now be able to log in to your Drupal micro-site using their Convio credentials.

4. Code Links to Social Networks

Now it's time to write some JavaScript and PHP! We'll mainly use template customization to hook our site into the social grid. For the demonstration we used Drupal's default Garland theme, the page template for which is **\opt\bitnami\apps\drupal\htdocs\themes\garland\page.tpl.php**.

Now that both systems are configured, we will add JavaScript code to our Drupal theme's page templates to capture "Like" and "Tweet" actions.

We will implement the Facebook "Like" feature using the Facebook JavaScript API and FBML. This feature uses OpenGraph to obtain additional information about your site. Replace the opening <html> tag in the page template of your site's theme with the following tag:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:fb="http://www.facebook.com/2008/fbml"
      xmlns:og="http://opengraphprotocol.org/schema/"
      xml:lang="<?php print $language->language ?>"
      lang="<?php print $language->language ?>" dir="<?php print $language->dir ?>">
```

In the page template's header, we also add the OpenGraph meta tags, replacing the values for **og:site_name** with the name of our micro-site and **fb:admins** with the Facebook ID of the owner of the FBML application.

```
<meta property="og:title" content="<?php print $title ?>"/>
<meta property="og:type" content="non_profit"/>
<meta property="og:image"
   content="<?php print 'http://'.$_SERVER['HTTP_HOST'].check_url($logo) ?>"/>
<meta property="og:site_name" content="EmuAdvocate"/>
<meta property="fb:admins" content="1000000000"/>
```

Also in the page template header, add references to the JavaScript libraries we plan to use:

```
<script type="text/javascript" src="http://ptnr9.demo2.convio.com/api/ConvioApi.js"></script>
```

ADD SCRIPT TO PAGE TEMPLATE

First we add a hidden to the template's page content where we can stash some constants that we will pass as values to the **logInteraction** API.

```
<div style="display:none">
  <form method="get" action="/" id="customInteractionForm">
    <input type="hidden" id="cons_id" name="cons_id" value="" />
    <input type="hidden" id="interaction_subject" name="interaction_subject" value="" />
    <input type="hidden" id="interaction_type_id" name="interaction_type_id" value="" />
  </form>
</div>
```

We also add an empty to be used by the Facebook FBML API:

```
<div id="fb-root"></div>
```

CAPTURE SOCIAL SHARE EVENTS

It is important to remember that the "A" in AJAX stands for "asynchronous" and that asynchronous means "not synchronous." That sounds perfectly obvious, doesn't it? Overlooking that perfectly obvious fact cost us a few frustrating hours of wondering why strange things were sometimes happening, and then again sometimes not happening. It also explains the crude but effective daisy-chaining from one function to the next in the code below.

We are doing several things in this script: pulling the "source" parameter out of the URL query string for link tracking purposes, checking for a Convio session and retrieving the user's **cons_id**, setting up Like and Tweet buttons on the page and logging events as custom interactions.

```
<script type="text/javascript">
<!--
var api = new ConvioApiClient('api_key', '/drupal/api_client.html', 'xml');

function logInteractionComplete() {
    return false;
}

function logInteraction(cons_id) {
    document.getElementById('interaction_cons_id').value = cons_id;
    api.callConsAPI("logInteraction", logInteractionComplete,
                    document.forms['customInteractionForm']);
    return false;
}

function getLoggedInComplete(responseXML, status) {
    if (status == 200) {
        var cons_id = responseXML.getElementsByTagName(
            "cons_id").item(0).firstChild.nodeValue;
        logInteraction(cons_id);
    }
}

function getLoggedInCons() {
    //must make this call on the insecure channel,
    //since that's where the login session is
    var cb = function(facade) {
        getLoggedInComplete(facade.responseXML, facade.status);
    };
    api._insecureRestCall('POST', 'http://ptnr9.demo2.convio.com/site/CRConsAPI',
                cb, 'method=loginTest&v=1.0&api_key=api_key&response_format=xml');
}

function logLike() {
    //set the form interaction type ID to "facebook like" type
    document.getElementById('interaction_type_id').value = "1000";
    document.getElementById('interaction_subject').value = "Like";
    getLoggedInCons();
}

function logTweet() {
    //set the form interaction type ID to "twitter tweet" type
    document.getElementById('interaction_type_id').value = "1001";
    document.getElementById('interaction_subject').value = "Tweet";
    getLoggedInCons();
    return true;
}

function logPost() {
    //set the form interaction type ID to "message post" type
    document.getElementById('interaction_type_id').value = "1010";
    document.getElementById('interaction_subject').value = "Post";
    getLoggedInCons();
    return true;
}

