

IBM UrbanCode Deploy



Worklight Plug-in

Version 2.0 This information is under development and can change at any time; the released version will be available on ibm.com.

Note

Before using this information and the product it supports, read the information in “Notices,” on page 23.

This edition applies to version 2.0 of Worklight plug-in for IBM UrbanCode Deploy and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 2013.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Worklight plug-in 1

Software requirements and compatibility for the	
Worklight plug-in.	1
Building and deploying mobile applications	1
Building mobile applications	2
Adding mobile artifacts to UrbanCode Deploy .	10
Configuring the Worklight Server	11
Deploying mobile applications	13
Rolling back mobile applications	15
Process steps in the Worklight plug-in	17

Deploy Adapter to Worklight Server	17
--	----

Deploy Worklight Application to Worklight	
Server	18

Remove Application from Application Center .	19
--	----

Upload Application to Application Center . .	20
--	----

Appendix. Notices 23

Trademarks	25
----------------------	----

Worklight plug-in

The IBM® Worklight® plug-in includes steps that manage the deployment of artifacts to the IBM Worklight Server.

To use this plug-in, the target environment must have an installation of IBM Worklight Server, with a Console and Application Center configured.

The plug-in includes steps that are related to deploying adapters and Worklight applications to the Worklight Server, such as the following steps:

- “**Deploy Adapter to Worklight Server**” on page 17
- “**Deploy Worklight Application to Worklight Server**” on page 18

The plug-in also includes steps that are related to removing and uploading applications to the Application Center, such as the following steps:

- “**Remove Application from Application Center**” on page 19
- “**Upload Application to Application Center**” on page 20

To install this plug-in, see Installing plug-ins.

Software requirements and compatibility for the Worklight plug-in

Each version of the IBM Worklight plug-in has specific software requirements and compatible software.

The following table lists the software that is required for the IBM Worklight plug-in, as well as the software that the plug-in is compatible with.

Version	Required software	Compatible software
IBM Worklight plug-in Version 1.0	IBM UrbanCode Deploy Version 6.0.0 or later	IBM Worklight Server Version 6.0.0
IBM Worklight plug-in Version 2.0	IBM UrbanCode Deploy Version 6.0.0 or later	IBM Worklight Server Version 6.0.0 or later

Building and deploying mobile applications

You can set up your development environment so that you can build your mobile applications and, by using the IBM Worklight plug-in for IBM UrbanCode Deploy, deploy the build results to the IBM Worklight Server.

Before you begin

Ensure that the following software is installed and running:

- IBM UrbanCode Deploy
- IBM Worklight Server with the Application Center and Console running.
- IBM Worklight Studio

Extra software might be required, such as:

- Source control management (SCM) system.

- Build engine.
- Application server.
- Database.

About this task

Before you can build and deploy mobile applications to the Worklight Server, you must complete the following configuration steps:

1. Configure the build system.
2. Configure UrbanCode Deploy, including the following steps:
 - Create components.
 - Create component processes or application processes that include steps from the Worklight plug-in to deploy the mobile application.
3. Configure Worklight Server Console, including the following steps:
 - Create and configure a database.
 - Configure the Worklight project Web Archive (WAR) file.

Procedure

After you set up the build, UrbanCode Deploy and Worklight Server console, you can build and deploy mobile applications by using the following high-level steps:

1. Check in (commit) changes from IBM Worklight Studio into a source control management (SCM) system.
2. Build the application and add a new version to UrbanCode Deploy.

Tip: Assign a version to the mobile application that is deployed to the Application Center. This version must match the version that is assigned in UrbanCode Deploy. For example, if the mobile application has a commercial version of 1.0 on the Application Center and the internal version from the latest build is 16, assign version 1.0.16 to the application in UrbanCode Deploy. Keeping the version numbers synchronized helps you to recover if you encounter a problem. For example, if the latest version of the mobile application was not deployed successfully to the Application Center.

3. Request deployment in UrbanCode Deploy.
4. View the mobile artifacts in the Worklight Console, install, and test the application from the Application Center.

Results

The mobile application artifacts are deployed to the Worklight Server and can be installed on the target device.

What to do next

Optionally, create extra component and application processes in UrbanCode Deploy to roll back deployments (for example, to recover from an error condition or an incomplete deployment.)

Building mobile applications

To set up a continuous integration and delivery cycle for your mobile applications, you must build the mobile application artifacts before you deploy them to the IBM

Worklight Server. The IBM Rational® solution for Collaborative Lifecycle Management (CLM), IBM Worklight Studio, and IBM UrbanCode Deploy integrate to help automate the build and deployment of mobile applications.

CLM contains the Rational Team Concert™ product that is delivered as an application together with a Jazz™ Team Server. Rational Team Concert and the Build System Toolkit work together to build your mobile applications. When Rational Team Concert is installed on the Jazz Team Server, it manages the workspaces, projects, source files, and builds of your mobile applications. The Build System Toolkit runs the actual build tasks.

Related information:



Building with Jazz Team Build

Build computer resources

Before you run a mobile application build script on a build computer, you must ensure that the required resources exist on the build computer.

Workspace resources

The following workspace resources must exist on the build computer:

- The mobile application project source code that you want to build.
- The Ant build scripts that direct the build.

