

Worklight® 4.2.1

# Release Notes

**January 2012**

Document Version 2



# Table of Contents

<b>Document History</b> .....	<b>3</b>
<b>New Features</b> .....	<b>4</b>
Application Development.....	4
Runtime, Security, and Management.....	5
<b>Changed Behavior</b> .....	<b>6</b>
Database Structure.....	6
Desktop Environments .....	6
Worklight Server's HTTP Interface .....	6
<b>Upgrading from Previous Versions</b> .....	<b>7</b>
Upgrading the Development Environment .....	7
Upgrading AIR Applications from Worklight Version 4.0.3.72 .....	7
Upgrading the Production Environment .....	7
<b>Deprecated API and Unsupported Features</b> .....	<b>8</b>
Authentication of Desktop Widgets and Gadgets .....	8
<b>Bug Fixes and Closed Issues</b> .....	<b>9</b>
<b>Known Issues</b> .....	<b>11</b>
<b>Appendix A: Upgrade Instructions for AIR Applications from Version 4.0.3.72</b> .....	<b>12</b>

# Document History

Version	Comment
2	Adobe AIR Packaging Tool No Longer Embedded with Worklight
2	Support for Android Options Menu Icons for Different Device Densities
2	Bug fixes: 3840, 3875

# New Features

## Application Development

### Support for PhoneGap 1.2

Release 4.2.1 includes PhoneGap 1.2 as part of its mobile client run time. Note that PhoneGap changed its plug-in architecture and interface in the transition from version 1.0 to 1.1 and 1.2. If you developed custom PhoneGap plug-ins, you may need to adjust your code to the new interface. Refer to the Worklight training material for detailed instructions of writing PhoneGap plug-ins.

### Command-line Application and Adapter Deployer

Worklight 4.2.1 includes a command-line utility to deploy applications and adapters. The utility is available on Windows and Linux. Developers and administrators can write scripts calling the utility to automatically deploy adapters and applications to Worklight Servers. This is particularly useful when porting these artifacts from a development environment to testing and production environments. Refer to the *Worklight 4.2.1 Developer's Reference Guide* for details.

### Transportability of Mobile Application's Web Resources across Server Environments

In previous versions of Worklight, the web resources of a hybrid mobile application included the address of the Worklight Server with which the app should communicate. As of version 4.2.1, the address of the Worklight Server is only included in the native application shell of iOS and Android apps. This enables developers and administrators to test the application in one environment and then move its web resources directly from the test environment to production, without re-building it.

### Connectivity Settings Screen Controlled with Flag

Release 4.2 introduced a Settings screen that was automatically added to iOS and Android apps. In 4.2.1, developers can control whether the settings screen is included with the application by configuring the application descriptor. This allows developers to include the screen in development and testing phases, and remove it in production. Refer to the *Worklight 4.2.1 Developer's Reference Guide* for details.

Note: If you mistakenly upload an iOS app with the Worklight Settings screen to the Apple app store, then to remove the settings in a subsequent version you must re-build the app with an *empty* settings bundle. Just removing the settings bundle from the Xcode project will not erase the settings from the device.

## **Ability to Reset to the Default Server Address in the Connectivity Settings Screen**

The Connectivity Settings screen now allows users to reset the server address to the default address defined in the application. This enables users to recover from bad settings and to restore the application connectivity to its original stage.

## **Application Version Automatically Propagated to Native Projects**

When building an application in version 4.2.1, the application version in each environment (defined in the application descriptor) is automatically updated in the applicable native project.

## **Support for Android Options Menu Icons for Different Device Densities**

Developers can now provide Android options menu icons for specific device densities that are supported by their apps. This ensures that the icons are displayed well on every device, and follows Google's guidelines for multi-screen support. Refer to the *Worklight Developer's Reference Guide* for details.

# Runtime, Security, and Management

## **Direct Update: Only skin used by application is pushed to device**

In Worklight 4.2.1, when an update is available for an application, only the skin used by the application is downloaded to the device.

