Worklight Mobile Platform 4.2

Technology Overview

Contents

The Worklight Mobile Platform – Overview

Worklight allows organizations to extend their business through HTML5, hybrid and native apps for a variety of smartphones and tablets, and support the entire application lifecycle from development, through back-end integration and authentication, to post-deployment management.

Development: The Worklight Studio and the Worklight SDK simplify the development of web, hybrid and native apps across multiple mobile platforms, including iOS, Android, BlackBerry and Windows Phone. Worklight's optimization framework enables the delivery of a rich user experience that matches the user expectations and styling requirements of multiple target environments. In the process, Worklight maximizes the sharing of the code-base from one environment to the other, effectively reducing costs of development, time to market and ongoing management efforts.

Integration: Worklight's server architecture and Adapter technology simplify the integration of applications with back-end enterprise systems and cloud-based services. The Worklight Server has been designed to seamlessly fit into the organization's IT infrastructure and leverage its existing resources wherever possible. The standalone back-end integration layer can be customized and shared among multiple applications. Furthermore, Worklight Adapters support two types of data delivery mechanisms: device requests and Push notifications.

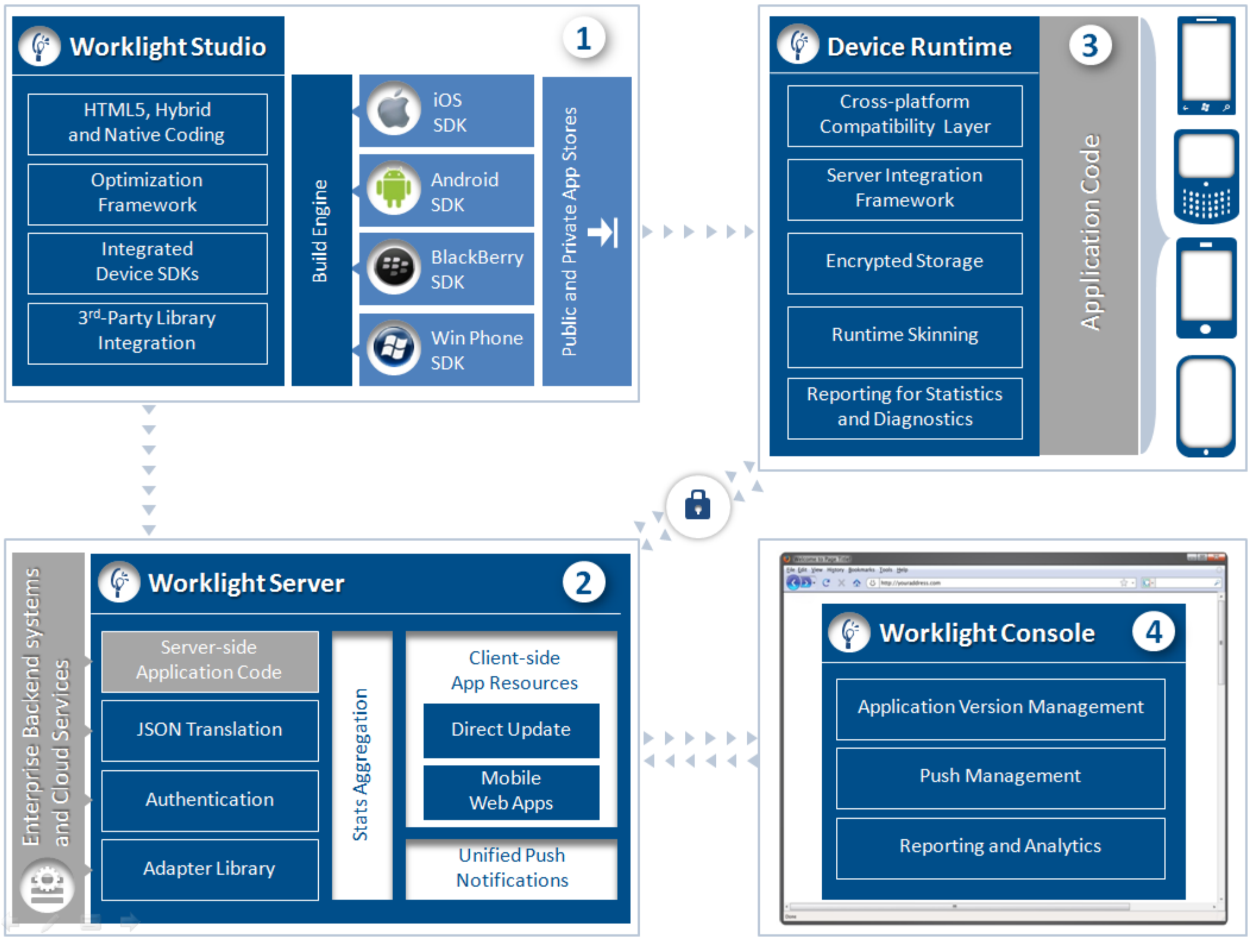
Runtime: The Studio prepares application files for upload to the various public app stores and private distribution repositories. Running applications communicate securely with any enterprise back-end systems and cloud-based services through the Worklight Server, which secures sensitive services and optimizes data for mobile delivery.

Manage: Once deployed, administrators can monitor and control the access of different apps and devices to back-end systems; directly update and disable them based on predefined rules; and control all push services and event sources from one centralized web interface called the Worklight Console. In addition, administrators can access analytics information about the installed app base and its users using built-in and customized reports.

The Worklight Mobile Platform – Components

The Worklight architecture consists of four main components;

- The Worklight Studio – the platform's IDE
- The Worklight Server – a secure gateway between apps, back-end systems and cloud services
- Worklight's Device Runtime Components – complementing the Server with client side functions
- The Worklight Console – a web-based administration interface.


