



# TouchTest™ iOS Tutorial



---

SOASTA TouchTest™ iOS Tutorial

©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>Why Mobile App Testing?</b> .....	<b>1</b>
<b>CloudTest® Basics</b> .....	<b>1</b>
Recording and Refining a TouchTest .....	2
<b>Initial Project Setup</b> .....	<b>4</b>
<b>Using the MakeAppTouchTestable Utility (Developer Only)</b> .....	<b>8</b>
<b>Static vs. Dynamic Instrumentation</b> .....	<b>8</b>
Applying MATT to an IPA or APP (Dynamic Instrumentation) .....	8
Making an APP file TouchTestable.....	9
Making an IPA file TouchTestable .....	11
Applying MATT to an Xcode Project (Static Instrumentation).....	12
<b>Inspecting the Mobile App in CloudTest® (Static/Dynamic)</b> .....	<b>15</b>
<b>Deploy the TouchTestable App</b> .....	<b>17</b>
Install from Xcode (Static Instrumentation).....	17
Deploy from the Command Line (Dynamic Instrumentation).....	18
<b>Install TouchTest Agent &amp; Register Device to Use TouchTest</b> .....	<b>19</b>
Associating Mobile Apps with a Device .....	23
<b>System Alerts for New URLs (iOS 9 and later)</b> .....	<b>24</b>
<b>Using TouchTest with iOS 10</b> .....	<b>25</b>
Setting Team Profile Name .....	25
<b>Recording a TouchTest Scenario</b> .....	<b>27</b>

---

<b>Create a TouchTest™ Clip</b> .....	<b>27</b>
Pause Recording .....	36
Adding an Interval Delay between Each Action .....	37
<b>Create a Composition</b> .....	<b>38</b>
<b>Playing a Composition</b> .....	<b>39</b>
<b>Result Details</b> .....	<b>40</b>
<b>Advanced Clip Editing</b> .....	<b>43</b>
Inspecting App Action Details .....	44
App Action Properties .....	45
Adding a Text Validation.....	48
Adding verifyElementPresent on an App Action.....	50
Adding an Output.....	51
<b>Identifying and Analyzing Common Errors</b> .....	<b>53</b>
Network or Communication Errors.....	53
App Action and Other Errors .....	54
Correcting Validation Errors using Touch Locator .....	55
<b>Support for Accessibility Locators</b> .....	<b>58</b>
Accessibility Identifiers .....	58
Accessibility Labels.....	58
<b>Analyzing Results</b> .....	<b>59</b>
<b>Appendix I: Using TouchTestIDs in Your Project Source Code</b> .....	<b>I</b>

---

---

**Adding TouchTest™ IDs to An iOS App ..... I**

## Why Mobile App Testing?

CloudTest<sup>®</sup> Mobile, featuring TouchTest<sup>™</sup> technology, delivers for the first time, complete functional test automation for continuous multi-touch, gesture-based mobile applications. TouchTest<sup>™</sup> technology delivers fast, precision functional testing while increasing the stability of automated tests across releases.

TouchTest controls mobile devices through a lightweight web service called TouchTest<sup>™</sup> Agent. Devices can be dedicated to testing in the lab, used as part of an external test, or crowd-sourced as part of a high volume, globally distributed test.

TouchTest support is provided for recording user actions within any iOS 5.0 or greater device including iPhone, iPad, and iPod Touch. There is no need to jailbreak the iOS device and the device can be untethered.

### CloudTest<sup>®</sup> Basics

SOASTA CloudTest<sup>®</sup> provides fast, effective performance, load and functional test automation of any modern Web application, Web service, or mobile application in a lab, staging or production environment using unique visual programming and multi-track user interfaces. The CloudTest platform can utilize both public and private cloud resources to assure any web or mobile application won't fail under peak user traffic.

Central is the primary user interface in which CloudTest<sup>®</sup> runs, and contains CloudTest's primary features, organized by sections.

The **Composition** is the test itself as presented in the **Composition Editor**, and contains one or more **Clips** arranged on **Tracks** and governed by user-specified sequence and tempo. The Composition Editor is a player, debugger, and the dashboard where results are analyzed.

The **Clip** is the basic building block of a test as built in the **Clip Editor** and has a **Target** such as a web site, or in the case of TouchTest<sup>™</sup>, a mobile app. A clip can be thought of as a visual script that is composed of a series of timed or sequenced events, which correspond to gestures performed on the mobile device. It can contain

messages, browser or app actions, and scripts, as well as delays and checkpoints—all of which can be organized into containers (i.e. groups, chains, transactions, etc.)—and parameterized as required.

TouchTest clips are recorded directly from the mobile app and added to the Clip as you perform them on the mobile device.

### Recording and Refining a TouchTest

TouchTest™ records the details of actual gestures and events that iOS invokes on the app that is tested. These gestures and events are represented within the **Clip Editor** as App Actions. Precision recording captures and plays back all continuous touch gestures including pan, pinch, zoom and scroll.

Each gesture you perform on a TouchTest-enabled device is precisely, and automatically, added to the test clip as an App Action.

Like any clip element within CloudTest®, App Actions have inputs and outputs, as well as properties, waits, and validations that can be parameterized. Additionally, an App Action can be added to any containers (e.g. transactions, groups, etc.).

As a general guideline, your test should account for all the factors of the mobile app(s) you want to test, and include one, or, as many viable test cases as it will take to arrive at a good mobile app test case.

Once a test case is recorded a test designer can move quickly to validating captured app actions and other test design issues. Test design also takes into account the following factors:

- **The types of app actions to perform**

The test designer will consider the types of app actions that make up a test case for the given mobile app. These app actions should then be performed during recording.



- **The timing of app actions**

In addition to TouchTest's built-in detection of the duration of gestures, CloudTest® provides an additional set of Waits, which allow the tester to gain control of the pace within a test.

- **The validation of tests**

Verifying the behavior of a mobile app is another important step in successful testing. After each app action is recorded, the test designer can add as many validations as needed by picking from among built-in Verify commands.

- **The number of devices and their locations**

Typically, a single test clip defines a single test case that can be run on multiple tracks or devices. However, tests of great complexity can be quickly devised by introducing multiple test cases, multiple devices/repeats—in tandem with geographic location. Complex mobile app tests can be easily built utilizing one or all of these capabilities.

## Initial Project Setup

**Note:** For iOS, Objective C and Swift are compatible with TouchTest and no special actions must be taken.

In this section we will use git to retrieve the project, download the MakeAppTouchTestable utility (for use in the next section), and examine the Stockfish project by opening it in Xcode.

The Stockfish sample project is used throughout this tutorial. You can easily follow along applying all of the steps to your own project or app. In this section we will use git (or GitHub) to retrieve the source code project, after which we'll also open and examine the project as is—before we make any modifications to it.

1. Using git (or GitHub) retrieve the following project:

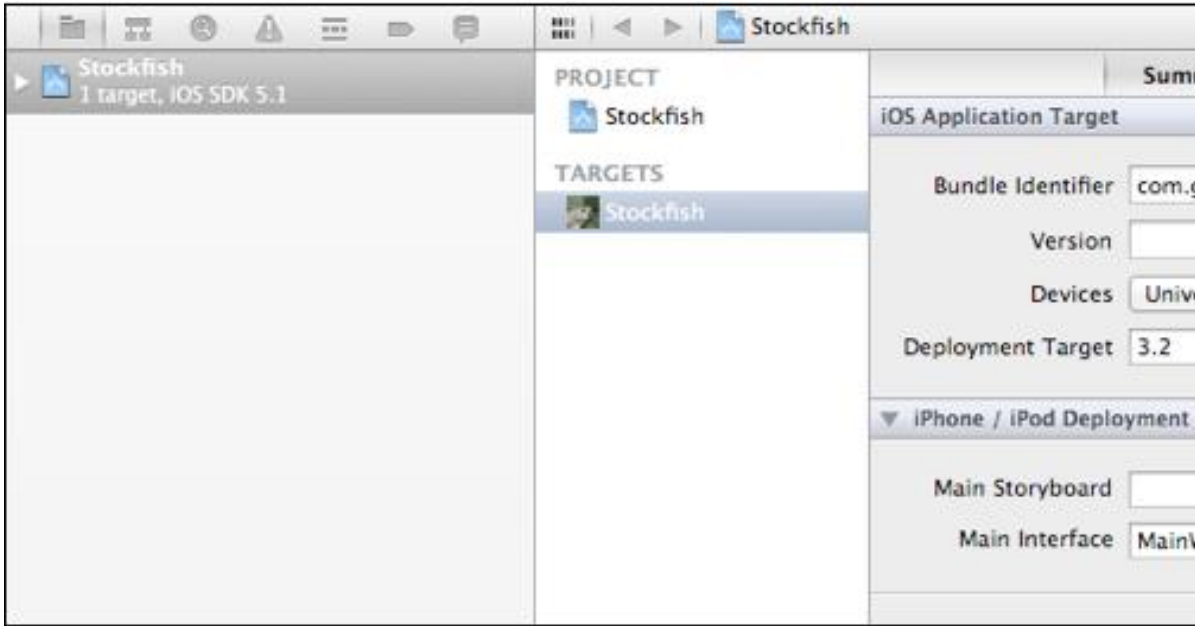
```
git clone https://github.com/elitecoder/stockfishchess-ios
```

**Note:** The git command must be installed on the given node and in the path to succeed.

2. Locate the Stockfish.xcodeproj file in Finder and double-click to open it.

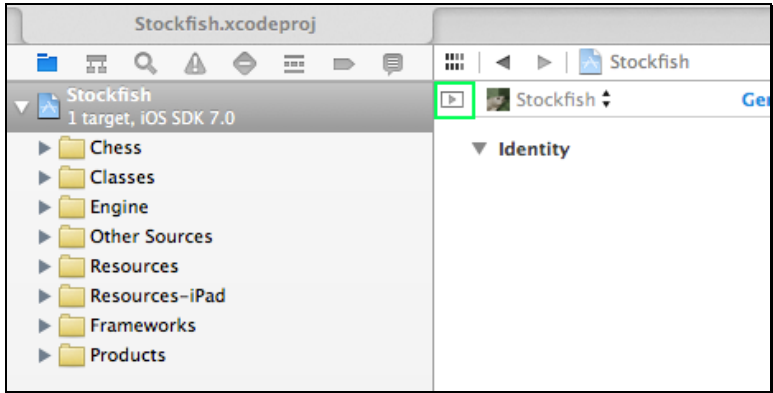
**TIP:** You can also open Xcode and then use File > Open... to select the Stockfish.xcodeproj.

The project's components prior to running the utility are shown below.



Note that out-of-the-box there is one target, *Stockfish*, defined in this project. We will now duplicate that target for use with TouchTest.

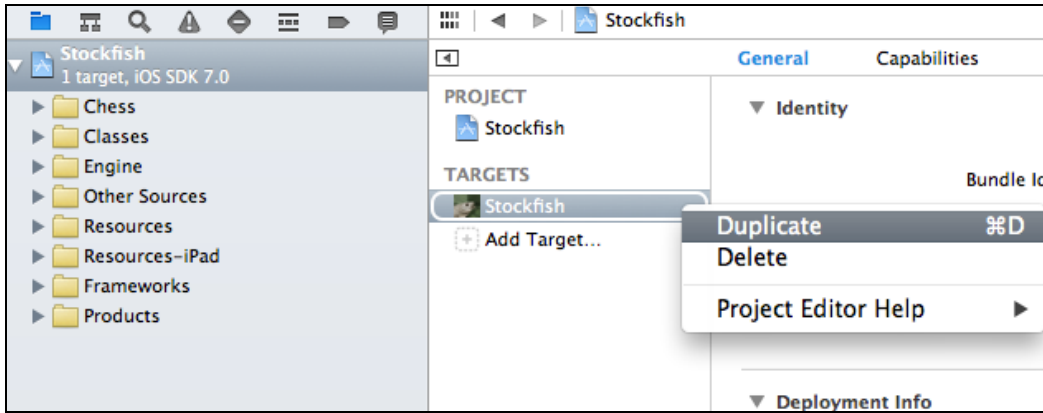
1. Expand the panel to reveal the underlying target "Stockfish."



Familiarize yourself with the Stockfish project before proceeding.

2. If you're using static instrumentation, right click the *Stockfish* target and choose "Duplicate".

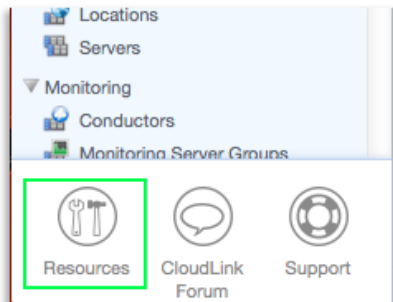
**Note:** The Duplicate target is used only with static instrumentation.