Using a Rational Team Concert repository workspace to manage your Worklight project source code and build scripts offers the following advantages:

Advantage	Description
Source control	Changes to source code and build scripts can be requested, developed, reviewed, approved, delivered, and tracked based on the requirements of your development project. Build scripts are living files, just like the source code.
Build automation	The Jazz Build Engine automatically loads the workspace to build onto the build computer early in the processing of a build request. You can create and use a dedicated build workspace for each build definition. Do not point a build definition directly to a stream or to a workspace that is meant for another purpose. For example, do not point a build definition directory to the personal workspace of a user or a team integration workspace. Note: The Jazz Build Engine is a component of the Build System Toolkit; it refers to the process that runs on a build computer and runs Ant scripts.

Static resources

The build administrator must manually install the static resources on each build computer.

Tip: Install these resources into the same relative locations on each build computer. You can specify the relative locations within either of the following types of build dependency resources:

Build property files

Specify the relative locations of the static resources within the build property files. If you install static resources into different locations on different build computers, a location that is specified within a build property file that works on one build computer might fail on another build computer.

Build definitions within Rational Team Concert

Specify the relative locations of the static resources within the build definitions in Rational Team Concert. If you install static resources into different locations on different build computers, a build definition that works on one build computer might fail on another build computer.

The following static resources must exist on the build computer:

Static resource	Description
Oracle JDK	Use this JDK for running the Ant scripts and Android SDK tools that are run by the build scripts. Ensure that you install a JDK, not a JRE, because some Ant tasks require Java™ tools that are available only in the JDK.
Apache Ant	Use Apache Ant to run the Ant scripts.

Static resource	Description
JAR library files	<p>The following JAR library files provide and enable the Worklight Ant tasks that are used in the build scripts:</p> <p>worklight-ant.jar Use the worklight-ant.jar file if you are building applications on the IBM Worklight Server Version 6.0.0. This file is contained in the WorklightServer folder of the IBM Worklight Server installation.</p> <p>worklight-ant-builder.jar Use the worklight-ant-builder.jar file if you are building applications on the IBM Worklight Server Version 6.1.0. This file is contained in the WorklightServer folder of the IBM Worklight Server installation.</p> <p>Important: Ensure that the version of the JAR library file that you use (worklight-ant.jar or worklight-ant-builder.jar) matches the version on the target server.</p> <p>Tip: An alternative approach to preinstalling the JAR library files on each build computer is to include them in your build workspace. This approach allows your build definitions and engines to build with different versions of Worklight. This approach also supports the generation of reproducible builds.</p> <p>The disadvantage of this approach is that the JAR library files can be large. The large file size might affect the performance of builds and build computers.</p> <p>If you share a build system and build computers across multiple teams, use this alternative approach to manage the JAR library files.</p>
Optional. Dojo Toolkit	<p>Install the Dojo Toolkit on each build computer in the following situations:</p> <ul style="list-style-type: none"> • The mobile applications under development use Dojo. • The mobile application projects either include the Dojo Toolkit (in the workspace project) or access it over a Content Delivery Network.

SDKs

Install one of the following SDKs on each build computer:

SDK	Description
Apple Xcode SDK	Install on OS X build computers that run builds to produce iOS IPA applications. For more information about installing the Apple Xcode SDK, see Getting Started with IBM Worklight Module 02.1 – Setting Up Your iOS Development Environment.

SDK	Description
Android SDK	Install on build computers that run builds to produce Android APK applications. For more information about installing the Android SDK, see Getting Started with IBM Worklight Module 02.2 – Setting Up Your Android Development Environment.

Related information:



Best practices: Setting up Jazz team builds

Build scripts

You can create Ant build scripts for Worklight projects that contain applications and adapters. By using these build scripts, you can automate your mobile application builds.

Build script tasks

You can create build scripts that use the following types of Ant tasks:

Type of Ant task	Description
Built in tasks from Apache Ant	Includes tasks such as: <ul style="list-style-type: none"> • <echo> • <report> • <mkdir> • <exec> • <replaceregexp>
Tasks from IBM Worklight	These tasks perform the following actions: <ul style="list-style-type: none"> • Build Worklight applications and adapters, such as <app-builder> and <adapter-builder>. IBM Worklight provides a set of Ant tasks that help you to build adapters and Worklight applications for your IBM Worklight Server. • Build IBM Worklight web archive projects. IBM Worklight provides the <war-builder> Ant task for building the Worklight project WAR file.
Tasks from the Rational Team Concert Build System Toolkit	These tasks provide information to the build results. Tasks include: <ul style="list-style-type: none"> • <startBuildActivity> • <linkPublisher> • <artifactPublisher>

Sample build script task flow




You can create build scripts for Worklight projects that contain different numbers of applications or adapters. The following sample task flow describes the overall design of a build script for a Worklight project that has a single Worklight application and a single adapter.

