



## TouchTest™ Developer Guide

SOASTA TouchTest™ Developer Guide

©2015, SOASTA, Inc. All rights reserved.

The names of actual companies and products mentioned herein may be the trademarks of their respective companies.

This document is for informational purposes only. SOASTA makes no warranties, express or implied, as to the information contained within this document.

---

## Table of Contents

<b>About the Developer Guide</b> .....	<b>1</b>
<b>TouchTest™ Documentation</b> .....	<b>2</b>
TouchTest iOS Documentation .....	2
TouchTest Android Documentation .....	3
TouchTest Continuous Integration Documentation .....	3
MakeAppTouchable Documentation .....	4
<b>Device and Architecture Support</b> .....	<b>5</b>
iOS Supported Devices .....	5
Android Supported Devices .....	5
TouchTest Mobile Apps and Agents .....	6
<b>Prerequisites for iOS using Xcode</b> .....	<b>7</b>
CloudTest/TouchTest Utility Software .....	7
<b>Prerequisites for Android using Eclipse</b> .....	<b>9</b>
CloudTest/TouchTest Utility Software .....	9
<b>TouchTest Workflows (Static/Dynamic)</b> .....	<b>11</b>
Static Instrumentation of an Xcode Project .....	11
Dynamic Instrumentation of an APP file .....	12
Dynamic Instrumentation of an IPA file .....	12
iOSAppInstaller Utility .....	13
<b>Miscellaneous TouchTest Configuration</b> .....	<b>14</b>
Sharing a TouchTestable App with TouchTest Users .....	14
TouchTest and iOS Network Link Conditioners .....	14
<b>Install the iOS TouchTestable App from Xcode</b> .....	<b>15</b>
<b>Install the TouchTestable Android App from Eclipse</b> .....	<b>16</b>
<b>Adding TouchTest™ IDs to an iOS App</b> .....	<b>17</b>
Adding a Mobile App to CloudTest® Manually .....	2
<b>Appendix: MakeAppTouchable Reference</b> .....	<b>4</b>

## About the Developer Guide

This Developer Guide provides a compendium of developer-only references, additional details not presented in the TouchTest tutorials, and links to relevant documentation across the TouchTest documentation set (i.e. found in both tutorials & Knowledge Base).

Additionally, Developer prerequisites, device support, and other developer-centric issues are gathered here.

For mobile web recording and playback instructions, as well as details about configuration and workflow in a given iOS and platform, refer to the appropriate tutorial (e.g. from among those listed below).

In general, once you have identified the tutorial aimed at your specific environment, you should use that tutorial for recording and playback instructions.

If you are a Titanium Studio developer, using either iOS or Android, please refer to the separate tutorial for your environment (listed below).

When in doubt, refer to the tutorial for your environment.

## TouchTest™ Documentation

This Developer Guide serves as an index to developer-only issues in TouchTest such as device support, workflow, as well as a guide to the TouchTest™ documentation set— which includes the [CloudLink Community](#), [Knowledge Base](#), and [Documentation](#) nodes, as well as a complete set of tutorials for the various possible configurations using TouchTest™.

For user workflow, SOASTA recommends that you begin with either the [TouchTest iOS Tutorial](#) or the [TouchTest Android Tutorial](#) although in some cases it is appropriate to begin with a platform (Titanium Studio) or Continuous Integration (CI) tutorial [here](#).

Developers should be aware of the environment-specific tutorials presented below, all of which have been created for the convenience of users developing and testing in these environments. TouchTest tutorials are categorized by mobile platform and/or continuous integration platform below.

### TouchTest iOS Documentation

[TouchTest™ iOS Tutorial](#) – This tutorial provides a basic introduction to using TouchTest in the Xcode environment for both developers and end-users who may be testing a mobile app in an environment setup by a developer. Both static instrumentation and dynamic instrumentation are covered.

[TouchTest™ for Appcelerator iOS Tutorial](#) – This tutorial provides a basic introduction to using TouchTest in the Titanium Studio environment and is intended for both developers and testers working in a developer-led mobile app environment. The Titanium Studio overlap with Xcode is discussed where it occurs (for example, as it relates to provisioning a device).

[TouchTest™ Advanced Tutorial](#) – This tutorial presents the advanced case for using TouchTestIDs in iOS environments to enhance the readability of mobile app tests, and additionally, delves further into the use of accessors such as outputs, validations, and waits to enhance mobile testing. Additionally, a comprehensive App Action References is included in this document.

[TouchTest™ OpenGL Tutorial](#) – This tutorial presents the necessary steps for developers to expose app internal values, such as those used in mobile apps that utilize OpenGL and similar technologies, for use in TouchTest mobile app testing.