**SOASTA highly recommends that you create a new Xcode target for use with TouchTest for static instrumentation.** For now, create the duplicate target but note that if you use dynamic instrumentation (discussed in the next section) then you will not actually use the duplicate target. For static users, the target duplication technique allows you to easily build two versions of your app: a testable version that is linked with TouchTest Driver, and a production-ready version that does not include any TouchTest™ functionality.

Xcode will create a new target called *Stockfish copy*.

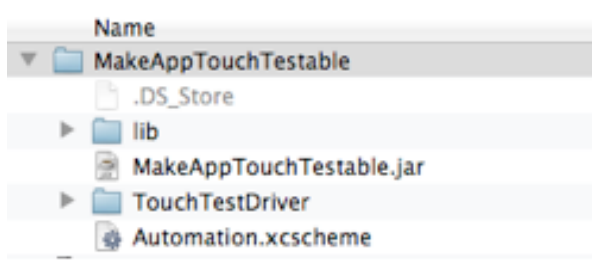
3. Login to your CloudTest instance and open the Resources page (found at the bottom of Central).



4. Download *MakeAppTouchTestable Utility* from the Downloads section.



5. Unarchive the ZIP file.



This archive contains the following:

**Note:** It is not necessary to open any of the files shown below, but it's good to know that they are there. Occasionally, you'll need to download a new MakeAppTouchTestable and this is where the updated, unarchived files will go.

- MakeAppTouchTestable.jar utility is the script that will make the necessary project modifications and create a mobile app in CloudTest®
- The TouchTestDriver folder. The contents of this folder will be automatically copied to your project

6. Open a Terminal window and navigate to the location where the ZIP was unarchived.

For example, `cd Downloads/MakeAppTouchTestable.`

Now that you've retrieved the project, and the MATT utility, and have become familiar with the sample project, the next thing is to decide whether to use static instrumentation or dynamic instrumentation to make the project TouchTestable.

## Using the MakeAppTouchable Utility (Developer Only)

TouchTest™ includes the MakeAppTouchable (MATT) utility, which will automatically add the necessary components to make an Xcode project file, app, or IPA "Touchable."

**Note:** For iOS, Objective C and Swift are compatible with TouchTest and no special actions must be taken.

### Static vs. Dynamic Instrumentation

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

- Dynamic instrumentation occurs when MATT instruments an IPA or APP bundle 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. Stockfish.xcodeproj).

Static instrumentation is available in all TouchTest releases and for all supported iOS versions.

### Applying MATT to an IPA or APP (Dynamic Instrumentation)

Before you can use dynamic instrumentation, you must have either an APP or IPA file with which to work. If you are not the developer, discuss where to get the required file prior to attempting the following steps.

**TIP:** Refer to MakeAppTouchable help (via the Command line using `java -jar MakeAppTouchable.jar -help`) for more information.

MATT users can dynamically instrument a compiled iOS file by specifying one of the two following flags:

- `-ipa` – The full path to the compiled IPA file
- `-appbundle` – The full path to the compiled APP file

**TIP:** Provide the additional MATT parameters (listed below) whenever they are required by your app. If you're not the developer, discuss the optional parameters with your developer.

### ***Optional parameters for IPA/APP***

The following optional parameters are also available for dynamic instrumentation of IPA and APP files (in some cases their use may be necessary to succeed).

<code>-signingidentity &lt;signingidentityname&gt;</code>	<p>The name of the signing identity to be used for codesigning the application (e.g. "iOS Distribution: Developer Name").</p> <p><b>IMPORTANT: MATT can only sign your app with a Distribution profile. This can be an Ad-Hoc Distribution or an Enterprise Distribution profile.</b></p>
<code>-provisioningprofile &lt;profilepath&gt;</code>	<p>Path of the Provisioning profile to be used for building IPA file.</p>
<code>-entitlementsfile &lt;entitlementsfilepath&gt;</code>	<p>Path of the entitlements file to be used for codesigning the application. In the vast majority of cases, the entitlements file is not required.</p>

### **Making an APP file TouchTestable**

The dynamic instrumentation method for APP uses the MATT parameter `-appbundle`.

1. To dynamically instrument an IPA using the MakeAppTouchTestable utility, run:

```
sh MakeAppTouchTestable/bin/MakeAppTouchTestable -appbundle <compiled APP> -url <CloudTest URL> -username <CloudTest user name> -password <CloudTest password>
```

where:

- `<compiled APP>` is the full path to the compiled iOS IPA file

Here is a complete example using `-appbundle`:

```
sh MakeAppTouchTestable/bin/MakeAppTouchTestable -appbundle
"~/Documents/Demo/Stockfish/ stockfishchess-ios/build/Release-
iphonesimulator/Stockfish.app" Stockfish.app -target "Stockfish" -url
http://10.0.1.44/concerto -username bob@acme.com -password secret
```

`MakeAppTouchTestable` will configure your app, and create a new Mobile App object in the CloudTest repository. The Mobile App object created will have the auto-created URL Scheme in its Launch URL field.

**TIP:** In this basic example, we do not use MATT's `launchurl` flag to create a launch URL. In which case, `MakeAppTouchTestable` will auto-generate the URL for us. If the flag is used, be sure to avoid spaces and underscores as they will cause an error.

You will see a message similar to the following:

```
Mobile App Object representing your Application "Stockfish" has been created in
CloudTest Repository.
```

Optionally, if you'd like to overwrite an existing mobile app repository object, use the `-overwriteapp` command in the above.



## Making an IPA file TouchTestable

The dynamic instrumentation method for IPA uses the MATT parameter `-ipa` and requires the `signingidentity` and `provisioningprofile` flags. Omitting these flags will result in an `.app` file being created rather than the expected `.ipa`.

1. To instrument an IPA using the `MakeAppTouchTestable` utility, run:

```
sh MakeAppTouchTestable/bin/MakeAppTouchTestable -ipa <compiled IPA> -url  
<CloudTest URL> -username <CloudTest user name> -password <CloudTest  
password> -signingidentity "iPhone Developer: <Developer Name>" -  
provisioningprofile "/Users/<username>/Documents/Name.mobileprovision"
```

where:

- `<compiled IPA>` is the full path to the compiled iOS `.ipa` file
- `<iPhone Developer: <Developer Name>` must be a valid iOS Developer signing identity
- The provisioning profile file path must refer to a valid provisioning profile

**Note:** One and only one of the project file, IPA, or app file can be specified when using the MATT utility.

where:

- `<compiled IPA >` is the full path to the compiled iOS `.ipa` file

Here is a complete example using `-ipa`:

```
sh MakeAppTouchTestable/bin/MakeAppTouchTestable -ipa  
"/Users/jgardner/Documents/Demo/stockfishchess-ios/build/Release-  
iphonesimulator/Stockfish.ipa -target Stockfish -url  
http://10.0.1.44/concerto -username bob@acme.com -password secret
```

`MakeAppTouchTestable` will configure your app, and create a new Mobile App object in the CloudTest repository.

```
Copying over TouchTestDriver library.
```

```
Injecting load command for TouchTestDriver library...
```

```
Load command successfully injected.
```

```
TouchTest enabled app is now available at:
```

```
/Users/jgardner/Documents/Demo/stockfishchess-ios/build/Release-  
iphonesimulator/Stockfish_TouchTest.ipa
```

The Mobile App object created will have the auto-created URL Scheme in its Launch URL field.

**TIP:** In this basic example, we do not use MATT's `launchurl` flag to create a launch URL. In which case, `MakeAppTouchTestable` will auto-generate the URL for us. If the flag is used, be sure to avoid spaces and underscores as they will cause an error.

If it's the first time creating this object, you will see a message similar to the following:

```
Mobile App Object representing your Application "Stockfish" has been created in
CloudTest Repository.
```

If the mobile app already exists in the repository, you will instead see:

```
Already Have A Mobile App Object By This Name. Please Change The Name And Try
Again.
```

```
You should manually create an object corresponding to your Application using
Application ID: touchtest-e8beec35-99bf-44b7-8ede-cbb4ba46485f://
```

Whenever you'd like to overwrite an existing mobile app repository object, use the `-overwriteapp` command in the above. Take note that doing so may affect the work of other users.

## Applying MATT to an Xcode Project (Static Instrumentation)

The static method will use the MATT parameter `-project`.

1. To instrument an Xcode project using the `MakeAppTouchTestable` folder, run:

```
sh MakeAppTouchTestable/bin/MakeAppTouchTestable -project <Xcode project
directory> -target <target name> -url <CloudTest URL> -username
<CloudTest user name> -password <CloudTest password>
```

where:

- `<Xcode project file>` is the full path including file name to the ".xcodeproj" file representing your project
- `<target name>` is the name of the Xcode target you would like to modify

Here is a complete example:

```
sh MakeAppTouchable/bin/MakeAppTouchable -project
~/Downloads/Stockfish/Stockfish.xcodeproj -target "Stockfish copy" -url
http://<CloudTest URL>/concerto -username bob@acme.com -password secret
```

2. MakeAppTouchable will configure your project, and create a new Mobile App object in the CloudTest repository. The Mobile App object created will have the auto-created URL Scheme in its Launch URL field.

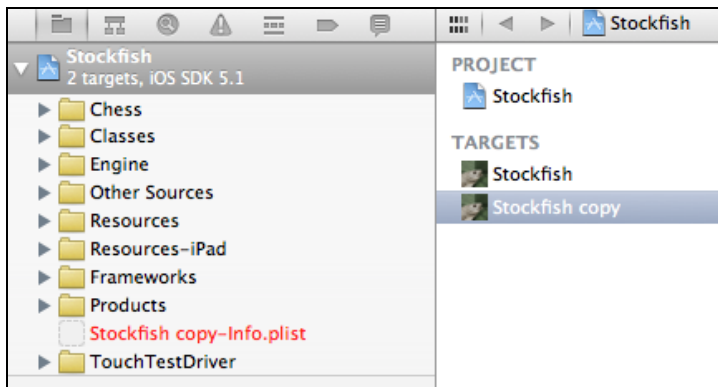
**TIP:** In this basic example, we do not use MATT's launchurl flag to create a launch URL. In which case, MakeAppTouchable will auto-generate the URL for us. If the flag is used, be sure to avoid spaces and underscores as they will cause an error.

You will see a message similar to the following:

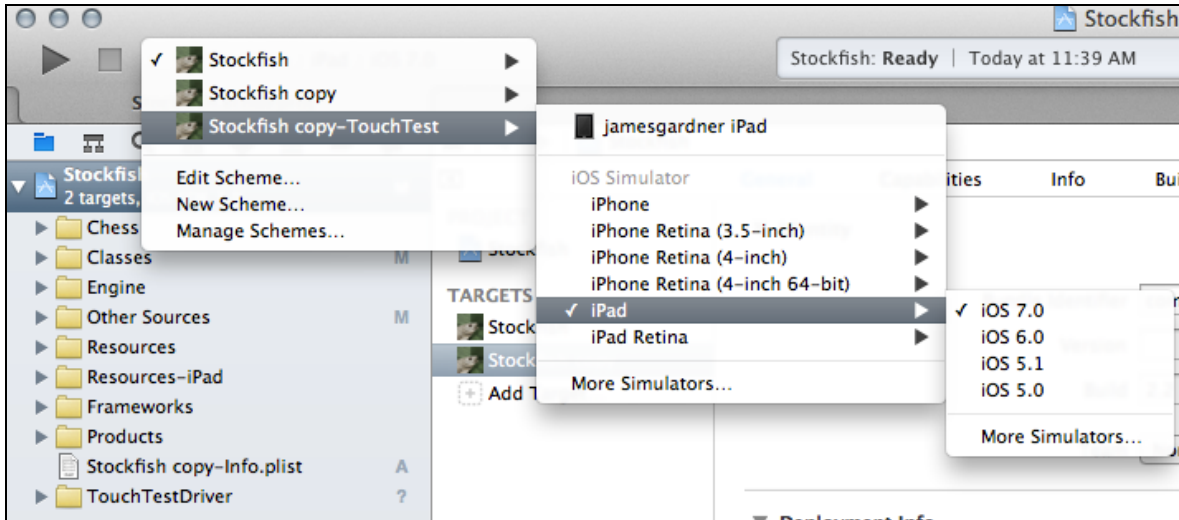
```
Mobile App Object representing your Application "Stockfish" has been created in
CloudTest Repository.
```

3. Optionally, if you'd like to overwrite an existing mobile app repository object, use the `-overwriteapp` command in the above.

