



Need for OpenSocial standards
Technical White Paper

Table of Contents

1	SYNOPSIS	3
2	KEYWORDS / CATEGORY	3
3	PROBLEM STATEMENT	3
4	ANALYSIS OF OBSERVATIONS	4
5	PROPOSED SOLUTION	4
5.1	ARCHITECTURAL CONSIDERATIONS	4
5.2	ADVANTAGES OF THE PROPOSED APPROACH.....	8
5.3	PORTABILITY OF THE SOLUTION.....	8
6	CONCLUSIONS	9

1 Synopsis

- The legacy social networking sites use its own container and content model
- This white paper broadly covers the necessity of Open Social
- It also covers the comparison between OpenSocial and the proprietary/legacy social networking sites like Facebook
- It also try to provide deep analysis of the social containers along with their maturity level.

2 Keywords / Category

- Social networking
- Open social
- Google open social
- Orkut apps
- iGoogle gadgets
- MySpace widgets
- Gadgets
- Widgets
- Open social gadgets
- Facebook widgets
- SNS (Social Networking Sites)

3 Problem Statement

- If there is one lesson that the Internet and the Web have taught us, it is the value of interoperability, i.e., the ability to connect systems and software through an open standard protocols. For example, at least a dozen early proprietary email systems were eventually superseded by the open Internet SMTP (Simple Mail Transport Protocol). The same happened with the open World Wide Web displacing proprietary hypertext systems
- There is an enormous incentive to find ways to interconnect [social] networks, since the members of each network can access a much larger set of potential transaction partners. This is not possible with privately held social networks like Facebook
- Missing interoperability among social networks

- Non standard functionalities
- Maturity level issues among social containers

4 Analysis of Observations

- A social network is a social structure made of individuals (or organizations) called "nodes," which are tied (connected) by one or more specific types of interdependency, such as friendship, kinship, financial exchange, dislike, sexual relationships or relationships of beliefs and knowledge or prestige
- Social networking applications leverage the social graph
- They should have ability to scale to other social containers
- There should be a common standard across social containers
- Big players like Yahoo and Google are stepping away to assure "neutrality" and standardization also
- They should have ability to adapt to other social containers without drastic changes
- Standardization can bring rapid growth of OpenSocial networks and agile development techniques.

5 Proposed Solution

5.1 Architectural Considerations

OpenSocial can bring forth standardization in the evolution of Social containers and can forge ahead as a real standard.

OpenSocial:

OpenSocial is a set of common application programming interfaces (APIs) for web-based social network applications, developed by Google along with MySpace and a number of other social networks. It was released on November 1, 2007. Applications implementing the OpenSocial APIs will be interoperable with any social network system that supports them, including features on sites such as iGoogle, Google Wave, Hi5.com, MySpace, orkut, Netlog, Sonico.com, Friendster, Ning and Yahoo.

A container is said to be OpenSocial supportive, if it supports gadgets.* API and opensocial.* API. These APIs are initiated by Google and developed by OpenSocial consortium. The complete reference of gadgets.* API is hosted at <http://code.google.com/apis/gadgets/>. The complete reference of opensocial.* API is available at <http://wiki.opensocial.org/>.

Google launched Open social to spread social applications across the web. Open social is a set of common APIs for building social applications for developers of social applications and for websites that want to add social features.

Google made OpenSocial as an open source project and invited people for App development, Container development and spec list refinement.

Cross-container Development:

There are millions of people using social networking sites, which support OpenSocial apps. To take advantage of the distribution that OpenSocial provides, you will want your app to run on more than just one site. There are a few things to consider when writing or maintaining an app that works across containers, which includes submission process, layout, implementation differences, policies and language support.

Most containers require that you submit your app to be reviewed before users can find your app in the directory. Each container has a different process for reviewing (and re-reviewing) your app. Check out the container-specific sections below to find out what to expect when submitting your app on a particular container.

OpenSocial defines four views - profile, canvas, preview, home. Containers tend to support a subset of these four views. For example, orkut supports a profile and canvas view, but not a preview or home view. The same view can have different sizes across containers.

