

# AT&T Developer Program

## Mobile Web 2.0

### White Paper

Revision           **1.01**  
Revision Date     September 4, 2009

This document and the information contained herein (collectively, the “**Information**”) is provided to you (both the individual receiving this document and any legal entity on behalf of which such individual is acting) (“**You**” and “**Your**”) by AT&T, on behalf of itself and its affiliates (“**AT&T**”) for informational purposes only. AT&T is providing the Information to You because AT&T believes the Information may be useful to You. The Information is provided to You solely on the basis that You will be responsible for making Your own assessments of the Information and are advised to verify all representations, statements and information before using or relying upon any of the Information. Although AT&T has exercised reasonable care in providing the Information to You, AT&T does not warrant the accuracy of the Information and is not responsible for any damages arising from Your use of or reliance upon the Information. You further understand and agree that AT&T in no way represents, and You in no way rely, on a belief that AT&T is providing the Information in accordance with any standard or service (routine, customary or otherwise) related to the consulting, services, hardware or software industries.

AT&T DOES NOT WARRANT THAT THE INFORMATION IS ERROR-FREE. AT&T IS PROVIDING THE INFORMATION TO YOU “AS IS” AND “WITH ALL FAULTS.” AT&T DOES NOT WARRANT, BY VIRTUE OF THIS DOCUMENT, OR BY ANY COURSE OF PERFORMANCE, COURSE OF DEALING, USAGE OF TRADE OR ANY COLLATERAL DOCUMENT HEREUNDER OR OTHERWISE, AND HEREBY EXPRESSLY DISCLAIMS, ANY REPRESENTATION OR WARRANTY OF ANY KIND WITH RESPECT TO THE INFORMATION, INCLUDING, WITHOUT LIMITATION, ANY REPRESENTATION OR WARRANTY OF DESIGN, PERFORMANCE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, OR ANY REPRESENTATION OR WARRANTY THAT THE INFORMATION IS APPLICABLE TO OR INTEROPERABLE WITH ANY SYSTEM, DATA, HARDWARE OR SOFTWARE OF ANY KIND. AT&T DISCLAIMS AND IN NO EVENT SHALL BE LIABLE FOR ANY LOSSES OR DAMAGES OF ANY KIND, WHETHER DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, SPECIAL OR EXEMPLARY, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF GOODWILL, COVER, TORTIOUS CONDUCT OR OTHER PECUNIARY LOSS, ARISING OUT OF OR IN ANY WAY RELATED TO THE PROVISION, NON-PROVISION, USE OR NON-USE OF THE INFORMATION, EVEN IF AT&T HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES OR DAMAGES.

All marks, trademarks, and product names used in this document are the property of their respective owners.

© 2009 AT&T Intellectual Property. All rights reserved. AT&T and AT&T logo are trademarks of AT&T Intellectual Property.

## Revision History

Date	Revision	Description
June 22, 2009	1.0	First release of this white paper.
Sept. 4, 2009	1.01	Copyedit of this white paper.

## Table of Contents

1. Forward.....	1
1.1 Audience.....	1
1.2 Contact Information .....	1
1.3 AT&T Resources .....	1
1.4 External Resources .....	2
1.5 Terms and Acronyms.....	3
2. Introduction .....	5
3. Web Applications Overview .....	6
3.1 Benefits of Mobile Web Applications .....	6
3.2 Web 2.0 .....	7
3.3 Web Standards and Specifications Entities.....	8
4. Mobile Browser Technologies .....	10
4.1 Open Mobile Alliance Browsing.....	10
4.2 High-End and Emerging Features .....	11
4.3 Ajax .....	11
4.4 Offline Operation.....	12
4.5 Faster JavaScript.....	13
4.6 Rich Internet Application Frameworks .....	14
4.7 Video Capabilities .....	14
4.8 HTML 5 .....	15
4.9 Features of Leading Mobile Browsers .....	16
4.10 Hybrid Applications .....	17
5. Widgets.....	18
6. GSMA OneAPI.....	20
7. BONDI .....	21
8. Security .....	23
8.1 Transport Security .....	23
8.2 Application-Level Security .....	24
9. Recommendations.....	25

9.1	High Level.....	25
9.2	Mobile Web Content Design.....	26
9.3	W3C Mobile Web Initiative.....	26
10.	Conclusion.....	28
11.	Acknowledgment.....	29

## Figures

Figure 1:	Google Gears Background Sync Architecture.....	13
Figure 2:	Rich Internet Applications.....	14
Figure 3:	Example of BONDI Test Widget.....	19
Figure 4:	GSMA OneAPI.....	20
Figure 5:	BONDI Architecture.....	22

## Tables

Table 1:	Terms and Acronyms.....	3
Table 2:	Native versus Web Development Approaches.....	7
Table 3:	Summary of Leading Capabilities in Today's High-End Mobile Browsers.....	11
Table 4:	Features of Leading Mobile Browsers.....	16
Table 5:	Widget Framework Components.....	18
Table 6:	First Phase of OneAPI Functions.....	20

## 1. Forward

This white paper discusses trends in Web technologies as applied to mobile devices. The emphasis is on handheld platforms including low- and mid-tier devices as well as smartphones.

### 1.1 Audience

The target audience of this white paper is software developers, IT developers, architects, and managers who are considering development or deployment of applications that use Web technologies.

### 1.2 Contact Information

E-mail any comments or questions regarding this white paper to [developer.program@att.com](mailto:developer.program@att.com). Please reference the title of this paper in the message.

