OPENi Graph API Framework in the Social Standards Landscape

Iosif Alvertis
National Technical University of Athens, Greece
alvertisjo@epu.ntua.gr

Timotheos Kastrinogiannis[1]
VELTI SA
tkastrinogiannis@velti.com

Fenareti Lampathaki
National Technical University of Athens, Greece
flamp@epu.ntua.gr

Michael Petychakis
National Technical University of Athens, Greece
mpetyx@epu.ntua.gr

Meletis Margaritis[1]
VELTI SA
mmargaritis@velti.com

Theodoros Michalareas[1]
VELTI SA,
tmichalareas@velti.com

Eric Robson
TSSG, Waterford Institute of Technology, Ireland
erobson@tssg.org

Robert Kleinfeld
Fraunhofer FOKUS, Germany
robert.kleinfeld@fokus.fraunhofer.de

Abstract. Today, the proliferation of social-enabled applications has disrupted the traditional enterprise mobile and desktop application value chain with three major phenomena: a) the migration of part of the generated value outside applications and onto social networks where social graph connectivity matters and there is limited support for context, b) the migration of part of the control of the use of these social data to multiple social networks outside the realm of control of the enterprise, the application developer and sometimes even of the end-consumer, c) the increasing number of old and new social networks that try to cover from a niche area (like Instagram) to the general public (like Facebook). This position paper presents the OPENi approach to a social web architecture that integrates data privacy and application developers data control in its design from its core and provides both: a) a social graph-rooted, OPENi Graph API framework that builds on existing social standards and allows developers to easily source into their applications a broad spectrum of existing cloud-based, user-centric functionality and content without hard-coding integration to specific social web service providers APIs and while retaining part of the generated social web data, b) a Context API framework that allows to build context to the developed applications and extend the social graph connectivity model so that consumers can express both weight and connectivity of their connection to other parts of the social graph. The paper also puts particular emphasis on discussing the key decisions, the challenges and the anticipated added value of the proposed approach that opens new perspectives on mobile and desktop social web enterprise applications development.

Keywords: Graph API, Context API, Social Web, Social Graph, Activity Streams, User-centric API Framework, Data Privacy

1. Introduction

We live in an era where internet tends to digitalize every part of our life; products, services, objects, social interaction have a digital representation or at least a unique URI to be addressable, can be retrieved through a web resource. The burst of web 2.0 with services like Netflix, Airbnb, Amazon.com or Pandora have transformed traditional industries and created a number of new application development ecosystems based on platform APIs with their own stakeholders (application developers, app stores, broker services, ad.networks). At the same time, social services have moved our digital life to the cloud, where we store our daily digital experiences; services like Facebook, Twitter, Blogger, RunKeeper and FitBit are used daily as part of our social activities allowing us to share our visual experiences, our thoughts, even our sleeping and exercising activities.

In this rapidly changing technological landscape, the problems the social enabled application development industry is facing relate to:

a) technological and cost/complexity challenges posed by the fragmentation of the cloud-based and social services industry

b) developing innovative features which can generate additional value once the social graph connectivity model is exploited

These challenges affect all aspects of the service delivery chain, from the consumer to the application developer, to the service provider, e.g.:

- The consumer's data gets fragmented and very often duplicated over various service providers. Additionally, most popular services today operate as "data silos", allowing users minimum or no flexibility in making any data they contribute available outside the provider's boundaries.
- As consumers demand applications that appear to seamlessly access and interoperate with multiple service providers, the application developers are struggling to deliver the additional complexity in the time frame and price point the consumers expect, while most of the value of social web generated data remains with social networks.
- Online and traditional 'bricks and mortar' service providers aim at attracting and incentivising new consumers to their services through desktop and mobile enterprise apps, yet as these applications are often released for little or no cost, the

[1] *Statements of facts and opinions expressed in this paper are made on the responsibility of the authors alone and do not represent the views of Velti.*

development teams for these applications are viewed as cost centres. This cost is compounded by the lack of open standards for cloud-based social web services which increases the development cost of creating be-spoke solutions from the ground up.

