Partner Design & Development : Ning Apps OpenSocial Documentation

This page last changed on Dec 23, 2009 by ernie.

Table of Contents

- 1 Table of Contents
- 2 Introduction
- 2.1 About OpenSocial
- 2.2 OpenSocial on Ning
- 2.2.1 What Ning Apps CAN do
- 2.2.2 What Ning Apps can NOT do at this time
- 2.2.3 Porting from other Social Network Containers to Ning
- 2.2.3.1 Submitting your Ported Application to the Ning App Directory
- 2.3 Feedback
- 3 Revision Log
- 4 Developer Guide to Ning Apps
- 4.1 Ning Apps Terminology
- 4.2 Anatomy of a Ning App
- 4.2.1 Canvas View
- 4.2.2 Profile View
- 4.2.3 About View
- 4.2.4 Latest Activity module
- 4.2.5 The Ning App Directory
- 4.3 OpenSocial Examples
- 4.3.1 Hello, World!
- 4.3.2 Hello, World! in Different Views
- 4.3.3 Sub-navigation in the Canvas View
- 4.3.4 Navigating Between Views and Passing Parameter Information
- 4.3.5 Displaying Profile Information
- 4.3.6 Displaying Friends' Information
- 4.3.7 Saving and Retrieving Persistent Data
- 4.3.8 Sending Messages Using requestSendMessage
- 4.3.9 Reading from and Writing to a Member's Latest Activity on their Profile Page via the Activity Stream
- 4.3.10 Accessing an External REST API via JSON
- 4.3.11 Accessing the OpenSocial RESTful/RPC Protocol
- 4.3.12 Ning Specific Methods
- 4.3.12.1 feature="ning"
- 4.3.12.2 Getting Person-Specific Information with ning.Person.Field
- 4.3.12.3 Getting Ning Container-Specific information with ning.core.AppStatus
- 4.3.12.4 Getting Network Information with ning.app.getInfo
- 4.3.12.5 Getting the Domain Prefixs and Unique Identifiers for a Social Network
- 4.3.12.6 ning.core.getContainerVersion
- 4.4 Validating Signed Requests Through OAuth
- 4.4.1 Sounds Great. How Do We Implement This in our App?
- 4.5 Policies
- 4.6 Testing your Ning App
- 4.6.1 Adding your Application
- 4.6.1.1 Adding an Application by URL
- 4.6.1.2 A Note on Caching
- 4.6.1.3 Removing an Application
- 4.6.2 Canvas View
- 4.6.3 About View
- 4.7 Submitting your Application to the Ning App Directory
- 5 Best Practices
- 5.1 What You MUST Do
- 5.2 What You SHOULD Do
- 5.3 What NOT to do
- 5.4 Support Resources and References
- 5.5 Ning API Specific Resources
- 5.6 Policy Based Links

Introduction

Welcome to Ning Apps! This document is designed to help developers get their OpenSocial applications up and running on the Ning platform as quickly and easily as possible.

About OpenSocial

The OpenSocial initiative was spearheaded by Google in the fall of 2007, and has since been adopted by a wide range of social media companies. It enables third-party applications to access member data, process friend relationships and perform network actions (such as activity feed updates) normally accessible only to a service's core features.

OpenSocial on Ning

With over a million Ning Networks created on our platform to date, OpenSocial applications on Ning have access to an expansive and ever-growing audience.

We're supporting a subset of OpenSocial 0.8.1 APIs and core Google Gadgets, with limited support for the legacy (iGoogle) APIs - you can find the list at our API page. If you're new to OpenSocial and would like more information about developing JavaScript applications using Google's Gadget API, click here -- CodeRunner is also a fantastic resource for beginner developers, and can be used in conjunction with the many tutorials we have in this document.

What Ning Apps CAN do

- **Build a functional application with different views:** Ning Apps lets you build a functional application that let's you integrate with different parts of the Ning Network including profile pages, as well as a dedicated separate page available from the main navigation.
- **Access & Integrate your third-party information with the members of a Ning Network:** If you have a database/content storage solution that is accessible via REST APIs, you can create a user interface that is directly accessible from all Ning Networks.
- **Get limited information about a Ning Network into your application using Javascript:** Ning Apps are based on OpenSocial standards which include methods to extract profile information.

What Ning Apps can NOT do at this time

- **Retrieve member data from custom profile questions or features such as photos, videos, forums, etc.:** This type of information falls outside the scope of the OpenSocial API. The Engineering team is currently looking into ways of expanding the OpenSocial specification by adding methods to access all member content on your Ning Network.
- **Integrate with existing features:** There is currently no filter or plug-in system for Ning Apps that would allow you to extend or enhance the built in features.
- **Let you query member or network information directly from your server:** Since this form of remote access is part of the OpenSocial v0.8 specification, we plan to provide OpenSocial REST APIs in a subsequent release.

Porting from other Social Network Containers to Ning

Ning is different from other social network containers in that members can create their own Ning Networks that other members can join; as a result, members can freely access pages of a network outside of member profile page, and the "Main View" has been created to reflect this; the object variable "ning" has been created to get specific metadata related to a Ning Network (See "Ning Specific Methods.") For maximum portibility, the OWNER object is the Network Creator.

If you've previously created a Profile App on Ning or an OpenSocial Application on another social network container, here's a handy chart outlining the differences between Profile and Ning Apps:

	Profile Apps	Ning Apps
Primary View	Profile	Canvas

Supplementary Views Supported	Canvas (1), About View	Profile (Optional, Defaulted Off), About View
Who adds an application	A member of a Ning Network	Network Creator of a Ning Network
Canvas view contains tabs in the navigation of Ning Network?	No	Yes
Activity Streams via	Writes to the "Recent Activity" module on a member's specific profile page	Writes to the "Recent Activity" module of the main page of a Ning Network
Alert Messaging via requestSendMessage	Has the ability to message all friends of a member of a Ning Network	Has the ability to message all members of a Ning Network via Alerts, the in-network messaging system. (Members of a network have the ability to get Alert notifications via e-mail.)

Submitting your Ported Application to the Ning App Directory

Ning allows networks to be created about anything and we realize that some subject matter that is acceptable to one group can be offensive to other people. We are currently in the process of building more controls for Network Creators to allow particular applications available for their members; until then, all applications must be reviewed by the Ning team to make sure the content is appropriate for all members.

Before you submit your application, please make sure that you have verified that the OpenSocial application has been tested properly on a Ning Network. Please refer to the "Testing your Ning App" section in this documentation for more information on how to test your OpenSocial Application on a network. You may also browse the Policies, Best Practices and our guides to testing your Ning App and submitting your Application to the Ning App Directory, and you can consult the Ning Developer Network at http://developer.ning.com for latest updates.

Feedback

We're actively looking for bug reports and enhancement suggestions to help make our OpenSocial implementation as friendly to developers as possible. Feel free to send feedback through our help center or Developer Forums. Thanks!

Revision Log

Date	version	Description
December 22, 2009	v1.0.8	Added footnote about Network Creators and Network Admins being able to install Ning Apps Added details about OAuth integration Added footnote about persistant data.
October 22, 2009	v1.0.7	Added screenshots of Tab Manager integration, Profile View "My Apps" screens There is no longer a need to create a separate development sandbox - just create a Ning Network. All references fo

		developer sandboxes have been removed. Added additional notes on Ning App caching. Added notes on how to remove a Ning App.
August 17, 2009	v1.0.6	Sandboxes must now be created by contacting the Ning Help Center. Removed ning.main view, UserPrefs and EnumValues sections. Added additional explanations on changes Network Creators and Admins can make to profile views.
July 7, 2009	v1.0.5	Added section "What Ning Apps Can and Can Not Do" Further clarified the section "UserPrefs and EnumValues" -- UserPrefs are unavailable in the Profile or Canvas View. All Ning Apps titles should have a maximum length of 24 characters. Fixed a broken PNG file related to the subnavigation. Added http://os.ning.com/tutorials/ningapps/docs/app-screen-template-455.psd (PSD file that designers can use to make screenshots appropriate for submission to the Ning App Directory) Removed Weather Sharer, to be replaced with a different sample application in a later doc release.
May 26, 2009	v1.0.4	Added section "Getting Ning Container-Specific information with ning.core.AppStatus" Changed XML sample for section "Navigating Between Views and Passing Parameter Information" Added content for the section "UserPrefs and EnumValues" author_email in the <ModulePrefs> header is now accessible; the "What you MUST do" section has been edited accordingly.
May 11, 2009	v1.0.3	Extended "Getting the Domain Prefix with ning.core.getNetworkDomain and a Unique Identifier with ning.app.field.ID"
May 8, 2009	v1.0.2	URL to create developer sandboxes released

May 6, 2009	v.1.0.1	Official document released.

Developer Guide to Ning Apps

Ning Apps Terminology

This document will use the following terms to describe components included in the Ning OpenSocial implementation.

Term	Description
Network	Ning allows anybody to create their own social network for anything, called a Ning Network. Each of these networks can be an OpenSocial container.
Ning Apps	Third party code that uses the Gadgets and OpenSocial APIs which are served on the page of a Ning Network available from the tab navigation. Formerly called Network Applications.
Profile Apps	Third party code that uses the Gadgets and OpenSocial APIs which are served on the profile page of a Ning Network. Formerly called Profile Applications.
Profile View	Application when displayed as a module on Ning member profile pages.
Canvas View	Full page view of an Application.
About View	Full page view of an Application's information, including description and screenshot.
Activity stream	Data feed that contains activities from one person.
Latest Activity module	The Latest Activity module displays the activity stream of a given user on his/her profile page.
Ning App Directory	The directory where Network creators can add a Ning App to their Ning Network, and will be accessible via the Manage menu.
Profile App Directory	The directory where members of a Ning Network can add a Profile Application to their profile page. This was previously known as the Application Directory.

Anatomy of a Ning App

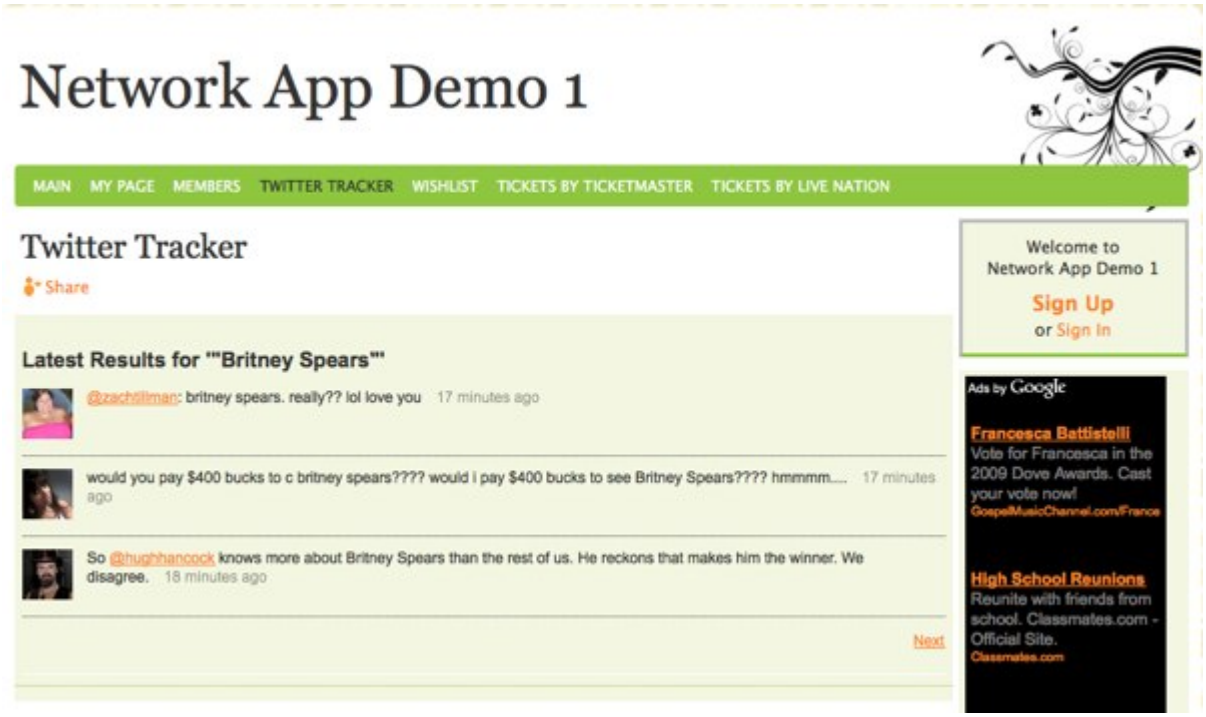
An OpenSocial Application on Ning is comprised of the following parts, which combine to form a feature to complement any of the hundreds of thousands of Ning Networks. As an Application Developer, here is a checklist of components which should be utilized, as well as an overview of the relationship between these components:

1. Profile View
2. Canvas View
3. About View
4. Latest Activity
5. The Ning App Directory

Canvas View

The Canvas View is the full-page view of an application. This view is required. Note that the Canvas view differs from the Canvas view in a Ning OpenSocial Profile Application in the following additional ways:

- The Ning App can optionally have its own tab in the top-level navigation. Clicking on that tab displays this Canvas View.
- Additionally, the application can have multiple canvas pages that can operate as closely as possible to the existing features of a Ning Network. Applications can specify sub-navigation links that live outside the OpenSocial container that allow for navigation between the different canvas pages.



