



□ WHITE PAPER

Mobile Web Apps vs. Mobile Native Apps: How to Make the Right Choice

INTRODUCTION

For any enterprise that uses a website to build brand awareness and sell its products, the explosive growth in mobile devices is impossible to ignore. But while many companies would love to extend their website or e-Commerce application to a mobile audience, they're often uncertain about how to proceed. Should you build a dedicated application for each mobile device? Mobile-enable your current website? Build some sort of hybrid solution? What are some of the things you should be thinking about when bringing content and applications to your mobile community?

This paper examines a topic at the heart of any mobile strategy: [whether to deploy a native application or a web application](#). It examines the key characteristics of each approach, their advantages and disadvantages, and what factors go into making a decision. It also explores an issue that's top of mind for mobile planners and software developers: how to support multiple platforms and devices in an industry that's highly fragmented and rapidly evolving. Do you need to create a separate app for each Android™, iOS, Windows™, or Blackberry™? With the constant emergence of new platforms, versions, and form factors, how do you even keep up?

Lionbridge has extensive experience meeting all of these challenges, and we can help you navigate the rapidly evolving mobile landscape. In this paper, we'll explore ways to optimize for the capabilities and limitations of different devices while minimizing development time and costs. The paper also examines some customer scenarios that put these concepts into practice, including examples of the creative thinking that goes into delivering a mobile solution that users will embrace.

COMPARING MOBILE WEB APPS AND MOBILE NATIVE APPS

Mobile applications come in two distinct formats: native apps and web apps. Due to differences in their underlying technology, each approach has inherent advantages and drawbacks.



The key advantage of mobile web apps over native mobile apps is cross-platform compatibility...

Mobile Web Apps

A mobile web app is a web application formatted for smartphones and tablets, and accessed through the mobile device's web browser. Like a traditional web application, a mobile web app is built with three core technologies: HTML (defines static text and images), CSS (defines style and presentation), and JavaScript (defines interactions and animations). Since web apps are browser-based, they're intended to be platform and device independent, able to run on any web-enabled smartphone or tablet. A mobile web app is normally downloaded from a central web server each time it is run, although apps built using HTML5 (described below) can also run on the mobile device for offline use.

Advantages:

The key advantage of mobile web apps over native mobile apps is cross-platform compatibility, allowing them to reach the broadest audience for the least effort. They're relatively cheap, easy, and fast to build, although some device-specific customization is usually required. Mobile web browsers are fairly standardized, making it much easier to create a universal mobile web app than a native one (with the caveat that the web app targets Apple/RIM/Android—see Disadvantages below).

- + Web apps are also cheaper and easier to maintain than native apps for the same reasons, using cross-platform applets rather than keeping up with changes across different devices.
- + Simple, ubiquitous access: Users don't have to download an application, but simply access a URL via their mobile browser which instantly delivers the most up-to-date application to their device. They can then bookmark the URL for repeat use.

Disadvantages:

- + Although improvements are ongoing, mobile browsers have limited capabilities compared to traditional desktop browsers. Functionality is similar for the major players (Apple, Android, RIM, Windows), but falls off sharply on other platforms. Depending on what platforms need to be supported, the solution may be limited to the lowest common denominator, giving the app a “clunky” look, or require time-consuming customization across browser versions.
- + Web apps generally¹ cannot access the on-board hardware and software on a mobile device. Requirements such as camera control, direct GPS control (there is limited access to current location), PIM integration, or control of the phone app will rule out web apps right away. Heavy/complex custom graphics (gaming, etc.) also cannot be supported.
- + Web apps generally require a connection to function, with performance issues if the website is slow or unavailable. When users are on the go without Wi-Fi coverage, they have to pay for connection time to network operators. In some countries the cost is minimal, but in others can be a showstopper. (This changes with HTML5, which makes web app content available in offline mode so users can still access the app even if they do not have cell reception or connection to a Wi-Fi network.)

Native apps have a major advantage over web apps—the ability to leverage device-specific hardware and software.

