Adding OpenSocial gadgets to VIVO

Instructions for connecting VIVO and Open Research Networking Gadgets

This document contains instructions on how to configure your VIVO installation to use OpenSocial gadgets.

VIVO uses an extension of the OpenSocial protocols called Open Research Networking Gadgets, or ORNG. ORNG is a project of the Clinical & Translational Science Institute at the University of California, San Francisco. You can find out more about the ORNG project at their web site, http://www.opengadgets.org/index.html

ORNG supports gadgets using a modified version of Apache Shindig. These instructions tell you how to install the Shindig-ORNG web application, and how to configure it to work with VIVO.

Note: these instructions assume that you will be installing VIVO on Tomcat using the standard installation procedure. VIVO does not yet support ORNG on containers other than Tomcat.

Installing and Configuring

Configuring VIVO to support ORNG requires several steps, including additions to the VIVO properties, modifications to Tomcat, creating a security key for safe network operations, and running a build script.

Create database tables and procedures

Shindig-ORNG uses several database tables in MySQL to store its data: which gadgets appear on which pages, how large are the gadgets, what information applies to each individual, and more. Shindig-ORNG also creates stored procedures in MySQL. These are small pieces of code that simplify the use of the database tables.

In the VIVO distribution directory, a file called `vitro-core/opensocial/shindig_orng_tables.sql` contains SQL commands that create the tables and stored procedures for Shindig-ORNG to use.

Tell MySQL to process this file with a command like this:

```
mysql -u username -p database < sql_file
```

So, if your current directory is the VIVO distibution directory, and your VIVO database is `vivoDb` and your MySQL user account is `vivoUser`, then you might use the command this way:

```
mysql -u vivoUser -p vivoDb < vitro-core/opensocial/shindig_orng_tables.sql
```

MySQL will prompt you for the password for your MySQL user account, and then process the file.

You may want to start your gadget collection with some example gadgets that have been developed by the ORNG group. The file called `vitro-core/opensocial/shindig_example_gadgets.sql` contains SQL commands that will add these gadgets to your system's configuration.

If you want to load these example gadgets, you can use a command similar to the previous one:

```
mysql -u vivoUser -p vivoDb < vitro-core/opensocial/shindig_example_gadgets.sql
```

As before, MySQL will prompt you for the password for your MySQL user account, and then process the file.

Create configuration directory and key file

In your VIVO home directory, create a directory called *shindig*. Under that, create directories called *conf* and *openssl*. Your VIVO home directory will look something like this:

```
     [VIVO home directory]
      |
      |--shindig
      |    |
      |    |--conf
      |    |
      |    |--openssl
      |
      |--solr
      |
      |--uploads
```

Shindig-ORNG uses an encryption key to insure that the communication between the gadget and the server is secure. You should create a file that contains the encryption key, and store that file in the *shindig/openssl* directory that you created.

On Unix-based systems (like Linux or Mac OS X), this command will create an encryption key from a random seed:

```
dd if=/dev/random bs=32 count=1 | openssl base64 > [key-file]
```

For example, if your VIVO home directory is */usr/local/vivo/data*, you might use the command this way:

```
dd if=/dev/random bs=32 count=1 | openssl base64 > /usr/local/vivo/data/shindig/openssl/securitytokenkey.txt
```

If your VIVO installation is installed on a machine that runs Microsoft Windows, you will need to find another way to create an encryption key. The easiest way might be to find a Unix-based machine, issue the command above, and copy the resulting file to your Windows machine.

Modify Tomcat settings

The Shindig-ORNG application must know where to find the configuration file that you created in Step I. It must also know its own URL, so that URL can be inserted into the gadgets.