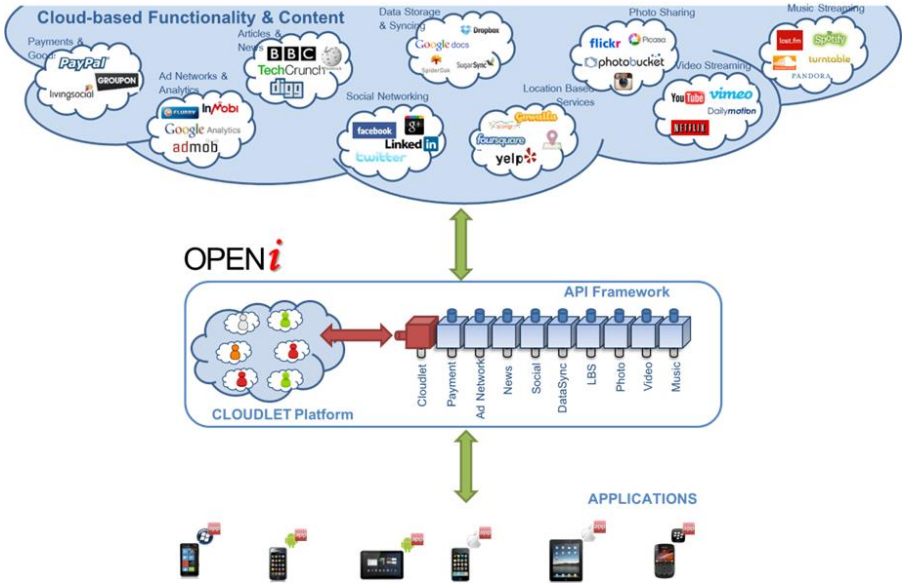

Figure 1-1: OPENi Approach

Furthermore while the social graph connectivity model has fuelled the initial wave of innovation for social web applications, as social networks reach their potential there is a need for a new innovative model of the social graph that can drive more innovation for the enterprise and the consumer and generate interest and more relevant applications.

In this context, the OPENi project [1] aims at delivering a consumer-centric, open source mobile cloud applications solution that will be the catalyst for mobile & desktop application innovation advancement. One of the key project outcomes is a common API framework that will allow Developers to build applications that integrate dynamically a broad spectrum of cloud-based services and at the same time can access end-user OPENi cloudlets (personal space in the cloud) to store and retrieve content and context data, while the end-user retains full control.

The position paper at hand presents the OPENi work-in-progress towards a Graph and Context API Framework that builds in a direct or implicit manner on top of existing social standards, like ActivityStreams and OpenSocial. In the next sections, the generic rules and the structure of the API Framework are explained while the role of the Context API is further elaborated.

2. A Graph API Framework for OPENi

Typically, a Graph API consists of objects, aggregations and connections, while objects can be connected with other secondary objects. Additionally, every object and aggregation is uniquely addressable through an id existing on the main path of an API. Facebook paved the way towards a Graph API design [2] based on objects with a unique index and a simplified dictionary for recurring, connected methods, while others, like RunKeeper, followed up with a similar approach. The basic advantages of a Graph approach are (a) the extendibility of the schema without affecting the main structure in the API and (b) a human readable, easily explorable API that requires less documentation. The main disadvantage is that it requires an extensive caching system, with the frequently retrieved objects, in order to improve the performance of the indexing system it uses, but also a complete redesign for existing data models to be extracted as Graph APIs.

The main advantage of the Graph API, a path agnostic way to model and access resources especially when the developer has an id for an object without knowing the path to that resource, **poses a great challenge: how to create a common dictionary of resource names that can be be used to avoid duplicates and allow for API interoperability and readability.** For that reason OPENi Graph API builds on top of the Activity Streams schema [3], which allows an API to have common names for objects across social media platforms. Nevertheless, there are objects that are not available in social media services at least through their basic APIs (i.e. the Facebook Graph API may be extended with developers' specific objects, like food, running performance), or through an Activity Streams approach.

To that end, an extensive API analysis over 221 different services was performed to identify most popular services through different categories, based on multiple resources through a well-defined approach based on the hints provided in developers' portals (i.e. Programmable Web) and in market predictions by analysts and in accordance with the OPENi project needs and requirements [4]. Through this analysis, 23 different services have been prioritized and analyzed, in order to identify objects; This driven-by-the-market analysis, allowed the identification of various objects that could not be found in any standards, while they

have also reached a level of API stability and common acceptance for names. The result of this analysis concluded in eight (8) different API categories that combine cloud-based functionalities in a set of functionalities that are similar to Phone SDKs and existing phone applications or functionalities:

- **Activity API** referring to a social, health, behavioural activity log as reflected in multiple cloud-based services ranging from Social, Photo and Video Sharing to Health and Location-based Services. In essence, it includes all the social and personal activities of a user, and is related with the logging activity of a device. *Relevant Categories***:** Gaming, Health. Location-based Services, Music, Photo, Shopping, Social, Video
- **Advertising and Analytics API:** enabling the collection, aggregation and analysis of end-user/customers needs, interests and preferences based on a) their interaction with advertising/marketing content, b) application behavioral data and c) social network interactions towards enabling personalized advertising services with enhanced end user added value. *Relevant Categories***:** Analytics, Advertising
- **Location API** enabling location awareness through Checkins, Direction, Events, Reviews and Tips. It is a strong contextual API with location, which can be extracted by a GPS sensor and can be mapped on a map. *Relevant Categories***:** Mapping, Location-based Services, Social, Music
- **Media API** bringing together photo, music and video sharing services with file transfer and syncing functionalities. It is related with the Gallery Application and the file system of a device. *Relevant Categories***:** File Transfer & Syncing, Location-based Services, Music, News, Photo, Social, Video
- **Products & Services API** embracing Payments and Shopping services. This special case of commercialization-relevant objects, requires a strong API with transactional capabilities, and enhanced security. It is related with the store application of a Phone. *Relevant Categories***:** Location-based Services, Payments, Shopping.
- **Profiles API** extrapolating information about persons based on Analytics, Advertising, Contacts, Gaming, Health, Location-based Services, Messaging & Chat, Music, Social, etc. services. Typically, people can be represented in various ways, with multiple profiles. This API brings all that information together: avatars, profiles, contacts and accounts, interrelating different profiles, in different services. It is directly related to the Agenda application of a smartphone. *Relevant Categories***:** Analytics, Advertising, Contacts, Gaming, Health, Location-based Services, Messaging & Chat, Messaging & Chat, Music, Social
- **Search API** in order to find and retrieve information from the cloud-based services. It brings together existing search engines with distributed search functionalities among various platforms, and is related to the search functionality found in a Phone OS. *Relevant Categories***:** Advertising, Contacts, File Transfer, Games, Health, Location-based Services, News, Photo, Places

Moreover, the challenge of designing a universal, graph API framework to control multiple resources around the web requires flexible handling of different and multiple profiles through different services. Again, a Graph API can handle different accounts under a single user, based on the account id. Thus, the user is uniquely addressable, an object connected with multiple accounts that represent the presence of the user in various services. By querying a user object, all the account objects can be discovered, and then be used through the API to query for additional information. Nevertheless, the central, hierarchical structure among user objects and account objects, allows developers to ask for an aggregation of objects from various accounts or even a multi-post process towards different services. Thus, a HTTP GET request to the path "*API_path/user_id/object_name*" will return a synthesis of same objects from multiple services, while a HTTP POST request with the account ids where the message is directed will post the message to multiple services. Additionally, objects are bound to every account, thus a GET or POST to the path "*API_path/account_id/object_name*" should return objects specific for a service, and no further parameters filtering is needed. In order to handle possible content from multiple services and keep a unified response format, every returned object includes the "actor" of the activity, an account-object, under the activity schema defined by Activity Streams. Nevertheless, the "verb" of the activity is omitted, as the used HTTP method with the requested path is adequate to give semantic meaning to a described activity.

In order to create a unified API framework, a minimum set of properties has been defined, to facilitate the documentation, the data modeling process and the easy extendibility of the API. The predefined properties, shown in Table 1: Minimum set of propertiesTable 1 give all the available information to identify the returned object; the id of the object may be used to explore further connections, the URL to retrieve resources from the service, the object-type property may inform a developer that has requested for an object based on an id for the type of the response, the "service" property facilitates the developer to easily identify and annotate the source of the object, while the "from" object gives the required information for the account profile that has generated the content. Additionally, a group of predefined sets of properties, like time properties, location properties, duration properties or object profiles (i.e. file, place, organization, person) has been described, and can be used from different objects when needed.

Property	Description	Type
Id	The id of the object, as it is uniquely defined inside the OPENi platform.	string
url	The web resource of the referenced object, can be an html page or a media file	string
object_type	The predefined name of the object, either by the platform or by the developer. e.g. "photo", "video"	string
service	The service where this object is located, e.g. "OPENi", "Facebook", "Flickr", "Foursquare"	string

Property	Description	Type
from	The owner of the specific object, if available (e.g. a user object has not such property)	object

Table 1: Minimum set of properties

On the other hand, no matter how deep analysis has been done, and a semantically powerful dictionary has been used, there will be continuously new objects introduced through cloud-based services, new services will be connected with the framework. With the extensibility and maintainability challenges coming to the foreground, such an API framework needs to be extended with objects, connections and activities, while new properties may apply to the objects. For that reason, an extension to the basic objects may apply to the Graph API framework; in any case, if it is compliant to the basic rules, there is no violation or distortion of the models. The rules to override objects, reuse property sets and extend the API with objects and activities, is work in progress towards the creation of a community-driven API Designer platform.