OpenSocial defines an API - it is up to the containers to implement that API, which can lead to some discrepancies in the functions supported or fields your app will have access to. Some containers also support special extensions, like access to photos, which you can't expect to work on all containers.

Social networking is popular all over the world and these sites tend to be popular in specific geographic regions. The more languages your app supports, the more users you can reach. OpenSocial lets you manage translations using *message bundles*, or XML files that contain the strings used in your app.

Container development:

For container developers, it introduced Apache Shindig. It is an open source project that helps you start hosting OpenSocial apps quickly by providing the code to render gadgets, proxy requests, and handle REST and RPC requests.

App development:

For app developers, Opensocial introduces [Social Application Tutorial](#), [Lightweight JavaScript API](#), [OpenSocial Templates](#) and [OSML Tags](#).

Spec list refinement:

The OpenSocial spec is always evolving and participation in the [process](#) is open to anyone. The current specification is located at <http://www.opensocial.org/page/specs-1>

REST protocol:

Representational state transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web. Conforming to the REST constraints is often referred to as being 'Restful'. The REST protocol is located at: <http://www.opensocial.org/Technical-Resources/opensocial-spec-v081/restful-protocol.html>

RPC protocol:

This proposal defines an RPC alternative to the [RESTful API](#) specification. The intent is to support the same data and operations as the Restful API in a form that is more natural for the JSON data format. Any batch of RPC should be programatically convertible to a sequence or batch of Restful calls and maintains the same semantics, any exception to this will be called out explicitly. The RPC protocol is located at: <http://www.opensocial.org/Technical-Resources/opensocial-spec-v081/rpc-protocol.html>

Containers and their support:**iGoogle Container:**

iGoogle gadget specifications was adopted to develop the OpenSocial specification. Google is one of the founder members of OpenSocial API; iGoogle supports version 0.8 and version 0.9 of OpenSocial specifications. It also supports legacy application development APIs that is `_IG_*`. This container supports few handy javascript functions such as `_trim`, `_esc`, `_hesc`, `_gel` and so on. It uses client side Restful API to access open social data. As of now, it does not support server side API. It supports gadgets with html content as well as URL type content. It supports localization of content (multi-language support). It supports user preferences.

MySpace Container:

MySpace container supports opensocial version 0.7 and version 0.8. It uses client side RESTful API to access open social data. This container has its own API reference along with OpenSocial features for creating and sharing Albums, Videos, and Photos. It also supports blogging, bulletin, sending private messages and so on. It has more person fields such as MOOD, ZODIAC_SIGN under container specific namespace (MyOpenSpace). It does not support localization of content (No multi language support as of now). It does not support user preferences. It has limited support in home and profile view of the application. It uses templates for content posting and creating activities. More details are found in <http://wiki.developer.myspace.com>.

Yahoo Container:

Yahoo Application Platform (YAP) v 1.6 supports the 0.8 version of the OpenSocial JavaScript APIs and the 0.8.1 version of the OpenSocial RESTful APIs. However, YAP

v1.6 does not support the OpenSocial Gadget XML definitions. In home view, it does not support Javascript. It supports YML tags, which is proprietary tags for Yahoo applications. Recently Yahoo joined OpenSocial, it will support OpenSocial features and OpenSocial XML definitions soon as mentioned in yahoo developer network site. It supports server side support including PHP, Python and Java for open social features. It provides PHP SDK for developing open social applications.

Wave Container:

Google wave container supports OpenSocial version 0.8 and version 0.9. It also adds wave specific API along with OpenSocial supports. It supports localization of content (multi-language support). It uses client side RESTful API to access open social data.

Facebook Container:

Facebook is an open source software service, which provides services like Social graphs, User profiles, Messaging, pages and Information aggregate. Facebook until now is not part of opensocial consortium and has its own platform. It widely uses FBML in place of XML used by Google gadgets. Facebook do not directly support Google gadgets. However, a gadget can be turned into a platform independent widget which can be installed on Facebook platform. It uses proprietary languages such as FBML, FQL, and FBJS.