1. Use Ant <property> elements to set the properties.
2. Use a hybrid target to build Worklight applications, adapters, and Worklight web archive projects. The hybrid target contains the following actions:

- a. URLs that point to the Worklight Server Console and the Application Center are published to either the Ant build log or the Rational Team Concert build results.
 - b. The Worklight <app-builder> Ant task builds the Worklight application.
 - c. The resulting .wlapp file is stored in the build output.
 - d. The Worklight <adapter-builder> Ant task builds the adapter.
 - e. The resulting .adapter file is stored in the build output.
 - f. The Worklight <war-builder> Ant task builds the Worklight web archive project.
 - g. The resulting WAR file is stored in the build output.
 - h. Optional. If you use Rational Team Concert, you can publish the .wlapp, .adapter, and WAR files to the Rational Team Concert build results.
3. When you build an Android application, include the following actions to build the native Android APK file:
 - a. Run the **android** command-line tool from the Android SDK to generate the Android build.xml file.
 - b. Run the generated Android build.xml file to build the APK file.
 - c. Optional. Publish the Android APK file to the location where you store your build output. For example, if you use Rational Team Concert, publish the APK file to the Rational Team Concert build results.
 4. When you build an iOS application, include the following actions to build the native iOS IPA file:
 - a. Run the **xcodebuild** command-line tool from the Xcode SDK to build the iOS application.
 - b. Run the **xcrun** command-line tools from the Xcode SDK to package the iOS application into an IPA file.
 - c. Optional. Publish the iOS IPA file to the location where you store your build output. For example, if you use Rational Team Concert, publish the IPA file to the Rational Team Concert build results.
 5. Add your Worklight application, adapter, Worklight web archive project (WAR file), and native application (Android APK file or iOS IPA file) to UrbanCode Deploy as a new version.

Tip: You can have multiple Worklight applications and adapters. If you have more than one Worklight application or adapter, repeat calls to tasks to build the mobile artifacts, add new property values, and then add the new artifacts to UrbanCode Deploy.

Related information:

-  Ant tasks for building and deploying
-  Building a project WAR file with Ant
-  Jazz build Ant task reference

Jazz Team Build

The Jazz Team Build system defines resources that are used to describe and manage builds.

The Jazz Team Build system in Rational Team Concert defines the following types of logical resources that are used to describe and manage builds:

Build definition

A build definition describes:

- The workspace to process.
- The Ant scripts and targets to run against the workspace.
- The schedule for the build.
- The properties to simplify the configuration of the build.
- The build engines that can handle build requests for the build definition.

Build engine

A build engine represents a Build System Toolkit Jazz Build Engine running on a designated build computer. A Jazz Build Engine running on a build computer correlates itself to a build engine in Rational Team Concert by specifying the unique identifier of the build engine.

Build request

A build request represents a scheduled or explicitly issued request to run a build according to a specified build definition. Build definitions are submitted to a build queue. A Jazz Build Engine receives and processes the build request if its corresponding build engine in Rational Team Concert is listed as a supporting build engine for the build definition.

Build result

A build result represents the output of a build.

Related information:

Building with Jazz Team Build

Build System Toolkit:

Each build computer contains an installation of the Rational Team Concert Build System Toolkit.

The Build System Toolkit consists of the following major components:

Jazz Build Engine

The Jazz Build Engine is a command-line tool that polls for and processes build requests from Rational Team Concert. When the Jazz Build Engine is started, it must identify a corresponding build engine in Rational Team Concert. The Jazz Build Engine can then accept any build request whose build definition is supported by the build engine. The Jazz Build Engine runs the Ant script and targets that are specified in the build definition. Each build is represented in Rational Team Concert by a build result.

Build toolkit

The build toolkit is a collection of Ant tasks. Ant scripts can use these tasks to send information (such as build progress, results, links, artifacts) to Rational Team Concert to include in the build result.

Build agent

The build agent is a lightweight process that handles agent-based builds that support z/OS® or IBM i build scenarios. For the Worklight plug-in, use the Jazz Build Engine instead.

Note: The Jazz Build Engine is a component of the Build System Toolkit; it refers to the process that runs on a build computer and runs Ant scripts.

Related information:

Building with Jazz Team Build

Build definitions:

In Rational Team Concert, a build definition describes the key components of a build.

The build definition describes the following components:

- The workspace to process.
- The Ant scripts and targets to run against the workspace.
- The schedule for the build.
- The properties that simplify the configuration of the build.
- The build engines that can manage build requests for the build definition.

The following sections describe the considerations that apply when you build IBM Worklight mobile applications.

Supporting build engines

When you specify a build engine to run build requests for the build definition, ensure that any required SDKs (such as the Android SDK or Apple Xcode SDK) are installed and configured on the build engine.

Properties

You can use properties to customize the build for a specific build definition. For example, you can set properties for the path to the build output or the native SDK.

Ant build file and targets

In the **Build file** field, use the following value to specify the location of the Ant build script that is loaded with the workspace: `load/${buildLabel}/project/folder/script`

Where:

project The name of the project that contains the build scripts.

folder The name of the folder within the project that contains the build scripts.

script The name of the build script XML file.

Tip: If you choose a different relative location for your build script in the workspace, you must change the value of the **loadDir** property.

In the **Build targets** field, specify any specific targets that you want to run in your build script. By default, build scripts run the **all** target.

Ant configuration

Ant configuration includes the following tasks:

- Select the option to include the Jazz build toolkit tasks on the Ant library path.
- In the **Ant home** field, specify the location on the build computer where Apache Ant is installed.