Now that the specified Xcode project has been modified, let's take a look at it. In the screenshot below, note that a new project folder now exists for the TouchTest Driver.



In addition, click the Scheme drop-down in the Xcode toolbar note the new entry with the suffix "-TouchTest". Select Stockfish-copy-TouchTest for now and the device or simulator to use. These will be explained in more detail later in this document.

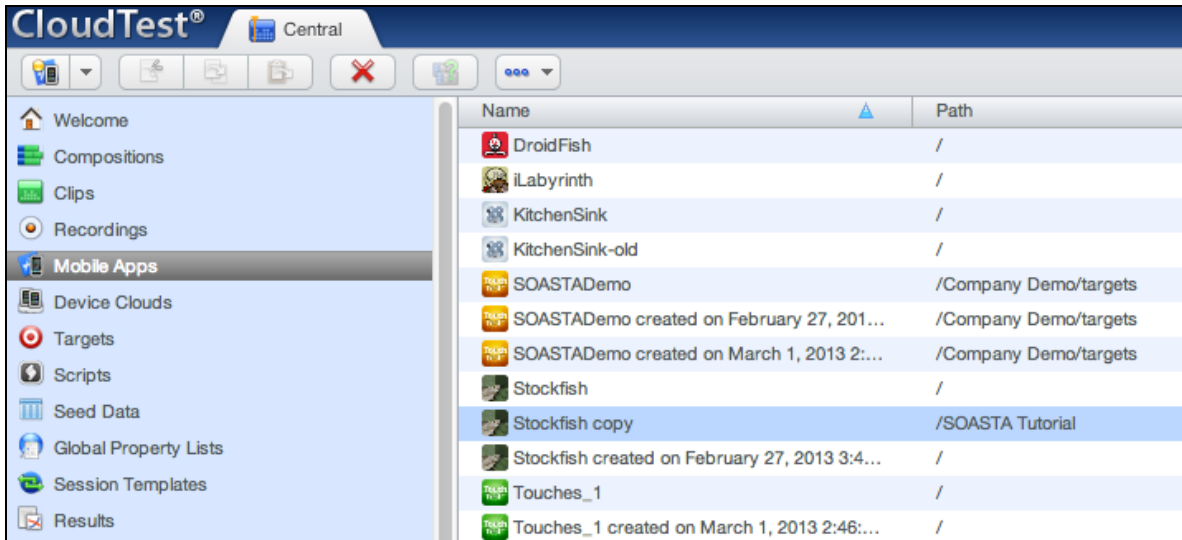


## Inspecting the Mobile App in CloudTest® (Static/Dynamic)

In the steps above at the end of each run of the MakeAppTouchable.jar we were notified that the “Mobile App Object” had been created in the CloudTest® Repository.

**TIP:** This mobile app will appear in the Choose Device Agent and Mobile App box whenever end-users start a mobile app recording. Selecting which mobile app to launch on which test device(s) is a crucial end-user step.

1. Optionally, verify that the Mobile App has been added by logging into CloudTest® and looking for its entry in the Central > Mobile Apps list. For example, in the screenshot below the *Stockfish copy* app appears as expected.

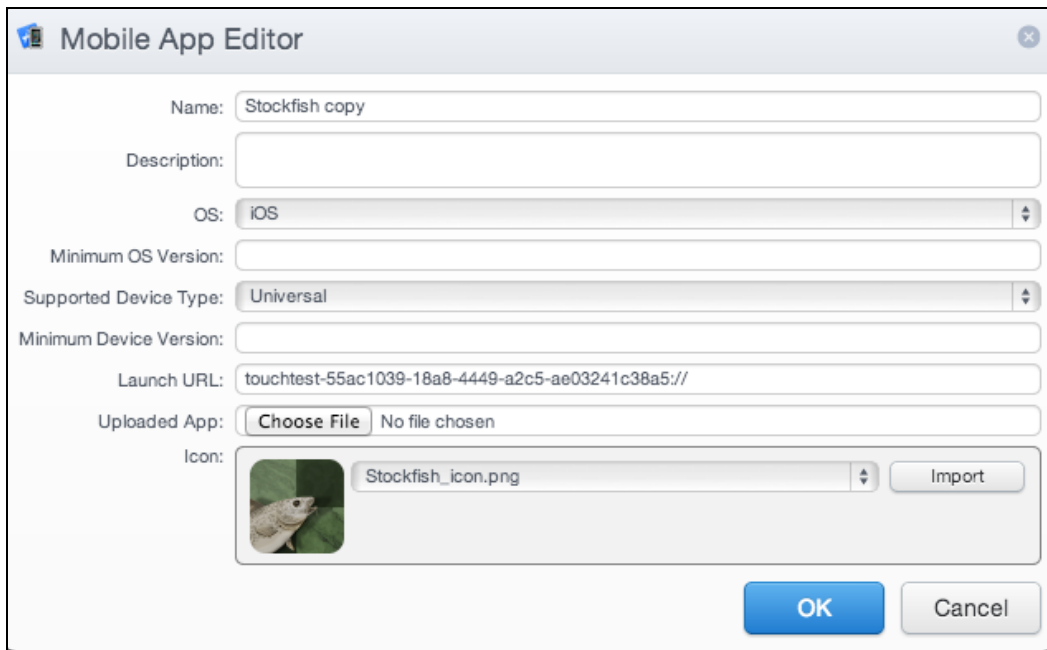


2. Double-click the Mobile App you created to inspect its details.

The Mobile App detail form appears.

- All of the fields shown were populated from the Xcode project, with the exception of Supported Device Type and Minimum OS Version.

- The default Supported Device Type is Universal (e.g. both iPhone and iPad). If desired, change it to be either iPhone- or iPad-specific.



The screenshot shows the 'Mobile App Editor' dialog box with the following fields and values:

- Name: Stockfish copy
- Description: (empty)
- OS: iOS
- Minimum OS Version: (empty)
- Supported Device Type: Universal
- Minimum Device Version: (empty)
- Launch URL: touchtest-55ac1039-18a8-4449-a2c5-ae03241c38a5://
- Uploaded App: Choose File No file chosen
- Icon: Stockfish\_icon.png (with a preview image of a stockfish)

Buttons: OK, Cancel

- The default Minimum OS Version supported in TouchTest™ is iOS 5.0.

## Deploy the TouchTestable App

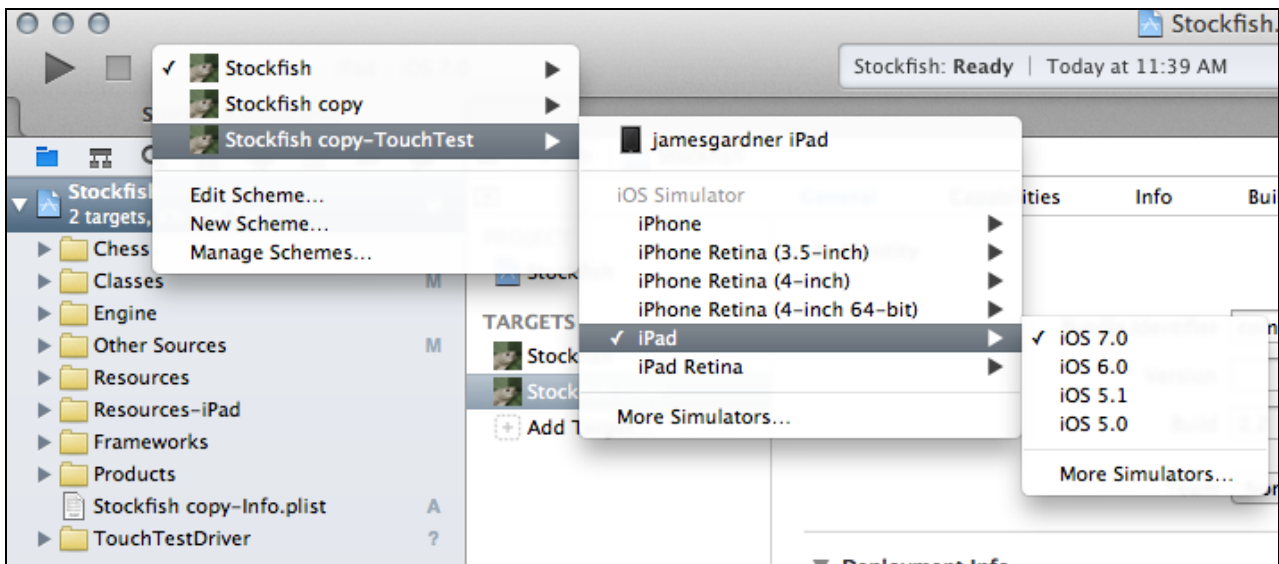
Choose either the Dynamic or Static method for making the mobile app TouchTestable.

### Install from Xcode (Static Instrumentation)

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.

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.



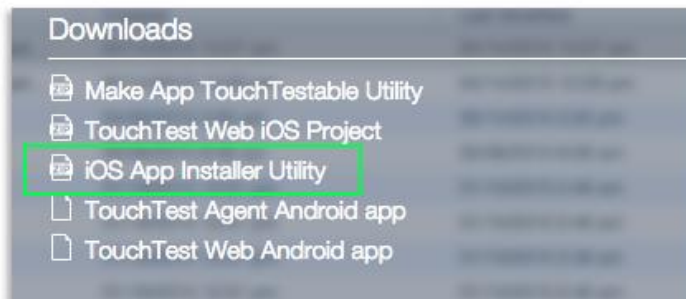
## Deploy from the Command Line (Dynamic Instrumentation)

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 > Resources](#) page.