//Initialize facebook api
window.fbAsyncInit = function() {
    FB.init({
        appId: '100000000000000', //Replace with your FB App ID
        status: true,
        cookie: true,
        xfbml: true
});
FB.Event.subscribe('edge.create', function(href, widget) {
```

```
    logLike();
});
};
(function() {
    var e = document.createElement('script');
    e.async = true;
    e.type = 'text/javascript';
    e.src = document.location.protocol + '//connect.facebook.net/en_US/all.js';
    document.getElementById('fb-root').appendChild(e);
}());

// Parse a parameter from the query string
function getParm(name) {
    var pattern = "[\\?&]" + name + "=([^&#]*)";
    var regex = new RegExp(pattern);
    var results = regex.exec(window.location.search);
    if (results === null) {
        return "";
    }
    else {
        return results[1];
    }
}

// Page initialization
$(document).ready(function() {
    var src = document.getElementById('sub_source');
    var frm = document.getElementById('node-form');

    // Parse and save the suource parameter if one appears in the url query string
    if (src !== null) {
        src.value = getParm('source');
    }

    // Add a submit listener to the Create Content form
    if (frm !== null) {
        frm.setAttribute("onsubmit", "return logPost();");
    }
});

//-->
</script>
```

ADD LIKE AND TWEET BUTTONS

We add the Facebook Like and Twitter Tweet buttons to the Open Groupware module's post theme, which we do by customizing the post template: **/opt/bitnami/apps/drupal/htdocs/sites/all/modules/og/theme/node-og-group-post.tpl.php**:

```
<!--facebook like -->
<fb:like class=" fb_edge_widget_with_comment fb_iframe_widget"
  width="95"
  show_faces="false"
  layout="button_count"
  href="<?php
    if ($node->nid) print 'http://'.$_SERVER['HTTP_HOST'].'/drupal/node/'.$node->nid;
    else print 'http://'.$_SERVER['HTTP_HOST'].'/drupal'; ?>?source=facebook" >
</fb:like>

<!--twitter tweet-->
<a target="_blank" onClick="logTweet();" href=http://twitter.com/share?url=<?php if ($node->nid)
print 'http:%2F%2F'.$_SERVER['HTTP_HOST'].'%2Fdrupal%2Fnode%2F'.$node->nid.'%3Fsource%3Dtwitter';
else print 'http:%2F%2F'.$_SERVER['HTTP_HOST'].'%2Fdrupal%3Fsource%3Dtwitter'; ?>&text=<?php
print drupal_urlencode($title) ?> >
   <img src="http://s.twimg.com/a/1287437169/images/goodies/tweetn.png" alt="Tweet" />
</a>
```

4. Code the Action Alert

The Action Alert on the demonstration micro-site does not use AJAX, but rather is implemented as a simple HTML form. It demonstrates the basic capability of the Convio Advocacy APIs but does not go into advanced features such as the use of **getAdvocacyAlert** method in conjunction with **takeAction** to create a more sophisticated user experience.

The JQuery Validate plug-in permits us to easily validate input fields by adding a **class="required"** attribute to the field inputs and hooking the form's **onsubmit** event.

We also use the Convio Drupal module's **getUser** method to pre-populate the form with information from the constituent's profile if a constituent logged in.

Though there are probably better alternatives, to keep our demo simple we simply copied code below into the body a Drupal page, In order for the page source to be interpreted rather than simply displaying the code, we had to change a couple Drupal settings:

1. From the Drupal admin menu, select **Administer->Site building->Modules**
2. Under **Core – optional**, enable **PHP filter**. This allows embedded PHP code/snippets to be evaluated.

When you create the page, copy the Action Alert Form code into the body and set the **Input format** to **PHP Code**.

ACTION ALERT FORM CODE

