AT&T API Platform Adapters for IBM® Worklight®

Installation and Setup Guide

Publication Date: November, 2014

Legal Disclaimer

Table of Contents

Contents

Table of Figures

Table of Tables

Table of Examples

1 Introduction

The AT&T API Platform Adapter for IBM® Worklight® provides a simplified way for Worklight developers to access the AT&T API Platform and RESTful APIs. By significantly reducing the complexity of building applications that use the AT&T API Platform, the adapter helps developers quickly bring robust hybrid mobile applications to market.

The adapter facilitates access to the following AT&T Platform RESTful APIs:

- Advertising
-

In App Messaging

- OAuth 2.0 Authentication Management
- SMS
- Speech to Text
- Text to Speech

Note: To learn more about the AT&T API Platform, see the AT&T Developer Program website.

2 Architectural Overview

Figure 1 shows the relationship between the mobile device where the app is running, the Worklight server, and the AT&T API platform.

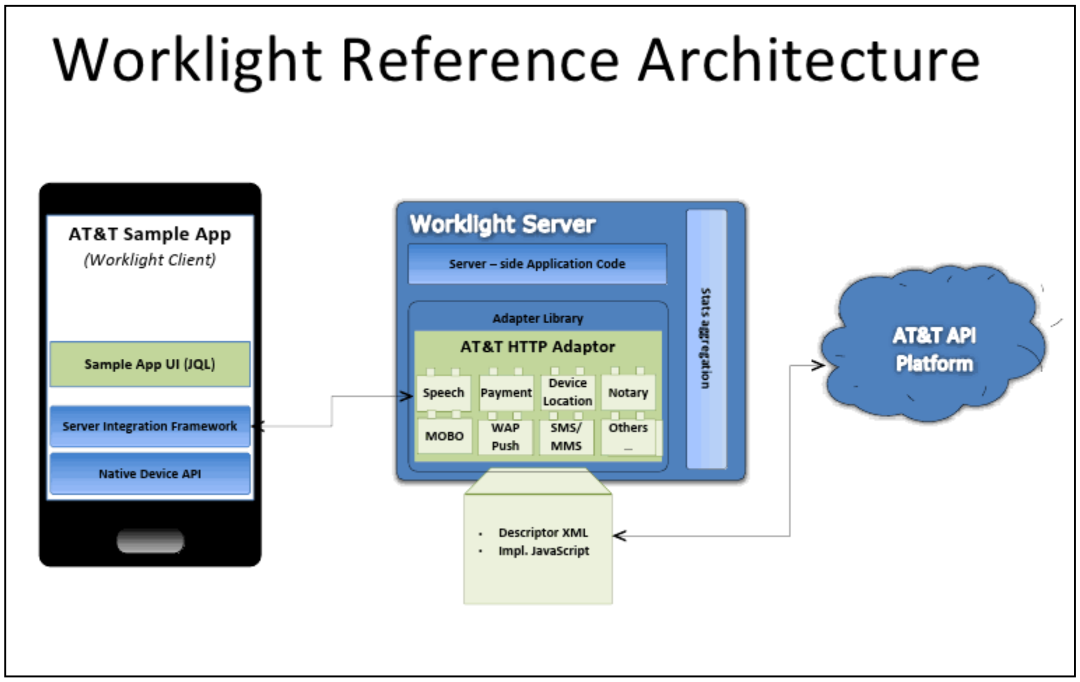


Figure 1: AT&T Worklight Architecture

3 Installing the Worklight Adapters Components

You must complete the following steps to create and test applications (apps) using the adapters on your development system.

- Download Worklight Studio
- Download the AT&T API Platform Adapters for IBM® Worklight®
- Register you application with AT&T and get application keys
- Build and deploy the adapters
- Use the adapters and keys in your mobile app
- Deploy your app to the Worklight server

3.1 Installing the Tools

Perform the following steps to install the required tools for creating an adapter app.

1. Download the AT&T Worklight project code from GitHub at:

   https://github.com/attdevsupport/ATT_APIPlatform_Worklight

2. If you have not already installed Worklight Studio, follow the "Setting up your Environment" section on the "Getting Started" page from IBM at http://www.ibm.com/developerworks/mobile/worklight/getting-started.html

3.1.1 Importing Projects

Perform the following steps to import the adapter and application projects into Eclipse

1. From Eclipse, click File, Import.
2. Expand the General folder, choose Existing Projects Into Workspace as shown in Figure 2.
3. Click Next.