[TouchTest™ Web \(Build\) for iOS Tutorial](#) – This tutorial introduces the use of the TouchTest Web app (which can be downloaded as an Xcode project on the TouchTest Welcome page). TouchTest Web is used to record web-based apps and sites on your iOS device. Once deployed, TouchTest Web works with TouchTest Agent and your CloudTest desktop to enable you to capture browser actions easily and play them back onto any iOS device(s) in just a few simple steps.

## **TouchTest Android Documentation**

[TouchTest™ Android Tutorial](#) – This tutorial provides a basic introduction to using TouchTest in the Eclipse environment for both Android developers and end-users who may be testing a mobile app in an environment setup by a developer.

[TouchTest™ for Appcelerator Android Tutorial](#) – This tutorial provides a basic introduction to using TouchTest in the Titanium Studio environment for both Android developers and end-users who may be testing a mobile app in an environment setup by a developer.

[TouchTest™ Web for Android Tutorial](#) – This tutorial introduces the use of the TouchTest Web app (available on the CloudTest Welcome Page), which is used to record;web-based apps and sites on your Android device. Once deployed, TouchTest Web works with TouchTest Agent and your CloudTest desktop to enable you to capture browser actions easily and play them back onto any Android device(s) in just a few simple steps.

## **TouchTest Continuous Integration Documentation**

[TouchTest™ Jenkins CI for iOS Tutorial](#) – This tutorial presents the necessary procedures for incorporating TouchTest using Xcode into continuous integration environments that utilize Jenkins/Hudson with iOS.

[TouchTest™ Jenkins CI for Android Tutorial](#) – This tutorial presents the necessary procedures for incorporating TouchTest using Eclipse into continuous integration environments that utilize Jenkins/Hudson with Android.

[TouchTest™ Appcelerator Jenkins CI Tutorial](#) - This tutorial presents the necessary procedures for incorporating TouchTest using Titanium Studio into continuous integration environments that utilize Jenkins/Hudson.

[TouchTest™ Bamboo CI for iOS Tutorial](#) – This tutorial presents the necessary procedures for incorporating TouchTest using Xcode into continuous integration environments that utilize Bamboo CI with iOS.

[TouchTest™ for Quality Center Tutorial](#) - This tutorial presents the necessary procedures for incorporating TouchTest into an HP Quality Center environment.

## MakeAppTouchTestable Documentation

Use the following list to find the TouchTest example scenarios you need.

- For a complete basic iOS scenario, using MATT to instrument an Xcode project, APP, or IPA, refer to the [TouchTest iOS Tutorial](#).
- For a complete basic Android scenario, using MATT to instrument an Android project in Eclipse, or APK, refer to the [TouchTest Android Tutorial](#).
- For an advanced iOS scenario using Xcode, with TouchTest ID examples, refer to the [TouchTest Advanced Tutorial](#).
- For extensive Continuous Integration examples of using TouchTest, including the CloudTest Jenkins Plugin, MakeAppTouchTestable module, in combination with xcodebuild and xcrun from within Jenkins to instrument an iOS project, APP, or IPA, refer to the [TouchTest Jenkins CI for iOS Tutorial](#).
- For extensive Continuous Integration examples of using TouchTest, including MATT, as well as Ant and ABD, to instrument an Android project, or APK, refer to the [TouchTest Jenkins CI for Android Tutorial](#).
- If you are using Android build.xml configuration, refer to the CloudLink, KnowledgeBase article, [Android build.xml configuration for MakeAppTouchTestable](#).

## Device and Architecture Support

Refer to [TouchTest Device Support](#) for the latest updates to iOS and Android Supported Devices.

### iOS Supported Devices

SOASTA TouchTest™ supports iOS 5.0 version or later and the following devices:

- iPhone 3GS
- iPhone 4
- iPhone 4s
- iPhone 5
- iPod Touch (3<sup>rd</sup> generation)
- iPod Touch (4<sup>th</sup> generation)
- iPad
- iPad 2
- The new iPad

SOASTA TouchTest™ Driver is not compatible with the armv6 architecture. The following architectures are supported:

- armv7
- armv7s

**Note:** For dynamic instrumentation, iOS 6 and up versions are required. Static instrumentation is supported in all supported iOS versions.

### Android Supported Devices

SOASTA TouchTest™ supports devices running Android SDK 2.3.3 (Gingerbread) or later.

## TouchTest Mobile Apps and Agents

The following mobile web components are crucial role players in TouchTest. See the linked documentation for more about TouchTest's mobile components.

- **TouchTest Agent Android App**

The TouchTest Agent app for Android (refer to [Configuring a TouchTest Agent](#) for a quick review of this mobile user agent).

- **TouchTest Agent (iOS)**

The TouchTest Agent URL for the given CloudTest instance (refer to [Configuring a TouchTest Agent](#) for a quick review of this mobile user agent).