- In the **Ant arguments** field, specify the **-lib** argument that includes the worklight-ant.jar required library on the Ant library path. If you are building applications on the IBM Worklight Server Version 6.1.0, you can use the worklight-ant-builder.jar file instead of worklight-ant.jar.

Important: Ensure that the version of the JAR library file that you use (worklight-ant.jar or worklight-ant-builder.jar) matches the version on the target server.

Use the following format for the **-lib** argument: **-lib *path\JAR file name***

Where:

path The path to the directory on the build computer that contains the JAR file. The path might be a location on the build computer where the JAR libraries were preinstalled. The path might also point to a location within the loaded workspace if you chose to include the JAR libraries in the build workspace.

JAR file name

The name of the JAR file that is included in the library.

- In the **Java home** field, specify the location on the build computer where the Oracle JDK is installed.

Related information:



Building with Jazz Team Build



Getting Started with IBM Worklight - Using Rational Team Concert to build your applications

Adding mobile artifacts to UrbanCode Deploy

You can use the build scripts to add your build artifacts to IBM UrbanCode Deploy for deployment to the IBM Worklight Server.

Procedure






You can use any of the following methods to add your build artifacts to UrbanCode Deploy:

Option	Description
Copy the files into a user-defined file system	Copy the build artifacts to a location on the UrbanCode Deploy server's file system for a versioned file.

Option	Description
Push the files to the UrbanCode Deploy server	<p>Use the Command-line client (CLI) to push the build artifacts to the UrbanCode Deploy server. The CLI is a command-line interface that provides access to the UrbanCode Deploy server.</p> <p>You can use the CLI to push the build artifacts to the UrbanCode Deploy server in the following scenarios:</p> <ul style="list-style-type: none"> • When the Jazz Build Engine and the UrbanCode Deploy server are not installed on the same build computer. • To support running the UrbanCode Deploy server on different operating systems. <p>Tip: You can use the following commands to deploy binary files to the UrbanCode Deploy server:</p> <p>createVersion Create the component version.</p> <p>addVersionFiles Upload the component files.</p>
Copy the files into a source-code management system	<p>Copy the build artifacts into a source-code management system, such as:</p> <ul style="list-style-type: none"> • Git • IBM Rational Asset Manager • Subversion

Tip: Assign a version to the mobile application that is deployed to the Application Center. This version must match the version that is assigned in UrbanCode Deploy. For example, if the mobile application has a commercial version of 1.0 on the Application Center and the internal version from the latest build is 16, assign version 1.0.16 to the application in UrbanCode Deploy. Keeping the version numbers synchronized helps you to recover if you encounter a problem. For example, if the latest version of the mobile application was not deployed successfully to the Application Center.

Related information:

-  Creating components from a versioned file system
-  Creating components from source-code management systems
-  Command-line client (CLI) reference
-  createVersion
-  addVersionFiles

Configuring the Worklight Server

You must configure the IBM Worklight Server before you can run the process steps to deploy mobile application artifacts.

About this task

For each Worklight project, you must configure the Worklight Server. The one time Worklight Server setup and configuration options for the Worklight project WAR file are described in the Worklight Information Center.

You can use any of the following methods to configure the Worklight Server:

- Use Ant tasks to configure the server:
 - Use the command line to run the Ant script.
 - Use the Ant process step within UrbanCode Deploy to run the Ant script with the Worklight Ant tasks. To use the Ant process step, you must download and install the Ant plug-in for UrbanCode Deploy.
- Use the Server Configuration Tool to configure the server.
- Manually configure the server.

Tip: Sample scripts are available for configuring the server and are in *Worklight installation directory/Worklight/WorklightServer/configuration-samples*. The sample script combines the two steps in the following procedure by creating a database and deploying the project WAR file.

Procedure

To configure the Worklight Server for a Worklight project:

1. Create and configure a database for your server:

Option	Documentation
Use Ant tasks to create and configure the databases	Creating and configuring the databases with Ant tasks
Use the Server Configuration Tool to create and configure the databases	Deploying, updating, and undeploying a Worklight Server by using the Server Configuration Tool
Manually create and configure the databases	Creating and configuring the databases manually

2. Deploy the project WAR file to the application server:

Option	Documentation
Use Ant tasks to install the Worklight project	Deploy a project WAR file and configuring the application server with Ant tasks
Use the Server Configuration Tool to install the Worklight project	Deploying, updating, and undeploying a Worklight Server by using the Server Configuration Tool
Manually install the Worklight project	Deploy a project WAR file and configuring the application server manually

Important: If you are configuring multiple projects on a single server, then see the Worklight topic Configuring multiple IBM Worklight projects. If you run multiple projects on a single server, install the .war files from multiple Worklight projects in the same application server, and then have them operate in parallel and independently.

Note: The Worklight project WAR file deployment that is described in this step is a one time configuration. When you deploy your mobile application, update the WAR file on the application server by using the plug-in process steps.

Deploying mobile applications

You can use the process steps in the IBM Worklight plug-in to deploy mobile applications to IBM Worklight Server.