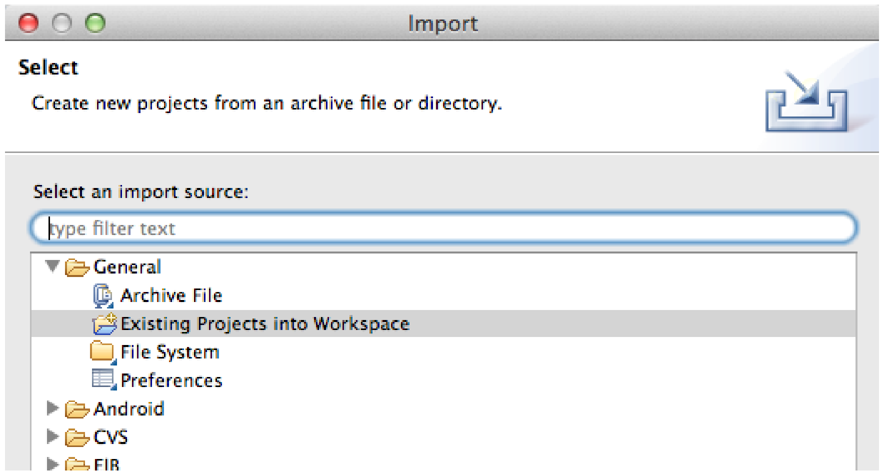
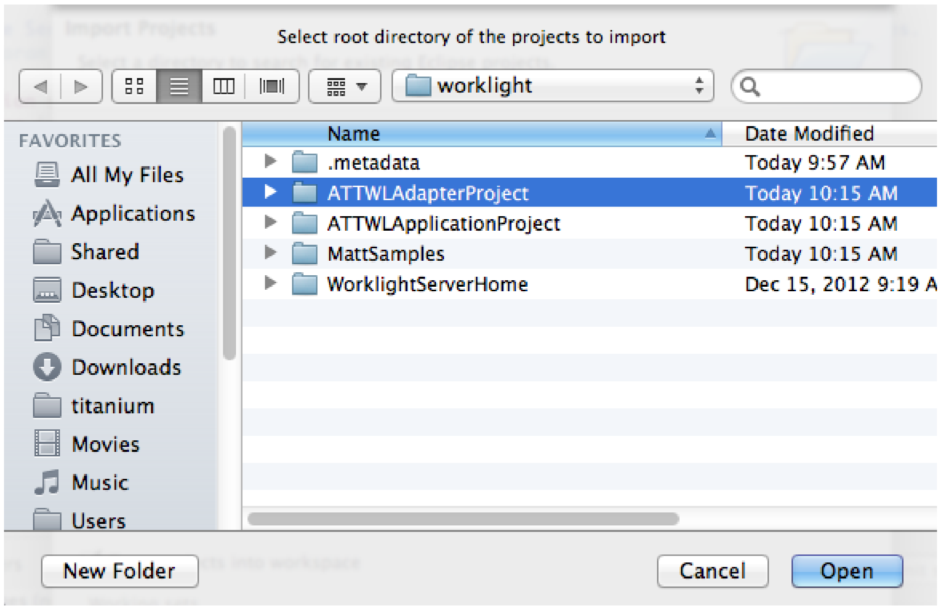


Figure 2: "Existing Projects and Workspace" location

4. Choose Select root directory, and click Browse.
5. Select the ATTWLAdapterProject folder, the ATTWLApplicationProject, ATT_WL_InAppMessaging, and the ATT_WL_Banking folder as shown in Figure 3.
6. Make sure the "Copy projects into workspace" is checked
7. Click Open.
8. Click Finish.



Name	Date Modified
.metadata	Today 9:57 AM
ATTWLAdapterProject	Today 10:15 AM
ATTWLApplicationProject	Today 10:15 AM
MattSamples	Today 10:15 AM
WorklightServerHome	Dec 15, 2012 9:19 A

Figure 3: Adapter Project and Application Project folder locations

3.1.2 Registering your Application

Before you can access the AT&T Platform services, you must register your application for the services that you want to access and to get an API key and secret key. These keys are necessary to call the underlying AT&T RESTful APIs that access the services.