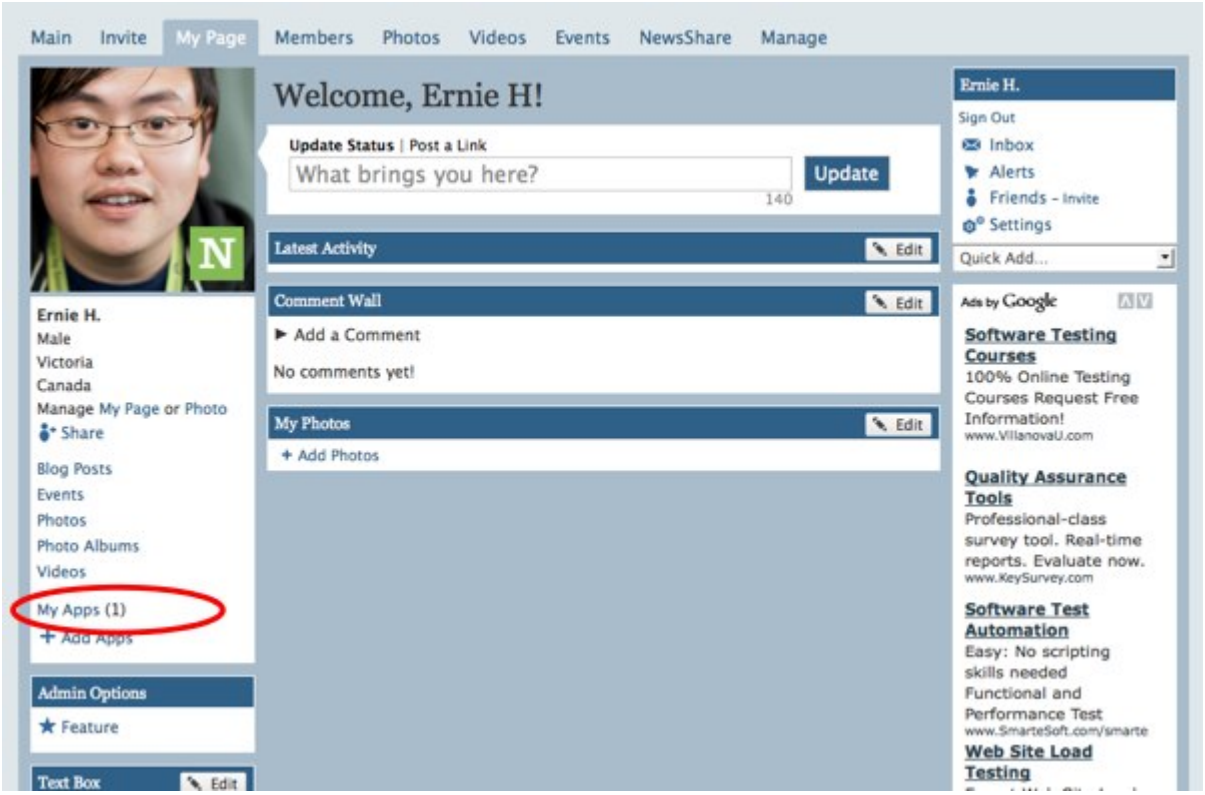
Specifically to Ning, Ning Apps have the ability to include a secondary navigation structure through <os:navigation> structure, which can emulate multiple canvas sub-pages through a parameter structure:

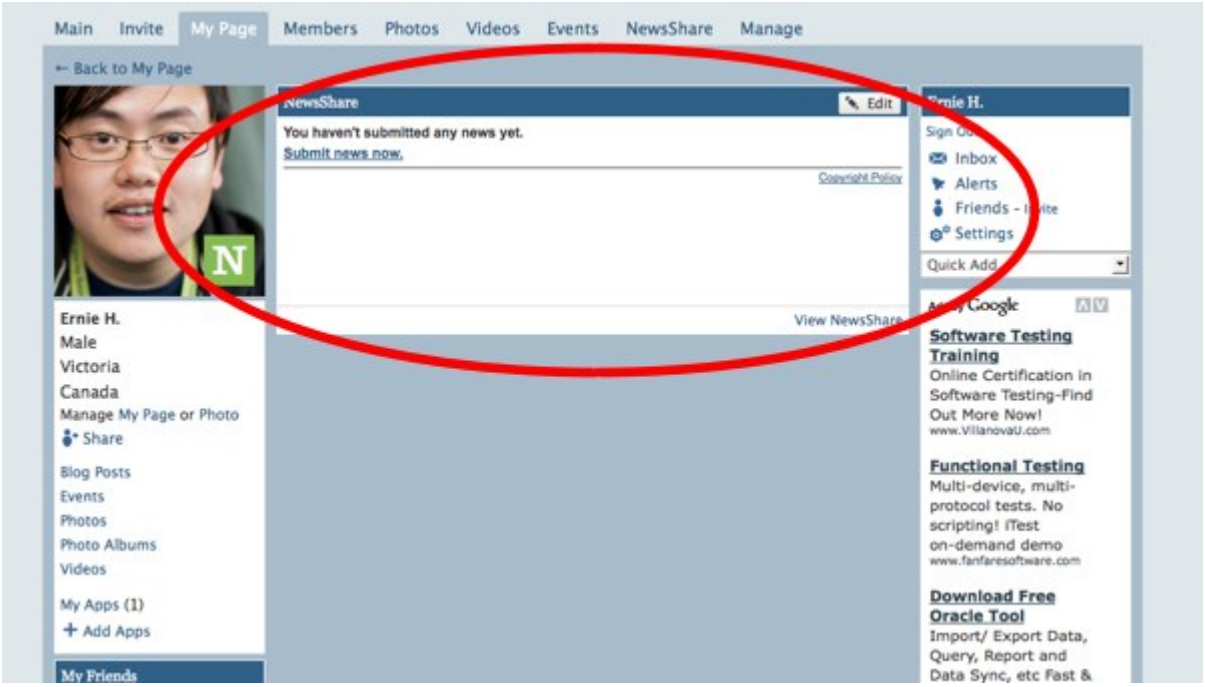
Here are the recommended dimensions:

Width: 737px (Any feature less than this width will be centered in the column)
Maximum height: There is no height restriction.

Profile View

The Profile View of an application gives the ability to add a profile-specific module in the profile page of every member in a Ning Network. Installing a Ning App with a Profile View will automatically add the Profile View component to the profile pages of all members on a Ning Network. This view is optional.

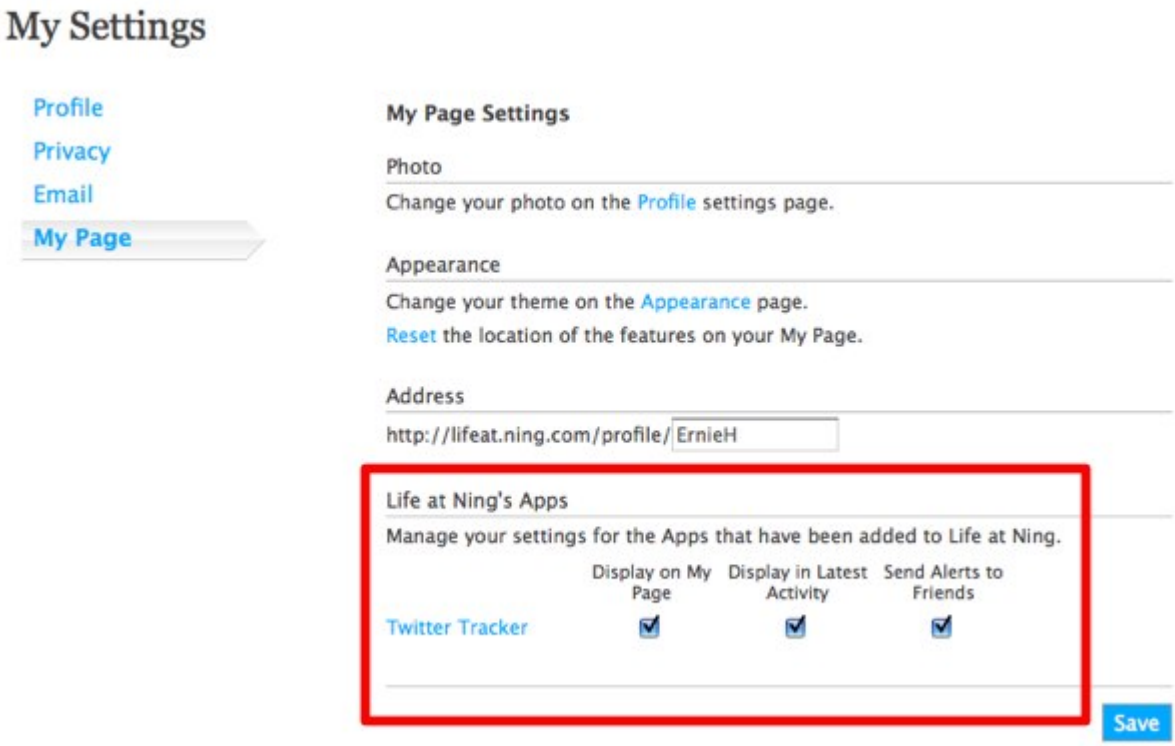




Note that members of a Ning Network can configure settings for their own Profile View of a particular Ning App by clicking on the "Edit" button in the upper-right hand corner, or the "Settings" link above the "Quick Add" toolbar in the upper-right hand corner. These include options to:

- Toggle updates from the Ning App to the Latest Activity section of their profile page.

- Set the ability to send alerts from their profile page to other members
- Display the Profile View module on their profile page.



TECHNICAL NOTE: The OWNER and VIEWER objects differ between the ~~main~~ canvas view and profile views -- while the OWNER object in the ~~main~~ canvas view belongs to the Network Creator, the OWNER object in the profile view refers to the member's profile page, similar to standard profile-based OpenSocial applications. Also note that the profile view of a Network App and a OpenSocial Application based on the profile view uses different application identifiers. As a result, if an application is installed on the profile page AND on the ~~main~~ canvas page, data will not overlap.

