

White Paper

The Emergence and Empowerment of Web 2.0

Dialogic White Papers

This white paper is brought to you by Dialogic as part of a continuing series on important communications topics.

Dialogic Corporation is a leading provider of world-class technologies based on open standards that enable innovative mobile, video, IP, and TDM solutions for Network Service Providers and Enterprise Communication Networks. Dialogic's customers and partners rely on its leading-edge, flexible components to rapidly deploy value-added solutions around the world.

Executive Summary

The emergence of Web 2.0 — the term associated with the web’s transition from individual web sites to a “platform” — has opened up many service creation possibilities to deliver new media and communication services for a new brand of service provider such as Google and Yahoo. This emergence has also empowered traditional service providers to seek new services that they can offer in addition to the “data pipe” so long associated with communications services.

This white paper discusses the current understanding of Web 2.0, the collaborative nature of web users who drive the requirements for new web services and open standards-based APIs, and why delivering the right APIs, supported by the right development environments is paramount to empowering media communications in a Web 2.0 environment.

Table of Contents

Introduction	2
Understanding Web 2.0	2
A Focus on the User as Content Creator	2
Web 2.0 Communication Services	3
The Service Provider’s Role	3
Hosted Application Services	3
Empowering Media Communications in “2.0”.	3
Standards-Based APIs.	4
Development Environments	5
Ruby.	5
PHP	6
Delivering the Most Useful Approach	6
Summary	6
References	7
For More Information	7

Introduction

Web 2.0 is the term associated with the transition of the World Wide Web from a collection of individual web sites to an emerging “platform” in its own right. This emergence is due largely to user collaboration, and these users have been the driving force for the requirements for new services. Many of the technologies and approaches discussed in this paper are evolving, and the purpose of this paper is to provide a general introduction to and thoughts on delivering media services and current trends in a Web 2.0 environment.

This white paper discusses the current understanding of Web 2.0 and the requirements for enabling service creation and delivery to end users, and covers how the emergence of Web 2.0 has empowered new and traditional service providers and developers to seek new services to offer. This white paper also discusses open-standard APIs and explains why delivering the right APIs, supported by the right development environments is paramount to providing media communications in a Web 2.0 environment.

Understanding Web 2.0

A Focus on the User as Content Creator

Web 2.0 does not yet have a specific definition or a set of technical standards or requirements that one can use to implement to deliver services into this model. Web 2.0 refers to the shift in how the web is currently used and perceived. The web has moved from being a data repository/database of individual web sites to a “platform”. The platform itself is being defined by its users in terms of how they are employing the web and how they wish to make use of it in the future. This shift to a platform provides opportunities for creating services and generating new sources of revenue from these services.

Many Web 2.0 “technology mechanisms” have already entered the mainstream, such as blogs, wikis, podcasts, micro-blogs (such as Twitter), RSS feeds, social networking software (myspace, Facebook), folksonomy, communications (VoIP, video calls, IM), and open APIs. The key focus is on the user — and specifically on the collaboration among users. These collaborators are now empowered to create content and services themselves, and are literally defining the kind of information that they want on the web and what services they want websites to provide. Content owners now share, socialize, network, and engage in e-commerce as they see fit. The user, as seen in Figure 1, is the central focus in Web 2.0.