Perform the following steps to register an application.

1. Create a new developer account on the AT&T Developer website, or login to an existing account.
2. Select My Apps from the bar at the top of the page
3. Click Setup a New Application.
4. Chose the APIs you want your application to support.

After your application is registered, you have an API key and Secret key. These keys are necessary to get your applications working with the AT&T Platform APIs.

3.1.3 Configuring your Worklight Application

Perform the following steps to configure your Worklight application and an AT&T sample application:

1. In the application project, open server/conf/worklight.properties.
2. Add the property values from the following table. All values are case-sensitive and should be separated by commas.

Property	Value	Description
appKey	App Key	API key that you received when you registered your application.
secretKey	Secret Key	Secret key that you received when you registered your application.
scope	ADS IMMN MIM SMS SPEECH TTS	Reference to the ATT service that is invoked by your application, for example, • Advertising is ADS • In App Messaging is IMMN,MIM • SMS is SMS • Speech To Text is SPEECH • Text To Speech is TTS

Table 1: Configuration properties for Worklight

4 Building Worklight Adapters

Perform the following steps to build an AT&T Worklight adapter.

1. Expand the ATTWLAdapterProject in the Eclipse project explorer.
2. Expand the adapters folder to see the available adapters, as shown in the following figure.

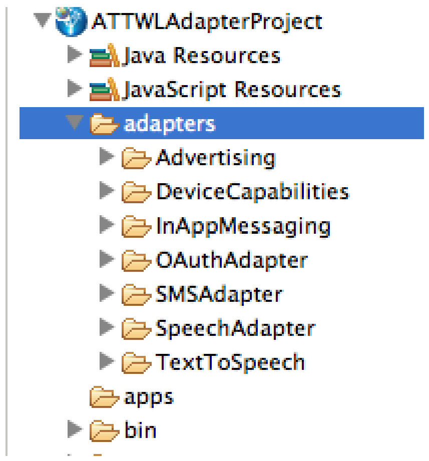


Figure 4: Available adapters.

3. Right click on an adapter folder and click Run As -> Deploy Worklight Adapter, as shown in the following figure. Repeat this for each adapter folder. This builds the .adapter file in the /bin folder.

   These steps build the Worklight adapter and deploy it to the enclosing project. If your application is in a separate Worklight project, the adapters need to be redeployed using the console. This is described in the next section.

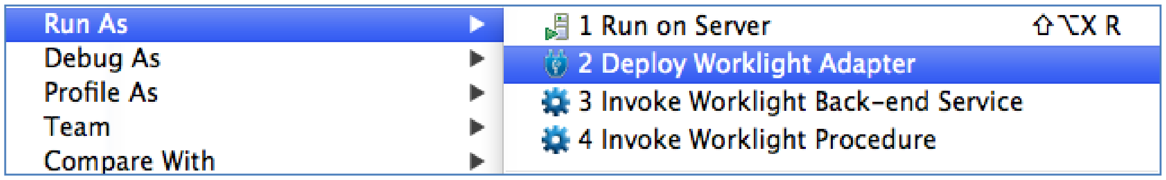
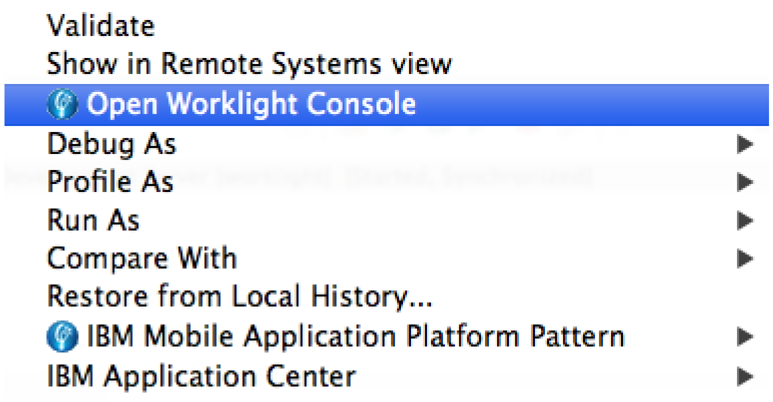