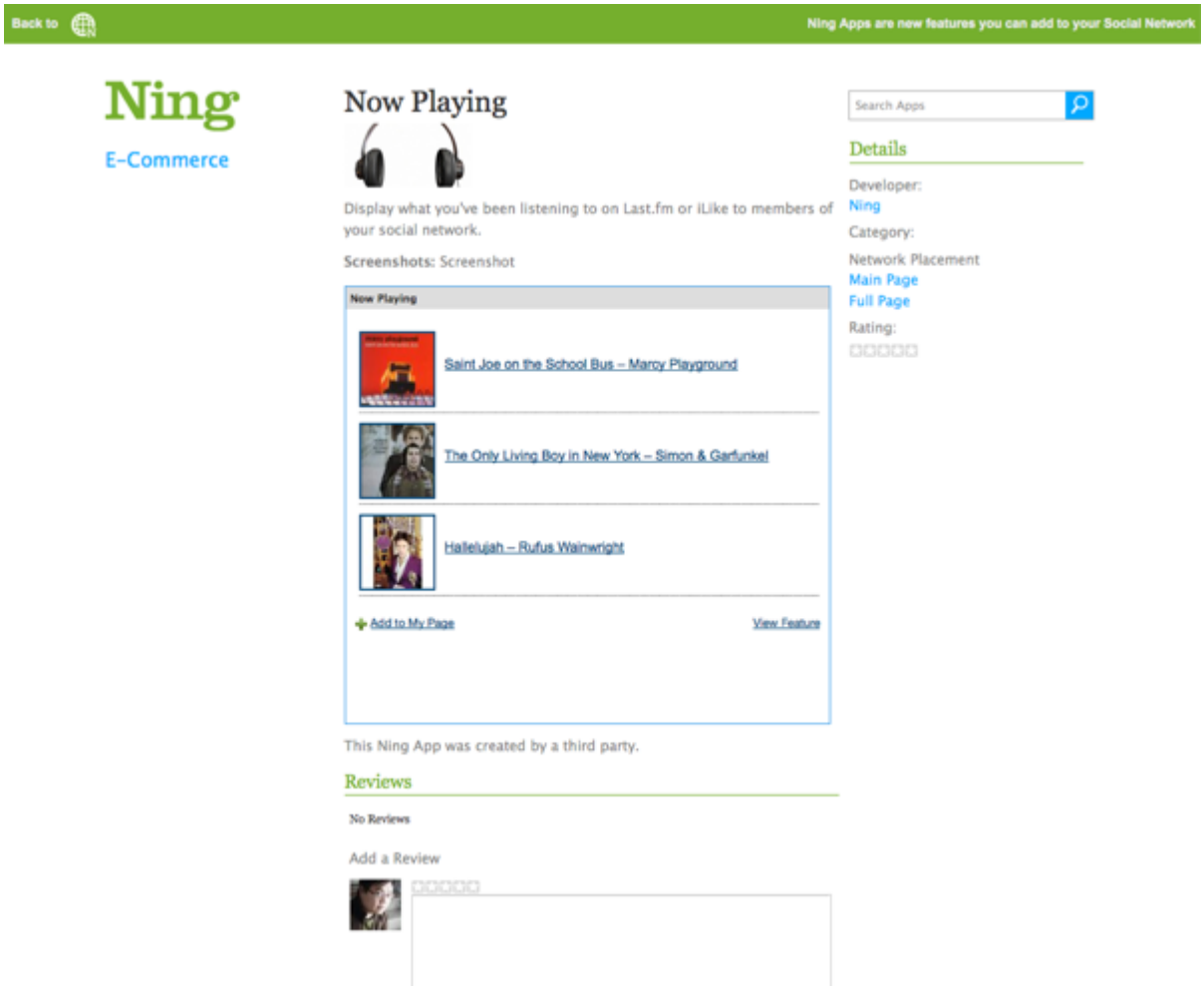
Here are the recommended dimensions:
Width: 492px (Any feature less than this width will be centered in the column)
Maximum height: There is currently no height restriction but will be in a future release.

About View

The About View is your Application's information page, and allows members of the Ning Network to view the description and screenshot of your application. In a subsequent release, we'll also include information about the developer and number of members using your application, as well as a button to add it to the viewer's profile page.

Here are the recommended dimensions:

Screenshot width: 502px
Screenshot maximum height: 1000px
Max file size: 150KB, non-transparent .gif/.jpg/.png



If a member has not added the OpenSocial Application, a member can click on the "Add to My Page" button in the upper right hand corner to add the application to their profile page:

Latest Activity module

Your Latest Activity module shows all recent activity of members on a Ning Network. As an Application Developer, you also have the ability to add updates relevant to your application in the Latest Activity module on the main page.

This component is still under development; screen shots will be rolled out at a latter date during the beta testing.

The Ning App Directory

This is where Network Creators or Network Admins of a Ning Network can add Apps to ~~the main page of~~ their Ning Networks. Contrary to Profile Apps, the Network Creator will no longer have the ability to directly add an OpenSocial Application by URL available through the user interface, but developers can add an XML file manually. (See the Testing Your Ning App section.)

OpenSocial Examples

Hello, World!

(Note: Some of the tutorials and tutorial descriptions provided are based on the OpenSocial Tutorial for Orkut. Due to the open nature of OpenSocial, you may find that a lot of the Ning OpenSocial Applications that you create can easily be ported to other OpenSocial containers such as Orkut, hi5 and MySpace with very little additional development time, and vice versa.)

OpenSocial Applications are essentially XML files that adhere to the OpenSocial specifications.

To create a quick "Hello World" application, paste the following code into the Google Gadget Editor, or an XML file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
<ModulePrefs title="Hello, world!">
<Require feature="opensocial-0.8" />
</ModulePrefs>
<Content type="html">
<![CDATA[
<script type="text/javascript">
function init() {
  document.getElementById("handler").innerHTML = "Hello, world.";
}
gadgets.util.registerOnLoadHandler(function() { init(); });
</script>

<div id="handler"></div>
]]>
</Content>
</Module>
```

You can view a live example of this on http://os.ning.com/apps/tutorials/hello-world.xml .

Once the OpenSocial Application is loaded inside a container, registerOnLoadHandler is called. In this case, the function calls the Javascript init() function which writes "Hello, world." to the "handler" HTML element.

A key element in this example is the use of <Require feature="opensocial-0.8"/> as a parameter of the module declaration. This specifies the release of opensocial required by the module. Containers may or may not implement all possible versions of the OpenSocial API, and when they don't the member will see an error. <![CDATA[PRODUCT:...]]> contains the bulk of the gadget, including all of the HTML, CSS, and JavaScript (or references to such files). The content of this section should be treated like the content of the body tag on a generic HTML page.

From there, you can simply save your gadget as a file in your network (through WebDav) in a new directory, OpenSocial, with the name for example helloworld.xml. You can also can develop your xml document off an external server. Once you have done that, you can add your XML file to the profile page of your Ning Network. Detailed instructions on how to add your OpenSocial Application are listed later in this document: Testing Your Ning App.

The process of editing an XML file, uploading it to a server and re-evaluating the code you just wrote can be a tedious process. Thankfully, Google has an OpenSocial Application called CodeRunner where you can cut and paste javascript code into a textbox, where the code is run client-side; this tool is pretty handly, especially with developers who are comfortable with Javascript and would just like to experiment with the more OpenSocial based methods. Due to the slightly different nature of Ning Apps, we have created an instance of the CodeRunner app that includes some Ning only properties. The file is available here: http://gadgetstore.ning.com/gadgets/coderunner/coderunner08.xml .

Hello, World! in Different Views

Once we understand the basic concept of the Content section and how it's used to show HTML, CSS and Javascript inside, we can expand that idea with the view attribute, used if you want to show conditional content based on if you're viewing the OpenSocial Application in the ~~Main View,~~ Canvas View or Profile View. This can be useful from a User Experience perspective, possibly showing less detailed information where the viewport is smaller.

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

```
 <ModulePrefs title="Test ning.main app">
  <Require feature="opensocial-0.8" />
<UserPref name="showAllPokes"
        display_name="Show all pokes"
        datatype="bool"
        default_value="true" />
 <UserPref name="numPokes"
        display_name="Number of pokes to display"
        datatype="enum"
        default_value="2">
  <EnumValue value="1" display_value="3"/>
  <EnumValue value="2" display_value="5"/>
  <EnumValue value="3" display_value="10"/>
 </UserPref>
 </ModulePrefs>
 <Content type="html" view="ning.main">
 <![CDATA[
  <p>Hello, ning.main view!</p>
 ]]>
 </Content>
 <Content type="html" view="canvas,profile">
 <![CDATA[
  <p>Hello, world!</p>
 ]]>
 </Content>
 <Content type="html" view="profile">
 <![CDATA[
  <p>Hello, Profile View!</p>
 ]]>
 </Content>
 <Content type="html" view="canvas">
 <![CDATA[
  <p>Hello, Canvas View!</p>
 ]]>
 </Content>
</Module>
```

In the following example, "Hello, Profile View!" and "Hello, Canvas View!" is shown is the Profile and Canvas Views, respectively. "Hello, world!" is displayed for the ~~Main,~~Canvas and Profile Views.

Sub-navigation in the Canvas View

Cannot resolve external resource into attachment.

Because of the distinct nature of Ning App, Applications can specify sub-navigation links that live outside the OpenSocial container that allow for navigation between the different canvas pages. Sub-navigation elements are specified via Ning namespaced elements in the gadget XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<Module xmlns:ning="http://developer.ning.com/opensocial/">
  <ModulePrefs title="NavLinks Test App">
    <Require feature="opensocial-0.8" />
    <Require feature="dynamic-height" />
    <Require feature="views" />
    <UserPref name="string_1" display_name="string display 1" datatype="string" required="false" />
    <ning:navigation>
      <ning:link>
        <ning:name>Tab 1</ning:name>
        <ning:URLParameter>param=tab1</ning:URLParameter>
      </ning:link>
      <ning:link>
        <ning:name>Tab 2</ning:name>
        <ning:URLParameter>param=tab2</ning:URLParameter>
```

```
        <ning:URLParameter>link22=22</ning:URLParameter>
      </ning:link>
    </ning:navigation>
</ModulePrefs>
  <Content type="html" view="ning.main">
  <![CDATA[
    <p>This is some HTML content in the Main View of a Ning App.</p>
  ]]>
  </Content>
  <Content type="html" view="canvas">
  <![CDATA[

      <!--
            Parameters are in the following format:
            <ning:URLParameter>key=value</ning:URLParameter>

            You can also set multiple URLParameter fields:
      <ning:name>Tab 2</ning:name>
      <ning:URLParameter>param=tab2</ning:URLParameter>
      <ning:URLParameter>link22=22</ning:URLParameter>
      -->

    <script type="text/javascript">
      // Un-comment the following if you are using the Firebug plug-in for Firefox.
      // console.dir(gadgets.views.getParams());
      // console.dir(gadgets.views.getSupportedViews());
       var o = gadgets.views.getParams();
            document.write("The param value is: " + o.param + " (If undefined then no parameters have been
 passed and may be the main page.)");
    </script>
    <p>This is some HTML content in the Canvas View of a Ning App!</p>

  ]]>

  </Content>
</Module>
```

Note that the canvas view will returned "undefined," as no parameters were passed to the canvas page.