### 1.3 AT&T Resources

devCentral, the AT&T Developer Program: <http://developer.att.com>

Mobile Application Development: <http://developer.att.com/mobiledevelopment>

Network Technologies:  
<http://developer.att.com/developer/index.jsp?page=toolsTechSection&id=7600074>

3G: <http://developer.att.com/3G>

Device Information: <http://developer.att.com/devicedetails>

Mobile Middleware: <http://developer.att.com/middleware>

Security Guidelines: <http://developer.att.com/security>

Certified Solutions Catalog: <http://developer.att.com/certifiedsolutionscatalog>

Wireless Reference Architecture: <http://developer.att.com/WRA>

Platforms and Operating Systems:

<http://developer.att.com/developer/index.jsp?page=toolsTechOverview&id=800048>

Mobile Web Applications:

<http://developer.att.com/developer/index.jsp?page=toolsTechSection&id=800084>

## 1.4 External Resources

W3C Mobile Web Best Practices Working Group:

<http://www.w3.org/2005/MWI/BPWG/>

Open Mobile Terminal Platform BONDI: <http://bondi.omtp.org>

OMA Browsing v2.2:

[http://www.openmobilealliance.org/Technical/release\\_program/browsing\\_v2\\_2.aspx](http://www.openmobilealliance.org/Technical/release_program/browsing_v2_2.aspx)

OMA Browsing v2.3:

[http://www.openmobilealliance.org/technical/release\\_program/browsing\\_v23.aspx](http://www.openmobilealliance.org/technical/release_program/browsing_v23.aspx)

OMA Browsing v2.4:

[http://www.openmobilealliance.org/technical/release\\_program/browsing\\_v24.aspx](http://www.openmobilealliance.org/technical/release_program/browsing_v24.aspx)

HTML 5 Differences From HTML 4: <http://www.w3.org/TR/html5-diff/>

Internet Explorer Mobile: <http://msdn.microsoft.com/en-us/library/bb159715.aspx>

W3C Geolocation API: <http://dev.w3.org/geo/api/>

BlackBerry Web Widgets:

<http://www.blackberry.com/DevMediaLibrary/view.do?name=webwidgets>

GSMA OneAPI: <https://gsma.securespsite.com/access/default.aspx>

GSMA OneAPI Reference Implementation: <http://oneapi.aepona.com>

## 1.5 Terms and Acronyms

The following table defines the acronyms used in this document.

**Table 1: Terms and Acronyms**

<b>Term or Acronym</b>	<b>Definition</b>
3GPP	Third Generation Partnership Program
AJAX	Asynchronous JavaScript and XML
Ajax	Same concept as AJAX but without specifying exact technologies
API	Application Programming Interface
BONDI	Browser/widget platform developed by OMTF and named after the famous Bondi surfing beach in Sydney, Australia
BREW	Binary Runtime Environment for Wireless
CDPD	Cellular Digital Packet Data
CSS	Cascading Style Sheets
DOM	Document Object Model
ESTI	European Telecommunications Standards Institute
GSM	Global System for Mobile communication
GSMA	GSM Association
HDML	Handheld Device Markup Language
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
MMS	Multimedia Messaging Service
MPEG-4	Motion Picture Experts Group version 4
OMA	Open Mobile Alliance
OMTF	Open Mobile Terminal Platform
PIM	Personal Information Manager
REST	Representational State Transfer
RIA	Rich Internet Application
RSS	Really Simple Syndication
RTSP	Real Time Streaming Protocol
SMS	Short Message Service
SQL	Structured Query Language

SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
VPN	Virtual Private Network
WAP	Wireless Application Protocol
W3C	World Wide Web Consortium
WhatWG	Web Hypertext Application Technology Working Group
XHR	XMLHttpRequest
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

## 2. Introduction

Mobile devices have been able to access the Web since 1996 when AT&T Wireless introduced the PocketNet phone, which employed Handheld Device Markup Language (HDML). This was a text-based microbrowser, operating over a Cellular Digital Packet Data (CDPD) network that had an effective throughput of about 10 thousand bits per second (kbps). Since then, there have been tremendous advances in mobile browser capabilities as well as the underlying networks. The result is that today's modern browsers can render at-large Internet content and support a multitude of elements, such as graphics and video. Advanced browsers on smartphones have most of the capabilities found in browsers developed for the desktop market. Because of this, developers can use Web technology for an increasing number of applications. Additionally, Web-based applications have become one way of addressing the fragmentation challenges of developing applications for the mobile platform.

Whereas desktop platforms have converged considerably, innovation and competition in mobile computing have led to a large number of mobile platforms. These include Android, Apple iPhone, Mobile Linux (as defined by the LiMo Foundation, as well as other implementations), Palm Garnet and Palm Pre, RIM BlackBerry, Symbian S60 and UIQ, and Windows Mobile.

Just as an increasing number of applications for desktops are Web-based, the same trend is occurring for mobile devices. By developing applications using Web technologies, developers can simultaneously address multiple mobile platforms. There are, however, significant differences between the mobile Web and the wireline Web. Developers must know when to use Web technologies and how to use appropriate mobile Web strategies, as discussed in this paper.

## 3. Web Applications Overview

To provide an overview about the development of Web applications, this section discusses the benefits of a Web-based approach and presents the pros and cons of employing a Web approach versus developing native applications. This section then discusses the meaning of the term *Web 2.0* and what the term means in a mobile context. Finally, this section lists the various standards organizations involved in mobile Web technology development.

### 3.1 Benefits of Mobile Web Applications

In the past, mobile Web applications ran slowly because networks had relatively low throughput and high latency (packet delays). Today's 3G networks, however, are significantly faster with users frequently obtaining 1 Mbps or higher throughput and improved latency rates. Web applications thus have become much more responsive and suitable for both consumer and business applications.

Advances in Web technology further improve the mobile Web experience by allowing selective updates of the screen (versus having to load a new page for every update) and offline operation.

Since nearly all mobile phones today come with a browser client, users don't have to install client software for new applications. For IT managers, this can be particularly beneficial as managing the software configuration of a large number of mobile devices can be a challenge.