3. The role of Context API

As examined in the previous section, OPENi Graph API can be used by businesses & applications to collect information about consumers/customers/end-users interactions/connectivity in the social graph correlating **actors** (**who), objects (what)** and **activities**. The value of the OPENi Graph API comes from providing enhanced flexibility and efficiencies into developing such applications (one API can be used to support social graphs that span across different networks) and from ensuring that Data Privacy controls are integrated in proposed framework with the end-consumer having control over his data (**OPENi cloudlet**).

Context: the new wave of innovation in social applications

In this section, the proposed OPENi Context API that aims at enabling the next wave of innovative applications in social networks is described. The main purpose of the OPENi context API is to allow businesses to use /build social-oriented applications where users are not restricted to describe **only** whether they are connected or NOT with another member of the social graph but also the **weight and the context** of that connection.

Some example cases from the OPENi requirements analysis, where this information can generate value, are the following:

a) **MyLife:** With regard to photo functionalities, two close family members are connected to share photos in a social network if one of them posts a photo it has increased value for the other family member; From a medical perspective, a patient is connected to his doctor in a social network, their connection has a trusted context, if a doctor posts an information article related to the patient condition it has increased value to that user.

b) **Personalized advertising:** where end-users contextual information, related to location, time, needs, mood and profile along with social-aware contextual information related to friends, recommendations and trend can be combined to order to maximize mobile marketing targeting efficiency and new advanced metrics of brand/advertising performance.

c) **Personalised In-store Shopping Experience:** A user enters an electronics retail store with a Personal Shopping Assistant (PSA) app on their smart phone. The PSA detects what store the user is into present the user with a set of personalised purchase recommendations such as what my 'shopping' friends bought.

This new type of information which is NOT currently captured in social networks can be the main driver for the next wave of innovation in social applications over the web allowing the users/application to express the context of their communication and businesses/social applications providers to use this information.

The Context API enables developers to build **context-aware applications and services that are aware of user's social graph and actions context, unlocking the potential of combining contextual information from one application with another** and thus, facilitating the goal of having rich-content, pervasive and added value social applications. Additionally, the context API deals with the issue of data privacy by allowing the end-users to have full control over the use of their contextual data. The issue of data privacy [7]with contextual data is even more pressing since such data reveal much more about user preferences and behaviors and need to be made available to social applications only on an opt-in/opt-out basis and under clear terms of service.

Social-aware Contextual Data – The OPENi Vision

Contextual property/data allows to further collect, manage and exploit information about why a particular action occurred as well as the context within which a particular activity was performed. In general, the environment, time, location, social context metadata provide valuable information on where, how, and when the interaction between the user and an application is exactly taking place. The exact entities represented in contextual information can be of a very dynamic nature, affecting application consumers' expectations and experience. Towards this direction standardization initiatives, such as W3C Ubiquitous Web Domain Delivery Context Ontology specification (DCO) [5], Activity Stream Working Group [3] and Intel Cloud Service Platform [6] aim at creating formal models of the characteristics of the environment in which devices, applications, and services are operating towards creating a proper and valuable Context API framework. Complementary to the above, the Customer Experience Digital Data Community Group [7], aims at addressing the existing problem of vendor-specific digital data objects, that is data objects tracked and captured by different vendors with vendor specific format requirements and code assignments, creating design complexity and

vendor dependency, via the definition of a vendor-agnostic Context API framework. Finally, several dedicated efforts related to social API exist aiming at the modeling of social networks driven user interaction under a common API framework such as Mozilla Social API [8], Yahoo! Social API [9] and Drupal Social API [10].

Moving beyond the current scope of contextual data, i.e., correlating end-user **real life** situations with proper metadata, the OPENi Context API offers extension to the latter for incorporating user's **digital life** situations, activities and history, as a result of their interactions with fixed and mobile social services and application. Web 2.0 popular applications, from blogs (e.g. Wordpress), feeds (e.g. Twitter), social tagging systems (e.g. Flickr) to social networks (e.g. Facebook), has led to a new era of creating and sharing content, socializing within virtual communities and exploring new information spaces. Moreover, mobile apps world is already exploiting social networking services either by adapting existing services and applications to mobile devices or by developing specific applications, which also integrate additional functionalities available to mobile phones like the usage of camera and GPS receiver. However, current applications either enable access to specific (CBS) services and thus, to fragmented content, or in few cases, allow myopic access to user's actions and user-contributed content within a social networks, revealing the lack and the need of a common **Social-aware Context API framework** that enables developers/customers/enterprises/service provides to access, aggregate and process social interactions (from multiple social CBS) via a common **one interface**.