```
<?php
  $user = ConvioLoginHandler::getInstance()->getUser();
?>

<style type="text/css">
#cnvActionAlert{
  width:520px;
}
#cnvActionAlert h1{
  font-family:Tahoma,Arial,Helvetica,sans-serif;
  font-size:1.9em;
  font-weight:normal;
  color:#555555;
  margin-top:0.5em;
}
#cnvActionAlert-description p{
  font-family:Arial,Helvetica,sans-serif;
  font-size:12px;
}
#recipients{
  font:normal 14px Tahoma, Arial, Helvetica, sans-serif;
}
#recipients ul li{
  padding-left:18px;
  background:url({secure hostname}/images/bullet_grey.gif) left 0.3em no-repeat;
  font:12px Arial, Helvetica, sans-serif;
}
#cnvActionAlert-formContainer, #cnvActionAlert-formContainer form, #cnvActionAlert-formContainer
p, #cnvActionAlert-formContainer label, #cnvActionAlert-formContainer input, #cnvActionAlert-
formContainer li{
  color:#444;
  font:normal 12px Arial, Helvetica, sans-serif;
  line-height:normal;
}
#cnvActionAlert-formContainer ul{
  margin:8px 0 0 10px;
```

```
  padding:0 0 0 15px;
  list-style:none;
}
#cnvActionAlert-formContainer ul li{
  padding-left:18px;
  background:url({secure hostname}/images/bullet_grey.gif) left 0.3em no-repeat;
  font:12px Arial, Helvetica, sans-serif;
}
#cnvActionAlert-formContainer h2{
  margin:0;
  padding:4px 12px;
  background:#555555;
  color:#ffffff;
  text-transform:uppercase;
  font-size:13px;
}
#cnvActionAlert-formContainer .label-field-wrap{
  clear:both;
}
#cnvActionAlert-formContainer span.req{
  float:left;
  display:block;
  width:0.9em;
  color:#f00;
  text-indent:-5000px;
}
#cnvActionAlert-formContainer span.req.true{
    background:url({secure hostname}/images/required.gif) no-repeat left 4px;
}
#cnvActionAlert-formContainer label{
  overflow:hidden;
  white-space:normal;
  display:block;
}
#cnvActionAlert-formContainer .field-wrap{
  margin-left:11px;
  padding-bottom:8px;
}
#cnvActionAlert-formContainer input, #cnvActionAlert-formContainer textarea, #cnvActionAlert-
formContainer select{
  width:100%;
  border:1px solid #999;
  color:#444;
  padding:5px;
  line-height:normal;
  margin:2px;
}
#cnvActionAlert-formContainer select{
  padding:5px;
  line-height:1.5;
}
#cnvActionAlert-formContainer fieldset{
  margin-bottom:15px;
  background:#ededed;
  border:1px solid #cccccc;
}

/*left column*/
#cnvActionAlert-formContainer #column1{
  float:left;
  width:52%;
}
#cnvActionAlert-formContainer #message{
  margin-bottom:15px;
  padding:10px 10px 0 5px;
  border:1px solid #ccc;
}
#cnvActionAlert-formContainer #message h2{
  margin:-10px -10px 3px -5px;
}
#cnvActionAlert-formContainer #message p{
```

```
  clear:both;
  margin:0;
  padding:0 0 10px 5px;
}
#subject{
  width:93% !important;
}
#cnvActionAlert-formContainer #message #body{
  font:13px Arial, Helvetica, sans-serif;
  width:93% !important;
  height:428px !important;
}
/*right column*/
#cnvActionAlert-formContainer #column2{
  float:right;
  width:44%;
}
#cnvActionAlert-formContainer #info{
  padding:0 !important;
}
#cnvActionAlert-formContainer .req-legend{
  padding:5px;
  color:#c66;
}
div#cnvActionAlert-formContainer #info .label-field-wrap{
  padding-left:5px;
  padding-right:6px;
}
#column2 select{
  width:96% !important;
}
#column2 input{
  width:90% !important;
}
#cnvActionAlert-formContainer .right-check{
  margin:0 0 10px 10px;
  clear:both;
}
#cnvActionAlert-formContainer div.right-check label{
  float:none;
  display:block;
  margin:0 5px 5px 20px;
  padding:0;
  text-align:left;
  width:auto;
  font-size:11px;
}
#cnvActionAlert-formContainer .right-check input{
  float:left;
  margin-top:1px;
  *margin-top:-2px;
  margin-bottom:5px;
}
#cnvActionAlert-formContainer .right-check input{
  width:auto !important;
  border:none;
}
/*submit buttons*/
#cnvActionAlert-formContainer .submit-wrap{
  clear:both;
  padding-bottom:10px;
  text-align:right;
}
#cnvActionAlert-formContainer .submit-wrap input{
  width:auto;
  padding:0.15em 0.6em;
  border:1px solid #cdcdcd;
  background:#e8e8e8;
  color:#444;
  font-size:13px !important;
  font-family:Arial, Helvetica, sans-serif;
```