- **TouchTest Web for Android App**

The TouchTest Web for Android app (refer to the [TouchTest Web for Android Tutorial](#) for more information).

- **TouchTest Web for iOS**

The TouchTest Web for iOS Xcode project is available on the Welcome page of your CloudTest instance (also refer to the [TouchTest Web for iOS \(Build\) Tutorial](#) for more information).

## Prerequisites for iOS using Xcode

An Xcode developer will need the following running on a Mac OS X desktop with access to CloudTest Lite or some other CloudTest instance:

- Xcode (version 4.2.1 or later)
- iOS Developer Program membership
- At least one properly provisioned device(s) (Xcode simulators can also be used)

## CloudTest/TouchTest Utility Software

A developer with Mobile Device Administrator privileges should perform the following prerequisites (if you're a CloudTest Lite user you're already a Mobile Device Administrator):

- Download and unarchive the MakeAppTouchTestable Utility from the CloudTest Welcome page.
- Your mobile app's Xcode project file (duplicate your project's target and run this utility against it using the instructions below).
- If you're using the Jenkins/Hudson continuous integration environment, refer to the [TouchTest™ Jenkins CI Tutorial](#) for a complete start to finish scenario using the CloudTest Jenkins/Hudson Plugin.



**Note:** This archive contains the necessary drivers for both the manual and automatic project modifications methods described below.

- Choose whether to make your app TouchTestable automatically or manually. The automatic method is highly recommended. Manual instructions are presented only in this Developer Guide as an Appendix.

SOASTA provides both methods to accommodate differing user concerns—while some developers prefer to perform each and every step as a matter of rigor; others are more interested in ease of use. If you'd like to try the utility, make a sandbox in a non-crucial location and try it out there.

- For most users, the “Using the MakeAppTouchTestable Utility” automatic method will provide an easier, alternate route.

- For intrepid users who want to know how everything works, TouchTest™ can be added manually using the steps in “Manually Adding TouchTest™ to an Xcode Project” appendix below. You can experiment by trying it out on a sandbox project.
- A Mobile App must be added (as a repository object) to CloudTest®. This is handled by the MakeAppTouchable utility in the case of the automatic method above, but must be done separately if using the manual method. In either case, the Mobile App repository object Launch URL field must minimally match the Xcode project URL scheme field (minimally, because it does take other arguments).

Once these developer prerequisites are met, TouchTest™ recording is performed by any user by deploying the following CloudTest® components:

- The TouchTest™ Agent; a per mobile device browser agent pointed at CloudTest® and reachable by the Conductor name above. Refer to the recording and playback instructions in the [TouchTest™ iOS Tutorial](#).
- If hybrid or web testing is used, then the TouchTest Web app is required. This is available as an Xcode project via the Welcome page of your CloudTest instance.
- The CloudTest® (or CloudTest® Lite instance) for which both are configured, and
- A CloudTest® mobile target configured to use all of the above (specified prior to recording)

## Prerequisites for Android using Eclipse

This guide presumes that the Android developer has a basic familiarity with the Eclipse development environment and has the following configuration:

- Eclipse (version 3.6 or later) with Android SDK installed along with the Eclipse plugin. The Minimum Android Version supported for use with TouchTest™ is 2.3.3 (Gingerbread).
- The basic Android tutorial also uses the Android NDK toolset in conjunction with the Android mobile app, DroidFish, although this is not a requirement of TouchTest itself.
- Unless you have a different Android mobile app you'd like to use with this tutorial, [install the Android NDK toolset](#) into your Eclipse environment before proceeding with the basic tutorial.

## CloudTest/TouchTest Utility Software

Developers who would also like to add TouchTest to an existing mobile app can find basic instructions for using the MakeAppTouchTestable utility in this guide.

- Download and unarchive the MakeAppTouchTestable Utility from the CloudTest Welcome page.
- If you're using the Jenkins/Hudson continuous integration environment, refer to the [TouchTest™ Jenkins CI Tutorial](#) for a complete start to finish scenario using the CloudTest Jenkins/Hudson Plugin.



**Note:** This archive contains the necessary drivers for both the manual and automatic methods described below.

▼	MakeAppTouchable	Today 5:28 PM
▼	android	Today 5:28 PM
	.DS_Store	Today 5:28 PM
▶	eclipse	Today 5:27 PM
	custom_rules.xml	Aug 23, 2012 6:52 PM
	.DS_Store	Today 5:28 PM
▶	TouchTestDriver	Today 11:53 AM
▶	ios	Today 11:53 AM
▶	lib	Today 11:53 AM
	MakeAppTouchable.jar	Today 11:53 AM
▶	titanium	Today 11:53 AM
	ReadMe.txt	Jun 19, 2012 5:52 PM
	.DS_Store	Today 5:28 PM

## TouchTest Workflows (Static/Dynamic)

When you make a mobile app TouchTestable, you do so by instrumenting it using either a static or dynamic method with the MakeAppTouchTestable utility.