Before you begin

- If it is not already installed, install the IBM Worklight plug-in. For more information about installing this plug-in, see Installing plug-ins.
- Install the plug-in that corresponds to the application server that is running your Worklight Server for deploying the WAR file to the Worklight Server. For example:
 - Apache Tomcat.
 - IBM WebSphere® Application Server.
 - IBM WebSphere Application Server Liberty Profile.
- Verify that your process steps use the path to the JAR files that correspond to your IBM Worklight Server version. For example, if you are using IBM Worklight plug-in Version 2.0 or later and a version of IBM Worklight Server later than 6.0.0, you must update your process steps.

Tip: The following JAR files are available for use from IBM Worklight plug-in Version 2.0:

worklight-ant.jar

Used to deploy artifacts to the IBM Worklight Server Version 6.0.0.

worklight-ant-deployer.jar

Used to deploy artifacts to the IBM Worklight Server Version 6.1 or later.

applicationcenterdeploytool.jar

Used to interact with an Application Center installed on an IBM Worklight Version 6.0.0 Server or later.

json4j.jar

Used with the IBM Worklight Server Version 6.0.0 or later.

About this task

In the process editor, you can modify component processes to include steps to deploy the following mobile application artifacts to your Worklight Server:

- Native applications (Android .apk or iOS .ipa)
- Worklight Adapters (.adapter)
- Worklight Applications (.wlapp)
- Worklight project (.war)

The following sequence is a suggested order for deploying the mobile application artifacts:

1. Deploy the .war file to the application server by using a process step from the corresponding plug-in for the type of application server that is running the Worklight Server. For example, for WebSphere Application Server use the Install or Update Application step.

2. The following artifacts can be deployed in parallel or in either order:
 - Deploy the Worklight Adapter (.adapter) file to the Worklight Server Console by using the “**Deploy Adapter to Worklight Server**” on page 17 step.
 - Deploy the Worklight Application (.wlapp) file to the Worklight Server Console by using the “**Deploy Worklight Application to Worklight Server**” on page 18 step.
3. Deploy the Android application package (.apk) or iOS application (.ipa) file to the Application Center by using the “**Upload Application to Application Center**” on page 20 step.

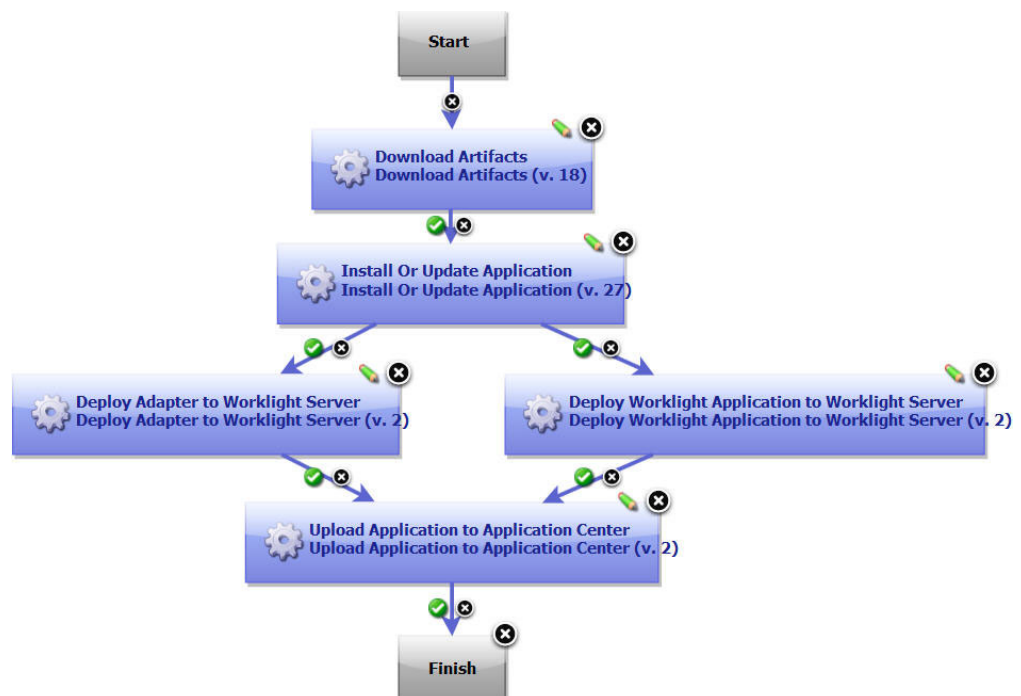
Example

The following simple example process deploys a mobile application to the Worklight Server Console and Application Center.

1. The **Download Artifacts** step retrieves the binary files.
2. The **Install or Update Application** step deploys the .war file to WebSphere Application Server (the application server that is used in this example).

Note: The **Install or Update Application** step in this example is provided by the Application Deployment for WebSphere plug-in (not the Worklight plug-in).

3. In parallel, the .adapter and .wlapp files are deployed to the Worklight Server Console by the **Deploy Adapter to Worklight Server** step and the **Deploy Worklight Application to Worklight Server** step.
4. The .apk (Android) or .ipa (iOS) file is deployed to the Application Center by the **Upload Native Application to the Application Center** step.