Mobile Native Apps

A native mobile app is built specifically for a particular device and its operating system. Unlike a web app that is accessed over the internet, a native app is downloaded from a web store and installed on the device. Native apps are written in Java, Objective C, or some other programming language.

Native apps have a major advantage over web apps—the ability to leverage device-specific hardware and software. This means that native apps can take advantage of the latest technology available on mobile devices and can integrate with on-board apps such as the calendar, contacts, and email. However, this is a double-edged sword: while mobile technology is wildly popular, it is also constantly changing and highly fragmented. This makes the task of keeping up with the pace of emerging technology onerous and costly, especially on multiple platforms.

¹This is changing with HTML5, but functionality is inconsistent and incomplete. Blackberry is a special case, with the RIM operating system allowing application components built using web technologies to be compiled into a native app with access to all device features.

Advantages:

- + A richer, more compelling user experience: Native apps can leverage the capabilities of the mobile device, including onboard hardware (such as GPS, camera, and graphics) and software (such as email, calendar, contacts, picture/video gallery, file manager, and home screen widget areas).
- + Ability to run offline: Since the application remains installed on the device from the original download, no internet connection is required. Users get peak performance at all times, with all graphics, images, scripts and data. Data transfers can resume (and re-sync with back end apps) when the connection is restored.
- + Better “front of mind” penetration: Native apps place a logo in the application list screen, providing visibility on a daily basis. In addition, app stores remind users to upgrade apps, so apps that update frequently (with real improvements) are more frequently brought to the user’s attention.
- + Native apps are “hot” right now. Users are continually combing app stores for the latest app they can’t live without, giving products there a higher likelihood of being discovered.
- + Native apps are also easier to monetize. You set a price, list the app on an app store, and when users buy it, you make money immediately (minus the commission). With a web app, you would have to set up a payment or subscription gateway if monetizing the app is something you want to do.

Disadvantages:

- + Content publishers have to share information about their subscribers with the app store, an arrangement that frustrates publishers (particularly media companies).
- + For a native app to work across multiple devices, separate versions of the app are required. The fragmented nature of the mobile industry means that developing, testing, and porting apps for different environments costs money—particularly with maintenance and promotion costs. Cross-platform frameworks can make things easier, but time and cost remain a fact of life when developing for rich functionality on multiple mobile platforms. (The next section explores development issues for mobile apps in more detail.)
- + Keeping the application up to date also means more work for the enterprise, requiring development, testing, and distribution for different platforms rather than simply updating a single website.

Hybrid Apps: Native Apps with Web Apps Inside

In addition to building custom native apps for different mobile platforms or creating a single less-capable web app that works on any device via its browser, there’s also a third option: blending the two approaches in a hybrid app. With a hybrid app, much or all of the user interface appears in a browser window, with a native app wrapped around it to provide access to device functionality not available via the browser.

This ability to combine standard web apps with native code can significantly reduce development time and cost, minimizing custom coding work. To the user, a well-designed hybrid app looks very similar to a native app: it is downloaded from an app store, stored on the mobile device, and launched just like a native app. But to developers there is a huge difference, because rather than rewriting the entire app for each mobile platform, they write at least some of the code in HTML, CSS, and JavaScript, and reuse it across different devices.

Hybrid apps have their own design issues. The web-based and native functionality should blend together seamlessly, rather than drawing attention to the fact that the app contains web content presented in a browser embedded in a native app. (We'll see how Lionbridge addressed this challenge in one of the customer scenarios later in this paper).

The Wild Card: HTML5

HTML5, the latest version of the HTML language, could be a game-changer in the world of mobile apps. By supporting on-board functionality, it will allow for a much closer experience to the look and feel of a native app. It also has a local storage feature that allows it to run offline without a network connection, just like a native application. As of this writing HTML5 is supported on the big three platforms (iOS, Android, RIM). As developers and software vendors realize the potential of this new technology we will see new tools, frameworks, and methodologies catering specifically to it. In fact this is already happening with tools such as Titanium and Rhodes.