## **Support for Tomcat 7.0.22**

Worklight Server 4.2.1 is based on Tomcat 7.0.22. The Tomcat server is embedded within the installation of Worklight, and does not need to be installed separately.

## **LSB-compliant scripts for Linux and Mac OS X**

Worklight 4.2 includes LSB-compliant init scripts for Linux and Mac OS X. Linux administrators can use these scripts to automatically start the Worklight Server upon machine reboot.

# Changed Behavior

## Database Structure

The column `PREVIEW_ENVIRONMENT` in the table `GADGET_INSTANCE` is obsolete, but kept in the database upgrade process.

## Desktop Environments

### Mac Dashboard Widgets

Building Mac Dashboard widgets was temporarily not supported in 4.2. In 4.2.1 this capability is restored.

### Adobe AIR Packaging Tool No Longer Embedded with Worklight

To build and sign Adobe AIR applications, developers must download the Adobe AIR development tools from the Adobe web site and install them on their computer. Refer to the *Worklight Development Environment Installation Guide* for details.

## Worklight Server's HTTP Interface

The HTTP interface that the Worklight Server exposes to apps was modified in version 4.2.1. If you set up firewall rules blocking some services, refer to *Worklight 4.2.1 Administration Guide* for details about the new interface.

Note: the 4.0 interface is still supported for backward compatibility.

# Upgrading from Previous Versions

## Upgrading the Development Environment

Worklight apps that were developed on previous releases must be upgraded before they can be built with version 4.2. To upgrade an app, you should simply install the new Worklight 4.2 Server and Worklight Studio 4.2 Eclipse Plug-in, and then build your application.

- If your apps are compatible with version 4.1.1, the upgrade process is completely automatic
- If your apps are compatible with version 4.0 or 4.1, you must perform some manual steps to upgrade your application. The Worklight Studio automatically detects the version of your app and instructs you as needed.

## Upgrading AIR Applications from Worklight Version 4.0.3.72

If you have AIR applications serviced by a Worklight 4.0.3 server, please follow the instructions in [Appendix A](#) for upgrading them to be served by a Worklight 4.2.1 server.

## Upgrading the Production Environment

The 4.2.1 Server can service apps deployed on a 4.0 Server. This allows customers to upgrade a production 4.0 server to 4.2.1 while still serving 4.0-based apps that were deployed to users and run on their devices.

In version 4.2.1, the `worklight.properties` file supports a property called `production`, which replaces the former `devmode` property. The new property must be set to `true` in production environments, in order to block access to development services in the production server: (1) Running a dynamic SQL query that has not been previously defined in an adapter; (2) Binding to the Worklight Server by the Rhino debugger.

Refer to the *4.2.1 Administration Guide* for details about upgrading a production Worklight Server.

# Deprecated API and Unsupported Features

## Authentication of Desktop Widgets and Gadgets

NTLM authentication for desktop widgets and gadgets is not supported in version 4.2.1.

## Bug Fixes and Closed Issues

ID	Applicable to	Description
816	4.1	The Worklight Studio Eclipse plug-in was not signed. As of version 4.2.1, it is signed.
934	3.3	No error is output if an invalid timeout value is provided in calls to the native iOS API
1618	3.1	Web to native API - No validation if the number of arguments is less than required
1633	3.1	Web to native API - 'null' is transferred from the web code to the native code as a string
1649	3.1	In iOS apps, hiding the tab bar after the user rotates the device resizes the WebView wrongly
2177	3.1	Events of type 'userprefs' are wrongly recorded as 'setuserprefs' in the raw reporting data
2314	4.0.1	Logged error is unclear when sending invokeProcedure parameters as string instead of array
2590	4.0.3	The iOS busy indicator has display problems upon application startup. In 4.2.1, the busy indicator is not displayed at all during application startup; only the application's splash screen is displayed.
2687	4.1	Skins not used by the app on the device still get downloaded when the app resources are directly updated
2715	4.1	If an app does not include skins and was installed on a device, newly added skins will not be available by directly updating the app's web resources.
2752	4.1.1	The New App wizard in the Worklight Studio allows naming apps with reserved keywords. The following keywords are now prevented: "app", "application", "auth", "messages", and "reset".
2905	4.1.1	It is not possible to increase the version of desktop and web widgets.
2918	4.1.1	Preview for the iOS and Android environments does not work on IE9
2936	4.1.1	When accessing the Reports tab in IE, there is a JavaScript error
3373	4.1.3	Data recorded in reports tables always contain a '00:00:00' timestamp