This information is provided through startup parameters in Tomcat. With most installations of Tomcat, the "setenv.sh" or "setenv.bat" file in Tomcat's bin directory is a convenient place to set these parameters. *If this file does not exist in Tomcat's bin directory, you can create it.*

Here is an example of the setenv.sh file, showing only the Shindig-ORNG requirements:

```
export CLASSPATH=/usr/local/vivo/data/shindig/conf
export CATALINA_OPTS="-Dshindig.host=localhost -Dshindig.port=8080"
```

Here is the equivalent file for an installation in Windows.

```
set CLASSPATH=C:\vivo\data\shindig\conf
set CATALINA_OPTS=-Dshindig.host=localhost -Dshindig.port=8080
```

This assumes that your setenv file was empty before starting this process, and that you used the default location for the Shindig-ORNG configuration file in Step I. In fact, it's more common for the setenv file to contain other parameters besides those used for Shindig-ORNG. In that case, it might look more like this:

```
export CLASSPATH=/usr/local/vivo/data/shindig/conf
export CATALINA_OPTS="-Dshindig.host=localhost -Dshindig.port=8080 -Djava.awt.headless=true -Xms1024m -Xmx1024m
-XX:MaxPermSize=128m"
```

Or, for Windows:

```
set CLASSPATH=C:\vivo\data\shindig\conf
set CATALINA_OPTS=-Dshindig.host=localhost -Dshindig.port=8080 -Djava.awt.headless=true -Xms1024m -Xmx1024m -XX:
MaxPermSize=128m
```

Configure VIVO

In the VIVO distribution directory, the file called *build.properties* contains configuration options for the VIVO application. You must set some additional parameters so VIVO will be able to communicate with Shindig-ORNG.

For now, these properties must be entered into both `build.properties` and the `runtime.properties` file in your VIVO home directory.

Property name	**`OpenSocial.shindigURL`**
Description	The base URL that VIVO will use when contacting the Shindig-ORNG application. Usually, this is the same host and port number as VIVO itself, with a context path of *shindigorng*
Default value	NONE
Example value	`http://localhost:8080/shindigorng`

Property name	**`OpenSocial.tokenService`**
Description	The host name and port number of the Token Service that Shindig-ORNG creates. Note that a value of *localhost* or *127.0.0.1* will not work. You must provide the actual host name of your machine, followed by *:8777*
Default value	NONE
Example value	`myhost.mydomain.edu:8777`

Property name	**`OpenSocial.tokenKeyFile`**
Description	The path to a key file that will be used to generate security tokens. This is the file that was created in Step I of this process.
Default value	NONE
Example value	`/usr/local/vivo/data/shindig/openssl/securitytokenkey.txt`

Run the deployment script

At the command line, from the top level of the VIVO distribution directory, type:

```
ant orng
```

to configure the Shindig-ORNG application and deploy it to Tomcat's webapps directory.

You must restart Tomcat so the main VIVO application will load the new settings.

Does it work?

Startup tests

Start VIVO, and verify that you can see VIVO's home page in a browser.

On startup, VIVO runs a series of self-diagnostics, or "smoke tests". If these tests find any problems with the OpenSocial configuration, you will see a warning message instead of the VIVO home page.

Some of VIVO's "smoke tests" are run after the startup is finished, and may take up to a minute to complete. If one of these tests fails, you will see the warning message as you navigate from one VIVO page to the next.

If one of the OpenSocial tests fails, you may continue to use VIVO, but it is likely that no gadgets will be shown. You can review the warning message by selecting the "Startup Status" link from the "Site Admin" page.

Search page

If you loaded the example gadgets, you should be able to see the "Google Search" gadget on the Search Results page in VIVO.

Every VIVO installation comes with a geographic data model, so type "Chile" in the search box, and view the results. Near the bottom of the page, you should see the "OpenSocial" section heading, and beneath it, a gadget offering "Full Text Search Results". This gadget does a google search at UCSF, using the search term that you entered. Again, this gadget is just an example, to show what is possible with OpenSocial gadgets and VIVO.

The first time you bring up the search page, it may take several seconds for the gadget to appear. After the first time, the gadget response should be much faster.

Profile page

If your VIVO installation contains profiles of people, you can see several gadgets on their profile pages. You must be logged in to VIVO, with authority to edit the profile you are viewing.

Go to a personal profile page in VIVO. If you loaded the example gadgets, you will see the "OpenSocial" section heading above the property lists, with an assortment of example gadgets available for experimentation.

As with the search page, the first appearance of the gadgets may be slow.