However, HTML5 is still under development and functionality remains somewhat limited and inconsistent. Different browsers interpret HTML5 differently, and are evolving at different timeframes. Off-line caching and on-board access to resources have also not been implemented in a standardized way. This current lack of consistency and standards means you may have to develop different HTML5 apps for different browsers or device classes.

Windows 8: Microsoft Responds to the Challenge

Naturally Microsoft isn't going to sit idly while a huge market takes shape around it. The software giant is responding to the growing shift toward mobile with its upcoming Windows 8 operating system, providing dramatic changes compared to previous Windows versions. Windows 8 is designed for a broad landscape of devices, with the ability to run on every conceivable form factor from PCs to smartphones and everything in between. It supports touch in addition to traditional mouse and keyboard input, automatically adapting to the environment of the device. In beta mode as of this writing, Windows 8 is already getting rave reviews—even from traditional Microsoft skeptics.

If its tactic succeeds, Microsoft could very well leapfrog the likes of Apple and Google. There are also other factors that could drive up Windows 8 market share: an army of developers who know Microsoft development technologies, and the excellent development frameworks from Microsoft that they're already familiar with. Microsoft has a history of outstanding come-from-behind victories (think Internet Explorer vs. Netscape) so this development is definitely one to watch. It's another indication that there's still no clear winner as different vendors battle for position—an obvious challenge for application development, but also an opportunity for enterprises able to build for multiple platforms.

HTML5 has enthusiastic supporters who claim it will become the universal standard for mobile platforms, as well as its critics who claim it never will match the capabilities of a native app. Of course, each faction has its own reasons for favoring a particular approach. For a company that just wants to engage in the best way possible with its mobile customers, HTML5 is something to watch closely as improvements take hold, without being overcome by marketing hyperbole. It has the potential to greatly improve the onboard capabilities (audio, visual, GPS, etc.) and is quickly closing the gap between web and native apps, and in the near future may bridge it entirely. This is an exciting prospect for stakeholders across the board in mobile development technology.

MOBILE DEVELOPMENT CHALLENGES

As mentioned earlier, one of the main challenges when moving to a mobile solution is software development in a technology landscape that's highly fragmented and rapidly evolving. Mobile apps require a fair amount of customization to run on diverse platforms and the steady stream of new hardware, OS versions and browsers. Even a single platform (Android, Windows, Blackberry, and to a lesser extent Apple) has numerous flavors that require some degree of customization. There are also other factors such as the overlay software from different manufacturers that can affect behavior of an app on a particular device.

In response, the mobile industry has spawned a rapidly growing ecosystem of cross-platform and cross-device frameworks², source code analyzers, libraries of reusable components, and other tools designed to accelerate and simplify multi-platform development. New tools are constantly emerging, with new functionality, different capabilities, and strengths and weaknesses. Developer preferences vary, particularly as new tools and capabilities become available. However, the basic themes don't change: to code less and accomplish more, to reuse and recycle across multiple platforms as much as possible, and consider developing from scratch as a last resort. In addition, any tool or framework should be able to work with current and future offerings, and not be locked into a particular platform or technology.

Web Apps

There are multiple tools to choose from (e.g. Pyxis™, Wapple™, MobileFrame™) that promise to let you build a mobile website in minutes, able to run on any mobile phone regardless of manufacturer or network. You select what elements of the original desktop site you want to migrate, preview how the web app looks on different devices, and then fine-tune for each device.

² The distinction between a cross-platform "framework," "development platform," "development environment," or "product" is tenuous at best. There's also overlap between the names of development products and the enterprises that provide them. For example, Titanium is a very popular cross-platform framework that is a product of Appcelerator. If you are using Appcelerator to build a multi-platform app, you are really using Titanium. Similarly Rhodes is a framework built and used by Rhomobile.