The MATT utility supports two instrumentation methods: static and dynamic.

- Dynamic instrumentation occurs when MATT instruments a compiled file (i.e. an IPA or APP file). This method requires SOASTA 51 (TouchTest 7040.11) and can be applied to iOS version 6 and 7 only.
- Static instrumentation occurs when MATT instruments a project file (i.e. .xcodproj). Static instrumentation is available in all TouchTest releases and for all supported iOS versions.

Because there are many possible workflows in a development environment, the *possible* steps are presented below *a la carte*. For each workflow there will be (minimally):

- A First step; used to retrieve the source project (all workflows)
- In between the first and last steps, each Jenkins job will have:
  - A CloudTest Plugin, MakeAppTouchTestable step;
  - with one or more Build steps,
  - and one or more Install/Run steps
- A Last step; to Play the Composition

**Note:** Provisioning and codesigning can be performed either using the CloudTest Jenkins Plugin, MATT, Advanced steps to enter the APP/IPA optional parameters (refer to the relevant sections). Or, authentication steps can be done using xcrun.

There is a one-time provisioning profile step that should be run prior to running the complete Jenkins job.

In general, you will delay the MATT step only until the last logical point (i.e., only after the GIT if you're using static instrumentation, after the APP step if for dynamic using a simulator(s), and after the IPA step for dynamic using physical devices.

### Static Instrumentation of an Xcode Project

In this workflow, you'll apply MATT to the Xcode project itself, using the `project` parameter.

- *Apply MATT* using the `project` parameter (static only)
- *Build the APP* using `xcodebuild`
- (optional) *Build the IPA* using `xcrun`
- Run/install using either Xcode or `iOSAppInstaller`

## Dynamic Instrumentation of an APP file

In this workflow, you'll apply MATT to the compiled APP file using the `appbundle` parameter.

- *Build the APP* using Xcode or `xcodebuild`
- *Apply MATT* to the compiled APP using `appbundle`
- *Run App on iOS Simulator* using Xcode or `iOSAppInstaller`

## Dynamic Instrumentation of an IPA file

In this workflow, you'll delay applying MATT until the IPA is created

- *Build the APP* using `xcodebuild`
- *Build the IPA* file using `xcrun`,
- *Apply MATT* to the compiled IPA using `ipa`
- Install the app on the device using Xcode or `iOSAppInstaller`.

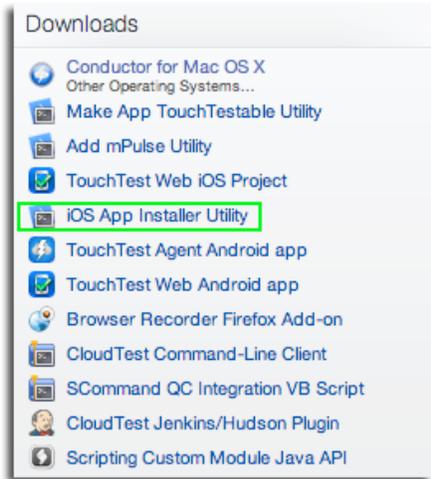
## iOSAppInstaller Utility

Using the newly TouchTestable APP or IPA you can now easily:

- Deploy the TouchTestable app to an iOS device or simulator.

SOASTA provides the iOSAppInstaller Utility for this purpose.

You can download the latest iOSAppInstaller Utility from the Central > Welcome page, Downloads section of your own server instance.



Unzip the utility at this time if you have yet to do so and note the contents of the resulting iOSAppInstaller folder.

- For a Simulator, use:  
`./bin/ios_sim_launcher -app`
- For an iPhone or iPad, use:  
`./bin/ios_app_installer -ipa`

## Miscellaneous TouchTest Configuration

### Sharing a TouchTestable App with TouchTest Users

Developers and testers can use TouchTest (via its CloudTest instance) to make the TouchTestable app available to other users. Refer to the CloudLink, KnowledgeBase article, [Uploading a Mobile App IPA or APK](#). Then, point your users to [Download a Mobile App IPA or APK](#)