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 --appbundle <path to IPA file>
```

- For an iPhone or iPad, use:

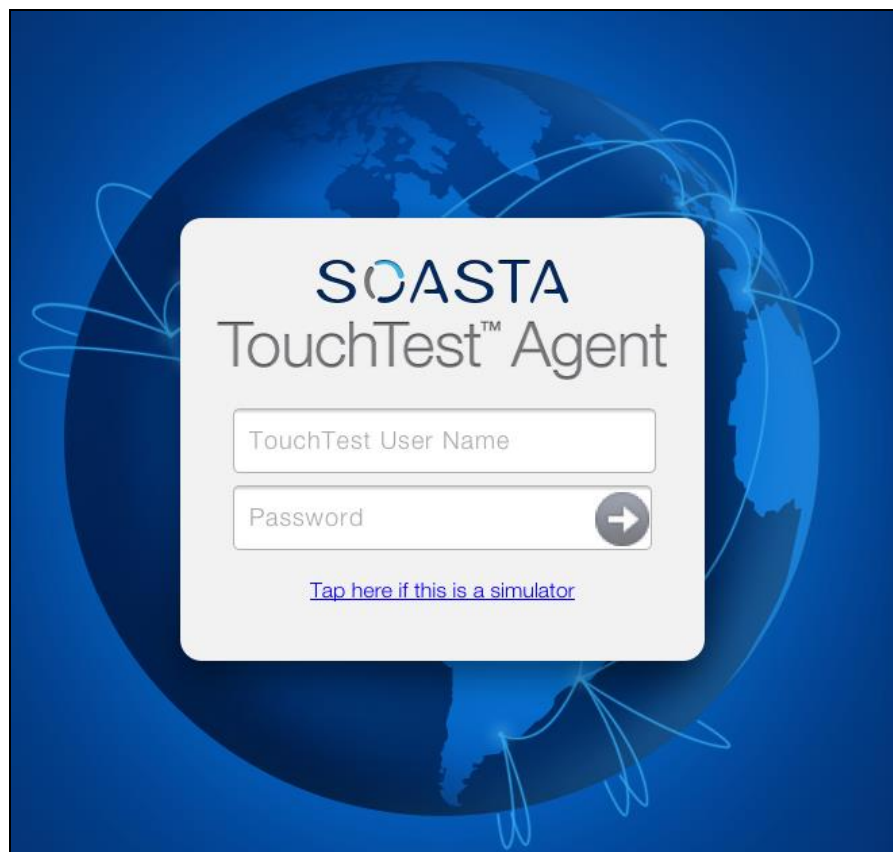
```
./bin/ios_app_installer --ipa <path to IPA file>
```



## Install TouchTest Agent & Register Device to Use TouchTest

The TouchTest™ Agent is responsible for launching the apps that are being tested on a given device. It is a web application that is runs in mobile Safari on iOS devices. To get started, browse to the TouchTest Agent URL below on the mobile device and then perform the one-time registration steps that will enable your device for use with TouchTest.

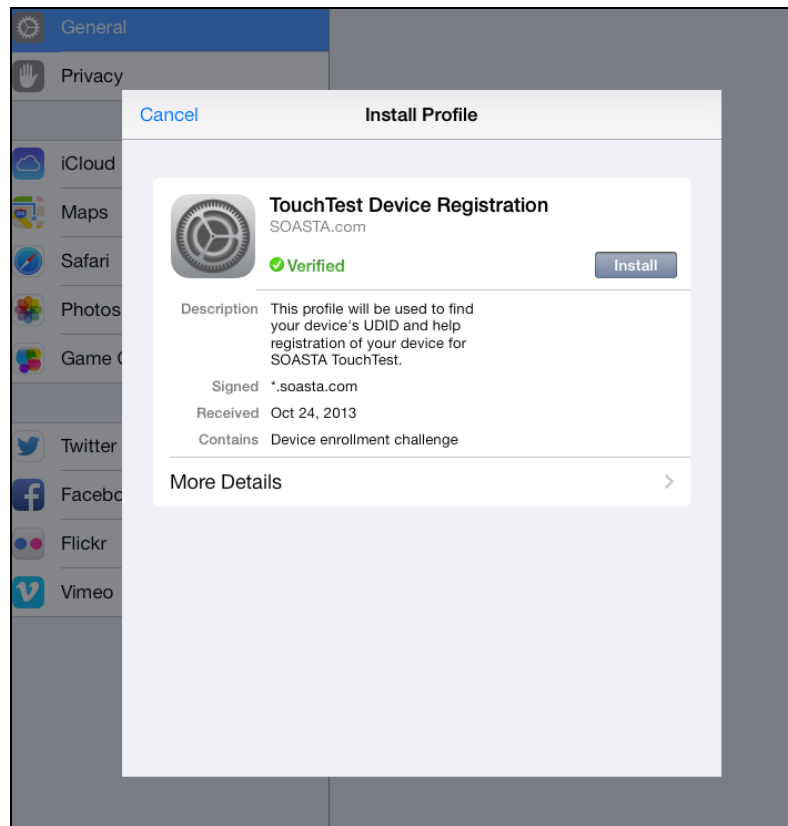
1. On the mobile device, enter the CloudTest URL with “/touchtest” appended. The complete URL form for TouchTest is http://<CloudTest URL>/concerto/touchtest/. If the device is not registered, it will prompt you to do so.



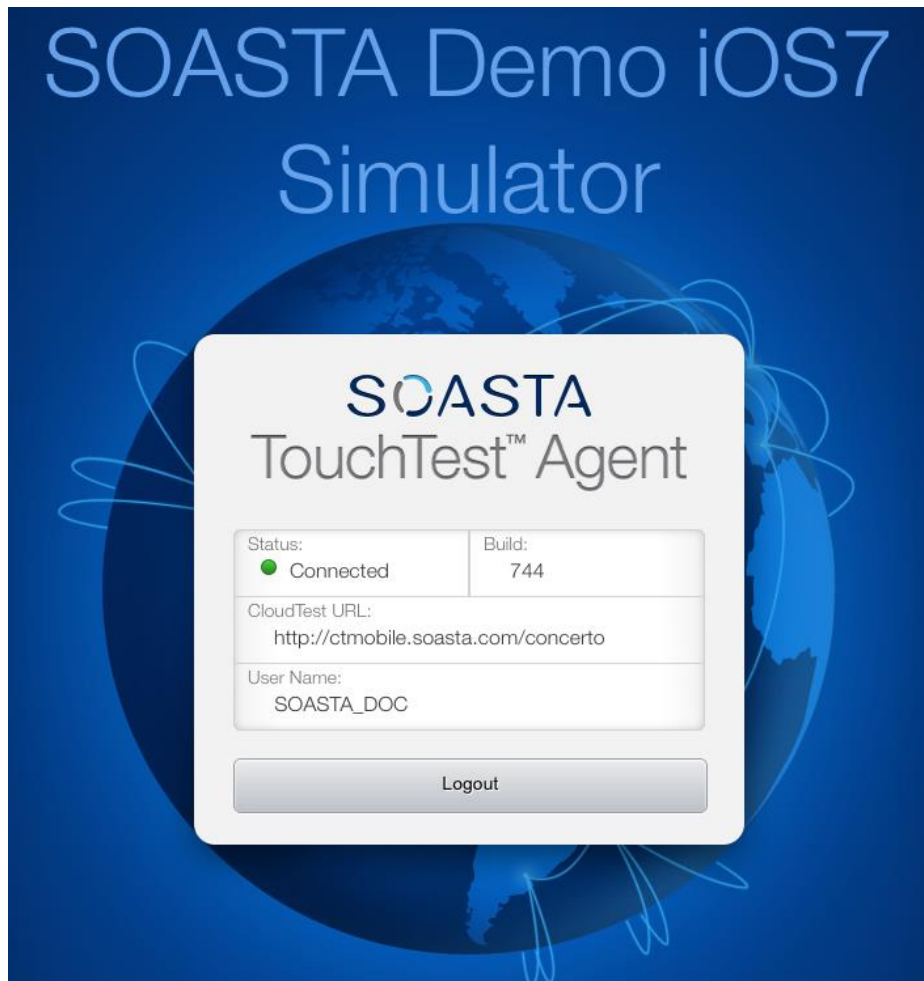
2. Log in using your SOASTA CloudTest User Name and Password. The Unique Device Identifier (UDID) will be used to register the mobile device for use with

TouchTest™ in iOS7 or after; for earlier iOS versions, the registration will also use a profile.

3. Click the Register Device button to continue (only applies to iOS 7 and earlier).
  - a. The Install Profile screen appears. Click the Install button to proceed.
  - b. The Install Profile alert appears to indicate that mobile device settings will be changed. Click Install Now to proceed.
  - c. If a passcode is in effect on the mobile device, an additional prompt will appear for you to authorize the profile installation.



4. When prompted, give the TouchTest Agent a name. For example *Tester iPad*. This name will be used throughout the product to refer to this device. Once entered, an Administrator is the only one who can change the device name.
5. The status on the device will change to *Connected*. If it is not *Connected*, refresh the page, and ensure that you are logged in.



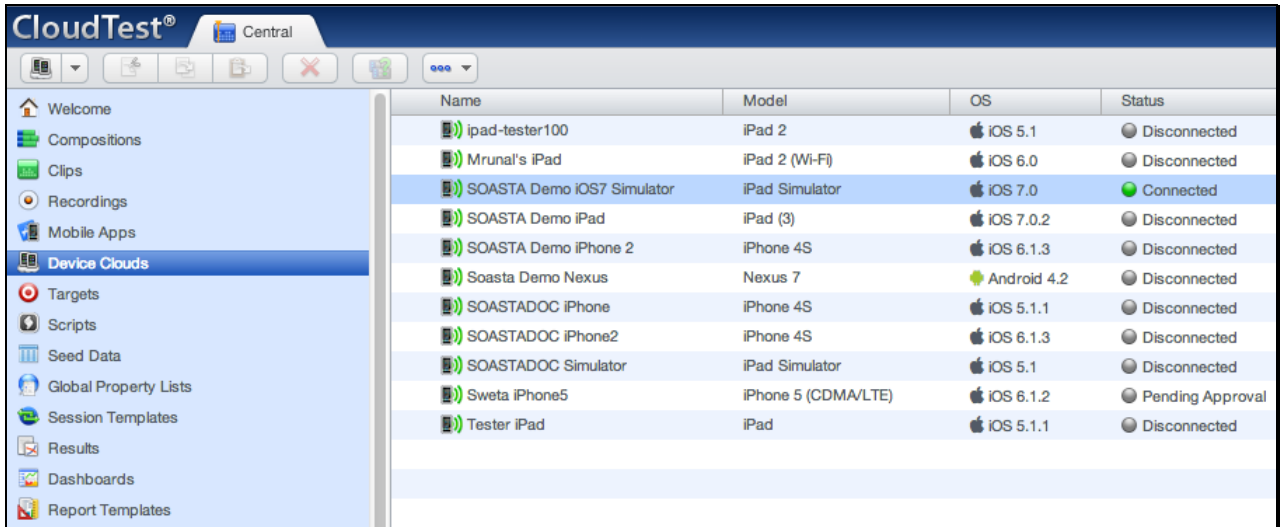
**Notes:** If you clear your cookies on the given mobile device after registration, you may need to register your device again so TouchTest can recognize it. This does not consume an additional license.

If you're using a Simulator prior to iOS 7, use the "Tap here if this is a simulator" link to proceed. This link will appear below the Login button in all configurations that require it. This link doesn't appear in iOS 7 or after.

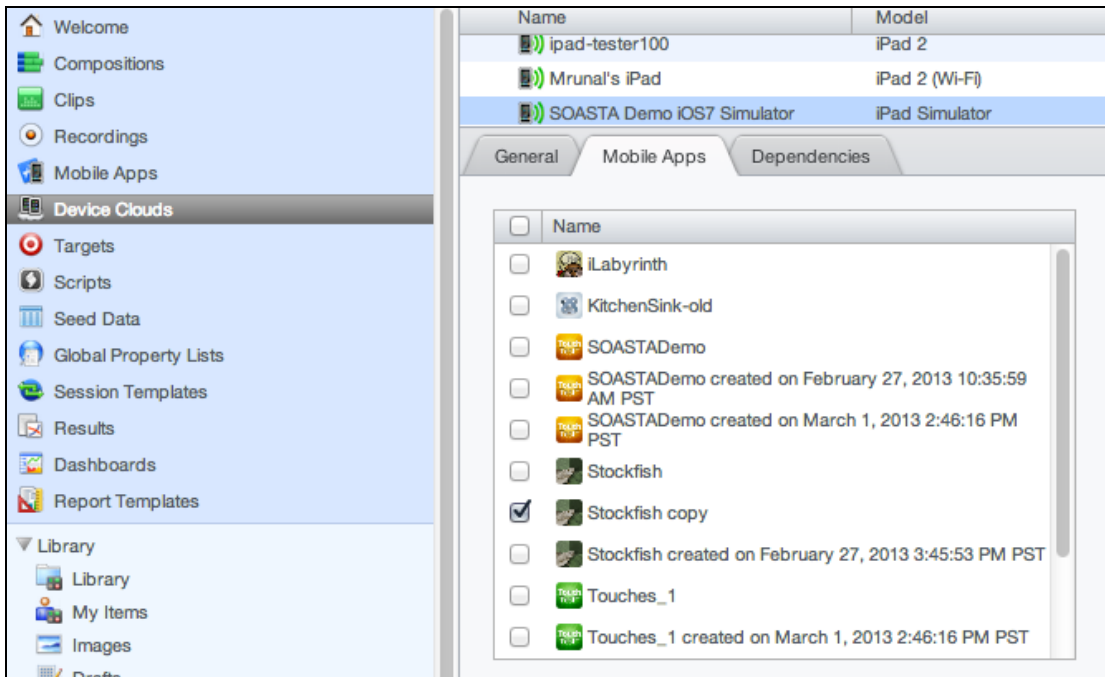
## Associating Mobile Apps with a Device

Once a device is approved, use the following steps to assign one or more mobile apps to that device.

1. In Central > Device Clouds, select the mobile device.



2. In the lower panel, click the Mobile Apps tab. If necessary, use the Maximize button to increase the workspace.

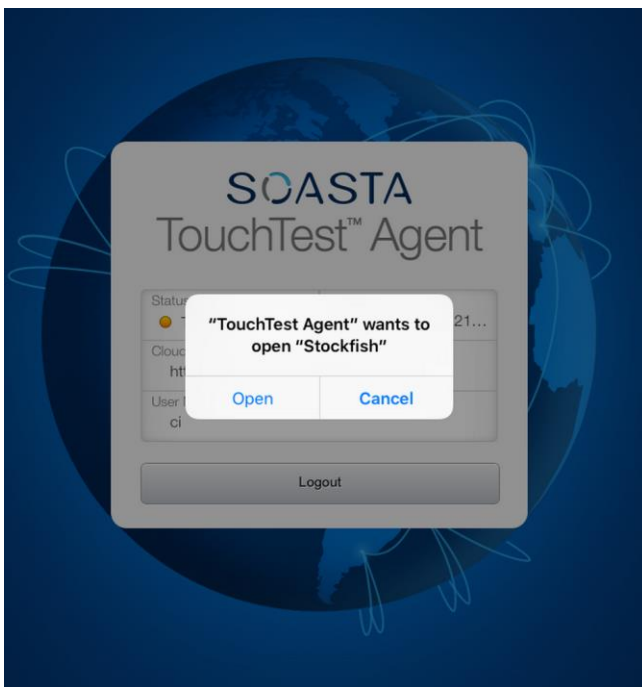


3. Locate and check the Mobile App(s) that you want to authorize this device to access. For example, Stockfish.
4. Click Save on the lower panel toolbar.

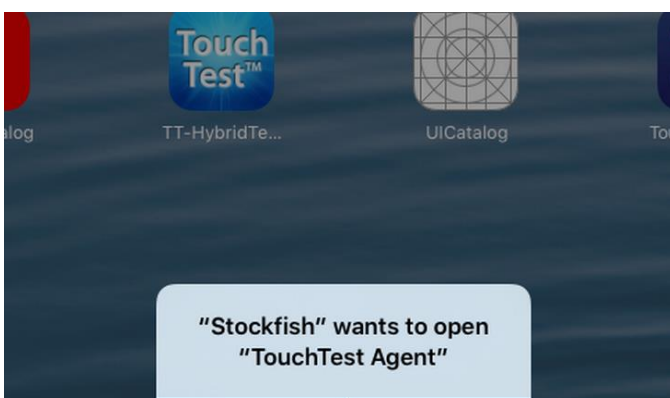
## System Alerts for New URLs (iOS 9 and later)

For iOS 9 and later, every time you attempt to launch a new URL scheme on your device for the first time, you must:

- a) Allow TouchTest Agent to open the new app by tapping **Open**.



- b) Allow the app you are launching to open TouchTest Agent by tapping **Open**.



## Using TouchTest with iOS 10

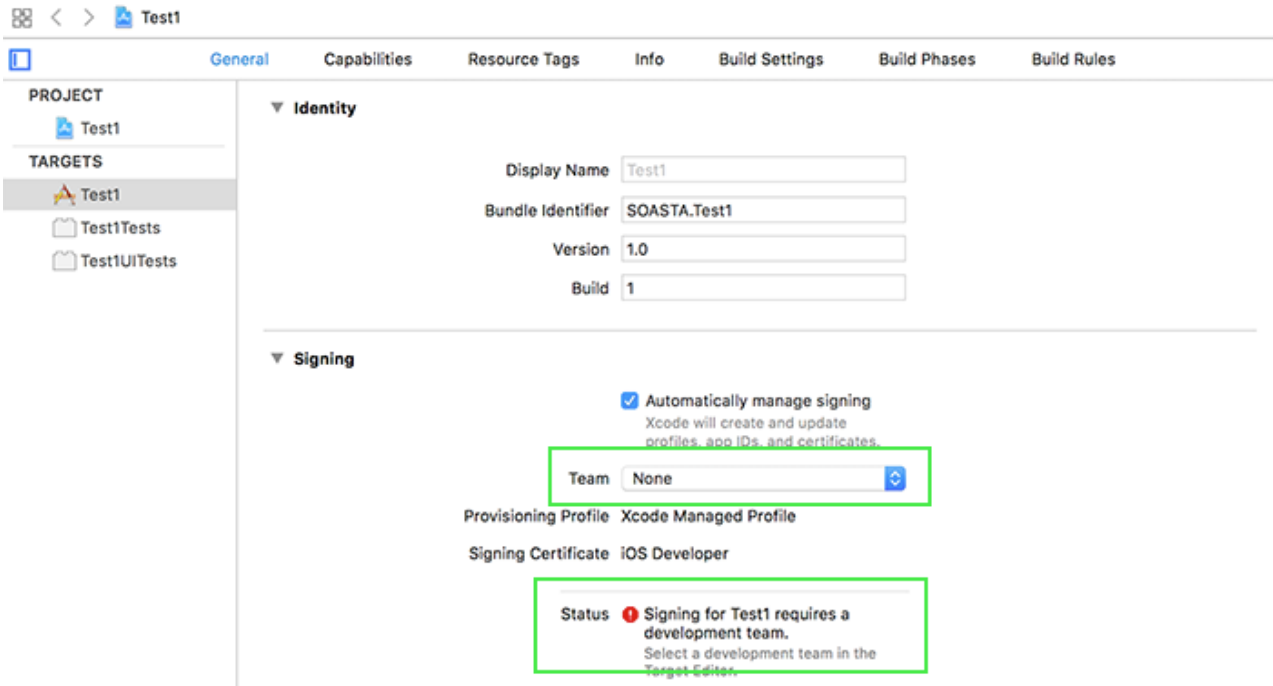
As of SOASTA 57.16, TouchTest supports iOS 10. Due to new restrictions brought by iOS 10, you must sign into Xcode with your development team profile name.

### Setting Team Profile Name

You must set your team profile name before instrumenting TouchTest using Xcode, an essential part of setting up TouchTest.

To set your team profile name:

1. Open any Xcode project.