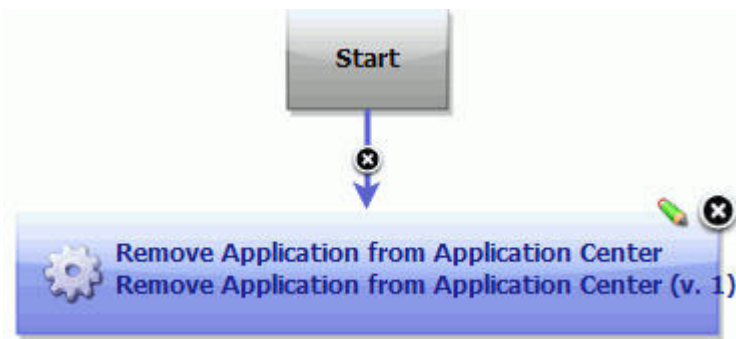


Rolling back mobile applications

There are a number of ways to roll back a mobile application that is deployed to IBM Worklight Server. One option is to remove the native application from the Application Center and then redeploy the application. Alternatively, you can manually roll back deployments.

About this task

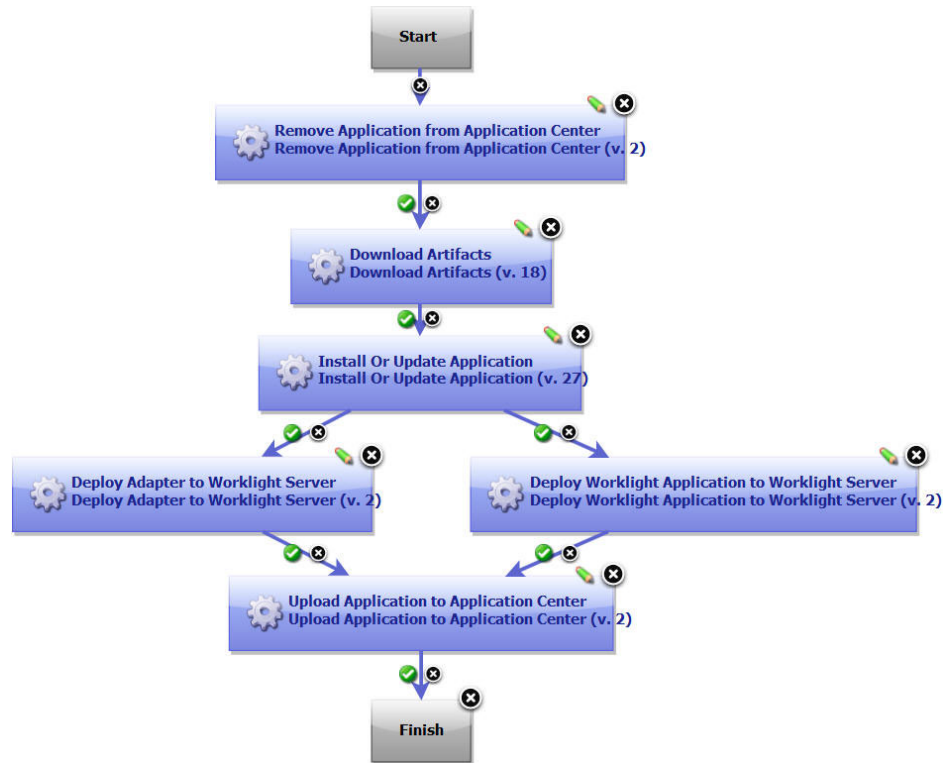
- To automate rolling back a mobile application deployment, create processes that use the following general steps:
 1. At the component level, create a process that removes the native application from the Worklight Application Center, and overwrite any deployed artifacts by redeploying the application:
 - a. To remove the native application from the Worklight Application Center, add the **"Remove Application from Application Center"** on page 19 step.



Tip: When you configure the Remove Application from Application Center step, specifying the Operating System and Version removes a specific native application, such as the version related to a failed deployment.