Another benefit of Web applications is that many developers already have Web skills, so the learning curve for developing mobile Web applications is generally lower than for developing native applications.

Web applications also have security architecture benefits as many business firewalls are already configured to permit HTTP traffic. Moreover, transport-layer security is readily available via the Secure Sockets Layer (SSL) that is built into nearly all browsers.

Finally, because TCP connections close after transfer of each page object, Web-based applications are relatively tolerant of temporary connectivity loss, which can occur if users are moving through the environment, such as in a car.

The benefits just described should not imply that a Web approach is superior to a native-application approach for every scenario. In fact, there are multiple pros and cons to consider, as summarized in the following table.

**Table 2: Native versus Web Development Approaches**

Issue	Native Application	Web Application
Application responsiveness	Excellent	Good with 3G and Wi-Fi; getting better with faster networks and new technologies such as Ajax, Gears, HTML 5
Capabilities	Excellent with access to device information and hardware	Good to excellent; new specifications, such as BONDI, will provide access to device information and hardware
Development	Very high learning curve	Leverages common Web tools and techniques
Multi-platform support	Separate versions required for each platform unless using Java or mobile middleware	Web content operates across multiple platforms, although browser features can vary
Offline operation	Excellent	Biggest limitation today, but being addressed by BONDI, Gears, and HTML 5
Security	Complex set of considerations with concern about local data and firewall traversal	Some advantages with server-side storage of sensitive data and easier HTTP-firewall traversal

Native applications can also leverage Web technologies, creating hybrid applications. This approach is discussed further below.

### 3.2 Web 2.0

The term *Web 2.0* is not strictly defined, but generally, it refers to Web-hosted, sophisticated applications that promote a high degree of user interaction, including user-created content. Some Web 2.0 concepts include use of the Web as the application platform, improved user experience, faster and more efficient Web applications, delivery of services as opposed to packaged software,

services paid for by advertising, users as co-developers, continuous improvement with no scheduled releases, and customer self-service.

Web 2.0 incorporates an ever-evolving array of technologies, development methodologies, and service models such as:

- CSS (Cascading Style Sheets), REST/XML (Representational State Transfer/Extensible Markup Language) APIs
- Data interchange formats such as JavaScript Object Notation (JSON)
- Semantically valid XHTML/HTML (Extensible Hypertext Markup Language/Hypertext Markup Language) markup
- RIA (Rich Internet Application) techniques such as Ajax, Adobe Flex, or Microsoft Silverlight
- Syndication, aggregation, and notification of data in RSS (Really Simple Syndication) or Atom feeds<sup>1</sup>
- Collaborative tagging, social classification, and indexing
- Client- and server-side mashups
- User-generated content via blogs, wikis, and forums

Unless guided by best practices and supported by standardized semantically-useful metadata, however, inconsistent service development and deployment can increase market fragmentation and inhibit scalability of services. Thus, developers should pick their Web technologies carefully and use standardized approaches when available. These technologies are discussed below.

### 3.3 Web Standards and Specifications Entities

A number of bodies develop Web-related standards and specifications. Of these, the World Wide Web Consortium (W3C) is the main international standards organization for the Web. Its work includes Web standards for both desktop and mobile use.

In the mobile arena, the Open Mobile Alliance (OMA) specifies mobile service enablers to ensure interoperability, and with respect to mobile browsing, has developed a number of specifications and recommendations which are discussed

---

<sup>1</sup> Atom Syndication Format is an XML language used for Web feeds and is an alternative to RSS.

in a subsequent section. Furthermore, OMA will develop specifications for OneAPI, which is also discussed in a subsequent section.

A relatively new group, Open Mobile Terminal Platform (OMTP), strives to simplify the customer experience for mobile data services and improve mobile device security. OMTP does not develop standards, but does issue requirements and specifications such as BONDI, which is discussed in a subsequent section. As a member of W3C, OMTP can make submissions to W3C activities that affect BONDI.

The GSMA (GSM Association), representing operators and mobile ecosystem providers, develops technical recommendations including OneAPI. GSMA collaborates with OMA and OMTP BONDI.

The Parlay Group develops telecom APIs. This group has worked jointly with the European Telecommunications Standards Institute (ETSI) and the Third Generation Partnership Program (3GPP) to develop the Parlay X 3.0 specifications, which define a wide range of telecom functions that are accessible using Web services.

The OpenAjax Alliance strives for successful adoption of open and interoperable Ajax-based Web technologies. The alliance provides feedback on projects such as BONDI.

## 4. Mobile Browser Technologies

Most mobile phones sold today include a browser for Web content. For all AT&T phones, a minimum set of browser capabilities are supported, plus additional capabilities are supported on higher-tier phones, such as smartphones. All AT&T phones support OMA browsing capabilities, which specify architecture, markup, scripting, and other functions. Most phones support version 2.2 of this specification, with some phones supporting version 2.3. With OMA browsing capabilities, developers must develop content specifically for mobile phones.

Higher-tier phones not only support OMA browsing, but have greater capabilities often comparable to desktop browsers, including the ability to render nearly all Internet Web content. These browsers are either native to the platform (for example, Pocket Internet Explorer for Windows Mobile) or available from third parties (for example, Firefox or Opera). These powerful browsers make a wide range of applications and services possible. Subsequent sections expand on these emerging capabilities.

### 4.1 Open Mobile Alliance Browsing

The OMA bases its browsing specifications on Internet technology, but limits profiles for constrained resources and user interfaces on mobile devices. For instance, it assumes reduced memory, processing power, bandwidth, and user-input methods. It defines application-level protocols, semantics, syntax, content formats, user-agent behavior, and use of hypermedia transfer protocols.