2. Click the **General** tab.
3. In the Signing section, specify your Team name in the dropdown menu.



**Notes:** An error message will appear if you don't specify a team. If you click the error message, it will take you to the Team dropdown menu Xcode.



## Recording a TouchTest Scenario

Once the TouchTest Agent profile is installed and device access approved you are ready to record and playback your TouchTest.

1. Create a new test clip within CloudTest®
2. Click Record within the Clip Editor and then choose Mobile App Recording and specify the Device Agent and the mobile app whose actions you want to record.
3. Perform the app actions on the mobile device to capture them in the new test clip.

These and the following additional configuration steps are described in the remainder of this tutorial.

- Composing a TouchTest Clip
- Playing back a TouchTest Composition
- Analyzing TouchTest Composition's Results

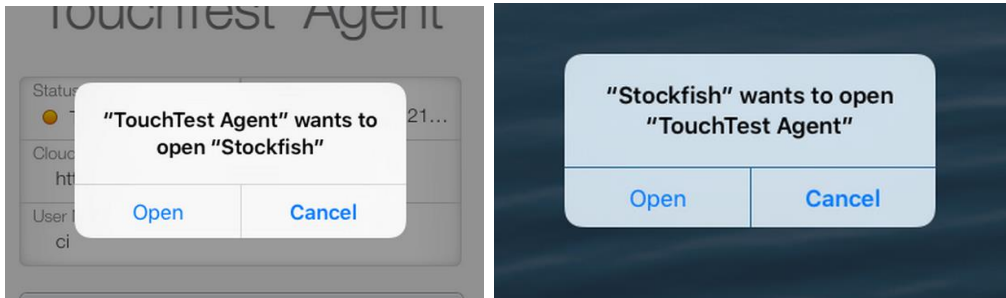
### Create a TouchTest™ Clip

Create a new clip that will be used to perform mobile app recording and serve as the basis for your TouchTest.

- Open the TouchTest Agent in Safari on your mobile device, and click Login (your previously entered username and password should be auto-populated).

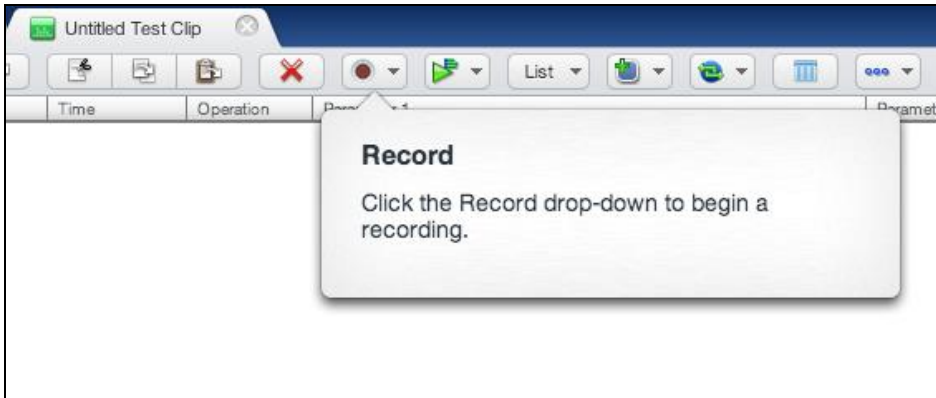
**Note:** Any time your device launches a new URL (for the first time only), you need to tap **Open** twice; first, to allow TouchTest Agent to open the new app and second, to allow the app to open TouchTest Agent.

These actions will resemble the pop-ups below:



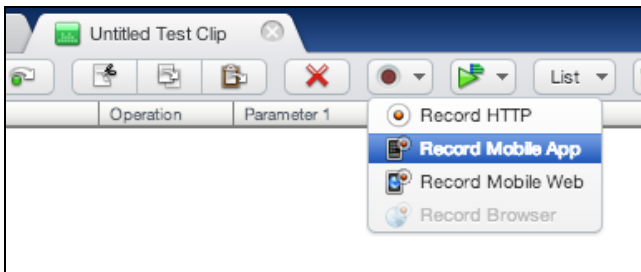
Once successfully logged on, the device Status will be *Connected*.

- Login to CloudTest on your desktop computer and select Central > Clips, and then click New on the Central toolbar.

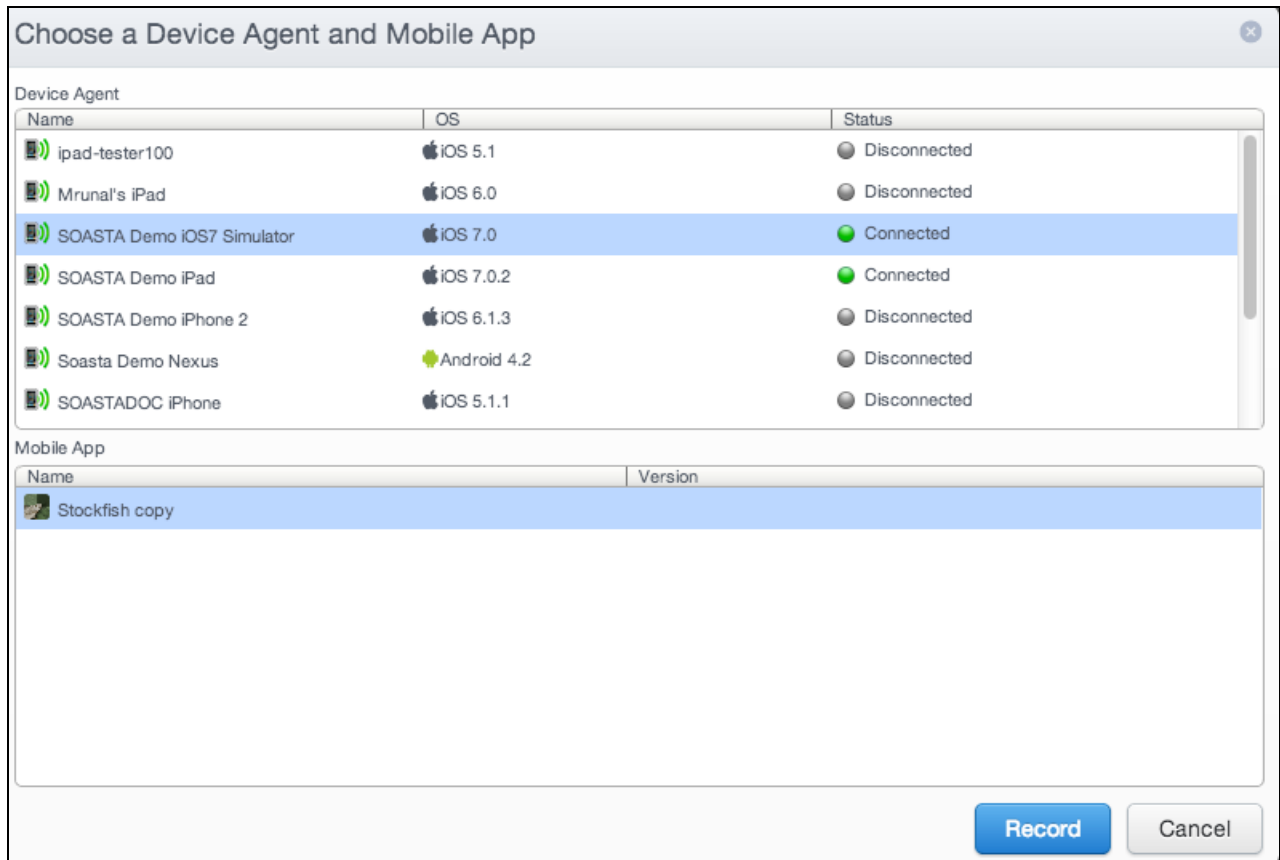


A new Untitled Test Clip opens in a Clip Editor tab. A Record pop-up identifies the Record drop-down.

- Once ready, click the Record button and then select Record Mobile App.

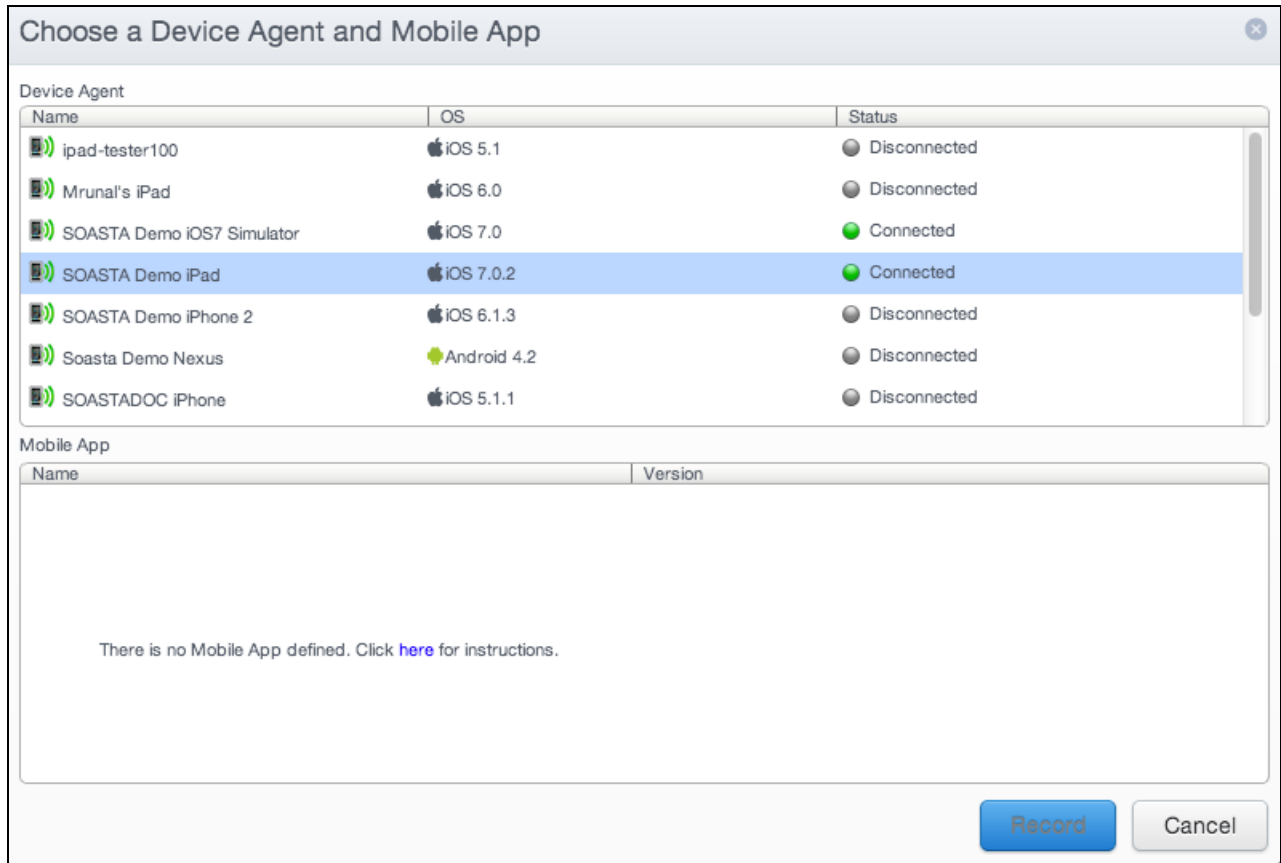


The Choose a Device Agent and Mobile App wizard appears.



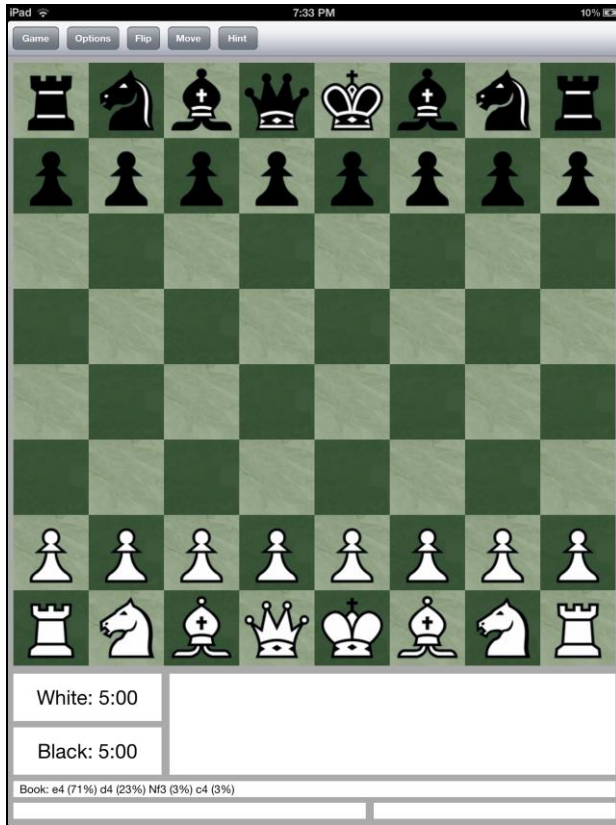
**Note:** If the Mobile Device Administrator has completed the steps above to associate one or more mobile apps with the device, those apps will appear in the Mobile App list whenever that device is selected. In the example above, both an actual iPad device agent and a Simulator device agent are shown. The test can be recorded with any configured TouchTest Agent on any device or simulator.

If not, the prompt shown below will appear. Perform the Make App TouchTestable steps in the section above and ensure to associate the mobile device with a mobile app. Note that the screenshot above and the screenshot below are showing different devices selected. Ensure that you are selecting the device or simulator you intend.



- Select the TouchTest Agent that you created above and also select the mobile app you'd like to test.
- Click the Record button in the wizard once your selection is made. TouchTest Agent will launch the selected app on the selected device.

The Stockfish app launches to its initial screen.



1. Perform the following menu actions on your mobile device.

- Select Game, New Game.
- Select Play both.



Make the following chess moves, known as King Gambit Declined:

- Move the white pawn to E4
- Move the black pawn to E5
- Move the white pawn to F4.
- Move the black pawn to D6.



2. Click the Record button again to end recording.



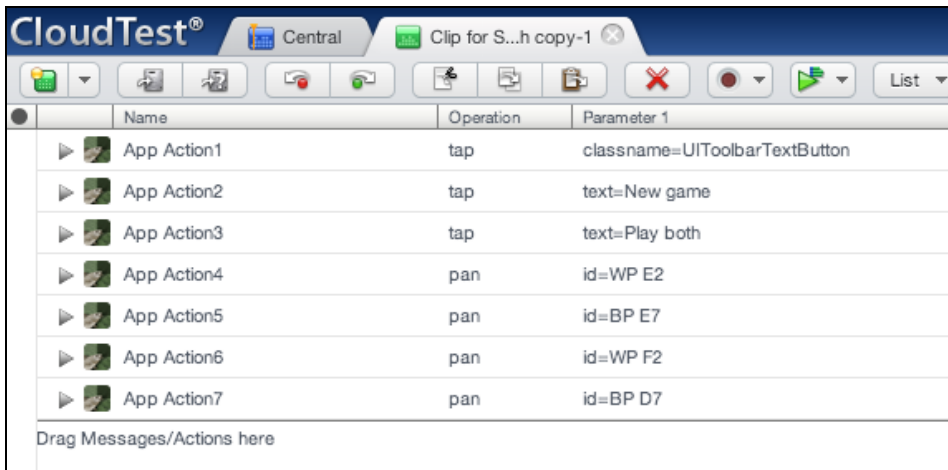


For each app action performed, the Clip Editor adds an app action to the clip.

3. Click Save on the Clip Editor toolbar.
4. Finally, switch to List view by clicking the Icon drop-down. This will provide a better view on clip element details.