```
  line-height:1.3;
}
/*errors*/
.ErrorMessage{
  font-family:Arial,Helvetica,sans-serif;
  font-size:12px;
  color:#ff0000;
  font-weight:bold;
}
span.ErrorMessage{
  padding-left:14px;
}
#cnvActionAlert-errors p{
  text-align:center;
}
/*success message*/
#cnvActionAlert-success p{
  font-family:Arial,Helvetica,sans-serif;
  font-size:12px;
}
#cnvActionAlert-success ul li{
  padding-left:18px;
  background:url(https//:demo2-secure.convio.net/ptnr9/images/bullet_grey.gif) left 0.3em no-
repeat;
  font:12px Arial, Helvetica, sans-serif;
}
#cnvActionAlert-share{
  margin-top:1em;
}
</style>

<script type="text/javascript" src="http://code.jquery.com/jquery-1.4.2.min.js"></script>
<script type="text/javascript"
src="http://dev.jquery.com/view/trunk/plugins/validate/jquery.validate.js"></script>
<script type="text/javascript">
function submitActionAlert(){
        //Validate the form before submitting
        if ($("form#cnvActionAlert-form").valid() == false)
           return false;
       return true;
}
</script>
<div id="cnvActionAlert">
  <img alt="" src="/drupal/themes/garland/images/flag.jpg" width="520" />
  <div id="cnvActionAlert-description">
    <!-- h1>Emus in Peril</h1 -->
    <div id="cnvActionAlert-errors"></div>
    <p>Tell your representatives in Washingon that the time has come to support the aspirations
of Emus in America and protect them from various indignities.</p>
    <div id="recipients">
      <p><strong>Send this message to:</strong></p>
      <ul>
        <li>Your Representitive</li>
        <li>Your Senators</li>
      </ul>
    </div>
  </div>
  <div id="cnvActionAlert-formContainer">
    <form id="cnvActionAlert-form" method="post" action="https://demo2-
secure.convio.net/ptnr9/site/CRAdvocacyAPI" onsubmit="submitActionAlert();">
      <input type="hidden" id="method" name="method" value="takeAction" />
      <input type="hidden" id="v" name="v" value="1.0" />
      <input type="hidden" id="alert_type" name="alert_type" value="action" />
      <input type="hidden" id="api_key" name="api_key" value="api_key" />
      <input type="hidden" id="redirect" name="redirect"
value="http://emuadvocate.info/drupal/node/2" />
      <input type="hidden" id="alert_id" name="alert_id" value="154" />
      <div class="submit-wrap">
        <input type="submit" value="Send Message" />
      </div>
      <div id="column1">
```

```
        <fieldset id="message">
          <h2>Message</h2>
          <input type="text" class="required" id="subject" name="subject" value="Support Pro-Emu
Legislation" />
          <p>Dear Senator,</p>
          <textarea class="required" id="body" name="body" rows="10" cols="45">Though a bit
peculiar, emus are sentient, inoffensive and occasionally amiable creatures who deserve our
protection. I urge you to support legislation protecting emus from the various indignities to
which emus may be subject.</textarea>
          <p>Sincerely,<br/>
          <em>[Your Name]</em><br />
          <em>[Your Address]</em><br />
          <em>[City, State ZIP]</em></p>
        </fieldset>
      </div>
      <div id="column2">
        <fieldset id="info">
          <h2>Your Information</h2>
          <label for="title">Title:</label>
          <select name="title" id="title">
            <option value="<?php if ($user != null) echo $user->getProperty("name.title");
 ?>"></option>
            <option value="Mr.">Mr.</option>
            <option value="Ms.">Ms.</option>
            <option value="Mrs.">Mrs.</option>
            <option value="Miss">Miss</option>
            <option value="Dr.">Dr.</option>
          </select>
          <label for="first_name">First Name:</label>
          <input type="text" class="required" id="first_name" name="first_name" value="<?php if
($user != null) echo $user->getProperty("name.first"); ?>" />
          <label for="last_name">Last Name:</label>
          <input type="text" class="required" id="last_name" name="last_name" value="<?php if
($user != null) echo $user->getProperty("name.last"); ?>" />
          <label for="email">Email:</label>
          <input type="text" class="required email" id="email" name="email" value="<?php if
($user != null) echo $user->getProperty("email.primary_address"); ?>" />
          <label for="street1">Street 1:</label>
          <input type="text" class="required" id="street1" name="street1" value="<?php if ($user
!= null) echo $user->getProperty("primary_address.street1"); ?>" />
          <label for="street2">Street 2:</label>
          <input type="text" id="street2" name="street2" value="<?php if ($user != null) echo
$user->getProperty("primary_address.street2"); ?>" />
          <label for="city">City:</label>
          <input type="text" class="required" id="city" name="city" value="<?php if ($user !=
null) echo $user->getProperty("primary_address.city"); ?>" />
          <label for="state">State:</label>
          <select class="required" name="state" id="state">
<?php
  $state="Choose a State";
  if ($user != null && $user->getProperty("primary_address.state") !="")
     $state=$user->getProperty("primary_address.state"); ?>
            <option value="<?php echo $state ?>"><?php echo $state ?></option>
            <optgroup label="United States">
              <option value="AK">AK</option>
              <option value="AL">AL</option>
              <option value="AR">AR</option>
              <option value="AZ">AZ</option>
              <option value="CA">CA</option>
              <option value="CO">CO</option>
              <option value="CT">CT</option>
              <option value="DC">DC</option>
              <option value="DE">DE</option>
              <option value="FL">FL</option>
              <option value="GA">GA</option>
              <option value="HI">HI</option>
              <option value="IA">IA</option>
              <option value="ID">ID</option>
              <option value="IL">IL</option>
              <option value="IN">IN</option>
              <option value="KS">KS</option>
```