The foundational elements of OMA browsing include:

- WAP Architecture (with or without proxy)
- XHTML Mobile Profile
- ECMAScript Mobile Profile (including Document Object Model) for local application scripting capability (akin to JavaScript)
- Wireless Cascading Style Sheets (Wireless CSS)
- Binary XML for efficient communications
- OMA Push for asynchronous server-initiated content delivery

## 4.2 High-End and Emerging Features

High-end mobile browsers are adopting most of the capabilities found in desktop browsers, as summarized in the following table.

**Table 3: Summary of Leading Capabilities in Today's High-End Mobile Browsers**

Technology	Summary
Ajax	Asynchronous retrieval of data, allowing for functions such as selectively updating one part of the display
Offline operation	Application interaction with local version of data
Faster JavaScript	Enables sophisticated applications
Video	Increasing number of Web formats supported; for example, Flash
Desktop integration	Seamless handover of sessions between desktop and mobile; for example, Firefox provides for handover via cloud of sessions, tabs, cookies, favorites, and passwords
Device and network capabilities	Emerging standards (such as OMA Device Profile Evolution (DPE), BONDI, and OneAPI) provide access to device and network information and functions
Developer tools	More powerful Web tools; for example, Google Web Toolkit (available for Android and iPhone) for converting Java to JavaScript

## 4.3 Ajax

Ajax, one of today's key technologies, enables rich Internet applications. Using Ajax, developers can create pages with selective update of on-screen information and functions such as mouse-over. Most current smartphone browsers support Ajax. The term "AJAX" was originally derived from "Asynchronous JavaScript and XML" but today, the lower-case term "Ajax" refers to the same concept without specifying the exact technologies used.

Ajax today usually employs:

- XHTML and CSS for presentation
- DOM (Document Object Model) for storing pages allowing programmatic access

- XML and XSLT (Extensible Stylesheet Language Transformations) for data interchange and display; alternatively, JSON (JavaScript Object Notation) may be employed
- XHR (XMLHttpRequest) for asynchronous communications
- JavaScript for programming

#### 4.4 Offline Operation

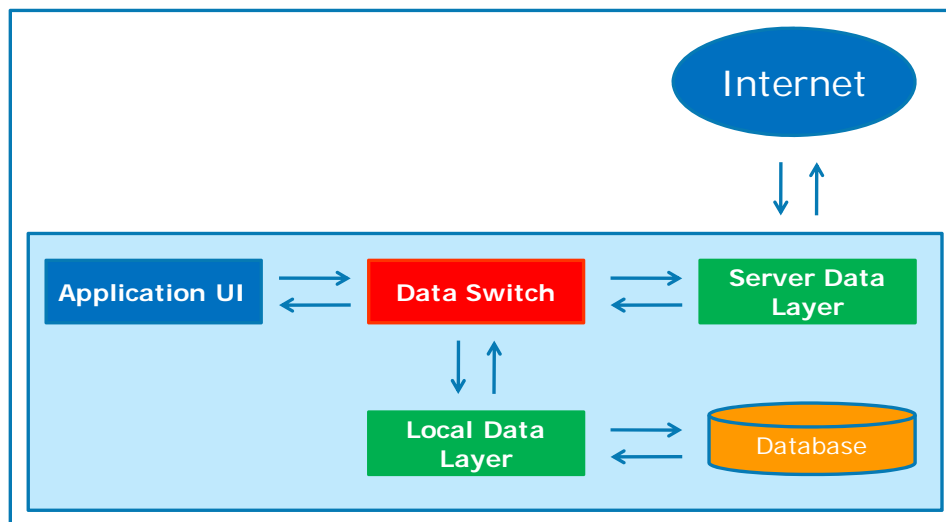
Browser-based applications generally require a network connection, which sometimes is not available for mobile users. New (and forthcoming) Web technologies, however, permit offline operation.

One example is the forthcoming BONDI filesystem API, a full-featured API that enables file system access on the local device. With such an API, a user could be viewing or composing responses to previously received e-mail addresses, even in the absence of a network connection. This API is available today as a Windows Mobile-based reference implementation and is being expanded to support LiMo, Android, and other platforms.

Gears is another Web technology with which developers can build JavaScript applications that operate offline. Gears synchronizes new data once the device has a connection. Gears was initiated by Google and is now an open-source project. It is available today on Android and Internet Explorer, and is forthcoming on BlackBerry 5.0 handheld release and Opera Mobile 9.5.

Gears also provides an API called WorkerPool in which JavaScript threads can execute without blocking the UI. Figure 1 shows the Google Gears architecture.

**Figure 1: Google Gears Architecture<sup>2</sup>**



Another Web technology that supports offline operation is HTML, with a feature called an application cache.<sup>3</sup> W3C has not yet completed its work on HTML 5, but browsers will support the defined features before the final specification.

Another offline operation, BlackBerry supports form queuing, in which users can fill out forms even when out of the coverage area.

#### 4.5 Faster JavaScript

Faster JavaScript execution not only improves general Web performance, but allows applications to approach the sophistication of native applications. Considerable emphasis on faster JavaScript execution, including methods like just-in-time compilation, is being put on new browsers such as:

- Firefox TraceMonkey
- Google Chrome V8
- Opera Carakan
- Safari Nitro
- WebKit SquirrelFish Extreme

<sup>2</sup> See <http://code.google.com/apis/gears/architecture.html>.

<sup>3</sup> See <http://www.whatwg.org/specs/web-apps/current-work/multipage/offline.html>.

Moreover, increasingly fast JavaScript execution will be available in future mobile browsers.