Run As
Debug As
Profile As
Team
Compare With
1 Run on Server ⇧⌥X R
2 Deploy Worklight Adapter
3 Invoke Worklight Back-end Service
4 Invoke Worklight Procedure

Figure 5: Building a Worklight Adapter

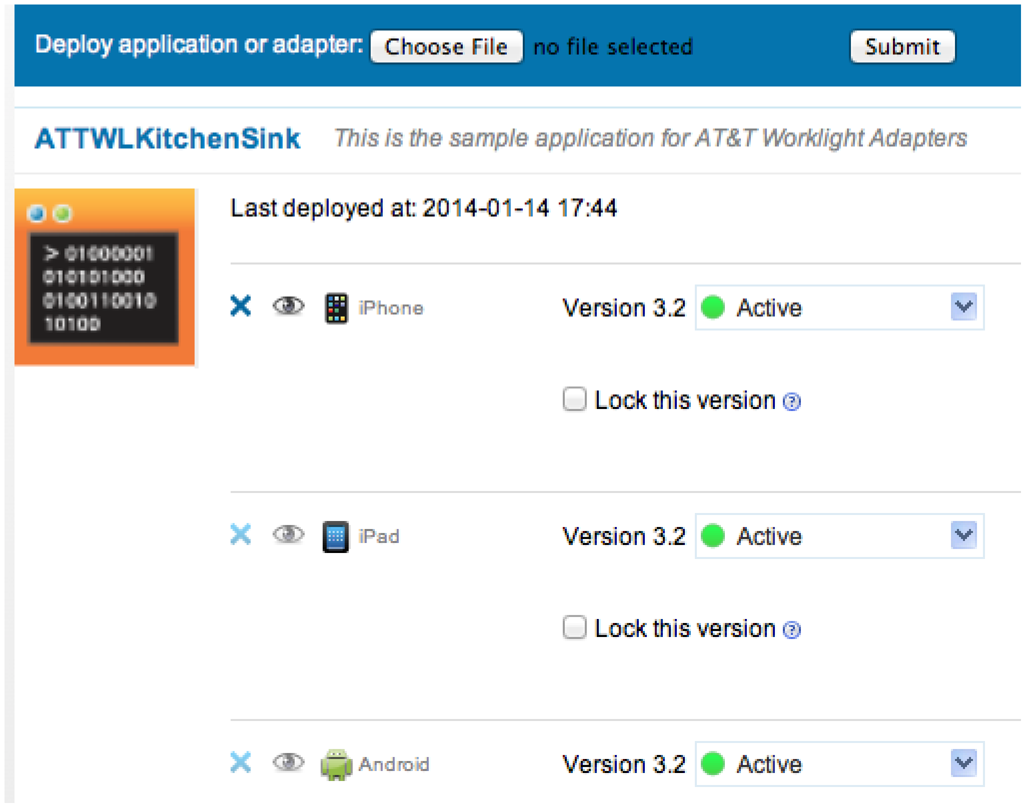
5 Deploying Adapters to a Worklight Application

Perform the following steps to deploy adapters to the sample application or other application.

1. Within the IBM Worklight Developer Studio, right click on the application project folder and chose Open Worklight Console as in this figure:


Validate
Show in Remote Systems view
Open Worklight Console
Debug As
Profile As
Run As
Compare With
Restore from Local History...
IBM Mobile Application Platform Pattern
IBM Application Center

The IBM Worklight Console opens in a Catalog page that lets you work with Applications and Adapters, as shown in the following figure:

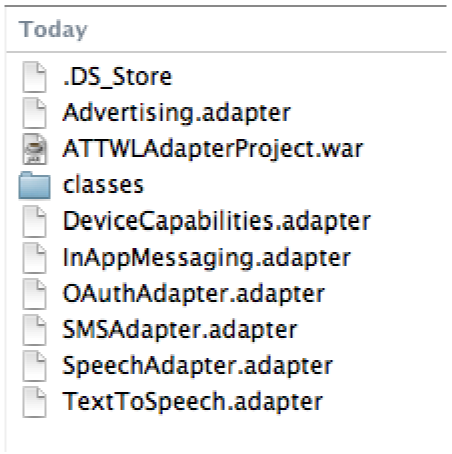


ATTWLKitchenSink *This is the sample application for AT&T Worklight Adapters*

Last deployed at: 2014-01-14 17:44

iPhone	Version 3.2	Active
	Lock this version	
iPad	Version 3.2	Active
	Lock this version	
Android	Version 3.2	Active

2. Click Choose File and select an adapter from the bin folder of the ATTWLAdapterProject:


Today

.DS_Store
Advertising.adapter
ATTWLAdapterProject.war
classes
DeviceCapabilities.adapter
InAppMessaging.adapter
OAuthAdapter.adapter
SMSAdapter.adapter
SpeechAdapter.adapter
TextToSpeech.adapter

3. Click Submit.

   Worklight displays a message that indicates whether the deployment action succeeded or failed. Repeat this procedure for each adapter.