### TouchTest and iOS Network Link Conditioners

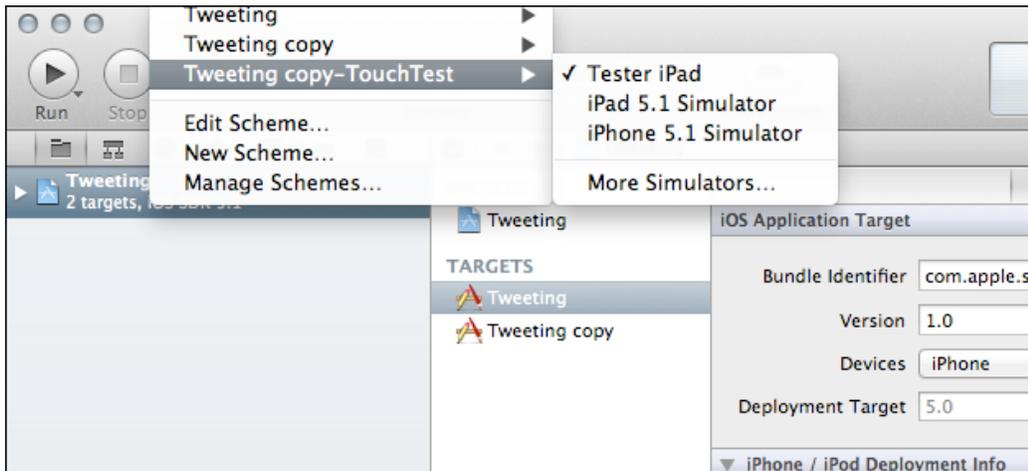
Refer to the CloudLink, KnowledgeBase article, [Using an iOS Network Link Conditioner with TouchTest](#) for instructions.

## Install the iOS TouchTestable App from Xcode

Using the new scheme that was added to your Xcode project by the MakeAppTouchTestable utility, you can now easily:

- Deploy the TouchTestable app to an iOS device or simulator.
- Prepare the iOS device for TouchTest™ recording or playback using the instructions in the [TouchTest™ iOS Tutorial](#).

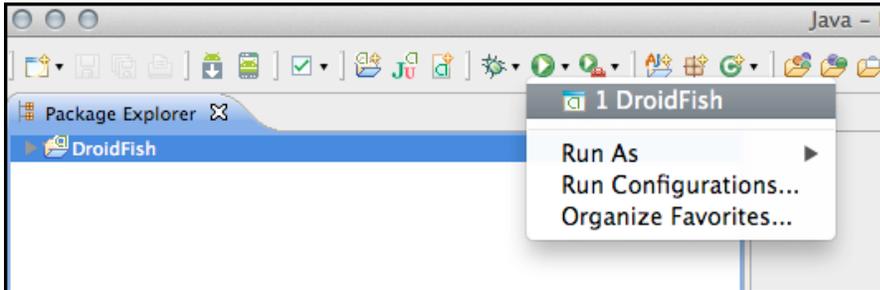
To deploy and run the TouchTestable app, select the “-TouchTest” scheme from the drop-down and the iOS device or simulator on which you’d like to run, then click the Run button.



## Install the TouchTestable Android App from Eclipse

After doing a Refresh on your Android project you are ready to install it.

- After installing to a device or simulator, prepare the Android device for TouchTest™ recording or playback using the instructions in the [TouchTest™ Android Tutorial](#).
1. Connect the Android Device to the desktop client running Eclipse via USB. You can also use a simulator. A physical device must have the following set:
    - The stock browser on the device should support launch of native apps
    - The Developer Options, USB Debugging box should be enabled
    - The Security, Unknown sources box should be enabled
  2. Click the Run button on the toolbar to build the project and push it to the Android Device.



- If the Android device is connected, and no Android Virtual Device (AVD) is running, the app is installed to the device.
- If more than one device or emulator combination is available, then a selection box appears for you to choose
- If no device is connected, the Android Virtual Device (AVD) will run and the Droidfish app will be installed to it instead. The AVD must be using SDK 2.3.3 or later.

When all of the conditions and steps above are completed, the app is pushed onto the Android Device. The Eclipse Console will indicate success and the app will launch.

```
[2012-09-19 20:05:21 - DroidFish] -----
[2012-09-19 20:05:21 - DroidFish] Android Launch!
[2012-09-19 20:05:21 - DroidFish] adb is running normally.
[2012-09-19 20:05:21 - DroidFish] Performing org.petero.droidfish.DroidFish activity
launch
[2012-09-19 20:05:21 - DroidFish] Automatic Target Mode: using device '015d15b4da23f411'
[2012-09-19 20:05:21 - DroidFish] Uploading DroidFish.apk onto device '015d15b4da23f411'
[2012-09-19 20:05:23 - DroidFish] Installing DroidFish.apk...
[2012-09-19 20:05:26 - DroidFish] Success!
[2012-09-19 20:05:26 - DroidFish] Starting activity org.petero.droidfish.DroidFish on
device 015d15b4da23f411
[2012-09-19 20:05:26 - DroidFish] ActivityManager: Starting: Intent
{ act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER]
cmp=org.petero.droidfish/.DroidFish }
```

## Adding TouchTest™ IDs to an iOS App

For iOS developers, SOASTA TouchTest™ provides TouchTest Driver to enhance mobile app locators as an integral part of touch-testing. This is not necessary for Android projects. Unlike the prior Xcode examples, this section uses the example app, Touches.