## 4.6 Rich Internet Application Frameworks

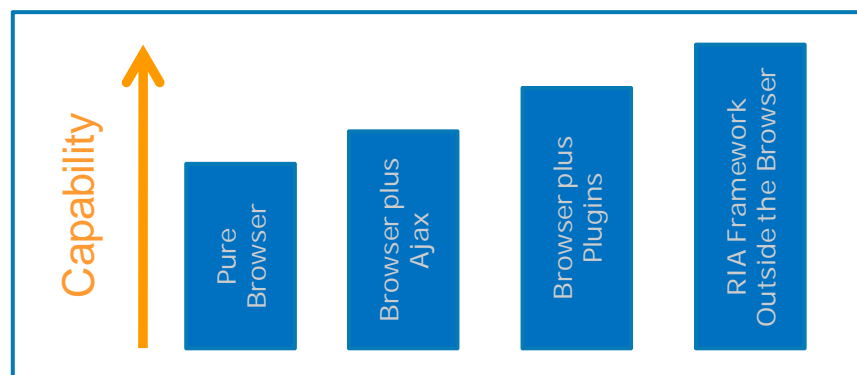
Rich Internet Application (RIA) frameworks operate above the OS layer. They provide a consistent runtime execution environment for Internet applications.

RIA frameworks can be deployed:

- As OS extensions (for example, Microsoft .NET RIA with Silverlight)
- As native applications (for example, Adobe AIR)
- As plug-ins for high-end browsers (for example, Adobe Flash)
- In conjunction with Java Virtual Machine (for example, JavaFX)

Figure 2 shows the increasing capability of different RIA frameworks.

**Figure 2: Rich Internet Applications**



## 4.7 Video Capabilities

Many AT&T phones have MPEG-4-capable video players with content accessible via HTTP and/or RTSP (Real Time Streaming Protocol). In addition, there are various other video technologies available to users. Some of these are part of the RIA frameworks discussed in the previous section.

Perhaps the best known is Adobe Flash, which is available today for Opera Mobile, Symbian, Windows Mobile, and BREW (Binary Runtime Environment for

Wireless.) A new version, called FlashPlayer 10 for Smartphones, initially supports Android, Symbian, and Windows Mobile.

Microsoft's Silverlight technology has a mobile version planned with initial support for Windows Mobile and Symbian S60. Silverlight uses Microsoft's .NET application framework and Visual Studio.

JavaFX, a new technology from Sun Microsystems, operates as a layer above Java Virtual Machine. Available for Android and Windows Mobile, the platform includes JavaFX Script, development tools, graphics media, audio support libraries, and the runtime environment. As discussed in the previous section, developers can use it as an independent application platform.

HTML 5 will allow tighter integration of video elements and other content without the need for a separate browser plug-in for video content.

## 4.8 HTML 5

HTML 5 is the next version of HTML and is currently in development, with the first working draft of the specification released in January, 2008 and completion expected around 2012.<sup>4</sup> Browser vendors, however, are already implementing some HTML 5 features as they become defined.

One objective of HTML 5 is to create a framework for applications with much more capability than previously possible, hence the original name of the specification work, "Web Applications 1.0." HTML 5 is being developed as a joint effort of W3C and the Web Hypertext Application Technology Working Group (WhatWG).

Some key features of HTML 5 include:

- Better page structuring through new elements like section, header, footer, article, nav, and dialog
- A canvas element with 2D drawing API for dynamic graphics and animation
- Direction provision for audio and video content
- Client-side persistent storage using key/value pairs and SQL
- Offline application APIs

---

<sup>4</sup> See <http://dev.w3.org/html5/spec/Overview.html> for more detail.

- Editing and drag-and-drop APIs
- Network Web Socket API
- Cross-document messaging

## 4.9 Features of Leading Mobile Browsers

The leading browsers today have many powerful features, as summarized in Table 4. Awareness of rendering engine is relevant as many Web developers are familiar with a particular rendering engine and its properties. Furthermore, each rendering engine functions consistently across different platforms.

**Table 4: Features of Leading Mobile Browsers**

Mobile Browser	Platforms	Rendering Engine	Ajax	Gears	Flash	Widget Engine	Comments
Android	Android	WebKit	Yes	Yes	Forthcoming	No	
Apple iPhone Safari	iPhone	WebKit	Yes	No	No	No	Excellent at rendering and navigating overall Web.
BlackBerry	BlackBerry	Mango	Yes	Forthcoming	No	Yes	Also includes BlackBerry security architecture, push capability, and optimization.
Firefox	Mobile Linux, Windows Mobile, Symbian	Gecko	Yes	No	Yes	Yes	Not released; expected Q2 2009. Synchronizes between desktop/mobile sessions.
Internet Explorer	Windows Mobile	Trident/MSHTML	Yes	Yes	Yes	No	Silverlight support planned by Microsoft.

Mobile Browser	Platforms	Rendering Engine	Ajax	Gears	Flash	Widget Engine	Comments
NetFront	Mobile Linux, Symbian, Windows Mobile, others		Yes	No	Yes	Yes	
Opera Mobile	Symbian S60, UIQ, Windows Mobile	Presto	Yes	Forthcoming	Yes	Yes	Opera Mini provides support for many other platforms.
Symbian S60	Symbian S60	Symbian S60	Yes	No	Yes	Yes via Nokia Web Runtime	Silverlight support planned by Microsoft.

## 4.10 Hybrid Applications

Developers may also wish to consider hybrid approaches, wherein a native application provides some degree of functionality and the browser provides the balance. There is a range of possibilities, such as a native application providing a Web wrapper, in which the Web application delivers all the functionality of the application. Alternatively, a highly functional native application can use the Web to extend its functionality.

An example is BlackBerry with `net.rim.device.api.browser.field`.<sup>5</sup> This provides access to browser-component functionality that can be incorporated into other applications. The API supports third-party applications that want the app's user interface to include browser fields that render Web content. RIM's rendering library handles all the rendering of Web content for the field, then hands the application back to the field for display.

<sup>5</sup> See <http://www.blackberry.com/developers/docs/4.6.0api/net/rim/device/api/browser/field/package-summary.html> for more details.