The **OPENi Context API** framework will enable developers to easily: a) design and integrate existing cloud-based features into their applications, and b) collect, manage, aggregate exploit user generated contextual data originated from users interactions/situations from both real-life environment (end users - mobile applications interface) and digital social life (end users - social networks (CBS) interface), focusing primarily on what they require in order to enhance their applications user experience and not caring about how the implementation parameters are met. This approach will reduce fragmentation, development and maintenance effort and time to deliver through a purely web compatible, platform-independent framework.

The OPENi Social-aware Context API goal is to provide services that will take advantage of contextual information and functionality otherwise only inherently available through implementing dedicated application-driven context data APIs and/or social-media APIs. To ensure interoperability between all social media, development of these services will be based on REST protocol that will allow heterogeneous data to be collected and exposed, so as to ensure the security of users' data and the anonymity of all users. Such services are:

- Accessing multiple social network accounts dynamically using the OPENi Graph API and corresponding social-aware contextual data using Context API;
- Accessing cross-applications contextual data in order to facilitate the design and functionality of context-aware OPENi applications;
- Retrieving location based services dynamically by multiple providers based on the current location and availability of suitable recommendation;
- Allowing OPENi application developers to create application- (business case-) driven contextual properties and manage them;
- Retaining, storing, editing and aggregating cross-applications and cross CBS contextual properties over all OPENi objects either in a static (predefined/structured) format or as extra attributes, extension to objects attribute.
- Accessing multiple multimedia providers dynamically and streaming/accessing content e.g., music, video and photo.

4. Conclusions

The OPENi Graph & Context API proposes a paradigm shift to user-centric and developer-centric design of social web applications (mobile and desktop). The proposed framework brings to social web application developers efficiencies in developing their applications, control to consumers and developers over a larger part of the generated social web data and a set of requirements for a new innovative social graph model that allows consumers to express the weight and context of their connection to the social graph.

In particular, the **OPENi Graph API** framework has been based on the analysis of multiple social and web 2.0 services, like Facebook and RunKeeper Graph API, and overcomes a number of challenges: (a) the description of objects and functionalities from different services and industries, (b) the description of a user hierarchy that will allow integration of multiple services and accounts under the same API rules and (c) the definition of a set of properties that can give a complete overview of the resource, in every way the resource has been accessed (i.e. through a direct call to the object id or through a call to an aggregation under a user or account object). The **OPENi Context API** framework builds on previous W3C work but relates context to the social graph and data privacy thus allowing the creation of a new set of context-aware social web applications harnessing the new type of information available by the proliferation of the mobile device.

Future steps along the proposed approach include: (a) the finalization of the specification of a community-driven API platform that tackles the issue of the API Framework sustainability and Context information modeling; and (b) the prototype implementation of these APIs (Graph/Context) and their consideration for standardization as part of W3C work on social standards.

Acknowledgements

This work has been created closely to research activities during the EU-funded project OPENi (Open-Source, Web-Based, Framework for Integrating Applications with Social Media Services and Personal Cloudlets, Contract No: FP7-ICT-317883).

References

1. OPENi (2013). http://www.openi-ict.eu/
2. Facebook Graph API. (8 July 2013). https://developers.facebook.com/docs/reference/api/
3. Acticity Streams. Activity Base Schema (Draft). (8 July 2013). http://activitystrea.ms/specs/json/schema/activity-schema.html
4. OPENI API framework (PART I): studying the landscape of cloud-based services. (30 April 2013). http://www.openi-ict.eu/openi-api-framework-part-i-studying-the-landscape-of-cloud-based-services/
5. W3C Ubiquitous Web Domain Delivery Context Ontology. http://www.w3.org/TR/2009/WD-dcontology-20090616/
6. Intel's Cloud Based Platform, http://software.intel.com/cloudservicesplatform/documentation/intel-cloud-services-platform-beta-context-services-rest-api-reference
7. W3C, Customer Experience Digital Data Community Group http://www.w3.org/community/custexpdata/
8. Mozilla Social API: https://developer.mozilla.org/el/docs/Social_API
9. Yahoo! Social API: http://developer.yahoo.com/social/
10. Drupal Social API: https://drupal.org/project/social_api