Navigating Between Views and Passing Parameter Information

The requestNavigateTo method is used to navigate between different application views. You can pass parameter data between these views by passing appropriate data as a parameter in the previously mentioned method. The following sample code illustrates one such example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs title="View Test">
    <Require feature="opensocial-0.8" />
      <Require feature="views" />
    <Require feature="dynamic-height"/>
  </ModulePrefs>
  <Content type="html" view="ning.main,canvas,profile">

  <![CDATA[
    <h1>Current view CDATA</h1>
  ]]>

  </Content>
  <Content type="html" view="ning.main">

  <![CDATA[
```

```
  <p>ning.main</p>
]]>
</Content>
<Content type="html" view="canvas">

<![CDATA[
   <p>canvas</p>
]]>
</Content>

<Content type="html" view="profile">

<![CDATA[
   <p>profile</p>
]]>
</Content>

<Content type="html" view="ning.main,canvas,profile">

<![CDATA[
<h1>View navigation</h1>
<div id="view_nav"></div>

<h1>Current view</h1>
<div id="curr_view"></div>

<h1>Available views</h1>
<div id="view_list"></div>

<script type="text/javascript">
   function test_curr_view() {
      var curr_view_html = document.getElementById('curr_view');
      var curr_view_ref = gadgets.views.getCurrentView();
      curr_view_html.innerHTML = "Current view: " + curr_view_ref.getName();

   };

 function test_supported_views() {
    var view_list_html = document.getElementById('view_list');
    var views = gadgets.views.getSupportedViews();
    var html_out = "<ol>";

    for(var key in views) {
       html_out += "<li>" + key + "<ul>";
       html_out += "<li><b>getName()</b>: " + views[key].getName() + "</li>";
       html_out += "<li><b>getUrlTemplate()</b>: " + views[key].getUrlTemplate() + "</li>";
       html_out += "</ul></li>";
    }

    html_out += "</ol>";
    view_list_html.innerHTML = html_out;
 };

 function test_view_nav() {
    var view_nav_html = document.getElementById('view_nav');
    var views = gadgets.views.getSupportedViews();

    html_out = "<select id='view_select'>"
    for (var view_name in views) {
       html_out += "<option value='" + view_name + "'>" + view_name + "</option>";
    }
    html_out += "</select>";

    html_out += "<ul><li>Value One: <input type='text' id='txtParamOne' /></li>";
    html_out += "<li>Value Two: <input type='text' id='txtParamTwo' /></li></ul>";
    html_out += "<input type='button' onClick='navigate_to_view()' value='Go!' />"
```

```
    view_nav_html.innerHTML = html_out;
  };

    function test_params() {
          var params1 = gadgets.views.getParams();
    // Un-comment the following if you are using the Firebug plug-in for Firefox.
     console.log("entering params:")
    console.dir(gadgets.views.getParams());

          if ('undefined' === typeof(params1) || 'undefined' === typeof(params1['param1'])){
                return;
          }
          document.getElementById('txtParamOne').value = params1['param1'];
          document.getElementById('txtParamTwo').value = params1['param2'];
    }

  function navigate_to_view () {
    var dest_view = document.getElementById('view_select').value;
    var views = gadgets.views.getSupportedViews();
    var dest_view_ref = views[dest_view];

    var params = {};
    params['param1'] = document.getElementById('txtParamOne').value;
    params['param2'] = document.getElementById('txtParamTwo').value;
    gadgets.views.requestNavigateTo(dest_view_ref, params);
  };


  test_curr_view();
  test_supported_views();
  test_view_nav();
  test_params();

  gadgets.window.adjustHeight();
 </script>
 ]]>
 </Content>
</Module>
```

For those needing to retrieve the parameters passed from the query string, the view-params object takes parameters in JSON format to be passed into applications. This is similar to the way that Orkut passes query string parameters via gadgets.util.getUrlParameters() in this URL: http://code.google.com/apis/orkut/docs/orkutdevguide.html. It is also possible to manually parse the parameters using location.href and using a string matching function or a regular expression.

Displaying Profile Information

Now you can begin creating the actual body of code that uses the OpenSocial API to display some useful information, such as the viewer's name and profile photo.

You can fetch this data by creating an opensocial.newDataRequest object, populating it with a request to fetch the viewer (that's you), and sending the request to the server.

This example uses the key opensocial.IdSpec.PersonId.VIEWER, or the enum value "VIEWER" for short - but you can also use the opensocial.IdSpec.PersonId.OWNER key. If Alice is viewing Bob's profile, then Alice is the viewer and Bob is the owner. You can access information about both users by specifying the appropriate key.

To see this in action, replace the CDATA content in your gadget with the following JavaScript:

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

```
<ModulePrefs title="Hello, world!">
<Require feature="opensocial-0.8" />
<Require feature="dynamic-height" />
</ModulePrefs>
<Content type="html">
<![CDATA[

<script type="text/javascript">
function response(data) {
  var viewer = data.get("viewer").getData();
  var name = viewer.getDisplayName();
  var thumb = viewer.getField(opensocial.Person.Field.THUMBNAIL_URL);

  var html = '<img src="' + thumb + '"/>' + name;
  document.getElementById('dom_handle').innerHTML = html;
};

function request() {
  var req = opensocial.newDataRequest();
  // req.add(req.newFetchPersonRequest('VIEWER'), 'viewer');
  var viewer = opensocial.IdSpec.PersonId.VIEWER;
  req.add(req.newFetchPersonRequest(viewer), 'viewer');
  req.send(response);
};

request();
</script>
<div id="dom_handle"></div>
]]>
</Content>
</Module>
```

Note that the send method within the request is asynchronous, and it takes a callback function - in our example, response - as a parameter. This callback function will be invoked when the response comes back from the server.

Displaying Friends' Information

This next portion of the tutorial builds on the previous, but in addition to showing your own information (as the viewer), it will describe how to fetch all of your friends and display their information within the gadget.

First you need to modify your request to ask for the data about the viewer and the viewer's friends. You can batch this request for two data sets by modifying the request function like this:

```
function request() {
  var params = { };
  // params[opensocial.DataRequest.PeopleRequestFields.MAX] = 50;
  // params[opensocial.DataRequest.PeopleRequestFields.FILTER] =
 opensocial.DataRequest.FilterType.HAS_APP;
  // params[opensocial.DataRequest.PeopleRequestFields.SORT_ORDER] =
 opensocial.DataRequest.SortOrder.NAME;

  var idspec = opensocial.newIdSpec({ "userId" : "VIEWER", "groupId" : "FRIENDS" });
  var req = opensocial.newDataRequest();
  req.add(req.newFetchPersonRequest("VIEWER"), "viewer");
  req.add(req.newFetchPeopleRequest(idspec, params), "viewer_friends");
  req.send(response);
}
```

Note that you are adding a newFetchPeopleRequest, which will return a collection of many people, as opposed to a newFetchPersonRequest, which only returns one person. New to OpenSocial v0.8 is the IdSpec, which takes a Javascript object passing information on the type of information you'd like, whether it be the OWNER or VIEWER of the application, as well as the collection of friends for the previously mentioned objects. Check out http://code.google.com/apis/opensocial/articles/datarequests/datarequests-0.8.html for additional information about this.

You can also add different parameters to the opt_params[] variable using constants from opensocial.DataRequest.PeopleRequestFields; adding the following line, for example, filters the set of VIEWER_FRIENDS to only include friends that have also added the same OpenSocial Application to your networks:

params[PRODUCT:opensocial.DataRequest.PeopleRequestFields.FILTER] = opensocial.DataRequest.FilterType.HAS_APP;

Now you can add code to the response method to print out your friends' profile photos and links to their profile pages.

```
function response(data) {
  var viewer = data.get("viewer").getData();
  var name = viewer.getDisplayName();
  var thumb = viewer.getField(opensocial.Person.Field.THUMBNAIL_URL);
  var profile = viewer.getField(opensocial.Person.Field.PROFILE_URL);

  var html = '<img src="' + thumb + '"/>' +
          '<a href="' + profile + '" target="_top">' + name + '</a>';

  html += '<hr>';

  var viewer_friends = data.get("viewer_friends").getData();
  viewer_friends.each(function(person) {
    var thumb = person.getField(opensocial.Person.Field.THUMBNAIL_URL);
    var profile = person.getField(opensocial.Person.Field.PROFILE_URL);
    html += '<a href="' + profile + '" target="_top" style="float:left">' +
          '<img src="' + thumb + '"/>' +
          '</a>';
  });

  document.getElementById('dom_handle').innerHTML = html;
  // This function changes the height of the module.  Make sure you add
  // <Require feature="dynamic-height"> to your XML component.
  gadgets.window.adjustHeight();

}
```

The object that contains your friends' data is a collection of Person objects. The above example uses the viewer_friends.each method to iterate through every Person in the collection.

Now your Application will show all your friends smiling back at you!

Saving and Retrieving Persistent Data

Most OpenSocial calls involve building a DataRequest object, sending it to the container, and processing the response. Reading and writing Application data is included in this flow. That said, an application can only
write to the data of the OWNER object. While this policy is fairly restrictive, it is to prevent malicious users from writing data to arbitrary users. With most OpenSocial-based applications such as Ning Apps, the best practice is to use a local persistent storage mechanism such as a SQL database and use persistent data for caching purposes.

To write a value to a specific key, create a DataRequest and add the result of calling newUpdatePersonAppDataRequest to it:

```
var req = opensocial.newDataRequest();
req.add(req.newUpdatePersonAppDataRequest("VIEWER", "myKey", "myValue"), "set_data");
req.send(set_callback);
```

The previous piece of code attempts to save the string myValue under the key of myKey, and the result is stored in set_data. It also specifies a callback function named set_callback that will get a DataResponse object indicating whether the update failed or succeeded. Note that as of this release, Ning allows **a maximum of 10k of serialized data stored**, including keys, values and serialization overhead.

A function to handle this response could look something like this:

```
function set_callback(response) {
  if (response.get("set_data").hadError()) {
    /* The update failed ... insert code to handle the error */
  } else {
    /* The update was successful ... continue execution */
  }
};
```

If you would like more information about making DataRequest requests, please check out the OpenSocial API Developers' Guide, http://code.google.com/apis/opensocial/articles/persistence.html or the OpenSocial Data Requests reference at http://code.google.com/apis/opensocial/articles/datarequests/datarequests-0.8.html for more information. To retrieve the value of myKey you must create another DataRequest. This time, include a newFetchPersonAppDataRequest result:

```
var req = opensocial.newDataRequest();
var idspec = opensocial.newIdSpec({'userId':'VIEWER', 'groupId':'SELF'});

req.add(req.newFetchPersonRequest(opensocial.IdSpec.PersonId.VIEWER), 'viewer');
req.add(req.newFetchPersonAppDataRequest(idspec, 'myKey'), 'get_data');
req.send(get_callback);
```

To specify who you want to retrieve data for, you can pass any of the following as the first parameter to the newFetchPersonAppDataRequest function:

- opensocial.IdSpec.PersonId.VIEWER
- opensocial.IdSpec.PersonId.OWNER
- opensocial.IdSpec.Group.VIEWER_FRIENDS
- opensocial.IdSpec.Group.OWNER_FRIENDS

The meaning of each of these values will be covered in depth later in this article.

The callback specified by this function gets a DataResponse object back that contains the data that was stored, or an error message if something went wrong. A callback function implementation could look something like this:

```
function get_callback(response) {
  if (response.get("get_data").hadError()) {
    /* the fetch failed ... insert code to handle the error */
  } else {
    var data = response.get("get_data").getData();
    /* the fetch was successful ... "data" contains the app data */
  }
```

```
};
```

Assuming that the request succeeded, the data variable in the function above will be assigned a JSON object with the following layout:

```
{
 XXXXXXXXXXXXXXXXXXX : { myKey : "<value of myKey for user XXXXXX...>" }
}
```

where XXXXXXXXXXXXXXXXXXXX is the ID number of the user who the data belongs to. If your request specifies many people to fetch data for, then the result will look like this:

```
{
 XXXXXXXXXXXXXXXXXXX : { myKey : "<value of myKey for user XXXXXX...>" },
 YYYYYYYYYYYYYYYYYYY : { myKey : "<value of myKey for user YYYYYY...>" },
 ZZZZZZZZZZZZZZZZZZZZ : { myKey : "<value of myKey for user ZZZZZZ...>" }
}
```

If you request multiple keys for each person, the keys will be scoped to each person's ID in the returned data object:

```
{
 XXXXXXXXXXXXXXXXXXX : { myKey1 : "<value of myKey1 for user XXXXXX...>",
                         myKey2 : "<value of myKey2 for user XXXXXX...>" },
 YYYYYYYYYYYYYYYYYYY : { myKey1 : "<value of myKey1 for user YYYYYY...>",
                         myKey2 : "<value of myKey2 for user YYYYYY...>" },
 ZZZZZZZZZZZZZZZZZZZZ : { myKey1 : "<value of myKey1 for user ZZZZZZ...>",
                         myKey2 : "<value of myKey2 for user ZZZZZZ...>" }
}
```

The following code snippet is a prototype that gets and saves a piece of persistent data in an OpenSocial Application. Note that the output function is a testing method used with an application such as CodeRunner, and any output function such as alert() can also work:

```
/**
 * DEFAULT CODERUNNER SAMPLE CODE
 */
function get_callback(response) {

// response.get("oViewer").getData().getId();
var userid = response.get("oViewer").getData().getId();

  if (response.get("get_data").hadError()) {
    /* the fetch failed ... insert code to handle the error */
    output("get callback failed");
  } else {
    var data = response.get("get_data").getData();
    output(data[userid]["myKey"]);