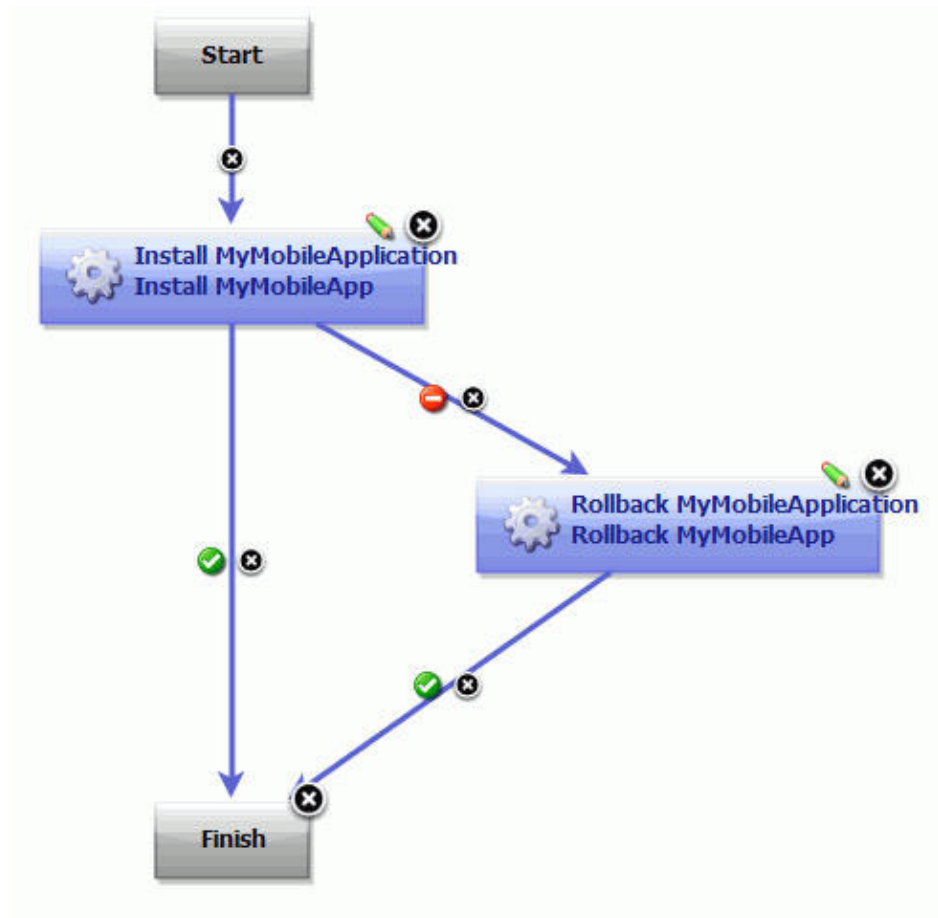
- b. Any artifacts that were successfully deployed to the Worklight Console are not removed. To overwrite the deployed artifacts, add process steps to redeploy the mobile application as described in the topic "Deploying mobile applications" on page 13.

The following example process removes the native application from the Worklight Application Center and redeploys the application:



2. At the application process level, create a process that includes the Rollback Component process step, and configure the step to call the component process that you created in the preceding steps. The **Rollback Component** step replaces the component version with an earlier version.

In the following application process example, if the **Install Application** step fails, then the **Rollback Application** step runs.



- To manually roll back a mobile application deployment:
 1. Delete the native application from the Worklight Application Center.
 2. In the Worklight Console, delete the adapters and applications. For details, see the topics in the section Administering adapters and apps in Worklight Console in the Worklight Information Center.
 3. Redeploy the previous version of the mobile application from UrbanCode Deploy.

Process steps in the Worklight plug-in

The IBM Worklight plug-in includes steps that deploy adapters and Worklight applications to the Worklight Server. The plug-in also includes steps that remove and upload applications to the Application Center.

Deploy Adapter to Worklight Server

Deploy an adapter to the IBM Worklight Server.

Remember: Before you run this step, you must ensure that the **Download Artifacts** process step is run so that it can retrieve the files for deployment.

Table 1. Properties for the Deploy Adapter to Worklight Server step

Name	Type	Description	Required
Server Path	String	The URL of the Worklight Server and the context root of the Worklight project. For example: http://myworklightserver/JKEBankMobile	Yes
Adapter File	String	The Worklight Adapter (.adapter) file to deploy to the Worklight Server.	Yes
Worklight Ant JAR File Path	String	<p>The path to the Worklight Ant Deployer JAR (worklight-ant-deployer.jar) file that is used to deploy artifacts to the Worklight Server.</p> <p>Note: An Ant JAR file is included in this plug-in for deploying artifacts to the Worklight Version 6.0.0 Server. The default path to use this file is <code>\${PLUGIN_HOME}/lib/worklight-ant.jar</code></p> <p><code>PLUGIN_HOME</code> The location where the agent extracted the IBM Worklight plug-in during deployment on the agent machine.</p> <p>Important: Ensure that the version of the Ant JAR file that you use matches the version on the target server.</p>	Yes

Deploy Worklight Application to Worklight Server

Deploy a Worklight application to the IBM Worklight Server.

Remember: Before you run this step, you must ensure that the **Download Artifacts** process step is run so that it can retrieve the files for deployment.

Table 2. Properties for the Deploy Worklight Application to Worklight Server step

Name	Type	Description	Required
Server Path	String	The URL of the Worklight Server and the context root of the Worklight project. For example: http://myworklightserver/JKEBankMobile	Yes
Application File	String	The Worklight Application (.wlapp) file to deploy to the Worklight Server.	Yes

Table 2. Properties for the Deploy Worklight Application to Worklight Server step (continued)

Name	Type	Description	Required
Worklight Ant JAR File Path	String	<p>The path to the Worklight Ant Deployer JAR (worklight-ant-deployer.jar) file that is used to deploy artifacts to the Worklight Server.</p> <p>Note: An Ant JAR file is included in this plug-in for deploying artifacts to the Worklight Version 6.0.0 Server. The default path to use this file is <code>\${PLUGIN_HOME}/lib/worklight-ant.jar</code></p> <p><i>PLUGIN_HOME</i> The location where the agent extracted the IBM Worklight plug-in during deployment on the agent machine.</p> <p>Important: Ensure that the version of the Ant JAR file that you use matches the version on the target server.</p>	Yes

Remove Application from Application Center

Remove a native application from the IBM Worklight Application Center.