## 5. Widgets

There are various definitions of how people view widgets. For our purposes, we define widgets as stand-alone Web applications that use browser technologies such as HTML, Ajax, JavaScript, and CSS. Widgets represent a repackaging of Web content, which can be more appealing or easier to use than the browser. A widget can actively present information on the default display screen, or be easily executed from an application menu or from an icon on the default display screen. Like an application, a widget is downloaded and is then available without subsequent downloads.

There are various browsers that have built-in widget support such as Access NetFront, Opera Mobile, and Symbian S60. Some popular stand-alone widget frameworks include Nokia WidSets, SurfKitchen, and Yahoo! Go. There are also handset-specific frameworks from companies including Motorola, Sony Ericsson, and RIM. Other frameworks are available from Bling, Mojax, Plusmo, Snaptu, Spime, and Webwag. Finally, Sun Microsystems has a framework for Java Platform, Micro Edition called Sun Java On Device Portal.

Widget requirements are being standardized by the W3C WebApps Working Group. One example is “Widgets 1.0: Requirements” which standardizes packaging.

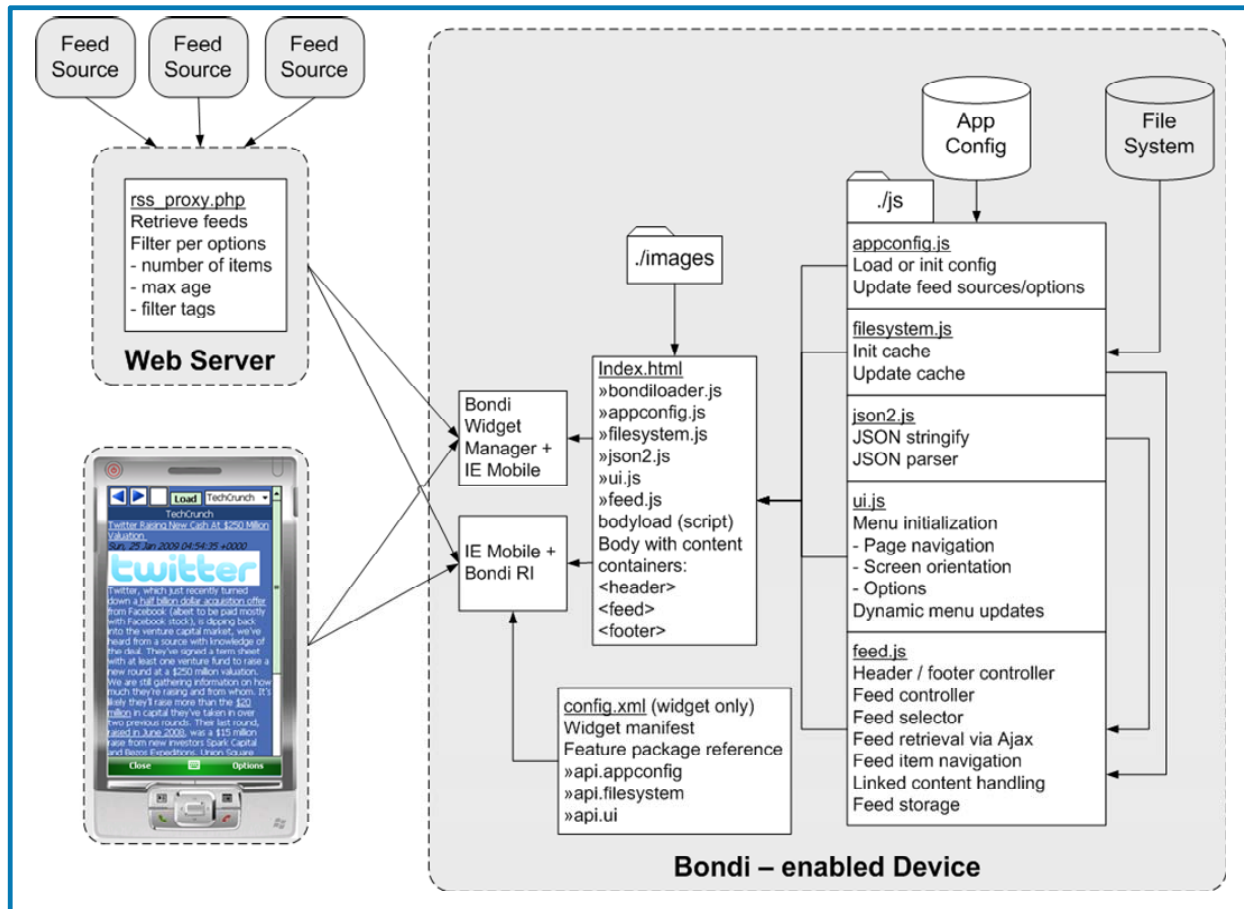
Table 5 summarizes the items typically found in a widget framework.

**Table 5: Widget Framework Components**

Item	Details
SDK	Widget emulator, device API simulators, JavaScript debuggers, CSS and DOM inspectors, XHR/HTTP loggers and debuggers
Administrative tools	Certification/signing; capabilities for widget polling, notification, updates, revocation
Widget client	Widget runtime; separate executable or part of browser
Web sites	Public- or operator-hosted Web sites that contain a library of widgets for that framework

Figure 3 shows an example of a widget developed using BOND I.

**Figure 3: Example of BOND I Test Widget**



## 6. GSMA OneAPI

The GSM Association has an initiative, called OneAPI, to define a commonly supported API for mobile operators to expose network information to Web (and other) application developers. These APIs will use both RESTful and Web services interfaces. The basis for this work is Parlay X, as defined by 3GPP in TS29.199. The first APIs implemented will be for messaging and location functions.

As shown in Figure 4, developers will benefit because their applications will be able to obtain information in a consistent fashion across multiple operators that support OneAPI. Applications leveraging OneAPI can operate either on a server or on a mobile device, though most applications will favor server implementations.