    /* the fetch was successful ... "data" contains the app data */
    output("get callback suceeded");
  }
};
```

```
function set_callback(response) {
  if (response.get("set_data").hadError()) {
    /* The update failed ... insert code to handle the error */
    output("set callback failed");
  } else {
    /* The update was successful ... continue execution */
    output("set callback succeeded");
    get_request();
  }
};

function get_request() {
  var req = opensocial.newDataRequest();
  req.add(req.newFetchPersonRequest("VIEWER"), "oViewer");
  req.add(req.newFetchPersonAppDataRequest("VIEWER", "myKey"), "get_data");
  req.send(get_callback);
};

function request() {
  var req = opensocial.newDataRequest();
  req.add(req.newUpdatePersonAppDataRequest("VIEWER", "myKey", "myValue"), "set_data");
  req.send(set_callback);
};

// Sets a value, and then gets that same value.
request();
```

Sending Messages Using requestSendMessage

Ning actively supports OpenSocial's requestSendMessage API, which requests Ning to send a message to a specific user or users. Rather than the application sending a standard e-mail, all messages will be directed to a section of the Ning Network's messaging system, called alerts. A dialog box will appear for the OpenSocial Application user to confirm the message.

We previously referenced Google's CodeRunner Application, a way to test OpenSocial APIs in real time. For this example, I am friends with one friend on a Ning Network and ran the following code:

```
var params = {};
params[opensocial.Message.Field.TYPE] = opensocial.Message.Type.EMAIL;
params[opensocial.Message.Field.TITLE] = "This is my test subject";
var message = opensocial.newMessage("This is a test message with some random text.", params);

// VIEWER, OWNER, VIEWER_FRIENDS, OWNER_FRIENDS or ids as a string or an array.
opensocial.requestSendMessage("VIEWER_FRIENDS", message, function(resp){
   if (resp.hadError()) {
      // put your error handling code here
      alert('There was an error!\nResponse code: '+resp.errorCode_+'\nResponse message:
 '+resp.errorMessage_);
   } else {
      // handle successful send message logic here.
   }
});
```

Which brings up the following dialog box:

Cannot resolve external resource into attachment.

Once the message has been sent, the person who received the message will receive the message in the alerts portion of their Inbox.

Cannot resolve external resource into attachment.

NOTE: The requestSendMessage API has a quota of 35 invocations per member, per day. This means that a member can use the send message functionality 35 times per day across all apps. If a user enters multiple people in the 'to' field, that counts as a single invocation. The message body has a maximum character limit of 1000 characters and strips out HTML.

Reading from and Writing to a Member's Latest Activity on their Profile Page via the Activity Stream

The following code will let an OpenSocial Application on Ning write to the Latest Activity module on a member's profile page, provided that the Application is installed on your profile page and you are viewing it:

```
<script type="text/javascript">
function postActivity(text) {
  var params = {};
  params[opensocial.Activity.Field.TITLE] = text;
  var activity = opensocial.newActivity(params);
  opensocial.requestCreateActivity(activity, opensocial.CreateActivityPriority.HIGH, callback);
};

function callback(data) {
  /* callback is an optional function which is called after requestCreateActivity. */
  /* do what you would like with the data attribute. */
  document.getElementById("dom_handler").innerHTML = "Latest activity will be shown when you refresh your
 profile page.";
};

postActivity("This is a sample activity, created at " + new Date().toString());
</script>
<div id="dom_handler"></div>
```

NOTE: The requestCreateActivity API has a quota of 5 invocations per member, per application, per day. This means that a member can call the functionality five times a day for each application. New Activity Feed data can only be written to a member's Profile Page, not the main page of a Ning Network. With the exception of the <a> tag, all HTML is stripped from the text before the API is invoked.

Accessing an External REST API via JSON

In order to allow your OpenSocial Application to work with data located on remote servers, you can use a function called makeRequest. makeRequest allows you to get data from remote servers, and send data to remote servers.

Last.fm is an example of a site that uses web services to retrieve metadata, including artist photos. Here's an example of fetching Cher's photo from Last.fm:

```
<script type="text/javascript">

function request() {
  var params = {};
  // Content types available at http://code.google.com/apis/opensocial/docs/0.7/reference/
gadgets.io.ContentType.html
  params[gadgets.io.RequestParameters.CONTENT_TYPE] = gadgets.io.ContentType.JSON;
  var url = "http://lastfm-api-ext.appspot.com/2.0/?
method=artist.getinfo&outtype=json&api_key=b25b959554ed76058ac220b7b2e0a026&artist=Cher";
  gadgets.io.makeRequest(url, response, params);
};

function response(obj) {
  /** obj.data contains an object corresponding with the JSON-encoded response **/
```

```
  output("Cher");
  output(obj.data.image_large);
};

request();
</script>
```

Accessing the OpenSocial RESTful/RPC Protocol

Opensocial 0.8.1 offers a Server side API to communicate using a RESTful or RPC protocol. This component is still under development and will be rolled out at a latter date during the beta testing.

Ning Specific Methods

feature="ning"

If you would like to use the Ning specific features, add the following line to your gadget.xml header:

```
<Optional feature="ning" />
```

If you would like your application to soley run on Ning Networks, use this code instead:

```
<Require feature="ning" />
```

Running the XML above will fail with other social network containers. Additionally, you can use the hasFeature method to specifically check and implement Ning-specific code.

```
if (gadgets.util.hasFeature("ning")) {
  [... you are running on ning ...]
} else {
  [... you are *NOT* running on ning ...]
}
```

Getting Person-Specific Information with ning.Person.Field

```
var env = opensocial.getEnvironment();

function response(data) {
  var user = data.get("req").getData();
  output("User: " + user.getDisplayName());

  if (env.supportsField(opensocial.Environment.ObjectType.PERSON, ning.Person.Field.CREATOR)) {
    var creator = user.getField(ning.Person.Field.CREATOR);
    output(creator ? "User is Network Creator!" : "User is not Network Creator!");
  } else {
    output("Container does not support " + ning.Person.Field.CREATOR);
  }

  if (env.supportsField(opensocial.Environment.ObjectType.PERSON, ning.Person.Field.ADMIN)) {
    var creator = user.getField(ning.Person.Field.ADMIN);
    output(creator ? "User is Network Admin!" : "User is not Network Admin!");
  } else {
    output("Container does not support " + ning.Person.Field.ADMIN);
```

```
  }

  gadgets.window.adjustHeight();
};
```

Getting Ning Container-Specific information with ning.core.AppStatus

The ning.core.AppStatus object is a method for Developers to easily find out whether their OpenSocial code is being run as a Ning App, a Profile View of a Ning App (internally called a Child App) or a Profile App.

```
status = ning.core.getAppStatus();

if (status === ning.core.AppStatus.INDIVIDUAL) {
  output("I am a profile application");
} else if (status === ning.core.AppStatus.NETWORK) {
  output("I am a NingApp");
} else if (status === ning.core.AppStatus.CHILD) {
  output("I am a NingApp descendent");
} else {
  output("Status is unknown (you should assume individual app");
}

gadgets.window.adjustHeight();
```

To test the above code sample, add the Ning version of Coderunner (http://gadgetstore.ning.com/gadgets/coderunner/coderunner08.xml) as a Ning App and as a Profile App. The module should show up once on the main page of your Ning Network and twice on the profile page - once as a Profile App, and once as a Child App.

Three different messages will be displayed from the above code sample, depending on the where the module is installed.

Getting Network Information with ning.app.getInfo

It is possible to get a basic subset of information specific to the Ning Network. Because this information is stored in the OpenSocial core code, the fields are retrieve through the method ning.app.Field, but through the callback function of the ning.app.getInfo method.

```
function callback(foo) {
  if (foo.hadError()) {
    output("Error found!");
  } else {
    for (var x in ning.app.Field) {
      if (ning.app.Field.hasOwnProperty(x)) {
        if (foo.getField(ning.app.Field[x])) {
          output("Field " + x + ": " + foo.getField(ning.app.Field[x]));
        }
      }
    }
  }
  gadgets.window.adjustHeight();
}

if (gadgets.util.hasFeature("ning")) {
  ning.app.getInfo(callback);
} else {
  output("Not running on Ning!");
```

```
}
```

As of this time, ning.app.Field supports the following read-only attributes:

Field	Description	Sample Output
ID		2962556
NAME	The name of the Ning Network prefix. This field will also return the prefix for networks that have domain forwarding enabled.	erniesandbox17
LOCALE		en_US
PUBLISHED	Boolean value of	true
BAZEL	Boolean value of if this is a Ning Network.	true
SEARCHABLE	Boolean value that states whether the network has been indexed in Ning Network searches.	true
PUBLIC		true
CREATED_DATE		Wed Mar 04 00:39:39 GMT 2009
UPDATED_DATE		Fri Mar 06 00:49:39 GMT 2009
TZ_OFFSET		480
USING_DST		true
TAGS		
DOMAINS		
SUB_DOMAIN		erniesandbox17
MEMBER_COUNT	Contains the total number of members in the Ning Network.	1

Getting the Domain Prefixs and Unique Identifiers for a Social Network

ning.core.getNetworkDomain is a Ning-specific method call which returns the domain name on which your Application has been installed.

```
ning.core.getNetworkDomain()
```

Running this code on "mynetwork.ning.com", returns 'mynetwork.ning.com'. Note that if the network has the domain-mapping premium service installed (e.g. mynetwork.com), the method will returned the domain-mapped name (mynetwork.com), NOT the Ning subdomain (mynetwork.ning.com).

Note that using the common OpenSocial method:

```
opensocial.getEnvironment().getDomain();
```

Will return "ning.com" in all cases. This is to offer developers a way to differeniate between Ning and other social containers.

That said, the ning.core.getNetworkDomain() function, should NOT be used to uniquely identify a network, as this will return the mapped domain if the network is using it, and since domain purchases are in the hands of the Network Creator, can change over time.

The ning.app.Field.ID, ning.app.Field.SUB_DOMAIN and ning.app.Field.DOMAINS fields returned from the getInfo call:

- ning.app.Field.ID returns an unique numeric ID for each network. This is a unique identifier.
- The network is mapped to the value in ning.app.Field.SUB_DOMAIN (+ ning.com). This can also be a unique identifier.
- The network is optionally mapped to all values in ning.app.Field.DOMAINS:

```
function callback(foo) {
  if (foo.hadError()) {
    output("Error found!");
  } else {
    output("Network ID: " + foo.getField(ning.app.Field.ID));
    output("Subdomain: " + foo.getField(ning.app.Field.SUB_DOMAIN));
    output("Domains: " + foo.getField(ning.app.Field.DOMAINS));
  }
}

if (gadgets.util.hasFeature("ning")) {
  ning.app.getInfo(callback);
} else {
  output("Not running on Ning!");
}
```

If the following code sample above is pasted to the Ning-modified OpenSocial Dev App available at http://gadgetstore.ning.com/gadgets/coderunner/coderunner08.xml and run on a Ning Network with domain-mapping enabled, you will get a result similar to the following:

```
Network ID: 2330813
Subdomain: networkappsandbox2
Domains: examplenetwork.com
```

ning.core.getContainerVersion

```
if (gadgets.util.hasFeature("ning")) {
  output("Current Container Version is " + ning.core.getContainerVersion());
} else {
  output("Not running on Ning!");
}

gadgets.window.adjustHeight();
```

Validating Signed Requests Through OAuth

Ning Apps, are web applications - applications that send information of members of Ning Network to third-party servers and vice versa. Because these applications are from a third-party, how can we be guaranteed that someone won't try to impersonate web request using a tool like, say, Firebug?

One solution is to use a signed request, based on OAuth's parameter signing mechanisms. When someone does an action that triggers a web request, your Ning App can attach information about the social network container (Ning), the specific social network and the user themselves; this is combined with a cryptographic hash, so any impersonated requests with different information will return an error. You can also read a little bit more about signed requests on the OpenSocial wiki: http://wiki.opensocial.org/index.php?title=Introduction_To_Signed_Requests

Sounds Great. How Do We Implement This in our App?

We'll show you some code that we use for one of our own Profile Apps, Status. The application is a basic status display application, which takes input from the user, appends some network information to it in the form of the Ning Network ID and timestamp, and saves and retrieves data to the backend. The data is stored on a third-party server which runs PHP, and has a file and directory structure similar to what's shown:

```
/keys/ning.com.cert
/lib/OAuth.php
/helpers/SecurityHelper.php
/actions/create.php
gadget.js
```

Applications that run the OpenSocial 0.7 spec - such as Ning Apps, use Javascript to call social methods and send and receive requests. This is the makeRequest method, and in the Status Profile App, the makeRequest method is called as shown, in the file gadget.js:

```
var url = appUrl + '/gadget.php'
 var params = {};
 var postdata = {
    action : 'create',
    viewer_id : viewer.getId(),
          container: network,
    status_textarea : status_text
 };

 params[gadgets.io.RequestParameters.AUTHORIZATION] = gadgets.io.AuthorizationType.SIGNED;
 params[gadgets.io.RequestParameters.CONTENT_TYPE] = gadgets.io.ContentType.JSON;
 params[gadgets.io.RequestParameters.METHOD] = gadgets.io.MethodType.POST;
 params[gadgets.io.RequestParameters.POST_DATA] = gadgets.io.encodeValues(postdata);

 gadgets.io.makeRequest(url, statusSubmitted, params);
```

Without going into too much detail, while the makeRequest method calls gadget.php, the file takes the contents available in /actions/create.php:

```
if (SecurityHelper::isSignatureValid()) {
      $payload["query"] = array_merge($_GET, $_POST);
      if (SecurityHelper::requesterIsOwner()){
         $status_text = strip_tags($_POST['status_textarea']);
         try {
            // Saves the status here ...
            $payload["result"] = 'created';
```

```
            } catch ($e) {
                $payload["result"] =  $e->getMessage();
            }
        } else {
            $payload["result"] = 'not authorized';
        }
    } else {
        $payload["validated"] = "This request was spoofed";
    }
```

In /helpers/SecurityHelper.php:

```
require_once("lib/OAuth.php");

class SecurityHelper{
    /**
     * Check if the OAuth signature in the request is valid
     * code based on http://wiki.opensocial.org/index.php?title=Validating_Signed_Requests
    **/
    public static function isSignatureValid(){
        //Build a request object from the current request
        $request = OAuthRequest::from_request(null, null, array_merge($_GET, $_POST));

        //Initialize the new signature method
        $signature_method = new SignatureMethod();

        //Check the request signature
        @$signature_valid = $signature_method->check_signature($request, null, null,
 $_GET["oauth_signature"]);

        return ($signature_valid == true);
    }

    /**
     * Do we know if the request came from the gadget owner
     **/
    public static function requesterIsOwner(){
       return ( (!is_null($_REQUEST["opensocial_owner_id"])) && ($_REQUEST["opensocial_owner_id"] ==
 $_REQUEST['opensocial_viewer_id']));
    }