Worklight Studio
1
HTML5, Hybrid and Native Coding
Optimization Framework
Integrated Device SDKs
3rd-Party Library Integration
Build Engine
iOS SDK
Android SDK
BlackBerry SDK
Win Phone SDK
Public and Private App Stores
Device Runtime
3
Cross-platform Compatibility Layer
Server Integration Framework
Encrypted Storage
Runtime Skinning
Reporting for Statistics and Diagnostics
Application Code
Enterprise Backend systems and Cloud Services
Worklight Server
2
Server-side Application Code
JSON Translation
Authentication
Adapter Library
Stats Aggregation
Client-side App Resources
Direct Update
Mobile Web Apps
Unified Push Notifications
Worklight Console
4
Application Version Management
Push Management
Reporting and Analytics

Development Tools

The Worklight Studio

The Worklight Studio is an Eclipse-based Integrated Development Environment (IDE) that allows developers to perform all the coding and integration tasks that are required to develop rich employee- and customer-facing applications. The Studio augments the familiar tools of Eclipse with a wide variety of enterprise-grade features that are delivered by the Worklight Plug-in, enabling it to streamline application development and facilitate enterprise connectivity.

The following are some of the main features that are supported by the Worklight Studio:

Cross-platform Support

The Worklight Studio enables the development of rich hybrid and native mobile applications on iOS, Android, BlackBerry and Windows Phone smartphones and tablets. Mobile HTML web apps can be developed for a wider array of modern devices using web browsers capable of running HTML 4, CSS 2.1 and JavaScript 1.5 at a minimum.

Using its Optimization Framework, Worklight differentiates itself from other technologies in the market that deliver a lowest-common-denominator solution, by allowing developers to share the majority of the application code across multiple environments without compromising the user experience or the application functionality.

Using a unique file structure within the Studio, the Optimization Framework enables developers to adjust applications to the different mobile environments, while maximizing the reuse of the code-base. Developers can share the common app code among multiple environments, while isolating environment-specific code in designated folders that can overwrite or augment the commonly shared code. As a result, application logic remains consistent among the different environments, while the user interface behaves natively and adheres to end-user expectations.

Application developers can directly access all of the APIs that modern devices offer, as well as easily integrate publically available or customized 3rd-party libraries, frameworks and tools, resulting in advanced mobile applications built according to the unique and specific needs of the organization.

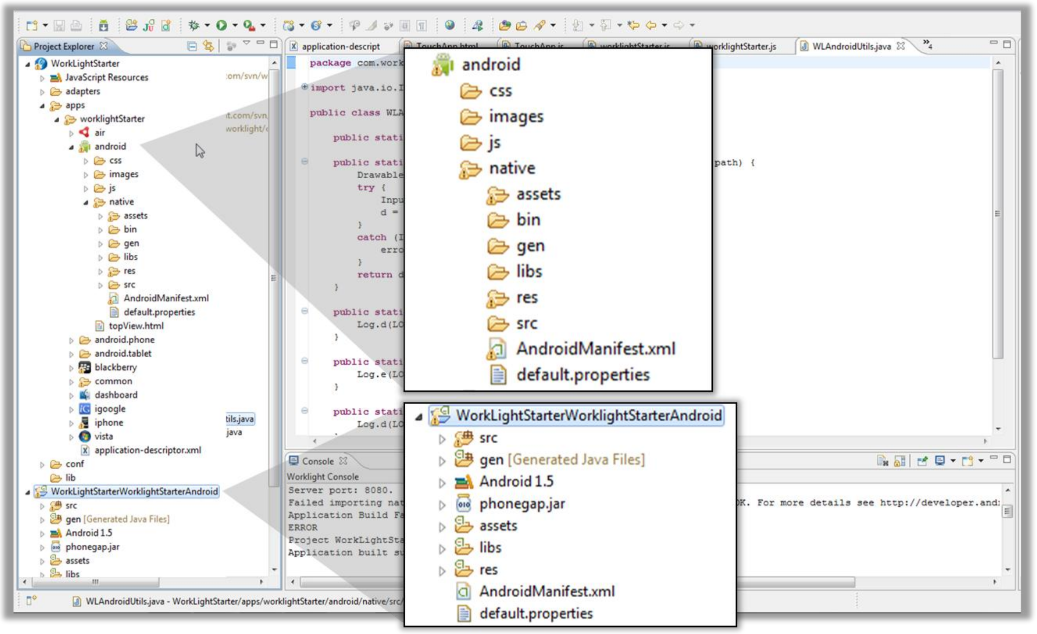
Because developers are not dependent on an intermediary build- or run-time layer, such as a cross-compiler or interpreter, using Worklight, native APIs are immediately accessible upon release of new mobile OS versions or 3rd-party libraries. Furthermore, the application's web code is executed directly by the mobile browser, so developers have direct access to the HTML DOM and are free to use any JavaScript API or third-party JavaScript toolkits and frameworks.