```
              <option value="KY">KY</option>
              <option value="LA">LA</option>
              <option value="MA">MA</option>
              <option value="MD">MD</option>
              <option value="ME">ME</option>
              <option value="MI">MI</option>
              <option value="MN">MN</option>
              <option value="MO">MO</option>
              <option value="MS">MS</option>
              <option value="MT">MT</option>
              <option value="NC">NC</option>
              <option value="ND">ND</option>
              <option value="NE">NE</option>
              <option value="NH">NH</option>
              <option value="NJ">NJ</option>
              <option value="NM">NM</option>
              <option value="NV">NV</option>
              <option value="NY">NY</option>
              <option value="OH">OH</option>
              <option value="OK">OK</option>
              <option value="OR">OR</option>
              <option value="PA">PA</option>
              <option value="RI">RI</option>
              <option value="SC">SC</option>
              <option value="SD">SD</option>
              <option value="TN">TN</option>
              <option value="TX">TX</option>
              <option value="UT">UT</option>
              <option value="VA">VA</option>
              <option value="VT">VT</option>
              <option value="WA">WA</option>
              <option value="WI">WI</option>
              <option value="WV">WV</option>
              <option value="WY">WY</option>
            </optgroup>
            <optgroup label="US Territories">
              <option value="AS">AS</option>
              <option value="FM">FM</option>
              <option value="GU">GU</option>
              <option value="MH">MH</option>
              <option value="MP">MP</option>
              <option value="PR">PR</option>
              <option value="PW">PW</option>
              <option value="VI">VI</option>
            </optgroup>
          </select>
          <label for="zip">ZIP / Postal Code:</label>
          <input type="text" class="required"  minlength="5" id="zip" name="zip" value="<?php if
($user != null) echo $user->getProperty("primary_address.zip");
 ?>" />
          <label for="phone">Phone Number:</label>
          <input type="text" id="phone" name="phone" />
          <input type="checkbox" id="optin" class="checkbox" name="optin" value="true"
checked="checked" />
          <label for="optin">Yes, I would like to receive periodic updates.</label>
          <!-- input type="hidden" name="preview" value="true" / -->
          <input type="hidden" name="response_format" value="xml" />
        </fieldset>
      </div>
      <input type="hidden" id="source" name="source" value="emuadvocate.info" />
      <input type="hidden" name="sub_source" id="sub_source" value="unknown" ?>
      <div class="submit-wrap">
        <input type="submit" value="Send Message" />
      </div>
    </form>
  </div>
  <div id="cnvActionAlert-success"></div>
</div>
```

FOR ADDITIONAL INFORMATION

For additional information on provisioning Bitnami Drupal images on Amazon EC2 refer to: bitnami.org/tutorials/amazon_machine_images (video: www.filearchive.net/tpn/TPN-2009-08-29/TPN-2009-08-29.html)

For technical reference documentation, examples and downloads for developers using Convio Open APIs, refer to open.convio.com.

Downloads of useful remote console tools are available on the web. **WinSCP** is available from winscp.net/eng/index.php **Putty** and **PuTTYGen** (used to convert Amazon PEM keys to PPK) are available from www.chiark.greenend.org.uk/~sgtatham/putty/download.html