}

class SignatureMethod extends OAuthSignatureMethod_RSA_SHA1 {
    protected function fetch_public_cert(&$request) {
        $filename = 'keys/' . $_REQUEST['xoauth_signature_publickey'];
        if (!file_exists($filename)) {
            throw new Exception('Certificate not found');
        }
        return file_get_contents($filename);
    }
}
```

The file /helpers/SecurityHelper.php includes our OAuth file and includes some additional methods, including fetch_public_cert, which returns the Ning Public Key that we make available at https://opensocialresources.appspot.com/certificates#container11. You can find the OAuth file at the URL http://oauth.googlecode.com/svn/code/php/OAuth.php - which is taken from an OAuth PHP library provided by Google. If you would like to look at more examples, you can view http://oauth.googlecode.com/svn/code/php/ .

A xoauth_signature_publickey certificate should now be available to validate signed requests using the makeRequest method, along with oauth_consumer_key, opensocial_owner_id, opensocial_viewer_id and opensocial_app_id parameters. The certificate is available at http://developer.ning.com/certificates/ning.cert. (NOTE: Even though the ning.cert name is used, xoauth_signature_publickey will return "ning.com.cert" - make sure to test the rename your certificate names appropriately when testing out your application.)

In the case of the oauth_consumer_key, Ning differs from other social networking containers in that each Ning Network maps to a unique consumer key. For example, "examplenetwork.ning.com" will be returned rather than "ning.com". Also note that domain-mapped Ning Networks also will point to its corresponding non-mapped domain. The way OpenSocial on Ning handles signed requests is based off of Orkut's methods of validating signed requests at http://wiki.opensocial.org/index.php?title=Validating_Signed_Requests, with two major changes:

- Replace Orkut's certificate with our certificate, tenatively available at http://developer.ning.com/certificates/ning.cert
- Note that the variables sent are of the form opensocial_owner_id, not opensocial_ownerid.

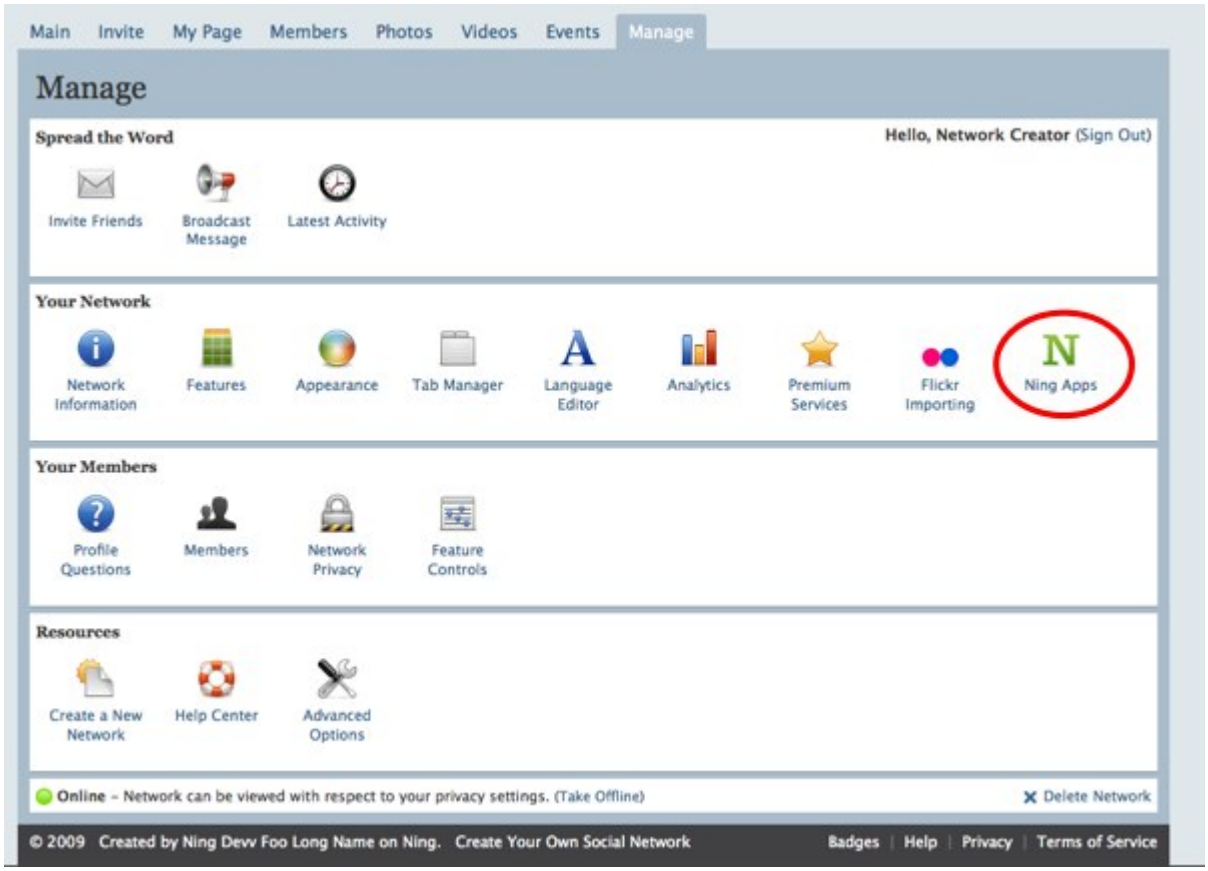
Policies

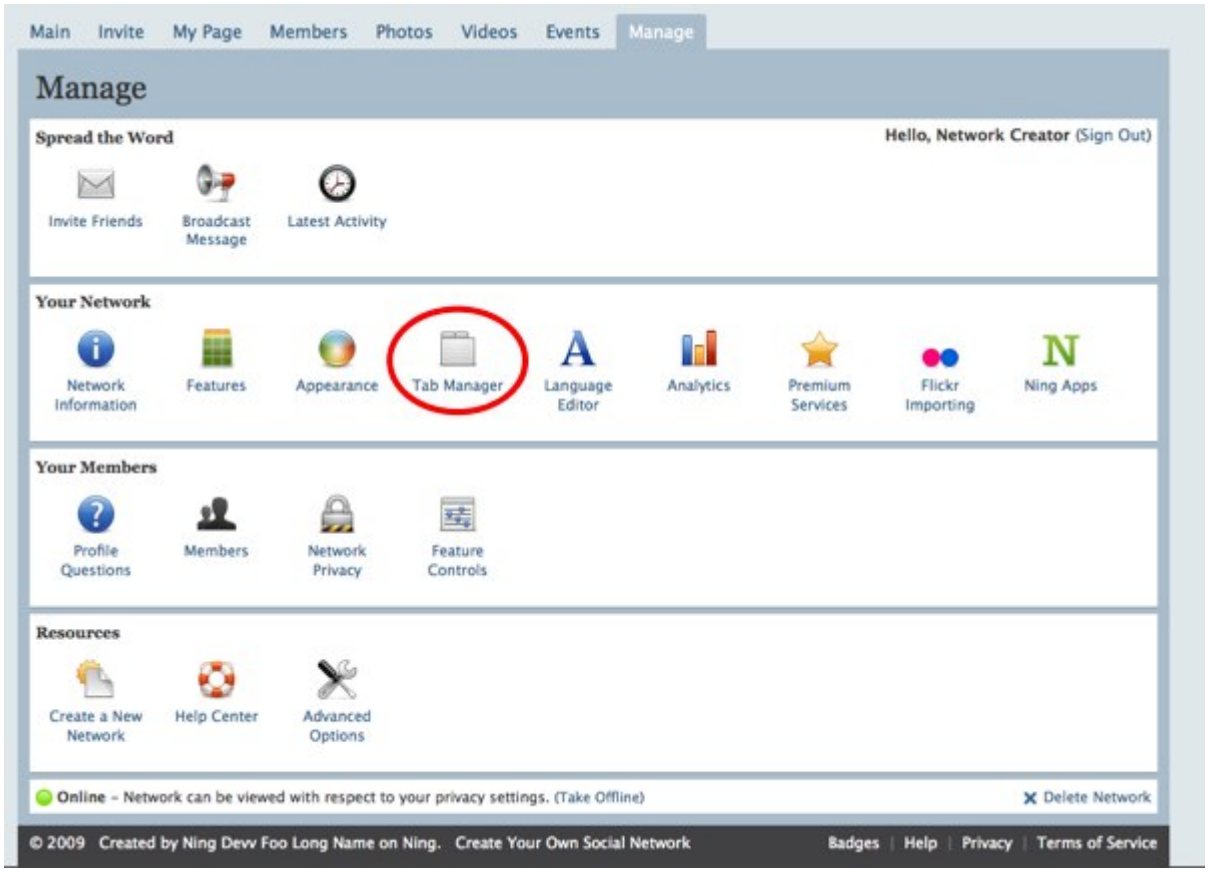
- The method opensocial.getEnvironment().getDomain(); will return "ning.com" in all cases; this is to offer developers a way to differentiate between Ning and other social containers. Developers should use ning.app.Field.ID or ning.app.Field.SUB_DOMAIN if they would like a way to generate a unique primary key.
- The requestSendMessage API has a quota of 35 invocations per member, per day. This means that a member can use the send message functionality 35 times per day across all apps. If a user enters multiple people in the 'to' field, that counts as a single invocation.
- The requestCreateActivity API has a quota of 5 invocations per member, per application, per day. This means that a member can call the functionality five times a day for each application. New Activity Feed data can only be written to a member's Profile Page, not the main page of a Ning Network. With the exception of the <a> tag, all HTML is stripped from the text before the API is invoked.
- As a Ning OpenSocial Application developer, you can access a Viewer's profile photo, profile name, profile link and friends list. All information on the platform is not exposed for privacy reasons.
- On the Ning platform, a network member can have friends across many Ning Networks on the platform. Because of this unique nature, VIEWER_FRIENDS and OWNER_FRIENDS apply to the set of friends on the specific Ning Network where the OpenSocial Application is hosted; they are not a collection of your friends over all Ning Networks.
- If a member of Ning has a default profile name and photo and a different profile name and photo on a specific Ning Network, the information on that network takes precedence when calling profile-based methods such as getDisplayName().
- Ning allows a maximum of 10k of serialized data stored, including keys, values and serialization overhead.
- No HTML is allowed when writing to the Activity Stream.
- Although OAuth is supported for Ning containers, you can not use OAuth to view a member's content at the present time.

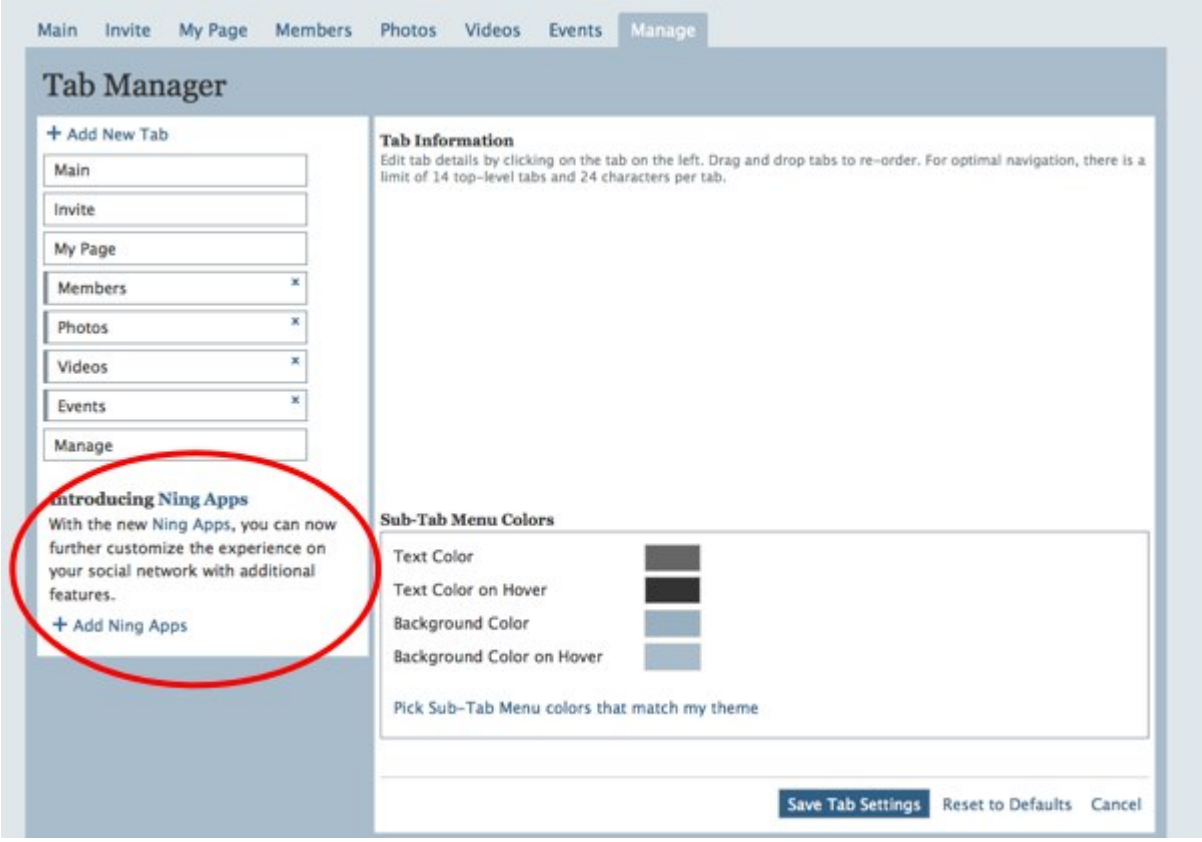
Testing your Ning App

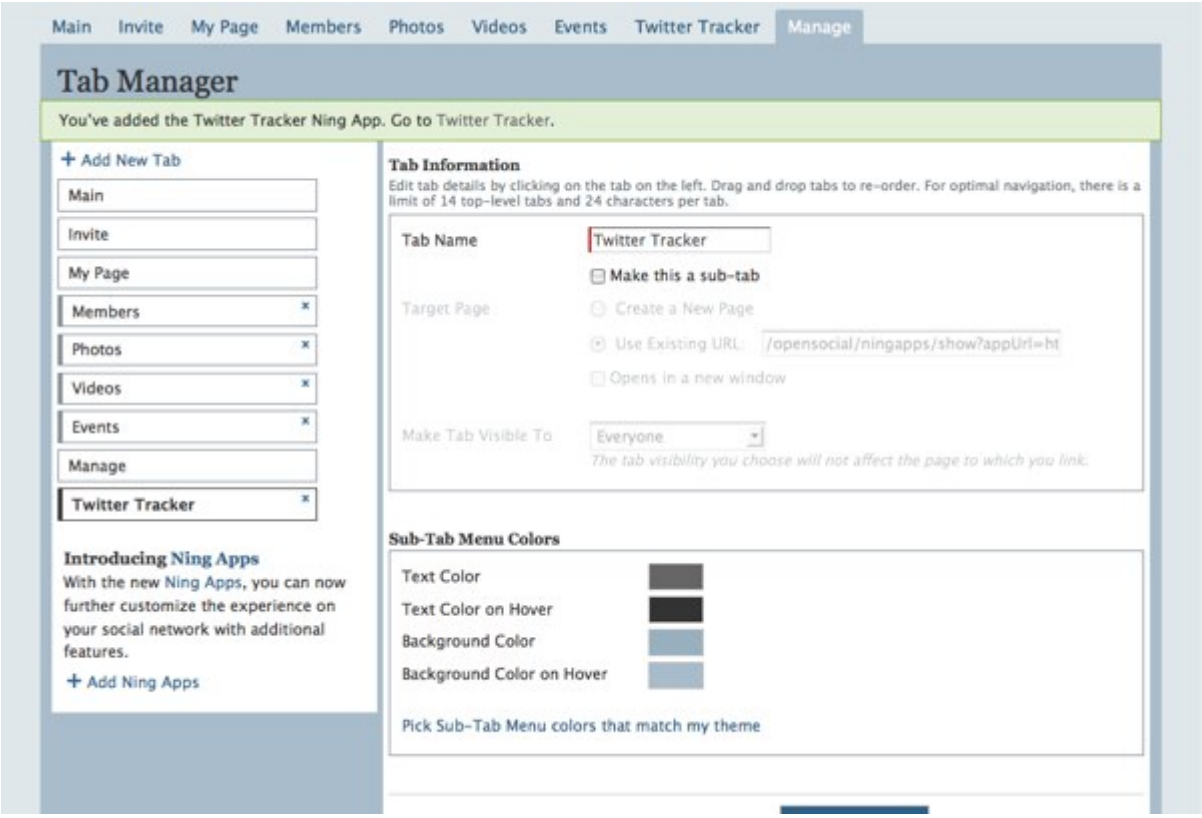
Adding your Application

The Ning Apps interface is available for Network Creators via the Ning Apps icon. This will display Ning Apps that have previously been accepted in the Ning Apps Directory.


Main
Invite
My Page
Members
Photos
Videos
Events
Manage
Manage
Spread the Word
Hello, Network Creator (Sign Out)
Invite Friends
Broadcast Message
Latest Activity
Your Network
Network Information
Features
Appearance
Tab Manager
Language Editor
Analytics
Premium Services
Flickr Importing
Ning Apps
Your Members
Profile Questions
Members
Network Privacy
Feature Controls
Resources
Create a New Network
Help Center
Advanced Options
Online – Network can be viewed with respect to your privacy settings. (Take Offline)
Delete Network
© 2009 Created by Ning Devv Foo Long Name on Ning. Create Your Own Social Network
Badges | Help | Privacy | Terms of Service


Main
Invite
My Page
Members
Photos
Videos
Events
Manage
Manage
Spread the Word
Hello, Network Creator (Sign Out)
Invite Friends
Broadcast Message
Latest Activity
Your Network
Network Information
Features
Appearance
Tab Manager
Language Editor
Analytics
Premium Services
Flickr Importing
Ning Apps
Your Members
Profile Questions
Members
Network Privacy
Feature Controls
Resources
Create a New Network
Help Center
Advanced Options
Online – Network can be viewed with respect to your privacy settings. (Take Offline)
Delete Network
© 2009 Created by Ning Devv Foo Long Name on Ning. Create Your Own Social Network
Badges | Help | Privacy | Terms of Service





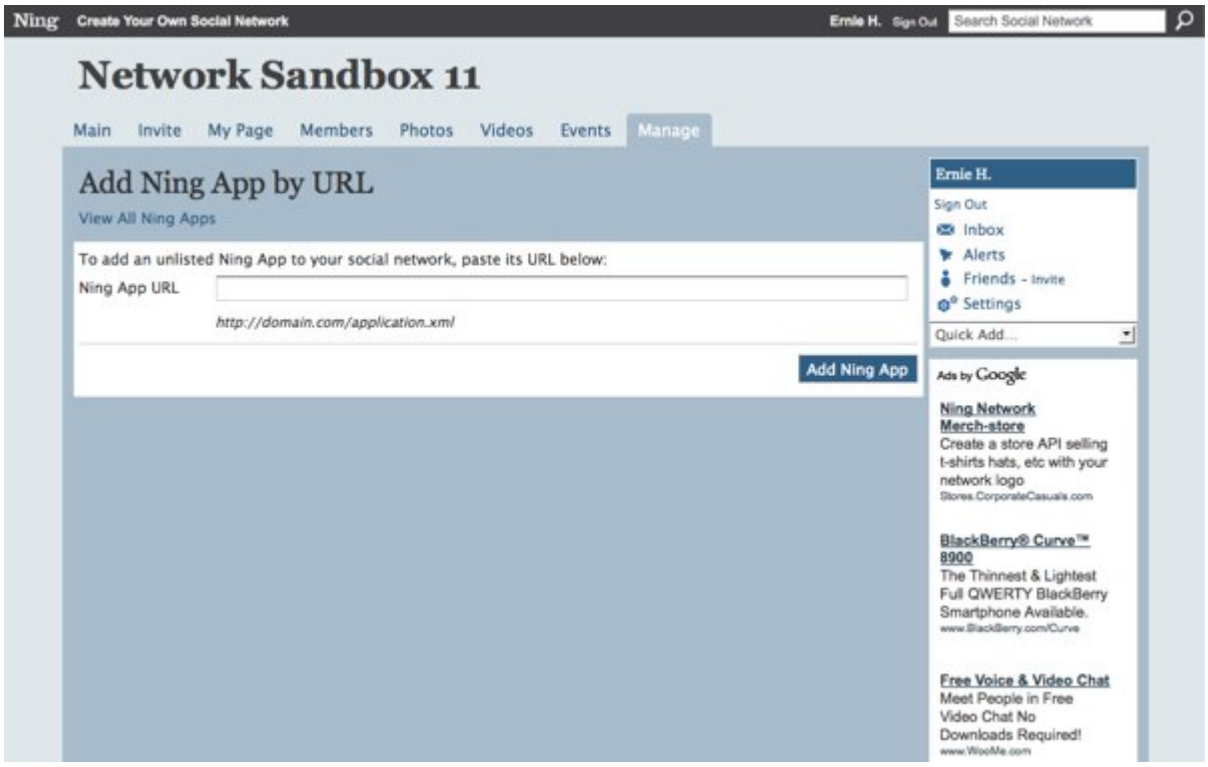
Adding an Application by URL

As an Application Developer, you can bypass the Ning App Directory and test your Ning App directly by creating a Ning Network per usual, and add your XML URL manually. To add your XML file, navigate to the following URL:

http://NETWORKNAME.ning.com/opensocial/ningapps/addByUrl

Note that NETWORKNAME is the name of the Ning Network.

In the text box, paste the URL of your Application XML file.



The following ports for supported for hosting Network Apps: 80 (HTTP), 443 (HTTPS), 81, 8080, 8081, 8443.

A Note on Caching

By default, all OpenSocial Applications will have caching enabled to ensure that all Ning Networks are running at its speediest. That said, if you are actively developing an OpenSocial Application and would like to temporarily turn caching off, you can bypass the caching mechanism by adding "?ningapp_nocache=1" to the URL of the Ning Network profile or canvas page that you're viewing.

Otherwise, the following caching logic is used:

- Is the cache over a week old? If so, fetch a new version right away.
- If the cache is not over a week old, is the cache over 24 hours old? If so, fetch a new version in the background; this request uses the stale version, but subsequent requests will not.
- If the cache is not over 24 hours old, go ahead and use the cached version.

Removing an Application

To remove a Ning App, go to the Tab Manager under your Manage page as a Network Creator, click the [x] icon to remove the app, and click on "Save."

Canvas View

Click the 'View Application' link to navigate to your Application's Canvas View.

About View

When viewing the Canvas View of an application you do not have installed, you can click the 'About this Application' link to navigate to the About View for that application. From a developer standpoint, the

About View page of an OpenSocial Application is generated by taking the title, description and screenshot image attributes of the OpenSocial XML file, located in the <ModulePrefs> tag.

Submitting your Application to the Ning App Directory

To submit your OpenSocial Application to the Ning OpenSocial Application Directory, log in to the Ning Developer Network (http://developer.ning.com) and click on "Create an Application" on the main page, or choose "Submissions" from the navigation menu. From there, enter the XML file of your OpenSocial Application.