Hybrid Coding

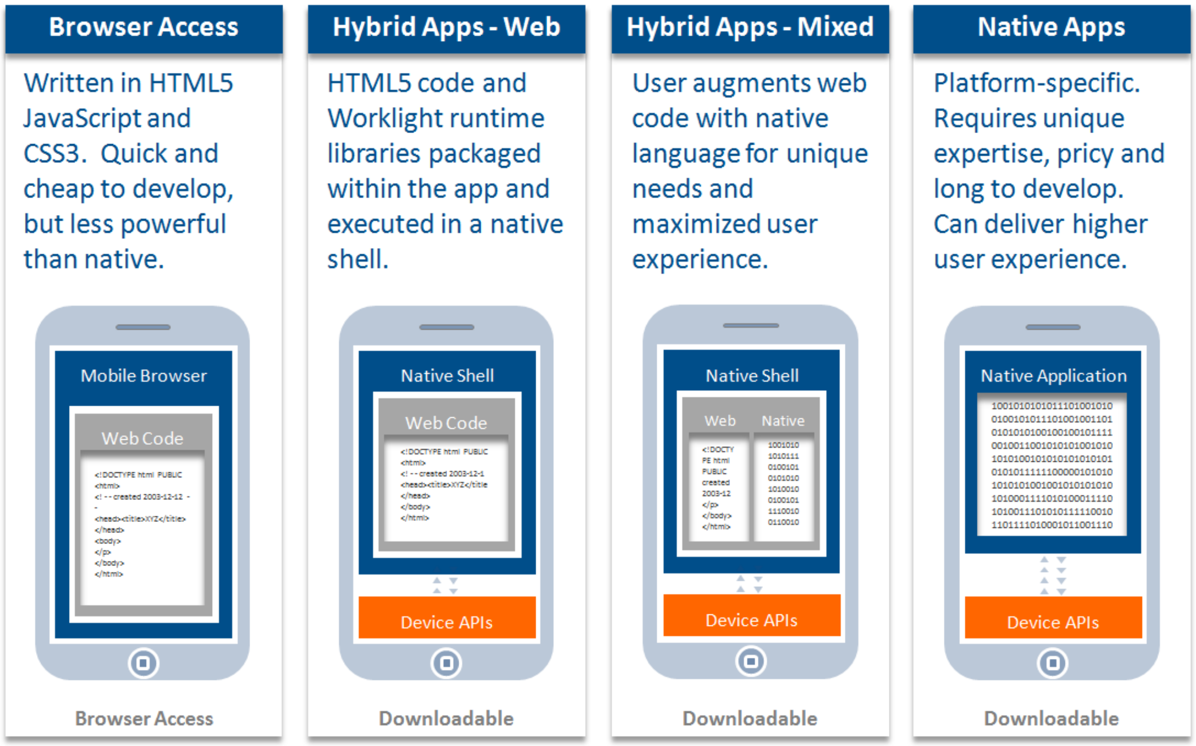
Facing the constantly evolving ecosystem of mobile devices and operating systems, application development has become a costly, yet unavoidable endeavor for the modern business. This challenge has created a market for cross-platform mobile development solutions that is rapidly growing.

But to achieve cross-platform capabilities, many solutions in the market relay on limiting proprietary tools, such as form creators, WYSIWYG tools, or simply pre-packaged apps. The result is an unavoidable tradeoff between user experience and multiplatform coverage. Using Worklight, developers can choose between using pure native code (Objective-C, Java, or .NET), standard web technologies (HTML5, CSS3 and JavaScript), or a combination of both within the same app, allowing them to strike the perfect balance between development efficiency and app functionality.

Developers can call native code from HTML-based pages, combine HTML and native-based pages in the same application, as well as display HTML and native components together on the same page.



No single development approach is a silver bullet, but using Worklight's unique support for Hybrid Coding, organizations are able to use the same mobile platform to develop, run and manage a variety of application types based on the specific needs of the project at hand.



Browser Access: Written in HTML5 JavaScript and CSS3. Quick and cheap to develop, but less powerful than native.

Hybrid Apps - Web: HTML5 code and Worklight runtime libraries packaged within the app and executed in a native shell.

Hybrid Apps - Mixed: User augments web code with native language for unique needs and maximized user experience.

Native Apps: Platform-specific. Requires unique expertise, pricy and long to develop. Can deliver higher user experience.

Runtime Skinning

Further optimization of apps is possible within the Worklight Studio by utilizing Runtime Skinning. These skins are applied to the mobile app during run-time and allow the app to automatically adjust to different devices from the same OS family. Common scenarios that benefit from Runtime Skinning:

Support for HTML5

Worklight’s support for HTML5 is unique. Leveraging its open approach, developers can write HTML5 code directly into the app without the limitation of proprietary interpreters or code translators, harness its full power and benefit from capabilities such as:

- A cleaner, more readable and consistent HTML code.
- Access to rich media types (audio and video) available before via native code only.
- Use of advanced UI components such as data pickers, sliders, edit boxes that automatically support ellipsis and others, implemented natively by the browser.
- Use of CSS3 styles and CSS3-based animation to reduce app size and improve its responsiveness.
- Application distribution channels that go beyond the different app stores and their time-consuming and limiting restrictions.
- Support for geo-location services.
- Offline storage capabilities.

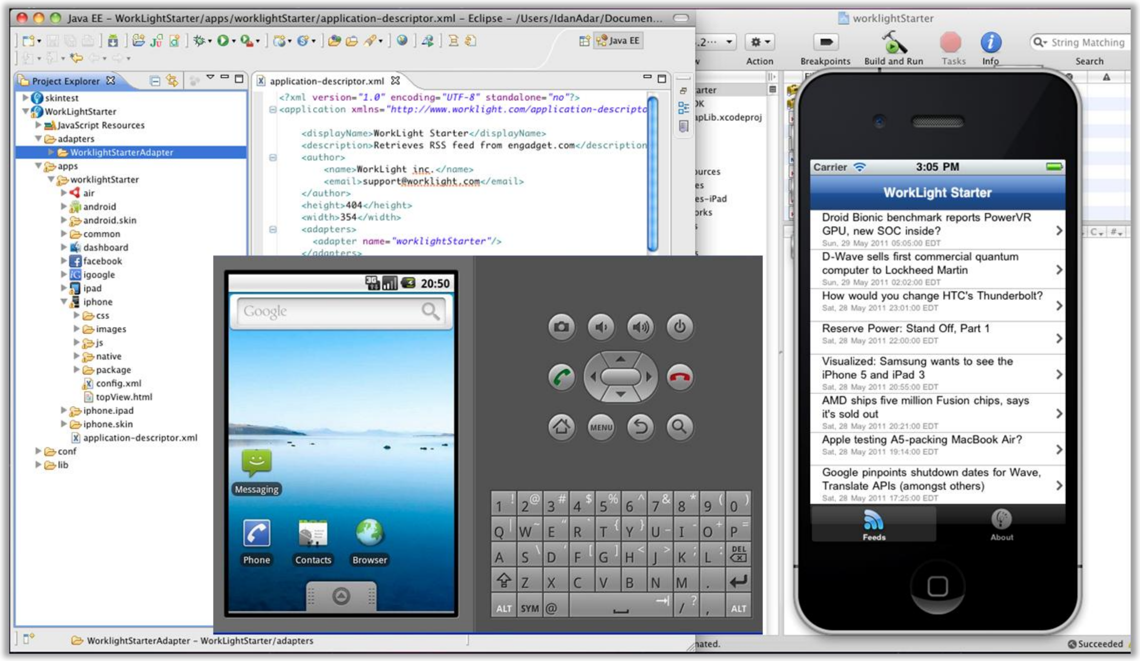
Worklight further augments these capabilities with enterprise-grade utilities such as on-device encryption and offline user authentication to increase application availability.