The process may seem straightforward, but creating an attractive, engaging experience can take considerable planning and experimentation. While web technologies provide a common architecture, results will vary with different screen sizes, device functionality, and operating environments. One major issue is support for a touch interface rather than a mouse one. Features such as hover pop-ups are not available, so you have to adjust the traditional web app accordingly. Also, the mobile web browser usually only supports a subset of the features of a PC browser, so a fair amount of time is spent finding alternative methods to convey and present data as well as re-designing for a more limited UI experience. Finally the resolution and screen real estate are significantly less, requiring care and consideration when laying out the web app.

Tablets are a special case: with more screen real estate, it can be tempting to overload the screen with too much data or a cluttered UI. Obviously, the form factor and screen resolution will require something in between mobile and PC while providing a touch-driven UI. It can be difficult to find a balance on this type of device.

Native Apps

Given the diversity of platforms and pressure to beat the competition to market, developers look for the fastest way to port native apps to different operating environments with minimal rework. Cross-platform frameworks (such as Sencha Touch™, Rhodes™, and Titanium) are used to build a platform-neutral core architecture for each of the environments in which the application will run (Android, iPhone/iOS, Blackberry/RIM, etc.). Platform-specific tools that come with the package are then used to customize for a particular platform and device.

Other tools (e.g. PhoneGap™, Appcelerator, Adobe Flash Builder™) strive to let you “write once and run everywhere,” with little or no customization required for different platforms. While this sounds like the Holy Grail of mobile development, in practice “write once” tools tend to fall short of their ambitions. There are simply too many variables, and you are inevitably relegated to

Getting Mobile Right

Whether building a native app or a smartphone-optimized website, mobile is the cutting edge of user interface design. New approaches for convenience and simplicity are required to provide a rewarding user experience on small screens and tiny keypads. The following requirements are typical, using a retail app as an example.

User Interface

- + Provide voice search, auto-fill-in, scanning
- + Enable one-touch checkout
- + Optimize for location-based searches
- + Trim navigation but ensure all product content (images, descriptions, ratings and reviews, tech specs, availability) can be surfaced
- + Ensure consistency between mobile app, tablet, PC, game console and TV experience
- + Support multiple streaming formats and compensate for small screen size
- + Extend to the next generation of mobile devices

Subscriber-Friendly Features

- + Automatic pause during incoming calls
- + Integrate with social media frameworks
- + Ability to gracefully handle in and out of coverage conditions

the “lowest common denominator” and making some hard choices about what functionality to support. Just as enterprises struggle to adapt to a rapidly evolving mobile landscape, framework and tool providers also struggle to keep up, with the result that they sometimes promise more than they can deliver.

Sometimes an app will need to access specific on-board functionality that can't be met by cross-platform development tools. This leads us to the final option, which is to build the application from the ground up for a specific mobile platform using the native operating system toolkit. While it takes longer (and costs more) when multiple platforms are involved, developers get complete flexibility to customize the look and feel. In addition, the risk of unknowns in porting can be ruled out.

Synchronizing Development with Testing

A major challenge in the mobile world is the unprecedented speed required to bring applications to market. Rather than the relative stability of the desktop PC, the average lifespan of a mobile device is now about one year, and mobile software is even less. Release cycles need to be measured in weeks, requiring Agile methodologies that enable development teams to define requirements, release software functionality, consume feedback, and make changes in extremely compressed timeframes.

This challenging environment also demands a new approach to software testing. Rather than waiting until development of a mobile app is complete, testing needs to be in step with development in order to detect and correct flaws as early in the project lifecycle as possible. Lionbridge has taken this tight coordination to a new level, using innovative tools and testing methodologies together with management frameworks that orchestrate projects end to end and reduce overall time/cost/risk.

MOBILE APP SOLUTIONS: THREE CLIENT SCENARIOS

Let's see how the issues described earlier play out in three mobile development scenarios, showing how Lionbridge analyzed project requirements and delivered effective solutions.