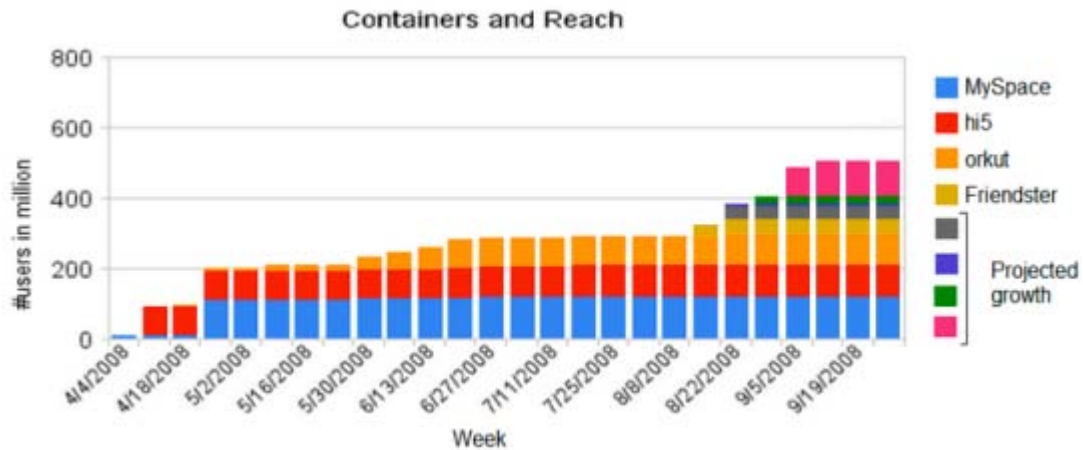
OpenSocial versus Proprietary applications:

- OpenSocial applications will have diverging look-and-feel, from each other and from the containers. This is because the containers do not provide common elements to blend the application into the container.
- OpenSocial applications may not be vertically resizable, since they will exist in an iframe. However, Google has an API For resizing that some or all of the networks may implement
- Proprietary apps has additional API functionality that is not present in OpenSocial
- The Proprietary apps API are generally server oriented, whereas the OpenSocial gadgets API is generally client-side JavaScript oriented authentication.

With the above observation, the OpenSocial is the proposed choice for developing social applications. Because of some implementation differences across containers we may not get completely "Write once, run anywhere" applications, but with some minor changes we can get it right now.

5.2 Advantages of the Proposed Approach

(Ref: <http://news.cnet.com/otl/?keyword=OpenSocial>)



As per David Glazer, Director of Engineering at Google, OpenSocial has to be made easier for developers to build applications, reach users, and make money. The platform has gone a long way in the right direction.

OpenSocial container can support diverse vendor applications like IM (Instant Messaging), mailing, photo sharing, and video sharing and so on. Standardization can lead to neutrality and quality of content available and can reduce spamming. Some of the future improvements to the OpenSocial platform will include better development tools (Visual Studio-like tool to speed development), payment platforms, analytics, cross-container portability, and mobile-application support.

The OpenSocial APIs allow developers to create apps that access a social network's friends and update feeds without modification for compliant platforms.

OpenSocial makes good economic sense too. More and diverse applications can bring large volume of traffic leading to better revenue generation.

5.3 Portability of the Solution

OpenSocial provides a useful piece of functionality, solving a developer problem by allowing applications developed with the APIs to run on different services without drastic modification. A photo-sharing application could tap into the social graphs of Orkut, Bebo, MySpace, Ning, or other services without any code changes.

It does not mean that the code written for one container will work in another container without any changes, but will work with minor changes, such as container specific functionality replacement.

6 Conclusions

Only a handful of OpenSocial "platforms," or supporting social networks, have gone live so far and already each is "customizing" in ways that are not interoperable. Some of these networks have user data fields that are not available on other networks, other are extending OpenSocial in ways that could be widely supported but aren't part of the platform right now.

The porting an app from one network to another is today "a matter of hours instead of weeks or months" like it would be if it were not for OpenSocial. As more and more containers go live, though, it could end up taking weeks to make something truly interoperable. OpenSocial could very likely just end up being a network of sites that happen to give people permission to build apps instead of offering meaningful advantages for developers compared to stand-alone sites. Cross site-data, portability would be the one remaining advantage. The vision of free flowing user information, with sophisticated and powerful granular controls, data-centric features and free love for all seems less central to OpenSocial every day.