Support for 3rd-party JavaScript Toolkits and UI Frameworks

On top of Worklight's support for HTML5, the platform provides seamless integration with UI frameworks such as jQuery Mobile, Sencha Touch, and dojox.mobile. Developers can pick the JavaScript UI framework of their choice and use it to develop their hybrid application within the Worklight Studio.

Native Device SDK Integration

The Worklight Studio tightly integrates with the SDKs of the mobile devices that Worklight Platform supports. This allows developers to fully utilize the native code capabilities and the best-in-class development tools, testing and debugging mechanisms that are indigenous to the mobile device without leaving the development environment.



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<application xmlns="http://www.worklight.com/application-descripto
    
    <displayName>WorkLight Starter</displayName>
    <description>Retrieves RSS feed from engadget.com</description
    <author>
        <name>WorkLight inc.</name>
        <email>support@worklight.com</email>
    </author>
    <height>404</height>
    <width>354</width>
    <adapters>
      <adapter name="worklightStarter"/>
    </adapters>
```

Droid Bionic benchmark reports PowerVR GPU, new SOC inside?
Sun, 29 May 2011 05:05:00 EDT

D-Wave sells first commercial quantum computer to Lockheed Martin
Sun, 29 May 2011 02:02:00 EDT

How would you change HTC's Thunderbolt?
Sat, 28 May 2011 23:01:00 EDT

Reserve Power: Stand Off, Part 1
Sat, 28 May 2011 22:00:00 EDT

Visualized: Samsung wants to see the iPhone 5 and iPad 3
Sat, 28 May 2011 20:55:00 EDT

AMD ships five million Fusion chips, says it's sold out
Sat, 28 May 2011 20:21:00 EDT

Apple testing A5-packing MacBook Air?
Sat, 28 May 2011 19:14:00 EDT

Google pinpoints shutdown dates for Wave, Translate APIs (amongst others)
Sat, 28 May 2011 17:25:00 EDT

Standardized Data Retrieval

The Worklight Studio enables developers to use XSL transformations and JavaScript code to convert retrieved hierarchical data from any back-end system to JSON format, thus preparing it for application consumption. Developers can invoke back-end services directly from within the Studio and receive raw results in XML, or processed results (after having converted to JSON using XSL transformations and JavaScript) in JSON format.

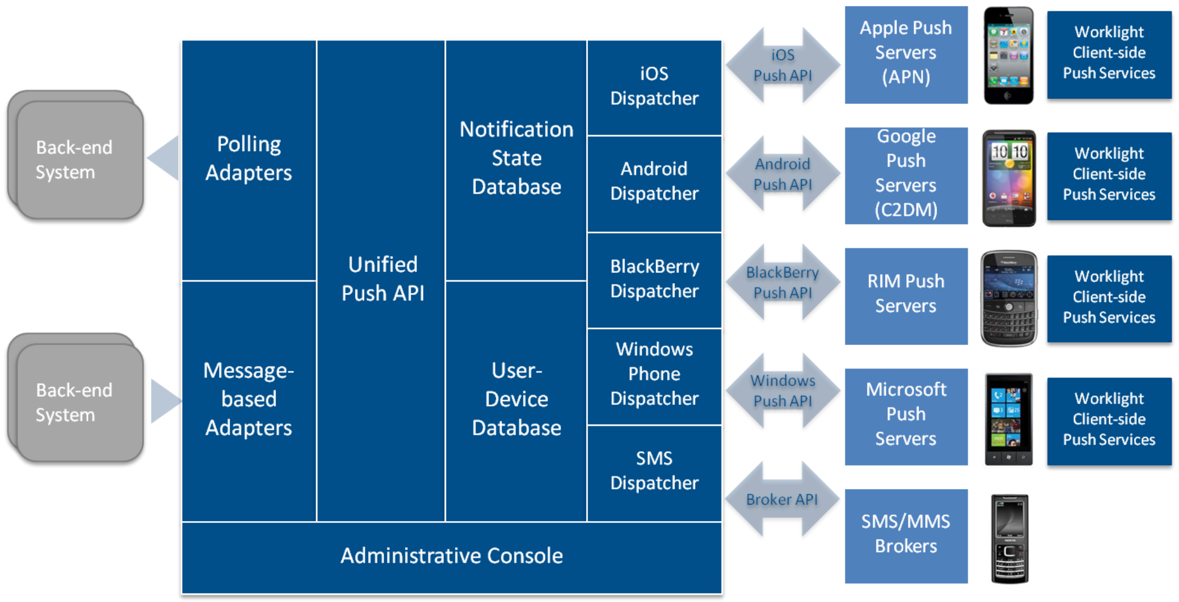
Developers can perform server-side mashups in JavaScript to collect data from various back-end applications and streamline them to the device, thereby reducing the number of requests made on the slow mobile network and greatly improving app responsiveness.

Furthermore, developers can also choose to implement server-side back-end integration and authentication code in Java, rather than in JavaScript.

Unified Push Notifications

In the process of creating the integration adapters, users can utilize Worklight's uniform push architecture to pre-configure automatic alerts from one centralized interface. Leveraging Worklight's

Unified Push API for its supported devices, the entire process of communicating with the mobile vendor becomes completely transparent to the developer.

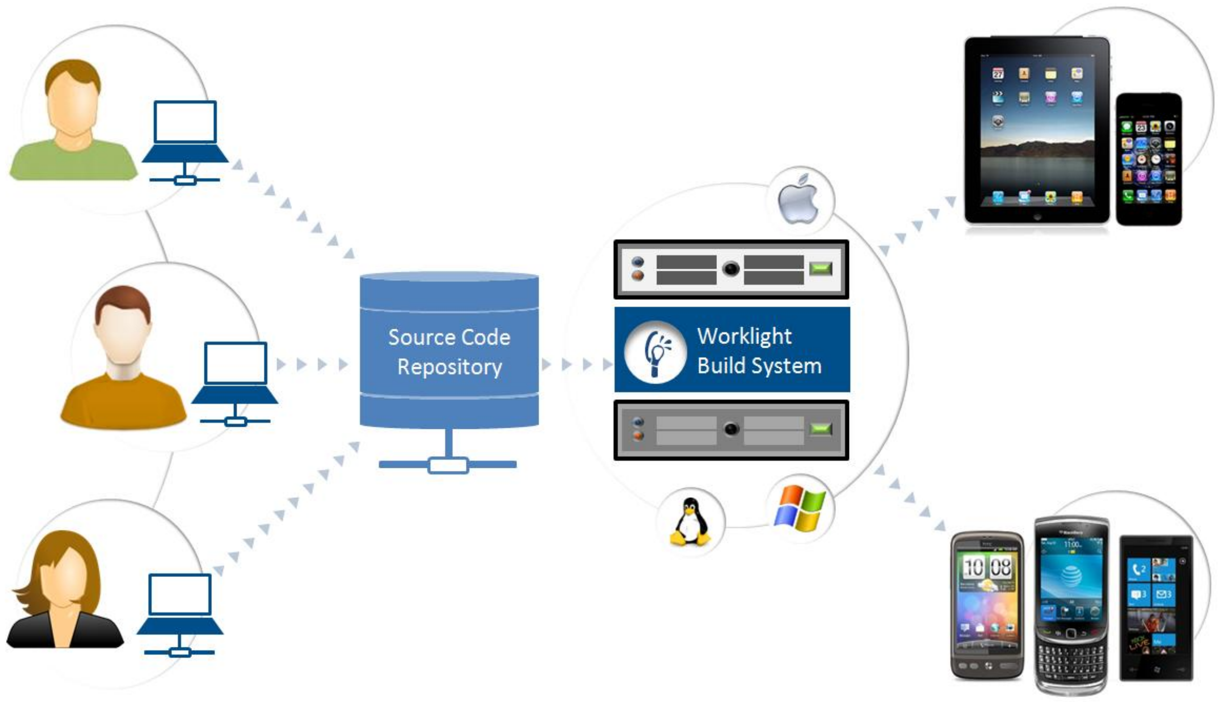


Distributed Development

Enterprise mobile development is rarely a simple process conducted by one developer. Most commonly, the complex enterprise development environment consists of multiple development, testing and QA teams all working on different portions of the app, sometimes even from different geographical locations. The Worklight Platform has been designed to fully support such scenarios through a variety of features and functions.

Centralized Build

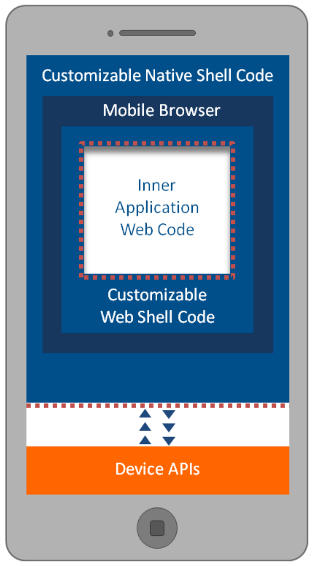
The Worklight Builder is a standalone application that can be easily integrated with common central build services such as IBM’s Rational Jazz Builder, Hudson and Luntbuild. Leveraging the Centralized Build, the different teams involved in the development, testing and QA phases can work off of one common version of the code, effectively enhancing the collaboration and automation of the internal application development process.