Scenario 1: Interactive TV and DVR Control (Native App)

This app was requested by a major cable media firm to allow subscribers to manage DVR, ISP, and inbox functions from a variety of smartphones. Capabilities would include delivery of live TV programming content (What's On), control over the subscriber's DVR, video streaming of movie and TV clips, access to subscriber emails from native email apps, integration between subscriber contacts and the device address book, and control of voicemails for VoIP phones. All of these individual capabilities would share a common overall architecture that could run on iOS, Android, and RIM devices.

Web App Considerations

The simplicity of a single code base for all required platforms would allow faster build time as well as potential support for additional platforms in addition to the three market leaders. However, the web app approach had one serious drawback: no support for integration with on-device email/contacts—a key client requirement. In addition, competitors were already introducing mobile apps with slick and rich user interface elements that mobile web apps could not match.

Native App Considerations

A native app would be able to support all requested features including integration with built-in PIM apps (email/calendar/contacts). It would also allow creation of an innovative and rich user interface to enhance the user experience, providing an opportunity to leapfrog the competition with a newer app with more advanced features. The downside: a native app would be more complex and take longer to build, requiring a separate code base for each platform.

Lionbridge Innovation

Because integration with built-in PIM apps (Email, Calendar, Contacts) was a “must have,” mobile web apps were ruled out. Since the native app route was selected, Lionbridge took a number of steps to mitigate the more complex development this type of solution would require:

- + While each platform had unique characteristics, the core architecture was very similar. Lionbridge split the platform-agnostic code from that which was platform-specific, minimizing dependencies and allowing for a more modular architecture that was easy to modify and extend.
- + A single communications protocol for all three mobile platforms eliminated the added complexity of handling different request types. Only slight code modifications were required to connect with each platform, avoiding redundant code for backend servers to know which platform was talking to them.
- + Since Android and RIM are both Java-based, they shared much of the same core code. This was facilitated by eliminating UI and platform dependencies on the core code and separating platform-specific APIs with an abstraction layer.

The result was a compelling, unique application that was rapidly ported to the major smartphone platforms while providing a consistent cross-device experience.

Development Stack

CATEGORY	TOOLS
Target Platform	iOS 3.1+, Android 2.1+, Blackberry 5.0
Source Code	Java, Objective C
Development IDE	Xcode, Eclipse
Source Control	Subversion™
Defect Tracking	Bugzilla

Scenario 2: Personal Health Record (Mobile Web App)

A major healthcare provider wanted to offer a mobile service providing up-to-date information about patients' conditions. Patients and authorized family members would be able to access many different types of personal health information, including lab results, messages from physicians about patient status, and home care instructions for patients and their loved ones. The app would also provide access to hospital bulletins, healthcare tips, upcoming events, and a hospital directory to help users find the services they need. The service was already available via a web portal for desktop users, and powered by an existing server infrastructure.

As a healthcare app, strict compliance was required with HIPAA privacy rules and other regulations. The app also had to be supported on a wide range of mobile device platforms so it could be used by the widest possible audience.

Lionbridge Innovation

Because functionality was straightforward, aimed at delivering relevant information in a format that was easy to read and use, a mobile app was the clear choice. There would not be significant gains in user experience from leveraging a native app, and the wide range of devices to support would mean a significant increase in development cost.

Existing web services built to support the web portal for desktops could be leveraged for the web app. These services needed adjustment to be usable on mobile devices but were otherwise portable. While the existing back end for the desktop web portal had similar functionality, there were also significant chunks of business logic handled on the client via JavaScripts. These scripts were portable for the most part, but they put an undue strain on smaller mobile devices. By leveraging the latest optimization techniques, Lionbridge was able to analyze and modify the scripts to make them friendlier to mobile devices, without the need for rewrites or developing concurrent mobile scripts.