**Figure 4: GSMA OneAPI**

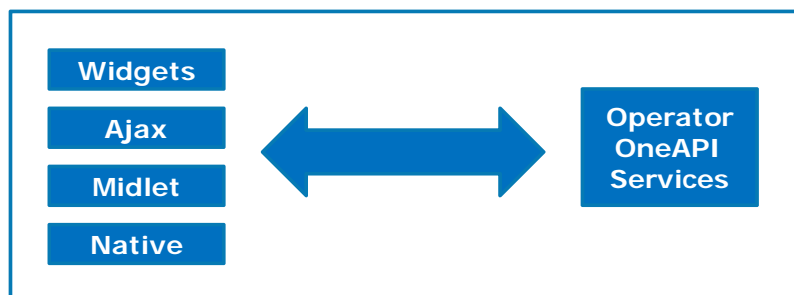


Table 6 summarizes the first phase of OneAPI functions.

**Table 6: First Phase of OneAPI Functions**

Function	Description
Short messaging	Send or receive SMS, WAP Push
Multimedia messaging	Send or receive MMS
Location	Determine the current physical location of a user/device
Payment	Charge a user for an application via the operator (bill/pre-pay account)

Further information is available at <https://gsma.securespsite.com/access/default.aspx>.

## 7. BONDI

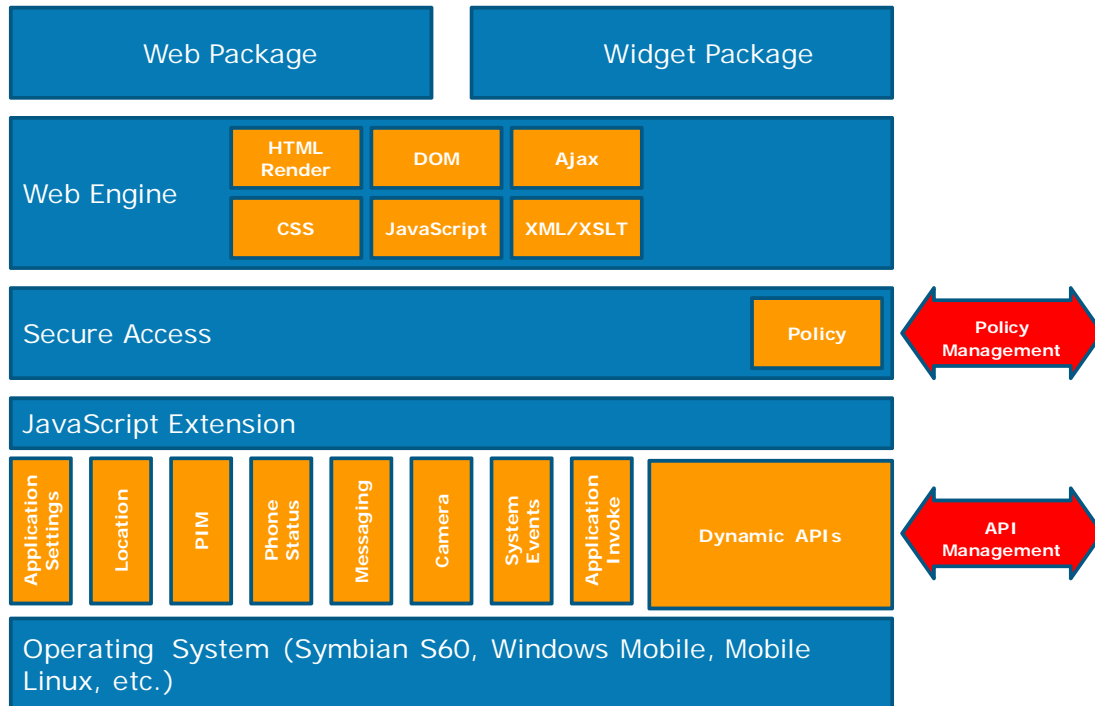
BONDI is a browser and widget specification developed by the Open Mobile Terminal Platform (OMTP). The core members of this group are operators and include AT&T, Hutchison 3G, Orange, Telefónica, Telenor, Telecom Italia Mobile, T-Mobile, and Vodafone.

BONDI standardizes access to a wide range of device features such as local-application invocation, messaging (SMS, MMS, e-mail), local file I/O, phone status (for example, signal strength), contacts, location, and camera functions.

Due to security concerns, JavaScript applications normally don't have access to these functions. BONDI addresses this concern by enforcing security through an elaborate structure of policies, certificates, and certification. Independent testing labs test and verify BONDI Web services and widgets for security and functionality so certificates can be issued. Version 1.0 of the BONDI specification became available in February, 2009. Currently, Windows Mobile devices can be used to test the BONDI stack.

Information and reference implementation are available at <http://bondi.omtp.org>. Figure 5 shows the BONDI architecture.

**Figure 5: BONDI Architecture<sup>6</sup>**



<sup>6</sup> Figure based on W3C chart at <http://www.w3.org/2009/Talks/03-mobileinternet/bondi-architecture.jpg>.

## 8. Security

One inherent security benefit of Web applications is that all or most of the data the application uses is stored on the server. Thus, if the device is lost or stolen, this mitigates the loss of sensitive data that might otherwise have been stored on the device.

As mobile browsers become more sophisticated, however, they will become vulnerable to various security exploits, just like desktop browsers, and may need to be updated as vendors issue new versions.

There are two fundamental security considerations: how to protect the data stored on the Web server and how to protect the customer's Web experience. In addition, developing secure Web applications requires addressing transport-layer security.

### 8.1 Transport Security

Transport security refers to the protection of communicated data. Although AT&T encrypts the radio link, data may travel over paths that are not encrypted such as through the Internet. Since most browsers support Secure Sockets Layer (SSL) or Transport Layer Security (TLS), securing communications is relatively straightforward. These security protocols are also compatible with SSL Virtual Private Network (VPN) concentrators, which organizations may already be using to secure employee remote access.