Troubleshooting

If the gadgets do not appear as you expect, look for these symptoms, and check for the corresponding possible causes.

Symptoms	Possible causes
• The "OpenSocial" heading does not appear on Individual page or in search results. • Tomcat log files do not contain errors.	• VIVO was not re-deployed with ant deploy after the OpenSocial values were set in *build.properties*
• Gadgets do not appear on Individual page or in search results. • Tomcat "localhost" log file contains an error message: `Unable to load properties: shindigorng.properties`	• Configuration file is not correctly named. • Tomcat's setenv file does not specify the correct CLASSPATH

• Dialog box appears in the browser with the message: "Error 500 reading application data: internalError"
• Tomcat "catalina" log file contains an error message: `java.sql.SQLException: Access denied for user`
	• Configuration file contains incorrect value for one or more of these:
	  ◦ orng.dbURL
	  ◦ orng.dbUser
	  ◦ orng.dbPassword

• "Smoke tests" fail at startup. `Token key file for Shindig does not exist` • Pages that display gadgets "hang" in the browser. • Tomcat "localhost" log file contains error messages, including: `com.google.inject.CreationException: Guice creation errors`	• *OpenSocial.tokenKeyFile* is not set in *build.properties*, or the file does not exist at the specified location.
• Gadgets do not appear on Individual page or in search results • vivo.all.log contains an error message: `MySQLSyntaxErrorException: Table 'vivo.orng_apps' doesn't exist`	• MySQL does not contain the shindig tables. `shindig_orng_tables.sql` was not processed.
• Gadgets do not appear on Individual page or in search results • vivo.all.log contains an error message: `java.net.ConnectException: Connection refused`	• In *build.properties*, *OpenSocial.tokenService* is not set correctly.

Changing the gadget configurations

VIVO will look at tables in MySQL to determine what gadgets should be made available, where they should appear, how big they will be, and much more. At this time, VIVO doesn't provide a user interface to edit the contents of these tables. Administrators will need to use a MySQL admin client, or SQL commands, to set these parameters.

The tables are named *orng_apps* and *orng_app_views*, and are described in the following sections.

The *orng_apps* database table

This table acts as a dictionary of the available gadgets. It includes the name and ID of the gadget and where the source code is stored on the web.

Field	Type	Usage
appid	int(11)	Identifies the gadget. In particular, this will be used to determine which rows in the *orng_app_views* table should apply to this gadget.
name	varchar (255)	A user-friendly name for the gadget. This will be displayed in the gadget's "Title Bar".
url	varchar (255)	The location where the gadget's contents and behavior are defined.
PersonFilterID	int(11)	deprecated - usually set to *NULL*
enabled	tinyint(1)	If set to 0, this gadget will never be displayed. If set to 1, it is displayed according to the rules in the *orng_app_views* table.
channels	varchar (255)	Keywords that identify the communication channels between the gadget and VIVO.

The *orng_app_views* database table

This table tells how, where, and when to display the gadgets that are described in *orng_apps*. Each row in this table is a "view", describing a single gadget and the rules that determine whether the gadget will be displayed on a particular page.

Note: If a gadget is described and enabled in the *orng_apps* table, but has no records in the *orng_app_views* table, the gadget will be displayed without restriction on all ORNG-enabled pages. This can be helpful when developing a new gadget. To avoid this, either

- remove the gadget from *orng_apps*, or
- set the *enabled* flag in *orng_apps* to *0*, or
- create a rule for the gadget in *orng_app_views.*