The Shell Approach

Most often, enterprises employ multiple development teams with different skills and expertise. The Shell approach allows such companies to reduce the internal barriers of mobile development, making it ubiquities across the organization by compartmentalizing skill-sets and responsibilities.

Worklight's Shell approach breaks down the development of the app into two portions: an external shell and an inner application.


Customizable Native Shell Code
Mobile Browser
Inner Application Web Code
Customizable Web Shell Code
Device APIs

The shell consists of a customizable container that provides JavaScript access to the native capabilities of the device. A devoted team of expert developers are responsible for its branding, security configurations, audits and authentication frameworks. The team can create a variety of shells, each carrying different policies and branding, forcing inner apps running within each shell to automatically comply with its parameters.

Such parameters could include restriction of access to data, use of certain APIs, different branding and more.

With the corporate policies enforced by the shell, the inner applications can be easily built by departmental development teams using nothing but web languages. Such teams are only required to focus on the user interface, the business logic and, potentially, data integration Distribution of the application or applications can be achieved by three different channels:

- An inner application can be fused into a shell by the centralized build server and uploaded to a private or public application store while new versions of the inner app are sent and updated directly on the end-user device.
- A shell can be packaged with a directory of corporate-sanctioned applications, allowing the user to choose a different inner app according to his or her needs.
- A shell can be distributed empty to the user who will then access a repository of applications stored on the Server.

The Worklight SDK

The Worklight SDK is a set of libraries and tools that support the development process and simplify integration of client-side code with back-end systems, cloud-based services and authentication mechanisms. The Worklight SDK consists of:

- JavaScript client API, providing access to Worklight services, environment-specific features and a set of cross-platform user interface utilities.
- Objective-C client API, providing access to Worklight services for iOS applications that were developed using Objective-C.
- Java client API, providing access to Worklight services for Android OS applications that were developed using native Java code.
- JavaScript server API, enabling pre-processing of requests from applications before forwarding them to back-end systems, post-processing of back-end data before sending it to applications and mashup of back-end data from multiple sources to reduce traffic loads and latency on the mobile network thus improving app responsiveness.
- Offline Access API, allowing developers to specify access failure routines, check connectivity state during runtime and automatically re-establish connectivity.
- Server- and client-side Unified Push API that support the dispatching of notifications, registering to push services and providing unified management.

- Localization API to detect the locale of the mobile device.
- Validating schemas for adapter configuration.
- Full documentation of the development, integration processes and of all APIs.
- Complete training material including presentations, exercises, and sample code.

Run-time Server Environment

The Worklight Server

Worklight's Java-based Server is a secure and scalable gateway between apps, external services, and the Enterprise. The Server facilitates secure connectivity, data manipulation, authentication, analytics and operational management functions. The Worklight Server:

- Provides adapter technology that connects to a variety of enterprise information systems over widely used integration technologies, such as SOAP, REST, SQL, LDAP, SAP and more, as well as cloud-based services.
- Enables multi-source data mashups to efficiently integrate several data streams into one and serve it to the application user. Multi-source data mashups are not only an effective way of servicing the right data to the user but also reduce overall traffic in the system.
- Allows developers to add custom server-side logic necessary for delivering back-end data for mobile consumption. This not only helps distribute processes between the client and server but also help address data security regulations within the organization.
- Integrates with the corporate authentication infrastructure to secure application and data access as well as transaction invocation. The Worklight authentication infrastructure is flexible enough to support different types of authentication – from multi-factor or multi-step login processes to non-interactive SSO integration as well as offline authentication of users to increase app availability. Furthermore, the Worklight Server simplifies the integration with HTTP-based services that require authentication. Integration with Kerberos, NTLM, Basic, and Digest authentication can be easily achieved by simple configuration of the HTTP adapter, without having to write server-side code. The Server also supports device-based application SSO, enabling apps to be automatically authenticated if an existing authenticated session is already available through the same mobile device.
- Employs standard and proprietary security mechanisms to thwart attacks (more about security).

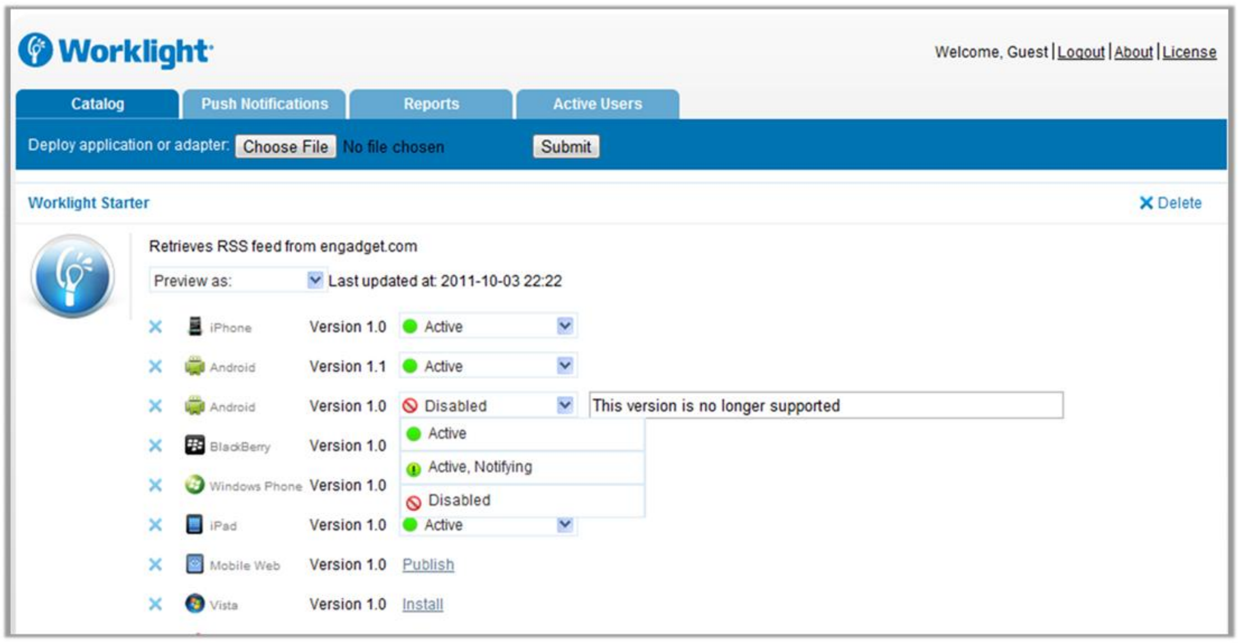
- Can easily scale to support millions of users and multiple applications through physical clustering.
- Provides application deployment and version control features that are managed and accessed by the Worklight Console.
- Can be integrated with IT monitoring and performance management systems that verify the vitality of the Worklight Server and the services it provides to applications.
- Automatically collects user adoption and usage data for auditing and reporting purposes and allows for the custom configuration of reporting metrics. Raw data can be easily exported for further analysis by the different BI tools used by the organization.

The Worklight Console

The Worklight Console is a web-based user interface dedicated for the ongoing administration of the Worklight Server and its deployed apps, adapters and push notification services. Through the Console, administrators can:

- Access administrative dashboards that monitor all deployed adapters and applications.
- Control and monitor all push notification services, event sources and related applications
- Manage multiple versions of the same application as well as remotely disable applications by version and mobile operating system type.
- Access built-in reports of application adoption and usage.
- Access the Worklight log and view key performance indicators (coming soon).
- Quickly download comprehensive system info for troubleshooting purposes (coming soon).
- Define device-based access control policies to control access of apps

Catalog Tab:


Welcome, Guest | Logout | About | License

Catalog | Push Notifications | Reports | Active Users

Deploy application or adapter: Choose File No file chosen Submit

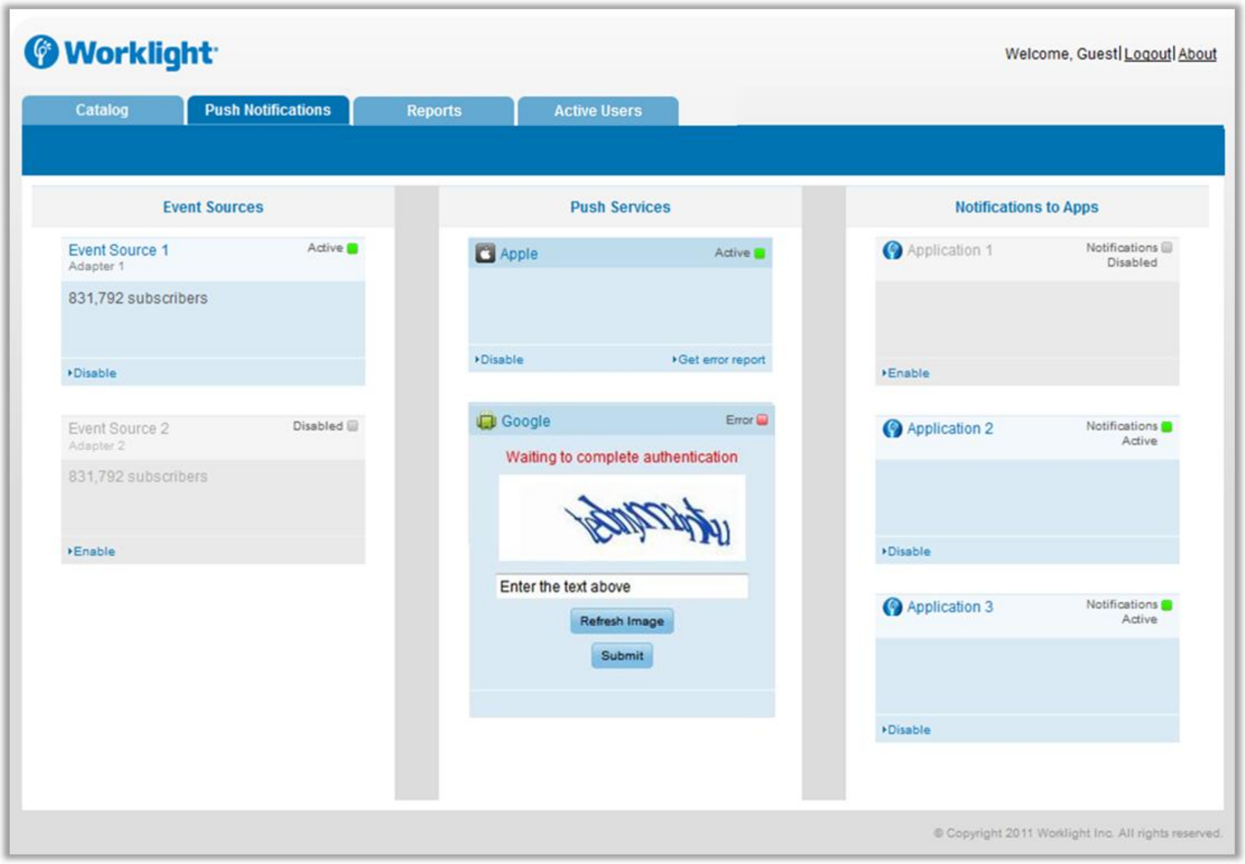
Worklight Starter ✕ Delete

Retrieves RSS feed from engadget.com

Preview as: Last updated at: 2011-10-03 22:22

✕	iPhone	Version 1.0	● Active	
✕	Android	Version 1.1	● Active	
✕	Android	Version 1.0	Disabled	This version is no longer supported
✕	BlackBerry	Version 1.0		
✕	Windows Phone	Version 1.0		
✕	iPad	Version 1.0	● Active	
✕	Mobile Web	Version 1.0	Publish	