Another advantage of using Web applications compared to other application architectures is that many organizations have already configured their firewalls to allow HTTP traffic, simplifying firewall traversal.

One consideration in using SSL, however, is that SSL handshakes are somewhat verbose and not necessarily ideal for exchanges with small amounts of data.

In the case of BlackBerry applications, Web communication is via a connection to the BlackBerry Enterprise Server or the BlackBerry Internet Service, each of which is already a secure path, making SSL unnecessary.

## 8.2 Application-Level Security

Web application developers should ensure their applications/sites are not vulnerable to various forms of attack. Some of these include:

Cross-Site Scripting (XSS). Vulnerabilities emerge when unescaped user data, such as malicious JavaScript, is included in HTML output. These vulnerabilities can be non-persistent (payload echoed in an immediate response), persistent (payload stored in the vulnerable system for later embedding in an HTML page sent to a user) or DOM-based (content stored in local Document Object Model and later reinterpreted as HTML that includes malicious script).

Cross-Site Request Forgery (CSRF). This vulnerability exploits the trust that a site has for a user's browser. This is the opposite of XSS where the user's browser trusts the site. In CSRD, the vulnerability consists of a Web application performing an action on a third-party site from an authenticated user without requiring user authorization.

Click Jacking. In this exploit, users are tricked into interacting with a transparent Web page, clicking on visible buttons, but performing actions on a hidden page.

Developers can enforce security via Secure Development Life Cycle<sup>7</sup> processes, and by using application-aware intrusion prevention systems or firewalls. In addition, Web applications should validate all user input, take advantage of Turing tests where appropriate, and make requests that are dependent on session sensitivity.

---

<sup>7</sup> See <https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/sdlc/326-BSI.html> for more information.

## 9. Recommendations

This paper makes a number of recommendations with respect to developing Web applications, first at a high level, then regarding content design, and finally, mentioning the W3C Mobile Web Initiative.

### 9.1 High Level

Developers should first determine whether to use a Web approach, create a native application, or perhaps combine both into a hybrid application.

The second most important step is deciding whether to develop mobile-specific content, as discussed in the subsequent section.

In making these decisions, it's important to note that though this paper has listed various advanced features available for mobile browsers, not every browser employs every feature. Developers should thus test their applications on all targeted browsers and devices.

One approach is to optimize high-usage applications for specific devices and browsers using advanced Web technologies, thus providing a highly efficient user experience. Other applications can then be designed on a lowest-common-denominator model, which can target a greater set of browsers, although with less efficient operation.

Next, developers should take advantage of broadly supported, standardized initiatives, such as BOND1, as handsets become available that support these initiatives.

Additionally, developers should test their applications in a variety of conditions and locations to assess the effects of mobility (for example, operation in a car), 2G versus 3G operation (for example, testing for different throughput and latency), and varying network conditions (for example, quiet versus busy locations or periods).

Finally, as with all mobile applications, Web or otherwise, it is important to implement appropriate device management, policy control, and security methods.

## 9.2 Mobile Web Content Design

Mobile usage is quite different from desktop usage, even for the same application, so developers should provide fast access to the most needed information.

There is a variety of approaches to addressing the small screen size of mobile devices. One is to use the same content as for desktop browsers, but to use different CSS styling or to transform XML content using XSLT. Both methods typically require that content be developed, from the outset, for both desktop and mobile browsers. Alternatively, developers can create mobile-specific content, which can be located at a different URL, or served after user agent-sniffing detects a mobile browser.

Developers must also accommodate user input limitations, keeping in mind that different devices have different types of keyboards or key entry, and that different users have different text-entering skill levels.

To improve page-loading speeds, pages should minimize the number of resources used and can use Ajax to intelligently or predictively pre-fetch information. Developers can also take advantage of local browser caching, which can be effective for objects, like images, that repeat across multiple pages.

## 9.3 W3C Mobile Web Initiative

The W3C has a project called the Mobile Web Initiative<sup>8</sup> and a specification called Best Practices 2.0, which is in draft stage. This specification follows in the footsteps of the Mobile Web Best Practices 1.0 (BP1), a W3C Recommendation. BP1's focus was the extension of Web browsing onto mobile devices. BP2 extends the focus to Web applications, generally defined as applications that are accessed and presented via Web technologies.

Web applications represent a spectrum of services and content. At the simple end are typical Web browsing sites presented in browsers; this was the focus of BP1. BP2 focuses on further issues of the delivery context and use of advanced Web technologies for browsers and non-browser Web runtime environments such as widgets.

BP2 provides a broad set of specific recommendations addressing:

---

<sup>8</sup> See <http://www.w3.org/Mobile/> for more information.

- Security and privacy
- User awareness and control of application behavior
- Conservative use of resources
- User experience
- Handling device capability variation

## 10. Conclusion

Mobile Web technologies are becoming progressively more powerful. On higher-end phones such as smartphones, nearly all of the features found on desktop computer browsers are becoming available. Consequently, Web applications increasingly are an effective alternative to native applications, following general Internet trends.

Key technologies and capabilities include Ajax, off-line operation, access to device network information, faster JavaScript execution, flexible video support, integration with desktop browsers, push capabilities, and better development tools.

Using the same powerful Web technologies, widget frameworks provide users with fast, intuitive access to Web-based information in convenient and easy-to-use applications.

Standardization efforts such as OMTB BOND and GSMA OneAPI will further increase functionality of Web applications, enabling applications to easily, and in a consistent manner across devices and operators, access a variety of device and network information and services such as location, PIM data, messaging functions, camera functions, billing functions, and user profiles.

## 11. Acknowledgment

Rysavy Research, LLC contributed to the content of this paper.