**TIP:** Please refer to the [TouchTest™ Advanced Tutorial](#) for a complete A-Z example of how to add touchTestIDs to a mobile app.

The use of `touchTestId` in Locators in the sample app, Touches, is described below.

**Note:** Use the `touchTestId` conditionally in a manner that guarantees it is not part of code that gets submitted to the App Store.

For example, by using:

```
#ifdef TOUCHTESTDRIVER
```

1. Identify the source file where the view is initialized. One example of where views can be initialized is in the method, `awakeFromNib`.
2. Include the TouchTest™ header file in the source file with the initialization of the view by using:

```
#ifdef TOUCHTESTDRIVER
#import "TouchTestDriver.h"
#endif
```

3. Next, call the `setTouchTestId` method on each view that will be located.

The string parameter to `touchTestId` is the value that you want to be used to locate the element, in this case *yellowSquare*, *blueSquare* and *pinkSquare*.

For example,

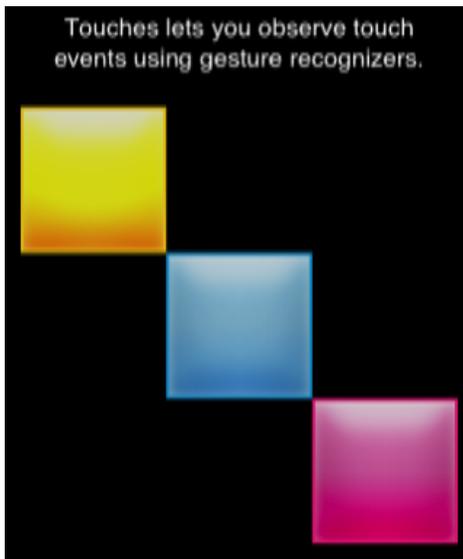
```
#ifdef TOUCHTESTDRIVER
    [firstPieceView setTouchTestId:@"yellowSquare"];
    [secondPieceView setTouchTestId:@"blueSquare"];
    [thirdPieceView setTouchTestId:@"pinkSquare"];
#endif
```

The blue lines in the screenshot below were added to the given source file.

```
- (void)awakeFromNib
{
    #ifdef TOUCHTESTDRIVER
        [firstPieceView setTouchTestId:@"yellowSquare"];
        [secondPieceView setTouchTestId:@"blueSquare"];
        [thirdPieceView setTouchTestId:@"pinkSquare"];
    #endif

    [self addGestureRecognizersToPiece:firstPieceView];
    [self addGestureRecognizersToPiece:secondPieceView];
    [self addGestureRecognizersToPiece:thirdPieceView];
}
```

The sample app, Touches, demonstrates the use of `setTouchTestId`. In this example, each TouchTestId corresponds to a square in Touches.



The clip below was recorded using Touches. Note that the `touchTestDriverId` is shown where present.

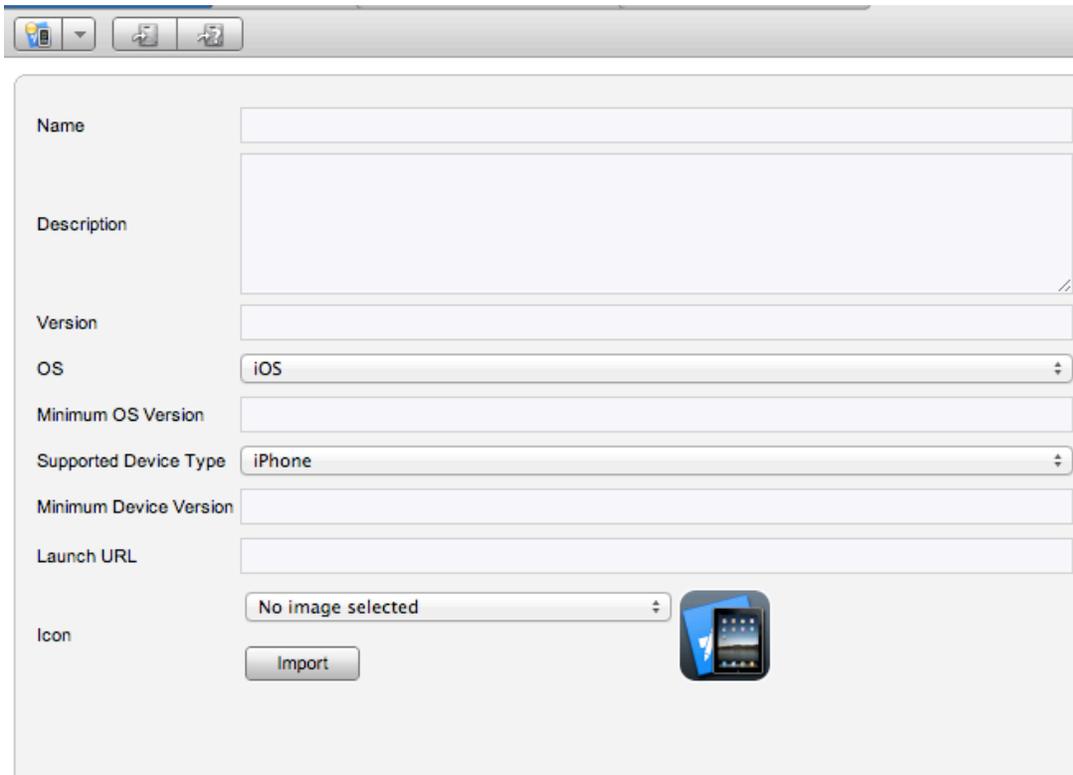
Name	Target Name	Operation	touchTestDriverId
App Action1	My Touches Target	tap	id=yellowSquare
App Action2	My Touches Target	tap	text=2x
App Action3	My Touches Target	pan	id=yellowSquare 101.000000,-5.000000
App Action4	My Touches Target	tap	id=blueSquare
App Action5	My Touches Target	tap	id=blueSquare
App Action6	My Touches Target	pan	id=blueSquare -105.000000,96.000000
App Action7	My Touches Target	pan	id=pinkSquare -86.000000,-89.000000
App Action8	My Touches Target	pan	id=blueSquare 108.000000,37.000000
App Action9	My Touches Target	pan	id=pinkSquare -18.000000,11.000000