   The deployed adapter is added to the console. Refresh the page to see the details.

6 Receiving SMS Messages

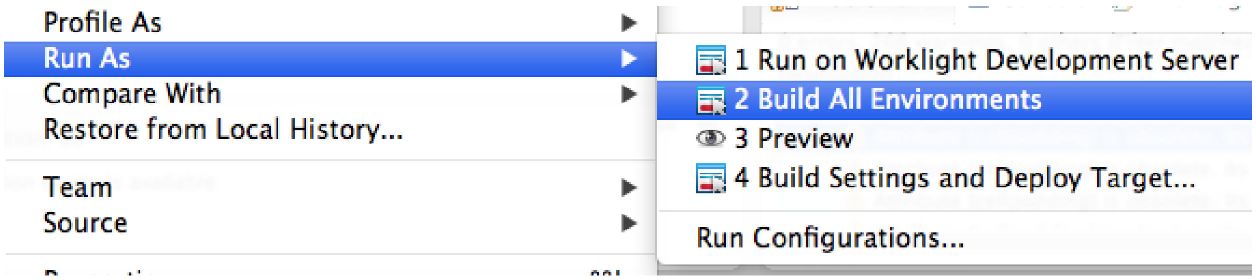
Optionally, perform the following steps to be able to receive SMS messages in the Kitchen Sink sample application.

1. Navigate to the /apps/ATTWLKitchenSink/common/js/sms.js file.
2. Specify your own short code in the shortCode variable that is declared at the top of the file.

7 Building and Deploying the Kitchen Sink Sample App

Perform the following steps to build and deploy the sample application.

1. Open the ATTWLApplicationProject folder in Project Explorer.
2. Expand the apps folder.
3. Select ATTWLKitchenSink.


Profile As
Run As
Compare With
Restore from Local History...
Team
Source
1 Run on Worklight Development Server
2 Build All Environments
3 Preview
4 Build Settings and Deploy Target...
Run Configurations...

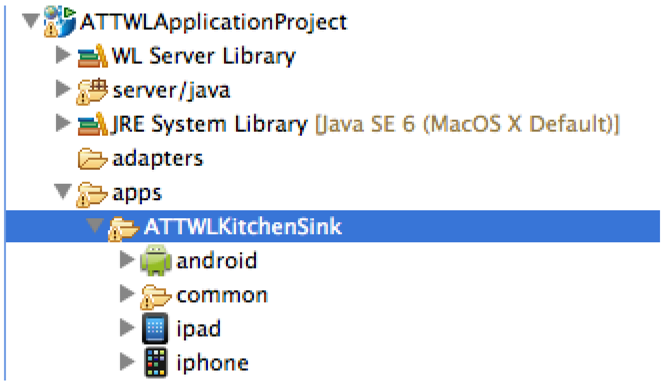
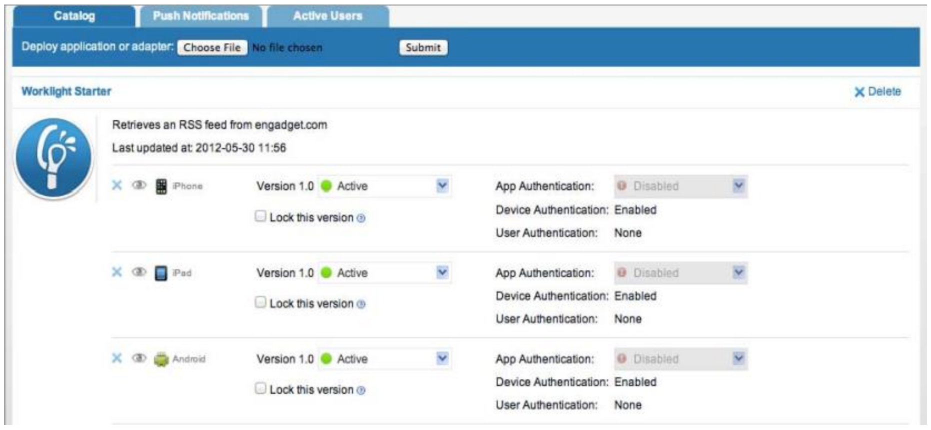


Figure 6: Selecting the ATTWLKitchenSink app.