Field	Type	Usage
appid	int(11)	Determines which gadget in *orng_apps* is affected by this rule.
viewer _req	char(1)	What requirements must the viewer satisfy in order to see this view? • *NULL* -- There are no requirements on the viewer. • *'U'* -- The viewer must be logged in to VIVO. • *'R'* -- The viewer must be logged in, and must be registered as a user of this gadget.
owner_ req	char(1)	What requirements must the owner of this page satisfy in order to see this view? • *NULL* -- There are no requirements on the owner of the page. • *'R'* -- The owner of the page must choose to display this gadget to the public. • *'S'* -- The viewer must be the owner of the page being viewed.
page	varchar (50)	What page does this rule apply to? A single gadget might have several views, but no more than one view per page. Recognized values are • *individual* -- The profile page of an individual, when it is not in "edit" mode. This applies when the viewer is not logged in, or does not have the right to edit the profile page. • *individual-EDIT-MODE* -- The profile page of an individual, when it is in "edit" mode. This applies when the viewer is logged in as an administrator or other privileged user, or as the owner of the profile page. • *search* -- The search results page. • *gadgetDetails* -- A page that contains only the selected gadget. This usually occurs when the user clicks on an icon or a link that expands the gadget to full-page mode.
view	varchar (50)	What is the view-mode of the gadget? These are defined as part of the OpenSocial standards. • *profile* -- The "standard" view, commonly used on the profile page of an individual • *small* -- The "condensed" view. • *home* -- The view which allows the user to change the gadget's settings. • *canvas* -- The "expanded", commonly used when the gadget is the only thing on a page.
closed _width	int(11)	How wide is the gadget when it is closed? (in pixels)

open_ width	int(11)	How wide is the gadget when it is open? (in pixels)
start_cl osed	tinyint (1)	When the page is first loaded, is the gadget open or closed (1 = closed, 0 = open)
chrome Id	varchar (50)	The gadget will be displayed on the page inside a <div> with this id. Note: the page must contain this <div> and its contents will be replaced with this gadget (or gadgets).
display _order	int(11)	If more than one gadget has the same chromeId, they will be displayed in order by this field.

Additional Considerations

Some things to be aware of when working with OpenSocial gadgets.

Re-running the deployment script

The OpenSocial framework relies on several of the settings in the *build.properties* and *runtime.properties* files, in addition to the ones that are explicitly linked to it.

Each time you change the settings in *build.properties* or *runtime.properties*, you should re-deploy the framework with

```
ant orng
```

Resetting the gadget cache

For efficiency, VIVO reads the gadget configuration only when it starts up. VIVO keeps a copy of the database tables in memory, for efficiency.

This means that if you change the gadget configuration in the database tables, you must either tell VIVO to read the tables again. Direct your browser to the orng/clearcache page within VIVO. For example,

```
http://localhost:8080/vivo/orng/clearcache
```

VIVO will re-read the gadget configuration, and display the VIVO home page.

You can achieve the same effect by restarting VIVO.

Issues with Linked Open Data

TBD

Disabling the OpenSocial gadgets

If you decide not to use OpenSocial gadgets in your VIVO installation, there are several ways to deactivate them, depending on how firm your decision is, and how thorough you wish to be.

Disable the gadgets

You can disable any or all of the installed gadgets by setting the *enabled* flag in *orng_apps* to zero (see section II.i. The *orng_apps* database table).

To make this change take effect, restart Tomcat, or clear the OpenSocial cache (see section III.ii. Resetting the gadget cache).

Disable the connection

Disabling the gadgets, as above, will remove essentially all of the OpenSocial processing within VIVO. To remove the remainder of it, you can disable the connection between VIVO and the OpenSocial service. Do this by removing or commenting the *OpenSocial* properties in *build.properties* (see section I.iv. Configure VIVO).

To make this change take effect, re-deploy VIVO and restart Tomcat.

Remove the OpenSocial webapp from Tomcat

Disabling the connection, as above, will remove all of the OpenSocial processing from your VIVO requests. However, you may still see that Tomcat takes longer to start up, and requires more memory.

To remove the OpenSocial webapp from Tomcat,

- stop Tomcat;
- in the [tomcat]/webapps directory, delete shindigorng.war and the shindigorng sub-directory;

- start Tomcat.

Clean up the remnants

To remove all traces of OpenSocial from your VIVO installation, you should take the steps outlined above, and also:

- Remove the *orng* tables from your MySQL database.
- Remove the changes to Tomcat by restoring your *setenv* file to its previous state.
- Remove the *shindig* directory and subdirectories from your VIVO home directory.

These steps will have no appreciable effect on the operation of VIVO, Tomcat, or MySQL. However, if these artifacts are not removed they could be a source of puzzlement for future VIVO maintainers.