## Adding a Mobile App to CloudTest® Manually

In cases where manual project integration has been used, it will be necessary to manually add a mobile app to CloudTest®.

1. Select Central > Mobile Apps and then click New. The Mobile App form appears.
2. Enter the app name as it will appear in the drop-down for user selection. Generally, this will also be the Xcode project name.



The screenshot shows a web form for adding a mobile app. The form has the following fields and options:

- Name:** A text input field.
- Description:** A large text area with a diagonal slash icon in the bottom right corner.
- Version:** A text input field.
- OS:** A dropdown menu with "iOS" selected.
- Minimum OS Version:** A text input field.
- Supported Device Type:** A dropdown menu with "iPhone" selected.
- Minimum Device Version:** A text input field.
- Launch URL:** A text input field.
- Icon:** A dropdown menu showing "No image selected" and a small image of an iPhone. Below it is an "Import" button.

3. Optionally, enter a description and an app version number. Version number will generally match Xcode project details.
4. Only iOS is supported currently. For this release, TouchTest™ supports iOS 5.0+ versions only).
5. Set the Supported Device Type to Universal, iPhone, or iPad.
6. In the Launch URL field, provide the unique URL Scheme you defined in XCode, plus any additional arguments relevant to your mobile app.

For example,

```
soasta-mobile-test-app-6ea1a5d1-8118-4135-9844-176f0c704262://  
key1=value1&key2=value2&key3=value3
```

where `soasta-mobile-test-appID://` is the name of your mobile app including the `://` and additional arguments are in the form `key1=value1&key2=value2`.

**Note:** Without a correctly formed Launch URL testing will not happen.

7. Optionally, import an app image for your mobile app to visually represent the correlation of TouchTest™ Agent with your app.

Supported image types include JPEG, PNG, and GIF. Images can be pre-edited to the requisite 57 pixels wide by 57 pixels tall. Images that are not cropped will be shrunk to fit within the requisite dimensions.

Click Save to create this mobile app object in CloudTest®.

## Appendix: MakeAppTouchTestable Reference

SOASTA 51 greatly improved command line help for the MakeAppTouchTestable utility. The complete reference now includes all of the following additional content for IPA and APP dynamic instrumenting.

### Updated MakeAppTouchTestable Help

This release features greatly improved command line help for the MakeAppTouchTestable utility. The complete reference now includes all of the following additional content for IPA, APP, and APK dynamic instrumenting.

### Usage:

```
sh MakeAppTouchTestable/bin/MakeAppTouchTestable -project PATH [OPTIONS]
```

### Required parameters:

<code>project &lt;projectpath&gt;</code>	Path of the Xcode project directory (e.g. ~/Documents/MyApp/MyApp.xcodeproj)
<code>target &lt;name&gt;</code>	the name of the Xcode target to modify
<code>url &lt;url&gt;</code>	the CloudTest URL (e.g. http://ctserver/concerto)
<code>username &lt;username&gt;</code>	The CloudTest user name
<code>password &lt;password&gt;</code>	the CloudTest password

### Example:

```
sh MakeAppTouchTestable/bin/MakeAppTouchTestable -project ~/Documents/MyApp/MyApp.xcodeproj -target MyApp -url http://ctserver/concerto -username bob -password pass
```

### IPA required parameters:

Choose only one of the following arguments: `-project`, `-ipa`, or `-appfile`.