4. Right click on ATTWLKitchenSink, click *Run As*, then *Build All Environments*. When that build completes, click *Run on Worklight Development Server* from the same menu, as shown in the following figure:

Figure 7: Building and deploying applications.

5. The details of the deployed application are added to the catalog and can be accessed at http://localhost:10080/ATTWLApplicationProject/console, as shown in the following figure.



Deploy application or adapter: No file chosen

Worklight Starter

Retrieves an RSS feed from engadget.com
Last updated at: 2012-05-30 11:56

iPhone	Version 1.0 ● Active	App Authentication:	Disabled
	Lock this version	Device Authentication:	Enabled
		User Authentication:	None
iPad	Version 1.0 ● Active	App Authentication:	Disabled
	Lock this version	Device Authentication:	Enabled
		User Authentication:	None
Android	Version 1.0 ● Active	App Authentication:	Disabled
	Lock this version	Device Authentication:	Enabled
		User Authentication:	None

Figure 8: Details of the deployed application.

7.1 Provisioning CA Certificates for Worklight Server

Starting in Worklight 6.2, Worklight stopped using the system wide JRE cacerts file for trusted CA certificates. This causes some of the AT&T Worklight adapters written in Java to fail with the following error:

```
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to
requested target"
```

This problem has been reported to IBM in PMR 14774,756,000. To date, only a manual work around has been provided:

1. From your Eclipse workspace directory, change to the following subdirectory:

   ATT_APIPlatform_Worklight/WorklightServerConfig/servers/worklight/resources/security

2. Export the Verisign CA certificate chain:

   ```
   keytool -export -alias verisignclass3ca -file verisign_c3.crt -keystore
   $JAVA_HOME/jre/lib/security/cacerts -storepass changeit

   keytool -export -alias verisignclass3g3ca -file verisign_c3g3.crt -keystore
   $JAVA_HOME/jre/lib/security/cacerts -storepass changeit

   keytool -export -alias verisignclass3g5ca -file verisign_c3g5.crt -keystore
   $JAVA_HOME/jre/lib/security/cacerts -storepass changeit
   ```

3. Import the Verisign CA certificate chain into Workight's keystore:

```
keytool     -import -trustcacerts -file verisign_c3.crt -alias verisign_c3ca -keystore key.jks
-storepass worklight
```

```
keytool   -import -trustcacerts -file verisign_c3g3.crt -alias verisign_c3g3ca -keystore key.jks
-storepass worklight
```

```
keytool  -import -trustcacerts -file verisign_c3g5.crt -alias verisign_c3g5ca -keystore key.jks
-storepass worklight
```

4. The above import commands give a message saying the Verisign CA certificates are already in the system wide CA keystore (a.k.a. TrustStore, cacerts) and do you still want to import them into your keystore. Type “yes” because Worklight Server no longer uses the system wide CA keystore.

   Example message:

   Certificate already exists in system-wide CA keystore under alias <verisignclass3ca>

   Do you still want to add it to your own keystore? [no]:

5. Restart the Worklight sever

   Notes: Due to the manual nature of this workaround, it may cease to work at unpredictable times. Some of the events that may break this work around are (but not limited to):

   a. IBM Worklight may at any time decide to recreate their self-generated keystore. It is unknown if they will include the imported CA certificates, so a re-import may be necessary.

   b. IBM Worklight may change the default keystore password or Java may change the default cacerts password.

   c. If you delete the Worklight Server in Eclipse Worklight DevStudio, the server will be recreated, but with a new keystore. You will have to re-import the Verisign CA certificates afterwards.

   d. AT&T API platform (api.att.com) may change the CA certificate signer without warning. If this happens, a different CA certificate chain would have to be imported from cacerts.

e. When these Verisign CA certificates expire, the new CA certificate chain or some portion thereof will need to be imported.