Once we've switched to List view, subsequent clips will be recorded and/or opened in that mode.



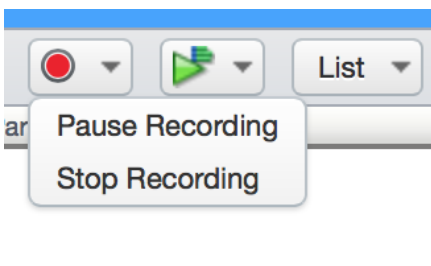
Note the first three actions pertain to the Menu selections we made to start a new game and to play both competitors.

## Pause Recording

You have the ability to pause at any moment during a recording to:

- View your app
- Eliminate unwanted actions
- Save time getting your app to a state where you want it to record actions
- Add more actions to a clip
- Correct locators in the middle of a clip
- Add more waits, outputs, or validations to an existing clip using touch locator
- Record screenshots to use in validations for all cases on a specific page

To pause at any moment, click the **Pause Recording** option.

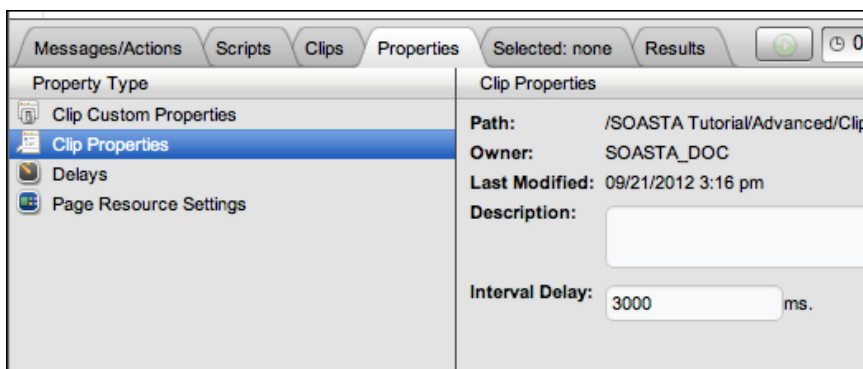


## Adding an Interval Delay between Each Action

In the following steps, we will add an interval delay to the test clip. This type of delay will stretch out the time between all the recorded app actions.

Imposing delays, either using the Interval Delay setting or by inserting Delay clip elements, can make the test more viewable during the editing phase, as well as during test playback (when viewing the test as it plays is most desirable).

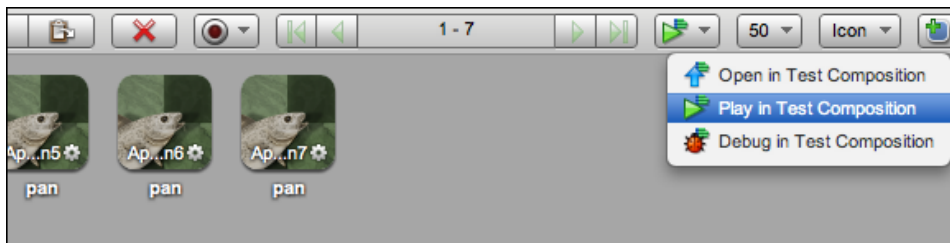
1. Click the Properties tab in the minimized sub-panel and then select the Clip tab at the top of the pane (the Clip tab may already be visible if properties are already open from the prior exercise).
2. In the Property Type list, click Clip Properties.
3. In the Clip Properties panel on the right, enter an Interval Delay in the given field. For example, *2000* ms. Entering *2000* adds a two second gap between each app action in the given test clip.
4. Click Save on the Clip Editor toolbar. When the Save Test Clip box appears, accept the default name, which takes the form "Clip for <Device Name> <Mobile App Name>.



## Create a Composition

With your test clip open in the Clip Editor, you are ready to create and play this simple, new test composition using this test clip.

1. To create a new composition from your test clip, click the Use in Test Composition drop-down in the upper-right corner of the Clip Editor toolbar and note the following commands:



- **Open in Test Composition**

Choose Open in Test Composition to add this clip to a new draft composition where additional composition parameters can be set in the Composition Editor, Edit tab before proceeding to play.

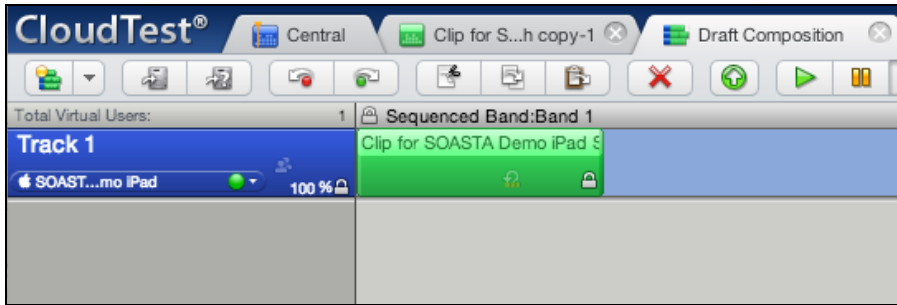
- **Play in Test Composition**

Choose Play in Test Composition to add this clip to a new draft composition where it will immediately be played in the Composition, Play tab before proceeding to edit parameters or play.

- **Debug in Test Composition**

Choose Debug in Test Composition to add this clip to a new draft composition where it can be debugged in the Composition, Debugging tab before proceeding to edit parameters or play based on debug actions.

1. Choose Open in Test Composition. When you do so, the Composition Editor appears with a draft test composition including the new test clip in Track 1.

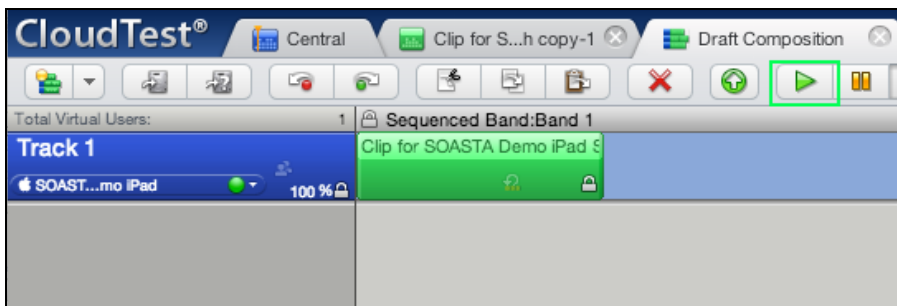


2. Click Save on the Composition Editor toolbar. Accept the default name, which takes the form: "Composition for <Clip Name>."

## Playing a Composition

Perform these additional steps while the test composition is open in the Composition Editor.

- Ensure that the TouchTest Agent status is still "Connected" on the mobile device via Safari.
- In the Composition Editor, click Play to run the test composition.



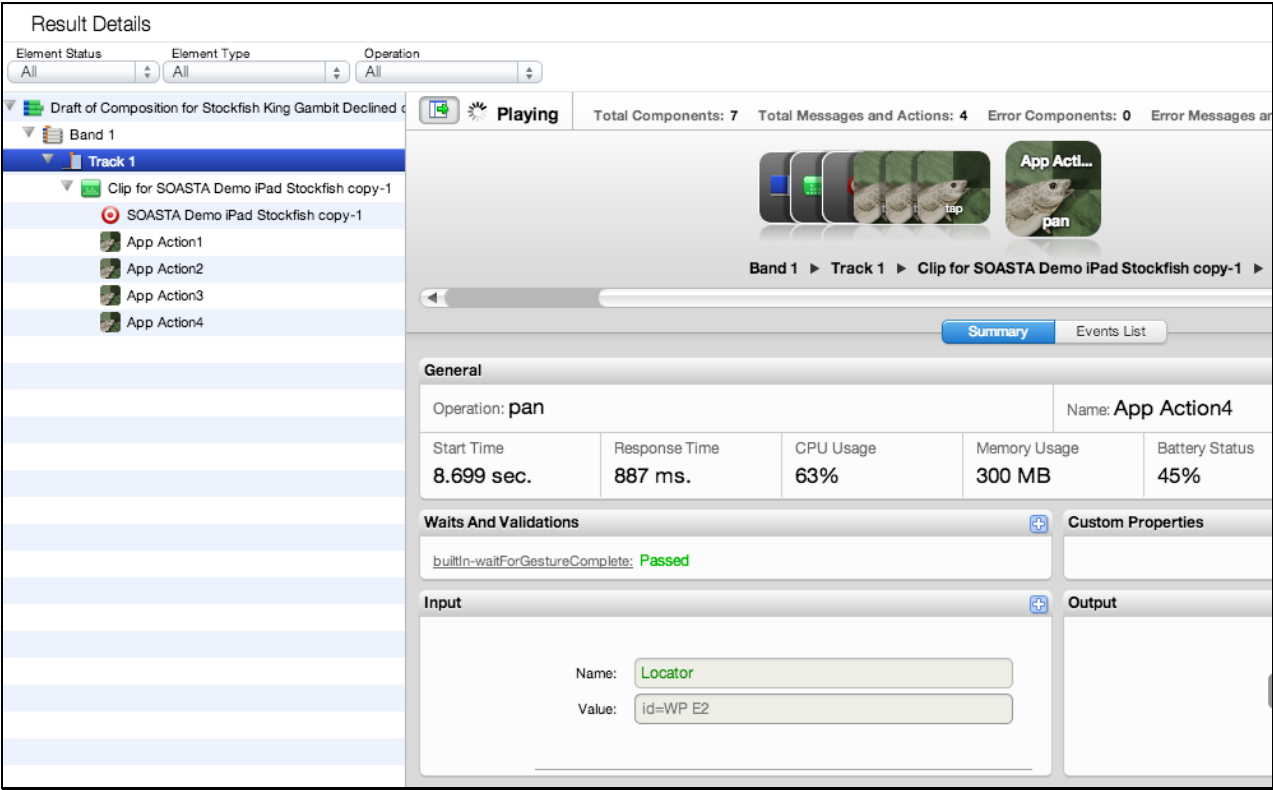
The Composition Editor's Status Indicator changes to "Playing," and the mobile app is launched on the specified mobile device(s) precisely as it was recorded.



While the test runs, the Composition Editor automatically switches to the Play tab, and by default, the Result Details dashboard displays.