ID	Applicable to	Description
3376	4.1.3	Too many Xcode schemes are available for an app generated by the Worklight Studio and imported to Xcode
3384	4.1.3	Some options of WL.BusyIndicator do not work on iOS
3396	4.2	To properly create the Windows Phone environment in the Worklight Studio, Eclipse's Build Automatically option must be enabled.
3406	4.2	The error output when deploying an app or adapter to the Worklight Server, while the Server is down, is not clear.
3423	4.2	When remotely disabling an app and specifying a URL for the new app version, iOS's itms:// protocol is handled correctly. Http:// must be used instead.
3452	4.2	For native iOS apps, the existence of the application version property in the application's .plist file is not enforced, enabling apps to run without indicating to which version they belong.
3538	4.2	Using Xcode 4.2, the Worklight native Objective-C SDK cannot be dragged into the application project. The workaround is to drag it to the project using Xcode 4.1, and then re-open the project in Xcode 4.2.
3553	4.2	Creating a BlackBerry environment requires the BlackBerry plug-in to be pre-installed on Eclipse.
3554	4.2	On startup of mobile apps on iPhone and iPad, a white screen and/or a busy indicator are displayed between the splash screen and the full application.
3557	4.2	On Windows Vista and 7 gadgets, the dock image is surrounded by incorrect background.
3634	4.1	In the Diagnostics screen, numbers can be tapped to open the phone dialer
3840	4.2	On iPad, the background of the Direct Update screen is half black and half white.
3875	4.2.1	An exception occurs if building an iOS project while Internet connection is not available

## Known Issues

ID	Reported In	Description
2205	4.0.1	The Daily Activity Distribution report does not aggregate activity on an hourly basis, but rather on a daily basis.
2286	4.1	Setting the address of one Worklight Server in the application descriptor, and then building it and deploying it to another Server causes an exception, instead of a proper error message
3730	4.2.1	The Worklight Console's Reports tab in IE7 is not operable, showing a 'Storage is undefined' error message.
3789	4.2.1	The name of some auto-generated iOS app resources contains also the Worklight project name, not only the app name. This requires providing the project name to the standalone builder when building an app that was previously built in the Worklight Studio.
3801	4.2.1	It is allowed to subscribe from the same app to event sources that accept re-subscriptions of a different user from the same device and to event sources that do not
3806	4.2.1	Push notification aliases are not handled correctly in native iOS apps, making re-subscriptions behave unexpectedly following user re-login.

# Appendix A: Upgrade Instructions for AIR Applications from Version 4.0.3.72

Follow these steps to upgrade AIR applications from version 4.0.3 to version 4.2.1.

1. Obtain the global unique ID of the 4.0.3 production Worklight Server. To do so, take the value of the `guid` property in production server's `worklight.properties` file. If your `worklight.properties` file does not explicitly define `guid`, take the value of the `publicWorkLightHostname` property instead.
2. In you 4.2.1 development environment, re-build your AIR application.
3. In the application descriptor file, add an `<air>/<guid>` element, containing the value obtain in Step 1. For example:

```
<air version="1.1">
  <guid>myGuid123</guid>
</air>
```

4. Rebuild your application's AIR environment.
5. Upload the newly built AIR application to your production Worklight 4.2.1 Server

When users download the new 4.2.1 application, it will replace the one already installed on their desktops.