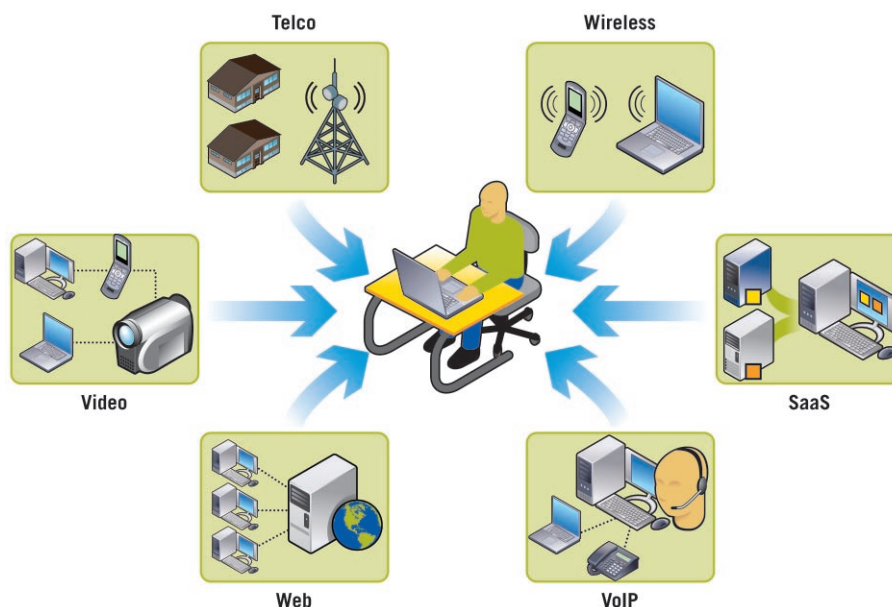


Figure 1. The User is the Central Focus in Web 2.0

Web 2.0 Communication Services

A user can subscribe to multiple services from different mobile, IM, VoIP, and other providers. What becomes interesting is not just the delivery of these services, but the ability to manage “presence information” to ensure that the right service is used at the right time. Many of the services available today relate to a single function, such as IM, or IM to SMS, and VoIP calls. Through social networking, users advertise their connectivity options. For example, people who are designated as “friends” by a user are able to see the communications options that the user has chosen, and can select which one to use. More sophisticated systems/services, which include basic presence, are also starting to become available, where — based on user rules and information — the appropriate communication technique is automatically selected; for example, a service could automatically be enabled to provide an office phone service from 9 am to 5 pm, a cell phone service from 5 pm to 9 pm, no phone services after 9 pm, and so on.

The Service Provider's Role

In delivering services, the ability to quickly construct dynamic web sites/pages/communication systems, including features for instant messaging, blogging, and so on, via tools and “mashups,” means that the main communication techniques are provided by a new breed of service providers, such as Google and Yahoo. The new service providers do not run a physical line into the home, but they can deliver fast, efficient, and inexpensive communication options, once the user is on the web.

Traditional service providers (who do run that line into the home) are no doubt wondering about the direction of service creation and their ability to deliver new and innovative services that their subscribers are willing to pay for. They do not want to be — and cannot afford to be — just a “data pipe.” To avoid being marginalized, they are looking to embrace and deliver new services themselves. These traditional service providers are also looking to balance their current (primarily TDM) business with IP convergence to create and enable services in a predominately IP environment. This new model also means that “data volumes” are being decoupled from “connection volumes,” as can be seen when customers demand faster connections and ready access to high data volume websites such as YouTube and other video and media sites.

The traditional data pipe is also being used to carry VoIP, though not necessarily through the traditional service provider, but instead through other competing providers, such as those enabling popular peer-to-peer communications services. This is driving the need for traditional service providers to be at the forefront and to generate new revenue streams through new services that can be competitive in the Web 2.0 world.

Hosted Application Services

As part of changes in the Web 2.0 environment, the use of hosted applications is on the rise. This means that the enterprise application for a corporation or an enterprise application suite for individuals is running on a hosted service. Using a hosted application removes some overhead for the corporation and can increase affordability for the individual because it has a pay-as-you-go model. Microsoft® may propel hosted application services even further by moving into this area to compete with current front-runners, such as salesforce.com and Google, as part of its “software plus services” strategy.

Larger players may not be alone in driving this market. Other companies can offer services that focus on addressing the multi-layered requirements of the applications. For example, at a lower level, a company might offer “access to capabilities” such as telecommunications through Web 2.0 APIs. This enables the development of services and applications that could deliver even better services by combining or “mashing” multiple capabilities.

What is helping speed the delivery of these services is the use of open standards, such as an XML suite of APIs, which allow the necessary interoperability across services.

Empowering Media Communications in “2.0”

Delivering a rich set of internet media types, while allowing the collaboration and linkage of multiple functions in a service offering, seems to be the answer for traditional service providers. Proprietary APIs are quickly disappearing because they restrict interoperability and mobility for service developers. Open standards-based APIs are gaining traction because they remove barriers and encourage cooperative application development.

APIs are now written in languages for developers comfortable with PHP, Ruby, XML, and other “web” languages, which are

not tied to more traditional telecom APIs based in languages like C/C++. Since these APIs are developed on simpler and more intuitive languages, they are much more suitable for collaborative development and a quick rollout of new applications and services.

Delivering the right APIs, supported by the right development environments, is paramount to empowering media communications in a Web 2.0 environment. In this section, different API types and development environments are discussed.

Standards-Based APIs

To fully embrace Web 2.0 and deliver a rich set of media types to the user, a provider needs to be willing to allow collaboration and the linking of multiple functionality offerings into a service. Proprietary development APIs do not fit well into this model, as they restrict interoperability and mobility for the developers of services. Using open standards-based APIs removes barriers and encourages acceptance. Today's Web developers require APIs developed using Java, HTML, XML, and other "web" APIs, and not with the more traditional C/C++. Using web APIs allows rapid prototyping and deployment of applications and services. However, as of today no defined standards exist in a one-size-fits-all scenario, and a number of methods are used to deliver the required functionality.