**Note:** IPA and APP instrumenting is unavailable for iOS 5 apps.

<code>ipa &lt;ipafilepath&gt;</code>	Path of the iOS IPA file (e.g. ~/Documents/MyApp.ipa)
<code>appfile &lt;appfilepath&gt;</code>	Path of the iOS app file (e.g. ~/Documents/MyApp.app).

### IPA optional parameters:

The following parameters are optional for IPA instrumenting (but may be required in some cases):

<code>-signingidentity &lt;signingidentityname&gt;</code>	The name of the signing identity to be used for codesigning the application (e.g. "iOS Distribution: Developer Name").
---	--

<pre>-provisioningprofile &lt;profilepath&gt;</pre>	Path of the Provisioning profile to be used for building IPA file.
<pre>-entitlementsfile &lt;entitlementsfilepath&gt;</pre>	Path of the entitlements file to be used for codesigning the application  <b>Note:</b> If a instrumented app produces the following error:  This crash is related to instrumenting : client has neither application-identifier nor keychain-access-groups entitlements  Then, the -entitlementsfile parameter (above) must be used to re-wrap the given app.

**IPA Example:**

```
sh MakeAppTouchTestable/bin/MakeAppTouchTestable -ipa ~/Documents/MyApp.ipa -url http://ctserver/concerto -username bob -password pass
```

### APK required parameters:

Choose either of the following arguments: `-project` or `-apk`.

<code>-apk &lt;apkfilepath&gt;</code>	Path of the Android APK file (e.g. ~/Documents/MyApp.apk)
<code>-androidsdk &lt;androidsdkpath&gt;</code>	Path of the directory where Android SDK is located.

### APK optional parameters:

The following parameters are optional for APK instrumenting (but may be required in some cases):

<code>-keystore &lt;keystorepath&gt;</code>	Path of the keystore to be used to sign the APK file.
<code>-storepass &lt;keystorepassword&gt;</code>	Password of the keystore to be used to sign the APK file.
<code>-keypass &lt;privatekeypassword&gt;</code>	Password of the private key (if different than the keystore password) to be used to sign the APK file.
<code>-alias &lt;aliasname&gt;</code>	Alias for the key to be used to sign the APK key. Only the first 8 characters of the alias are used.
<code>-donotconfiguremanifest</code>	Skip the Android manifest configuration step. Before using this option, please make sure that you have configured the AndroidManifest.xml file to insert changes required by TouchTest before building the APK file.

### APK Example:

```
sh MakeAppTouchTestable/bin/MakeAppTouchTestable -apk ~/Documents/MyApp.apk -androidsdk ~/Documents/android-sdk -url http://ctserver/concerto -username bob -password pass
```

## Other MATT Optional parameters:

-tenant <tenant>	The CloudTest tenant the user is associated with
-infoplistfile	The project's Info.plist file. If this parameter is not included, then MakeAppTouchTestable will try to automatically locate it.
-customrulesfile	The project's custom_rules.xml file. If this parameter is not included, then MakeAppTouchTestable will try to automatically locate it.
-launchurl	The URL that TouchTest should use to start the app. Example:  my-app://launch  If this parameter is not included, then MakeAppTouchTestable will auto-generate a URL.  <b>IMPORTANT:</b> Do not use spaces or underscores in your custom launch URL(s).
-appobjectname	Name of the Mobile App object created in the CloudTest server
-donotcreateapp	Don't create a Mobile App object in the CloudTest server
-universalapp	Create a Mobile App object compatible with both iOS and Android in the CloudTest server
-overwriteapp	If the Mobile App object already exists, replace it
-previewmode	Run in Preview mode to examine the console output for these parameters.  If this parameter is not included, then MakeAppTouchTestable runs in full mode and the project is changed.
-proxyserver <server>	HTTP proxy server name
-proxyport <port>	HTTP proxy port number
-proxyusername <username>	HTTP proxy user name (if any)
-proxypassword <password>	HTTP proxy password (if any)
-version	Print Utility Version
-reporterrors	Utility will automatically report errors to SOASTA along with any relevant files.
-donotreporterrors	Utility will not report any errors to SOASTA (Fogbugz).

<code>-addheadersearchpath</code>	Add TouchTest header files to "Header Search Paths" build setting.
<code>-useforceloadlinkerflag</code>	Add <code>-force_load</code> flag to the "Other Linker Flags" build setting.
<code>-removelibraryfrombuildphase</code>	If the user does not want the library to be added to the "Link Binary With Libraries" step of Build Phases, this argument will prevent that.



SOASTA, Inc.  
444 Castro St.  
Mountain View, CA 94041  
866.344.8766  
<http://www.soasta.com>