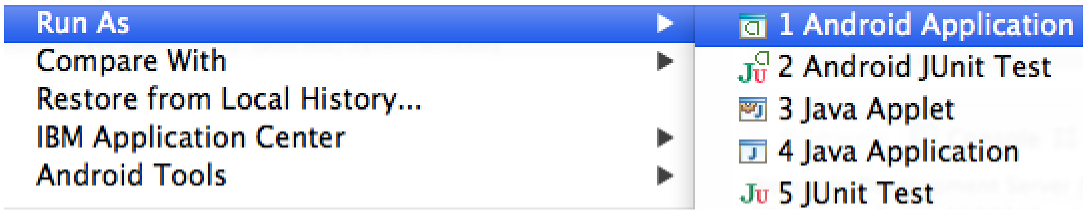
7.2 Running the Android App

For the android environment, the following new android project is created automatically after *Build All Environments* is run.

For the kitchen sink sample application, the name of the project is:

`ATTWLApplicationProjectATTWLKitchenSinkAndroid`

This project can be executed as a normal android application by right clicking on it, selecting Run As, then Android Application:



7.3 Running the iOS App

For the iPhone or iPad environments, right click on the iphone or ipad folder, and select Run As -> Xcode project, as shown in the following figure. This opens Xcode, where the app can be executed.

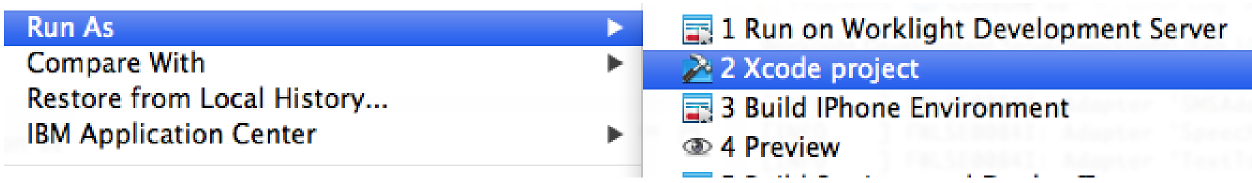

Run As
Compare With
Restore from Local History...
IBM Application Center
1 Run on Worklight Development Server
2 Xcode project
3 Build IPhone Environment
4 Preview

Figure 9: Running the app on iOS

8 Using the Adapters to Create a New Worklight Application

To create a new Worklight application, see Working with Worklight.

1. In your application, invoke the ATT adapters from Java Script. Worklight applications can invoke adapter procedures to communicate with any data source without being subjected to same origin constraints.
2. Invoke an adapter procedure to create an invocationData object.

invocationData Object Code Sample

```
var invocationData = {
     adapter : ‘SMSAdapter’,
     procedure : ‘sendSMS’,
     parameters : [{'body' : { “outboundSMSMessage”: {"Message" :
“Hello All”,  "Address" : “555-555-1212”}},
'contentType' : 'application/json', 'accept' : 'application/json',
'accessToken':'Bearer ' +  window.localStorage.accessToken}]
    };
```

Example 1: Creating an invocationData object

The invocationData object consists of the following JSON block of properties:

Property	Description
adapter	A string that contains the name of the adapter as specified in the <wl:adapter> element of the adapter xml file.
procedure	Procedure name as defined in the adapter xml file.
parameters	An array of parameters passed on to the remote procedure.

Table 2: invocationData Object Properties

3. Define the failure and success behavior in an options object.

Options Object Code Sample

```
var options = {
     onSuccess : yourSuccessCallback,
     onFailure : yourFailureCallback,
     InvocationContext {}
    };
```

Example 2: Creating an Options object

The options object consists of the following JSON block of properties:

Property	Description
onSuccess	The function to be invoked on successful completion of the asynchronous call.
onFailure	The function to be invoked on failure.
invocationContext	Optional parameter. object that is returned to the success and failure handlers.

Table 3: Options Object Properties

4. Invoke the procedure using the invocationData object and options object.

   ```
   WL.Client.invoke.Procedure(invocationData, options)
   ```

To learn more about installing Worklight adapters, see the server side development instructions at:

http://www.ibm.com/developerworks/mobile/worklight/getting-started.html#server-side-development