The following APIs provide access to media functionality:

- **CCXML** — Call Control eXtensible Markup Language is designed to provide call control support for dialog systems such as VoiceXML.
- **VoiceXML** — Voice eXtensible Markup Language is the W3 standard for specifying interactive voice human-computer dialogues.
- **MSCML** — Media Service Control Markup Language is a protocol used with SIP to enable the delivery of media conference services.
- **MSML** — Media Sessions Markup Language is currently not a formal standard, such as MSCML, but is defined for more general-purpose media server control.

MSCML and MSML are competing APIs, but that situation may soon be resolved when the IETF MediaCTRL working group delivers a new API for Media Server Control. Dialogic is involved in this work.

However, controversy still reigns in this area because some argue that these APIs are not in the least bit Web 2.0-oriented, but are more "Web 1.5-like" in that they are an evolutionary step, but do not deliver the overall broad-based, easy-to-use high levels of abstraction associated with Web 2.0.

Many of the Web 2.0 service applications being delivered today are defined as "RESTful" APIs. REpresentational State Transfer (REST) is not a defined API, but rather a programming style or methodology. REST identifies, within an HTTP context, how resources are defined and addressed, generally through stateless transactions such as PUT, GET, POST, and DELETE. Use of RESTful APIs allows for simple system distribution, and resources may easily contain further sub-resources held elsewhere. A good example of a RESTful-based API is the Facebook API. This API allows access to Facebook so that users can add social context to the application with profiles, friends, photos, and event data. Other social networking sites have similar APIs.

A RESTful API does not, however, map streaming data (voice, video, and others.) easily and needs to manage call control; this type of API is not media-related and does not contain call control functionality. Thus, when using this API to keep the stateless interface for status information RESTful, the options are either to force state management onto the application, or to keep state management as much as possible internal to the service. In keeping it internal to the service, programming becomes easier with higher API levels more along the lines of a "MakeCall" that has a success/failure indication, rather than a "MakeCall" that has indications that the remote is ringing. Although this does not remove the need for the application to understand the status, it does remove the need for the application to specifically manage the tracking of the information.

It is worthwhile to note what Google is doing in terms of its Google Talk service. The Google Talk service is built on the following open-source protocols [GoogleTalk]:

- **XMPP** — eXtensible Messaging and Presence Protocol. Now an IETF standard for instant messaging, XMPP was originally called Jabber, and the XMPP Enhancement Proposals (XEPs) were previously called Jabber Enhancement Proposals (JEPs).

XMPP defines a client and server architecture, delivering instant messaging and presence. It is not difficult for developers to take open source code and to connect their own XMPP server to the web.

- **Jingle** — A family of XMPP extensions that make it possible to initiate and maintain peer-to-peer sessions. Designed in a modular way, developers can easily add support for multimedia session types other than voice chat, such as video and file sharing.

AOL confirmed web reports in January 2008 that it is working with XMPP and has a test server up and running [Jensen]. Microsoft and Yahoo have reportedly not adopted XMPP yet, but it is believed that Yahoo is experimenting with it, and if AOL moves fully towards XMPP, then both companies may feel compelled to do so as well if they wish to provide the interoperability that the consumer will seek. If IM providers use XMPP, IM interoperability could be possible across systems, and XMPP is shaping up to be the preferred choice in this area.

Jingle allows for the initiation and management of peer-to-peer sessions and for interfacing to SIP and H.323 services that are delivering the predominately voice connectivity today. Jingle does have limitations for defined supplementary services, such as hold, transfer, and so on, and VoIP conferencing is not covered. These are, for the moment, being left to SIP to handle, which Google reportedly will support in the future [GoogleTalk Open].

However, Jingle, with its potential proprietary extensibility and modular design, could be extended to deliver features and services today.

While supporting XMPP/Jingle would provide an interconnection to services from these service providers, it does not necessarily provide full access to the overall set of functionality that developers may wish to provide to allow third parties to use XMPP/Jingle for development/mashup.

Although some APIs are based on standards, none of these as yet have the high level of acceptance in the Web 2.0 world that would indicate they can be considered as the de facto “standard API for media communications development.” They fall short because they do not deliver the features and functionality that are required above a base level, or they are seen as too complicated to implement. What is required is something that delivers the most useful of these requirements, something extensible, and built inherently on top of the common web interfaces of today.

Development Environments

In delivering an API, the development environment is an important consideration. This section discusses some environments available now for delivering functionality.

Ruby

One option is to deliver functionality accessible through a common web-based application language such as Ruby.