Table 3. Properties for the Remove Application from Application Center step

Name	Type	Description	Required
Server Path	String	The URL to the Worklight Server. For example: <code>http://localhost:9080</code>	Yes
Context	String	The context root for the Application Center Services enterprise application. For example: <code>/applicationcenter</code> .	Yes
User	String	The user name that is required to access the Application Center.	Yes
Password	Password	The user password that is used to connect to the Application Center. The default value is <code>\${p:environment/applicationCenterPassword}</code>	Yes
Disable SSL Security Checking	True or False	This option allows publishing on secured hosts without verification of the SSL certificate. Use of this flag is a security risk, but might be suitable for testing localhost with temporary self-signed SSL certificates.	No
Application Package	String	The package name of the application to remove from the Application Center.	Yes
Operating System	<ul style="list-style-type: none"> All Android iOS 	The operating system of the application to remove from the Application Center.	No
Version	String	The internal version (not the commercial version) of the application to remove from the Application Center.	No

Table 3. Properties for the Remove Application from Application Center step (continued)

Name	Type	Description	Required
Application Center Ant JAR File Path	String	<p>The path to the Application Center Deploy Tool Ant JAR (applicationcenterdeploytool.jar) file that is used to interact with the Application Center.</p> <p>Note: An Application Center Deploy Tool Ant JAR file is included in this plug-in for interacting with the Worklight Version 6.0.0 Server. The default path to use this file is <code>\${PLUGIN_HOME}/lib/applicationcenterdeploytool.jar</code></p> <p><i>PLUGIN_HOME</i> The location where the agent extracted the IBM Worklight plug-in during deployment on the agent machine.</p> <p>Important: Ensure that the version of the Ant JAR file that you use matches the version on the target server.</p>	Yes
JSON4J JAR File Path	String	<p>The path to the JSON4J JAR (json4j.jar) file.</p> <p>Note: A JSON4J JAR file is included in this plug-in for the Worklight Version 6.0.0 Server. The default path to use this file is <code>\${PLUGIN_HOME}/lib/json4j.jar</code></p> <p><i>PLUGIN_HOME</i> The location where the agent extracted the IBM Worklight plug-in during deployment on the agent machine.</p> <p>Important: Ensure that the version of the JSON4J JAR file that you use matches the version on the target server.</p>	Yes

Upload Application to Application Center

Upload an application to the IBM Worklight Application Center.

Remember: Before you run this step, you must ensure that the **Download Artifacts** process step is run so that it can retrieve the files for deployment.

Table 4. Properties for the Upload Application to Application Center step

Name	Type	Description	Required
Server Path	String	The URL to the Worklight Server. For example: <code>http://localhost:9080</code>	Yes
Context	String	The context root for the Application Center Services enterprise application. For example, <code>/applicationcenter</code>	Yes

Table 4. Properties for the Upload Application to Application Center step (continued)

Name	Type	Description	Required
User	String	The user name that is required to access the Application Center.	Yes
Password	Password	The user password that is used to connect to the Application Center. The default value is <code>\${p:environment/applicationCenterPassword}</code>	Yes
Description	String	The description of the application to upload to the Application Center.	No
Label	String	Normally, the label is taken from the application descriptor that is stored in the file to be uploaded. If the application descriptor does not contain a label, the fallback label is used.	No
Active	True or False	Store the application in the Application Center as an active application.	No
Installer	True or False	Store the application in the Application Center with the installer flag.	No
Ready for production	True or False	Store the application in the Application Center with the ready-for-production flag.	No
Recommended	True or False	Store the application in the Application Center with the recommended flag.	No
Disable SSL Security Checking	True or False	This option allows publishing on secured hosts without verification of the SSL certificate. Use of this flag is a security risk, but might be suitable for testing localhost with temporary self-signed SSL certificates.	No
File	String	The Android application package (.apk) or iOS application (.ipa) file to upload to the Application Center.	Yes
Application Center Ant JAR File Path	String	<p>The path to the Application Center Deploy Tool Ant JAR (<code>applicationcenterdeploytool.jar</code>) file that is used for deployment to the Application Center.</p> <p>Note: An Application Center Deploy Tool Ant JAR file is included in this plug-in for deployment to the Worklight Version 6.0.0 Server. The default path to use this file is <code>\${PLUGIN_HOME}/lib/applicationcenterdeploytool.jar</code></p> <p><i>PLUGIN_HOME</i> The location where the agent extracted the IBM Worklight plug-in during deployment on the agent machine.</p> <p>Important: Ensure that the version of the Ant JAR file that you use matches the version on the target server.</p>	Yes

Table 4. Properties for the Upload Application to Application Center step (continued)

Name	Type	Description	Required
JSON4J JAR File Path	String	<p>The path to the JSON4J JAR (json4j.jar) file.</p> <p>Note: A JSON4J JAR file is included in this plug-in for the Worklight Version 6.0.0 Server. The default path to use this file is <code>\${PLUGIN_HOME}/lib/json4j.jar</code></p> <p><i>PLUGIN_HOME</i></p> <p>The location where the agent extracted the IBM Worklight plug-in during deployment on the agent machine.</p> <p>Important: Ensure that the version of the JSON4J JAR file that you use matches the version on the target server.</p>	Yes

Appendix. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Intellectual Property Dept. for Rational Software
IBM Corporation
5 Technology Park Drive
Westford, MA 01886
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2003, 2013.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.