Development Stack

CATEGORY	TOOLS
Framework	Appcelerator Titanium™ 1.6
Target Platform	WebKit enabled device browsers
Source Code	JavaScript, HTML5, CSS3
Development IDE	Eclipse
Source Control	Subversion
Defect Tracking	Bugzilla

Scenario 3: One App, Five News Properties, Two Device Platforms (Hybrid App)

A large media organization wanted to provide access to a number of their online news properties using iOS and Android mobile devices. The following capabilities were required:

- + Keyword search
- + Availability of updated content from multiple news sites
- + Support for inline images and videos
- + Ability to save content to the device's local storage
- + Ability to share content via email/messaging
- + Integration with social media (ability to share via Facebook, etc.)
- + Community commentary support

Lionbridge Innovation

After analyzing the requested features and functional requirements, Lionbridge proposed a single application framework that would support all news properties on both platforms. This type of solution was feasible since there were very few differences in functionality between the sites. However, it required careful planning with minimal cross-component dependencies so functionality could be added for individual properties without disrupting the overall architecture. For example the financial news site required a stock ticker unique to that particular app—a feature that was easy to add because of the modular design.

Lionbridge developed an agnostic core architecture that minimized duplication of effort when porting between the iOS and Android platforms. The underlying design on the two platforms was similar enough that a developer familiar with one platform architecture could easily switch to the other and be in familiar territory. The design also made it easier for a single developer to add fixes/updates/changes to all platforms without having to hand off work to another developer.

A Hybrid Solution

The client required a native app because they wanted to leverage app stores as a new distribution channel. However, the content was already being served up from the client's website. What to do?

Lionbridge developed a hybrid approach, with a web app in a browser window with a native app wrapped around it to provide access to on-board device functionality. Web content runs within the browser frame, while all navigation, sharing, storage, and other device-specific functions are handled by the surrounding app. The approach provides the look and feel of a native app but delivers the heart of the application within the browser. The result was a much less complex app, with all heavy lifting of streaming videos, formatting images, and smart text (e.g. recognizing embedded email addresses, websites, and phone numbers) handled in the browser frame. The resulting combination app could then be submitted to an app store just like a regular native app.

Development Stack

CATEGORY	TOOLS
Target Platform	iOS 3.1+, Android 2.1+
Source Code	Java, Objective C
Development IDE	Xcode, Eclipse
Source Control	Subversion
Defect Tracking	Bugzilla

KEY TAKEAWAYS

With their platform-independent architecture and broad access, mobile web apps cost less to reach a wider audience. Native apps provide a richer user experience and stronger engagement, but can cost significantly more. Delivering this rich experience while minimizing cross-platform complexity and cost is a major challenge in mobile development projects. Beyond these inherent differences, here are some other key points to remember:

It's Not "Either/Or"

Native apps and web apps are not mutually exclusive; instead they complement each other and should both be part of the approach an organization uses to engage with its mobile customers. If you already have a web presence, you should also provide a mobile-optimized version, even if it's very basic. The question then becomes: do you still need a native app? If so, what shape should it take, and what are the issues? Even when your mobile roadmap is focused on rich native apps, it should include at least a basic web experience because of the almost universal coverage available for a relatively minor expenditure of time and effort.

For companies looking to enter the mobile space for the first time, the easiest way to start is by seeing how their website looks on one mobile device and then optimize as needed. There are also advantages to first designing around the mobile web: it allows a company to quickly test what works and what doesn't, and discover what mobile platforms are most commonly used by the audience they're trying to reach.

Development Cycles Measured in Weeks

With new mobile offerings emerging constantly, companies and their technology partners have to be careful not to get caught off guard. Complex long-running development projects have been known to exceed the lifespan of the products they were built for. There have already been fiascos where a product that seemed advanced at the start of a project was superseded by something more exciting by the time an application was released. (For example, even iPads were unaccounted for in some long-term development efforts.)

Accordingly, the application development cycle should be capped at about 12 weeks from whiteboard to release, and shorter if possible. This means you need to reuse and recycle existing frameworks whenever possible and consider building from scratch as a last resort. It also requires Agile processes, able to build for the latest technology with iterative rollouts, and testing that starts as early in the development cycle as possible.