✕	Vista	Version 1.0	Install	

Push Management Tab:


Welcome, Guest | Logout | About

Catalog | Push Notifications | Reports | Active Users

Event Sources
- Event Source 1 — Adapter 1 — Active 🟩 — 831,792 subscribers — Disable
- Event Source 2 — Adapter 2 — Disabled — 831,792 subscribers — Enable

Push Services
- Apple — Active 🟩 — Disable — Get error report
- Google — Error 🟥 — Waiting to complete authentication — Enter the text above — Refresh Image — Submit

Notifications to Apps
- Application 1 — Notifications Disabled — Enable
- Application 2 — Notifications Active 🟩 — Disable
- Application 3 — Notifications Active 🟩 — Disable

© Copyright 2011 Worklight Inc. All rights reserved.

Device Run-time Components

Worklight provides client-side run-time code that services HTML5, hybrid, or native application.

- **Access back-end data and transactions**: API for the invocation of Worklight services, retrieval of data and execution of transactions against back-end systems.
- **Authentication and security**: API and code for managing the authentication sequence and securing the application data and its link to the Worklight Server.
- **Application Management**: API and code for applying new application versions disabling applications according to policies defined in the Worklight Console.
- **Troubleshooting**: Code for detecting run-time connectivity problems in the app and collecting troubleshooting info about the app and the device
- **Usage reporting for audit and analytics**: API for collecting built-in and custom data from applications, to be recorded by the Worklight Server for audit and analytics purposes.
- **Cross-platform compatibility APIs**: Uniform API for device features and useful user interface tasks, hiding the differences across different environments.
- **Application of skins:** Allowing developers to adjust the look and feel of the app to the device's form factor in run-time, thus optimizing it for different versions of the same device family.

The run-time client environment consists of the following components:

- **JavaScript libraries**: A set of JavaScript libraries implementing the JavaScript APIs. These libraries are available in all run-time environments (with the exception of native iPhone and Android apps which are written in Objective-C and Java respectively and do not require JavaScript libraries).
- **Native libraries for hybrid apps**: A set of native libraries (for iOS and Android) that provide access to device-specific features. Apps written in JavaScript do not access these libraries directly, but through the relevant JavaScript APIs. In some cases, native code runs the web code provided by the developer.
- **Native libraries for native apps**: A set of native libraries for iOS and Android that provide access to Worklight Server functionality for natively written applications.

- **Native code templates**: For iOS, Android, BlackBerry, and Windows Phone devices, native code templates encapsulating a browser that runs the web code provided by the developer.

Security and Authentication Mechanisms

Worklight has been field-tested and proven to address the strict security standards of major financial institutions around the world. The system provides multiple mechanisms and tools that support the creation of secure applications.

The following is a list of the main security features of the platform:

Mechanism	Benefit	Details
On-device Encrypted Storage	Protect sensitive information from malware attacks and device theft	• Uses AES256 and PCKS #5-generated encryption keys for storing app-generated information on the device. • Allows offline user authentication. • Implemented in JS (highly obfuscated) with optional native performance enhancements.
Direct Update	Ensure timely propagation of updated app versions to the entire install base	• New versions of the code can be distributed without requiring the manual update of the app (applicable to web resources).