Created by Yukihiro Matsumoto of Japan in 1995, Ruby is a dynamic, reflective, general-purpose, object-oriented, programming language used by many developers of websites incorporating mashups. Ruby gained popular acceptance in 2006, and is a language that is relatively easy to learn. When combined with Rails, which is a web application framework, it allows the delivery of website and database content in a fast and efficient way. One Ruby implementation in progress is IronRuby, a Microsoft-based Ruby implementation targeted at the .NET framework.

Ruby also provides support for SOAP, which defines the use of XML and HTTP for access to servers in a platform-independent manner. Thus, providing the appropriate Ruby and SOAP support, a media server could be delivered that could be quickly and efficiently incorporated into a new service design within the Web 2.0 model.

Adhearsion

Adhearsion was created by Jay Phillips. It is an open source framework (Lesser General Public License [LGPL]), which was designed to improve VoIP development and sits above Asterisk and Freeswitch, abstracting the difficulties in working directly with either. It uses Ruby, but can support Java and C.

PHP

PHP is an alternative development environment. It is a computer scripting language that is now produced by The PHP Group. It formerly stood for “Personal Home Page” and was originally conceived in 1994 by Rasmus Lerdorf.

PHP is widely used and suitable for web development, and can be embedded into HTML. PHP language extensions may be written in “C” to add extra functionality. The PHP Extension Community Library (PECL) project is a repository for extensions. PHP is considered to have greater scalability than Ruby on Rails, and, as such, may have an edge for large scale deployments. It is possible to develop an API as PHP extensions to deliver on the media communications capabilities.

Delivering the Most Useful Approach

Arguably the most useful approach, which would satisfy the requirements of the broadest range of potential users, would be the provisioning of a Web 2.0 API set, which could be used in any of the development environments, such as Ruby, PHP, and Java, while abstracting the developer from the intricacies of lower-level communications.

Many developers have favorite development environments and are generally reluctant to move to others without an explicit need to do so. Providing the necessary capabilities in these favorite environments would enhance acceptance rates.

Furthermore, the higher the level of service an API can provide, the easier it becomes for the non-Telecom developer to use and to quickly incorporate and deliver applications and services. Care must be taken that the API provider does not abstract anything to which the developer will ultimately require access, as further features and functionality will be added to the applications and services over time. This can be achieved by providing the capability to work with some of the lower-level details, while not making it a requirement to do so. Using XML is a suitable method for enabling this and for providing backward compatibility that can be maintained as more functionality is added and customer requirements evolve.

Summary

The end user as content creator has changed the way the web is employed. This has created opportunities for delivering media-rich services with end users continually driving change. For the basic IM or peer-to-peer requirements of many users, a number of open-standards options are available today to deliver media services. However, for feature-rich conferencing and collaboration service offerings, no options are predominant, although a number of potential avenues have opened. Technology that is acceptable to the majority is required — a high-level API set capable of being used across broad ranges of web development environments. For traditional service providers in particular, delivering a rich set of media types, while allowing the collaboration and linkage of multiple functions in a service offering, seems to be the answer.

References

[GoogleTalk] Google Talk for Developers, “What is Google Talk?” —
<http://code.google.com/apis/talk/index.html>

[GoogleTalk Open] Google Talk for Developers, Open Communications, Client Choice —
http://code.google.com/apis/talk/open_communications.html

[Jensen] Florian Jensen’s Weblog, “AOL adopting XMPP aka Jabber”, January 17, 2008 —
<http://florianjensen.com/2008/01/17/aol-adopting-xmpp-aka-jabber/#comment-1088>

For More Information

“Overview of XML Communication Technologies,” Dialogic White Paper —
<http://www.dialogic.com/goto/?11039>

“Sophisticated Asterisk Development with Adhearsion,” Jay Phillips, June 19, 2007 —
<http://www.oreillynet.com/pub/a/etel/2007/06/19/sophisticated-asterisk-development-with-adhearsion.html>

“What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software,” Tim O’Reilly —
<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1>

XEP-0166: Jingle —
<http://www.xmpp.org/extensions/xep-0166.html>

XEP-0167: Jingle RTP Sessions —
<http://www.xmpp.org/extensions/xep-0167.html>

www.dialogic.com

Dialogic Corporation

9800 Cavendish Blvd., 5th floor
Montreal, Quebec
CANADA H4M 2V9

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH PRODUCTS OF DIALOGIC CORPORATION OR ITS SUBSIDIARIES ("DIALOGIC"). NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic® products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Dialogic may make changes to specifications, product descriptions, and plans at any time, without notice.

Dialogic is a registered trademark of Dialogic Corporation or its subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

Microsoft is a registered trademark of Microsoft Corporation in the United States and/or other countries. Other names of actual companies and product mentioned herein are the trademarks of their respective owners. Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement their concepts or applications, which licenses may vary from country to country.

This document discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.