Platforms Will Stay Fragmented (At least for a while)

Fierce competition between platforms, carriers, and manufacturers generates a strong drive to “differentiate” by creating unique technologies and standards, which only feeds the fragmentation problem. As mobile technology continues to accelerate in terms of innovation, capabilities, and network bandwidth, we will see an increasing need for a true solution able to bridge the fragmented landscape. HTML5 is the most promising proposition to date. However, like many potential silver bullets before it, there is a significant likelihood that HTML5 will be supplanted by another technology that more completely addresses multiple operating environments. While some platform convergence is possible, customer loyalties, intense competition, and widely varying requirements mean that multiple platform architectures—and the need to develop for them—will remain a fact of life, at least for the next few years.

Both Models Will Coexist

Native and web apps will coexist for the foreseeable future, even as both technologies continue to evolve and in some areas converge (think HTML5 and hybrid). Mobile browsing technologies will continue to improve as HTML5 matures, supporting web apps with richer, more native-like user experiences. Native apps will remain the choice for engaging users with the richest experiences that take advantage of the latest onboard functionality. However, for many other types of apps, the economics of software development will increasingly favor the web development route.

LIONBRIDGE MOBILE TECHNOLOGY SERVICES: DEVELOPING FOR THE PLATFORMS OF TOMORROW

- + Ground-up application development, from whiteboard to release
- + Portals and applications extended to mobile platforms
- + Existing mobile apps ported to new platforms
- + Experienced staff able to scale to any needs
- + Acceleration tools and management frameworks
- + Centers of excellence and vast testing network
- + In-country translation and localization, with tight quality control

In the fast-paced world of mobile development, speed to market with a superior product is essential. That's why enterprises worldwide depend on Lionbridge Mobile Technology Services to succeed in this dynamic environment. We provide technical expertise, global resources, and innovative project management to deliver compelling solutions for any mobile development challenge.

The Lionbridge approach helps clients rapidly scale up their mobile development initiatives while keeping a lid on costs. We use our Lionbridge Mobile Engineering Solutions framework to orchestrate resources worldwide, providing tightly synchronized workflows for cross-platform development, testing, and market-enablement. Blended onshore/offshore ("rightshore") engagements provide access to world-class engineering talent along with local management and accountability. We're also mobile development innovators, with cross-platform tools and reusable components that accelerate speed to market while reducing coding and porting requirements.

Lionbridge operates development and testing centers of excellence in strategic locations worldwide, backed by a global crowdsourced community of more than 100,000 testers for in-market usability testing. Globalization is in our DNA, and our in-country language professionals provide high-quality translation and localization services in 92 distinct geographies.

Please contact Lionbridge to learn how we can help you execute a successful mobile strategy for your organization.

Product Engineering Hotline

+1 877.342.5334 (toll-free in U.S.)

+1 503.305.8732

product.engineering@lionbridge.com

www.lionbridge.com/pes

ABOUT LIONBRIDGE

Lionbridge is the leading provider of translation, development and testing solutions that enable clients to create, release, manage and maintain their technology applications and Web content globally.

The Lionbridge product engineering group develops, re-engineers and optimizes software products, as well as tests to ensure quality, interoperability, usability, relevance and performance of clients' software, search engines, consumer technology products, web sites, and content. Lionbridge testing services, which are offered under the VeriTest brand, also include product certification and competitive analysis.

Lionbridge also provides specialized global crowdsourcing solutions for clients with international search engines and online marketing initiatives. To deliver these solutions, Lionbridge identifies, recruits and administers custom professional communities in over 100 local markets. We combine our global community of qualified freelancers with full program management, quality assurance and analytics to ensure our clients' search results and online marketing programs are consistent with customer expectations in international markets.

Fast Facts

Founded: 1996

Headquarters: Waltham, Massachusetts, United States

Employees: 4,200

Locations: Solution Centers in 26 countries

Revenue: \$405 million (2010)

NASDAQ: LIOX