Remote Disable	Enforce timely adoption of critical security updates to the entire install base	• Server-side console allows configuration of allowed app versions. Administrator can force users to install security updates to the native code.
Authentication Framework	Lower overall cost and complexity of integration with authentication infrastructure	• Server-side architecture designed for integration with back-end authentication infrastructure based on JAAS concepts, with Authentication realms. • Out-of-the-box integration with Kerberos, NTLM, Basic, and Digest authentication. • Ability to encrypt server-to-server SOAP communication with X509 certificates, following the WS-Security standard.

Mechanism	Benefit	Details
		• Client-side framework for asynchronous login requests on session expiration.
Server-side Safeguards	Prevent SQL Injection and protect against XSRF	• Prepared-statement enforcement. • Validation of submitted data against session cookie.
Enterprise SSO Integration	Leverage existing enterprise authentication facilities and user credentials and enable employee-owned devices	• Client side mechanism obtains and encrypts user credentials, sends to the server with requests • Encryption incorporates user-supplied PIN, Server side secret and device ID • Credentials cannot be retrieved from lost or stolen device
VPN Alternative	Enable secure delivery and operation of mobile apps for employee-owned devices or device types not allowed on the corporate network, as well as enable secure delivery when installation of VPN client on mobile devices is not possible or complicated to manage	• Client side and server side framework act as SSL based VPN • Network access control and policies pre-configured in the client side framework layer • Network access and security measures updated using server-side framework • On device encrypted storage to prevent compromise of sensitive data

System Requirements

Production Environment

The Worklight Server can be installed on the following operating systems:

- Windows Server 2008 64 bit
- Red Hat Enterprise Linux (REHL) Version 4 SP3 or later (SP4 recommended) 64 bit

The server requires the following databases to store metadata and cached back-end data:

- My SQL 5.0.22 or later (5.0.67 recommended)
- Oracle 10g and 11g

The Worklight Server can run on the following application servers:

  - Tomcat 7
  - WebSphere Application Server 7 and higher over Linux

The Worklight Server can be clustered to achieve high availability and scalability. In such case, a load balancer is required. This can be any commercial load balancer, software or hardware, which supports sticky sessions. The load balancer can optionally act as a reverse proxy and SSL accelerator.

Development Environment

The Worklight development environment includes the Worklight Server and the Eclipse-based Studio. The development environment is supported on the following operating systems:

- Windows 7, Vista or XP (32 or 64 bit)
- Macintosh environment

For development purposes, the following database is supported:

- My SQL 5.0.22 or later (5.0.67 recommended)

The Worklight Studio requires the following distribution of Eclipse:

- Eclipse for JEE developers, version Indigo

About Worklight

Worklight, an IBM company, enables organizations to efficiently create, run and manage HTML5, hybrid and native applications for smartphones and tablets with industry-standard technologies and tools. Worklight provides the most complete and extensible IDE with maximum code reuse and per-device optimization, next-generation mobile middleware and powerful management and analytics. Many of the world's largest companies rely on Worklight to provide optimal user experiences across more devices while reducing time to market, development cost and ongoing maintenance effort.

For further information please visit our website – www.worklight.com

Download the evaluation version of our platform – www.worklight.com/download