```
<Module xmlns:ning="http://developer.ning.com/opensocial/">
 <ModulePrefs title="My App" description="" author_email="@ning.com" thumbnail="">
   <ning:screenshot view="profile">http://remote/ss.png</ning:screenshot>
   <ning:screenshot view="tab">http://remote/tab.png</ning:screenshot>
 </ModulePrefs>
 <Content>
  ...
 </Content>
</Module>
```

The <ModulePrefs> tag of your XML file MUST have the following attributes filled out for an OpenSocial Application to be considered for the Directory:

Attribute	Description
title	The title of your OpenSocial Application. The title is used in many places, including the Ning App Directory, the module title area as well as the default value in the Navigation Tabs. All titles should have a maximum length of 24 characters.
description	The string that describes the OpenSocial Application. This description appears in About View of an Application page.
thumbnail	URL of the Application thumbnail. This must be an 120x60 image of a piece of logo or branding of your application or your company. The image should also be on a public web site that is not blocked by robots.txt. PNG is the preferred format, though GIF and JPG are also acceptable.
screenshot	URL of the Application screenshot. This must be an image on a public web site that is not blocked by robots.txt. PNG is the preferred format, though GIF and JPG are also acceptable. Screenshots should be 280 pixels wide, with a maximum width of 502 pixels wide. The height of the screenshot should be the "natural" height of the gadget when it's in use, with a maximum height of 1000px.

The <ning:screenshot> tag is a series of Ning-only tags that are used when Network Apps are submitted to the Ning App Directory, and are used for screenshots within the directory.

ning:screenshot view attribute	Description	Example

~~ning.main~~	~~Includes a screenshot of the main view of your Ning App. This field is required.~~	~~<ning:screenshot view="ning.main">http://remote/mainview.png</ning:screenshot>~~
profile	Includes a screenshot of the profile view of your Ning App, if your Ning App includes a profile view. This field is optional.	<ning:screenshot view="profile">http://remote/ss.png</ning:screenshot>
tab	Includes a screenshot of the tab view of your Ning App. This field is optional.	<ning:screenshot view="tab">http://remote/tab.png</ning:screenshot>

We have made a .PSD file available (Photoshop format) that you may use as a template to create your own screenshots, especially in the context of a Ning Network. It's available at the following URL:

http://os.ning.com/tutorials/ningapps/docs/app-screen-template-455.psd

Once you have added the URL, your Application will be in one of the following states, listed below.

Status	Description
Saved	An Application receives this state once you successfully add the URL of your XML File. Select a Category that best matches your application, and click 'Submit for Review'. Categories are chosen from a select box of the following values: Dating, Events, Food & Drink, Fun, Games, Messaging, Music, Movies & TV, News, Photos, Politics, Professional, School, Sports, Travel, Video. (This category is required.) Upon succesful submission, your application will be in a "Pending" state.
Pending	Once the Application is submitted and all required fields have been completed, the Application is in a "Pending" state awaiting approval from a member of the Ning Team. From there, the Application will be either "Active" or "Rejected," and an e-mail will be sent to the email address you used to log in to the Ning Developer Network.
Active	An "Active" Application indicates that the Application is in the Directory. Once in this state, Application Developers have the ability to delete or take their Applications offline.
Rejected	A "Rejected" Application indicates that the Application was rejected. Any additional notes will also be included with the status message.
Removed	A "Removed" Application indicates that an Application that was previously in the Directory has since been removed. Any additional notes will also be included with the status message.

Best Practices

Because of our committment to Ning's Network Creators, we have a thorough list of qualifications for a Ning Network to be included into the Ning App Directory. Here are a list of best practices that we recommend:

What You MUST Do

- **Your application must add value to networks.**
- **Make sure the ModulePrefs attributes of your Application are properly filled out and images are the proper sizes.**
  - Title: This will be the name of your application in the Directory, Profile Page, About Page and Canvas Page. Try to keep it under 40 characters.
  - Thumbnail: This should be 120x60. It is a logo that you want to associate with your application, and will be displayed in the Application Directory. The thumbnail must accurately represent the application.
  - Description: The description is displayed in the Directory, and also on the About Page above the screenshot. The description should accurately describe the application, make sense, and be free of spelling and grammatical errors.
- **Make sure the <ning:screenshot> attributes of your Application are properly filled out and images are the proper sizes.**
  Because the Ning Apps directory makes available additional screenshots, all Ning Apps taking advantage of this must use the <ning.screenshot> tags.
  - To maintain XML validation, use <Module xmlns:ning="http://developer.ning.com/opensocial/"> rather than <Module>.
  - Please see the "Submitting your App in the Ning App Directory" for further details.
- **The ~~ning.main,~~ profile and canvas view of your application must look and work as expected.**
- **Make sure the author_email attribute is filled out in <ModulePrefs>, and make sure it's pointed to an e-mail address where you can gather feedback from Network Creators and Ning Network members.**
  In the Ning Apps Directory detail pages, the author_email attribute will be exposed as a way for Network Creators to send questions and feedback. If you do expose the author_emails attribute, it should point to a place where you can gather appropriate feedback for your Ning App.
- **Members that aren't logged in may be able to view your OpenSocial Application. Give them a good experience.**
  Don't forget that both the Profile View and Canvas View of an OpenSocial Application can be viewed in a signed-out state for profile pages in public Ning Networks. The user ID for these users are given the id of _anonymous. If your OpenSocial Application depends on member-specific information, make sure the application can fail gracefully in this condition.
- **If your Application allows users to download, view, listen to or otherwise access or distribute third party content dealing with copyright, follow the instructions of Section 6 of the Application Developer Terms of Service.**

What You SHOULD Do

- **Make sure your application is free of spelling and grammatical errors.**
- **Test your OpenSocial Application in a variety of browsers.**
  For a good variety of appropriate browsers and platforms, we recommend Yahoo's A-Grade browser chart. This means testing for IE6, IE7, the latest versions of FireFox and Safari.
- **Be cognizant of the different type of Ning Networks.**
  Ning is different from other social network containers that run OpenSocial; while most social network containers only allow members to access and edit information from their profile pages, Ning allows its members to create Ning Networks that other members can join. With over half a million networks, there is great diversity: Some Ning Networks pay a premium service to remove promotional links to Ning. Other Ning Networks are geared towards education and are sensitive to content geared towards mature audiences.
- **Recognize that both Network Creators AND Network Admins can add a Ning App**
  It's important to know that Ning Apps can be added to a Ning Network by not only the Network Creator, but by a member of their network designated as a Network Admin, usually done on a smaller edge case of larger networks to facilitate community. That means the OWNER

and VIEWER IDs may not be the same. That said, you can use ning.Person.Field.ADMIN and ning.Person.Field.CREATOR to check if the user is a Network Creator or not.

- **Optimize your OpenSocial Application so it runs as speedy as possible.**
  Many optimization tips used in general web development will also apply to developing your OpenSocial Applications. You may want to take a look at the OpenSocial Latency Combat Field Manual for tips on optimizing HTTP Requests, Caching and other optimization tricks.
- **Ensure that your OpenSocial Application is caching properly.**
  By default, all OpenSocial Applications will have caching enabled to ensure that all Ning Networks are running at its speediest. That said, if you are actively developing an OpenSocial Application and would like to temporarily turn caching off, you can bypass the caching mechanism by adding "ningapp_nocache=1" to the URL of the profile or canvas page that you're viewing.
- **Give people using your OpenSocial Application a fun and meaningful social experience.**
  With the ability for Developers to use information such as a member's profile information and friends lists, the possibilities of creating an awesome OpenSocial Application are endless. The Social Design Best Practices document on Google has some great resources.
- **If appropriate, have your OpenSocial Application blend in as much as possible to a member's Profile Page.**
  You can use gadgets.skins.getProperty to get the CSS values of the default foreground, background and link colors - use this to your advantage for the best user experience. To see an example of this in action, take a look at the Weather Sharer application, or view the page on Google Code: http://code.google.com/apis/opensocial/docs/0.8/reference/gadgets/#gadgets.skins.Property.

What NOT to do

- **If your OpenSocial Application shares user generated content between all Ning Networks, profile thumbnail images can NOT be displayed.**

- **Don't require your users to download an browser/platform-specific external application.**
  We want to be sure that all people, regardless of the browser they use, have a great experience when using an OpenSocial application. For this reason, any OpenSocial applications that have severe limitations on particular platforms or require users to download external files to install on operating systems will not be admitted into the Ning OpenSocial Application Directory.

- **Don't violate our Terms of Service.**
  The Terms of Service we currently use for Ning Networks carry over to our OpenSocial Applications, and violators will be handled appropriately.
- **Don't do anything that will overly annoy the people using your OpenSocial Application.**
  If you have an OpenSocial Application that runs slow scripts, requires an external download, immediately navigates away from the page or blocks the execution of the Profile Page, you'll upset the people using your Application which will make your OpenSocial application less popular.

Support Resources and References

- OpenSocial APIs Supported by Ning
- OpenSocial Tutorials v0.7
- Gadgets XML Reference Page
- OpenSocial API Blog
- Social Design Best Practices
- Frequently asked Questions about OpenSocial
- OpenSocial Developer Forum
- Orkut's Guide to Localizing OpenSocial Applications
- Gadgets and Internationalization
- CodeRunner - "CodeRunner is an OpenSocial gadget that allows you to execute JavaScript calls against the container that it is installed in, without having to install or update a gadget XML file. CodeRunner is great for testing code snippets and mocking out test applications."
- Ning Certificate for Signed Requests - Read more about Validating Signed Requests

Ning API Specific Resources

- Ning Developer Network
- REST API JSON Operations for Ning
- XML Operations

Policy Based Links

- Ning Terms of Service
- 3rd Party Application Policy
- General Guidelines for Developing on the Ning Platform