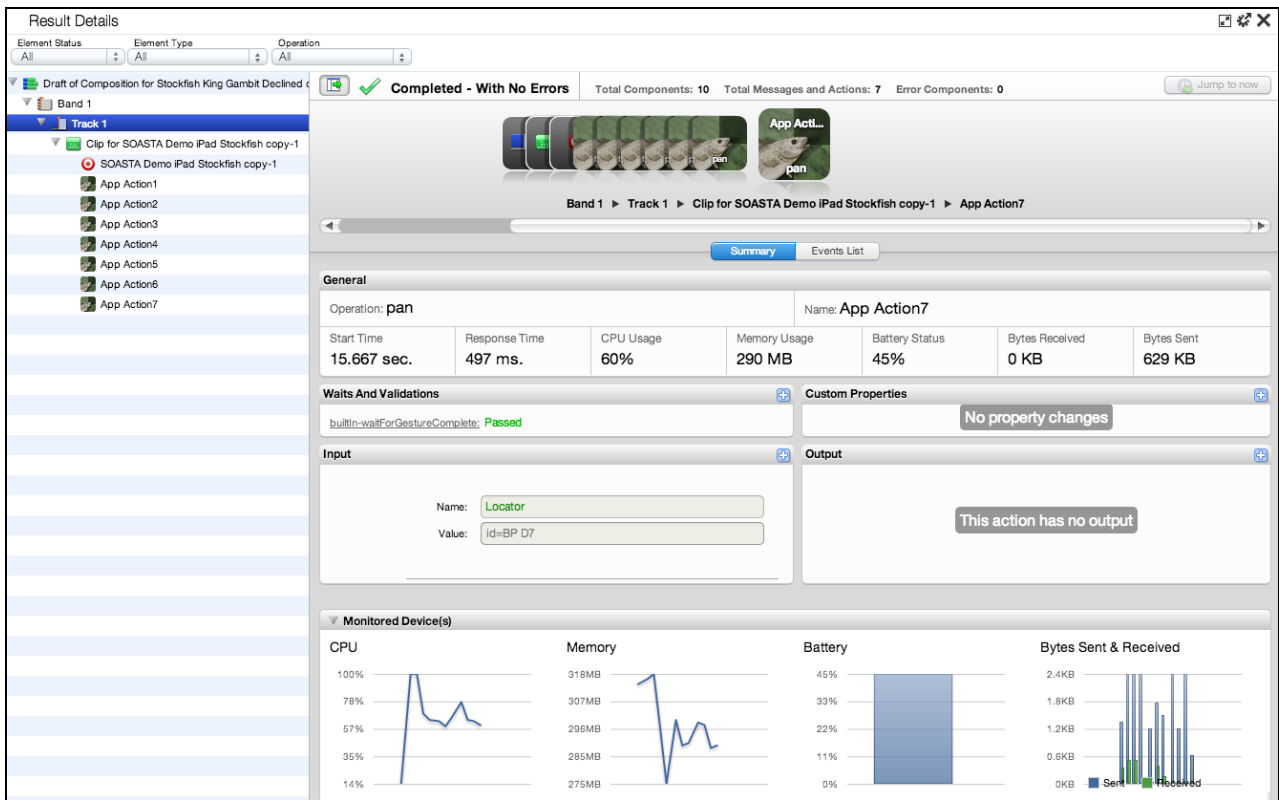
# Result Details

The Result Details dashboard helps to discover the cause of errors in your test, if any. While play continues results are posted in the Composition Editor, Play tab, Result Details widget.



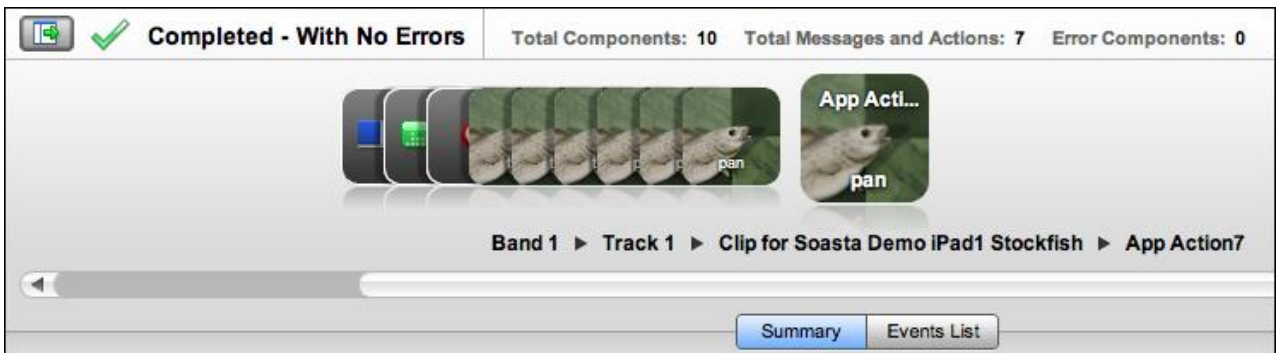
Once play completes, the final results are displayed in the Results tab (also in the Result Details widget).

If the test passed on all points, the status “Completed – With No Errors” is clearly posted in the Result Details dashboard.



The mobile app actions performed when the clip was created are played back on the device. Click to expand the nodes in the Navigation Tree on the left as they appear.

Result Details uses a Cover Flow (top panel to the right) to display the test composition’s stream as it occurs.

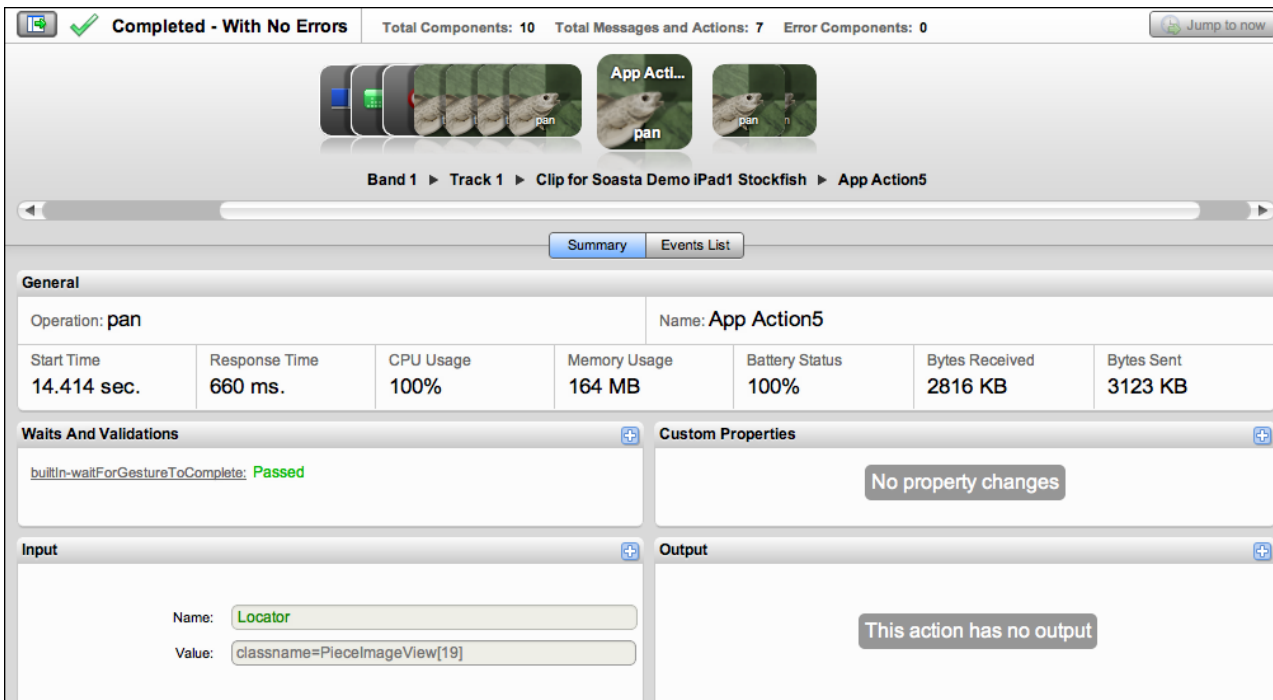


This stream is also shown in the Navigation Tree (on the left) as elements are executed. As play continues, the focus is set to the last executed element unless user

interaction prevents it. The current container is expanded while the prior containers are closed.

Clicking an element during play will halt this auto-focus-to-the-last-executed behavior. To resume auto-focus once interrupted, click Jump to Now in the upper right of the dashboard.

- Click any object in the Cover Flow at the top to center it and display its details and play statistics in the panes below.



- Use the scrollbar to browse the flow. Select any item to show its low-level details.











## Advanced Clip Editing

Now that we've played this simple test composition successfully, and learned how CloudTest will check the success or failure of a given composition, let's return to the test clip to inspect the clip elements and do some additional parameterization.

- Click the Clip Editor tab if it's still open, or right-click the test clip in the Composition Editor and choose Open in New Tab.

The List view is useful while clip editing, because it shows all the parameters and their corresponding inputs in one tabular view.

	Parameter 2		Target Name
UIButton	{"tapCount":"1","duration":"0.1...,31.000000","touchCount":"1"}		SOASTA Demo iPad Stockfish
	{"tapCount":"1","duration":"0.1...,19.500000","touchCount":"1"}		SOASTA Demo iPad Stockfish
	{"tapCount":"1","duration":"0.1...,20.500000","touchCount":"1"}		SOASTA Demo iPad Stockfish
	61.000000,50.000000		SOASTA Demo iPad Stockfish
	54.500000,40.500000		SOASTA Demo iPad Stockfish
	58.000000,46.500000		SOASTA Demo iPad Stockfish
	58.000000,56.000000		SOASTA Demo iPad Stockfish

When additional parameters are present they are displayed to the right of the Parameter 1 column.

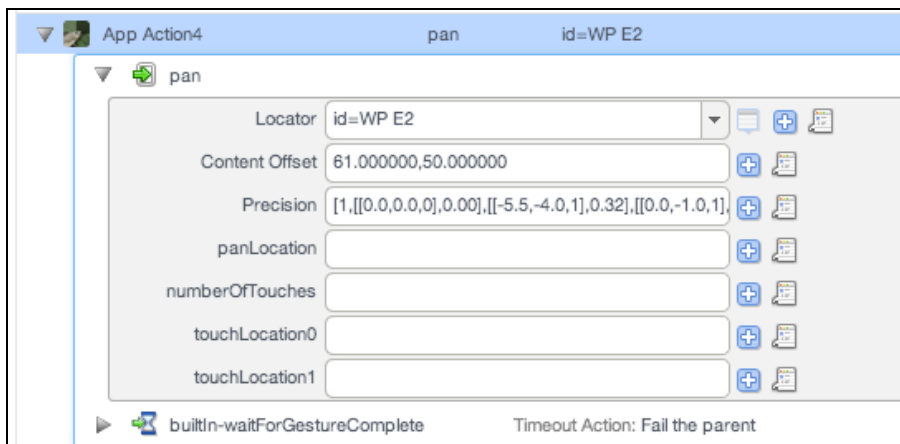
	Parameter 2		Target Name
ttton[0]	{"touchCount":...tapCount":"1"}	🔒	Soasta Demo iPad1 Stockfish
	{"touchCount":...tapCount":"1"}	🔒	Soasta Demo iPad1 Stockfish
	{"touchCount":...tapCount":"1"}	🔒	Soasta Demo iPad1 Stockfish
[12]	47.000000,48.000000	🔒	Soasta Demo iPad1 Stockfish
[19]	44.000000,45.000000	🔒	Soasta Demo iPad1 Stockfish
[12]	43.000000,51.000000	🔒	Soasta Demo iPad1 Stockfish
[17]	45.000000,45.000000	🔒	Soasta Demo iPad1 Stockfish

## Inspecting App Action Details

Examine elements and properties for any App Action by selecting it in the workspace above and then click its Gear icon to Show Info. When you do so, the Info Window appears.

In the test clip below the recorded App Action2 is open in the lower panel. The type of app action, *type*, represents the user name entered on the SOASTA Demo app login page.

1. Locate and expand the first app action in the clip that is a move in the game (WP E2) and then expand the enclosed action (e.g. a pan) to see its inputs.



In our sample clip, this is *Selected: App Action4*. This app action has multiple Inputs:

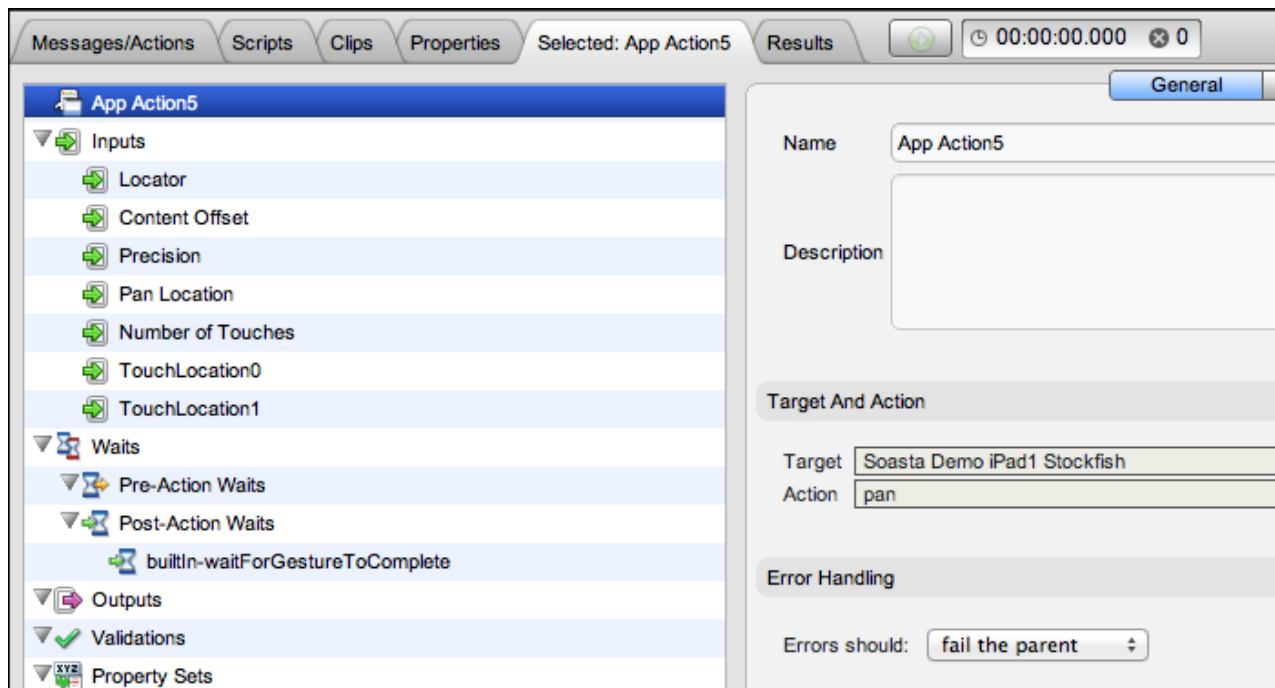
- *id=WP E2* is the physical location of the white pawn before the first move
- *Content Offset* is a physical location that uses *x,y* coordinates while the *Precision* value is an array.

## App Action Properties

Most of the accessors you will use in this exercise can be set using the Add toolbar in the expanded List view.

Some additional parameters, such as Custom Properties, can be set by double-clicking an app action to open it in the Clip Editor lower panel. Action level properties are shown in the tree on the left.

1. Double-click App Action5 to open it in the lower panel.
2. Select the top-level node in the tree (as shown below)
  - General, Repeat, and Custom Properties (for the action only; not for the entire clip) tabs appear on the right. Note that Error Handling here is set to *Errors should fail the parent* by default.



- Other settings, including Waits, Inputs, Outputs, Validations, and Property Sets that were seen in the expanded List view above can also be set in the lower panel by clicking that node in the tree and then performing the desired action on the right.
1. In the *Selected: AppAction5* tab (or for any selected app action), familiarize yourself with the available elements and properties.

-  **Inputs** (Locator, Scale, Precision, Content Offset)

Locators are unique characteristics that identify a specific element or object on a mobile device. Locators come in many forms, including links, IDs such as those defined within CSS, and XPath expressions.

-  **Waits** (Pre-Action Waits , Post-Action Waits )

Waits are commands that tell CloudTest not to execute an Action until a condition is met (pre-action waits), or to not continue processing the outputs, validations and property sets of the Action until a condition is met (post-action waits).

-  **Outputs**

Outputs specify what is to be shown in the Result Viewer for a given Action. Typical outputs include "captureScreenshot", "outputElementText", and "outputInnerHTML". A single Action can have an unlimited number of outputs.

-  **Validations**

Validations verify that some content or event occurred as expected and each validation has a corresponding Failure Action. App Action validations can range from simple true/false conditions to more complex ones. A single App

Action can have an unlimited number of validations. Any validation failures will be exposed in the Results Dashboard.

-  **Property Sets**

Property Sets give you the ability to take text or data from the app you are testing and store it in a custom property for use in a subsequent action or message.

SOASTA CloudTest includes three property sets, all of which have relevance for refining and editing a selected App Action.

- *Custom Properties*

Custom Properties are user-defined properties that are available to all clip elements, including Actions. Custom properties can be thought of as backdoors that allow access to portions of the object model more easily.

- *System Properties*

System Properties are available to all clip elements, including Actions. SOASTA CloudTest defines system properties. For example, a test clip has system properties such as name, repeat timing, label, and more.

- *Global Properties*

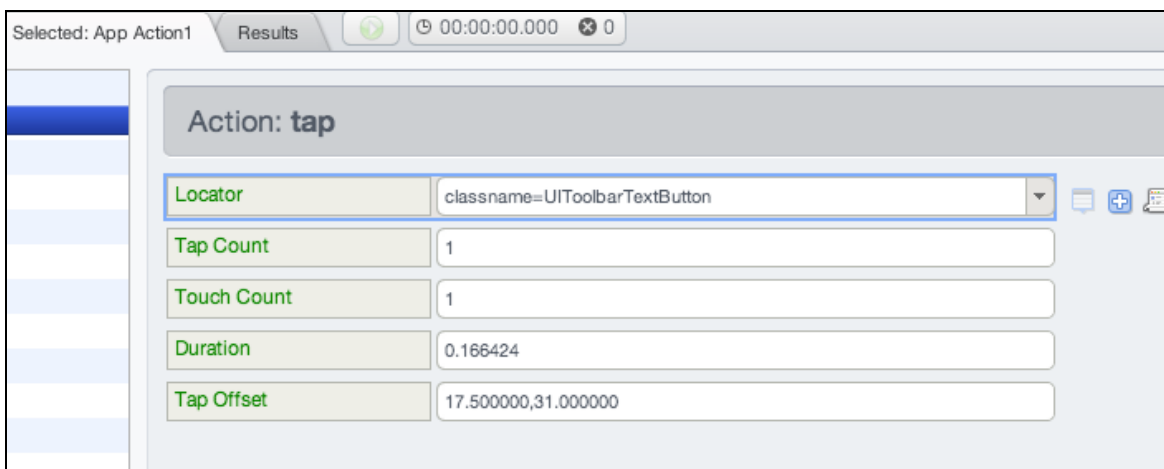
Global properties are defined within the Central > Global Properties List and are “global” within the entire SOASTA CloudTest environment—and can be used across compositions.

## Adding a Text Validation


Next, we will add a validation on App Action1 (this action corresponds to the first action we took while recording—which was to tap the Game button ). The response to this and other actions will contain information worth validating in many cases. The remaining steps demonstrate how to do simple validation in CloudTest.

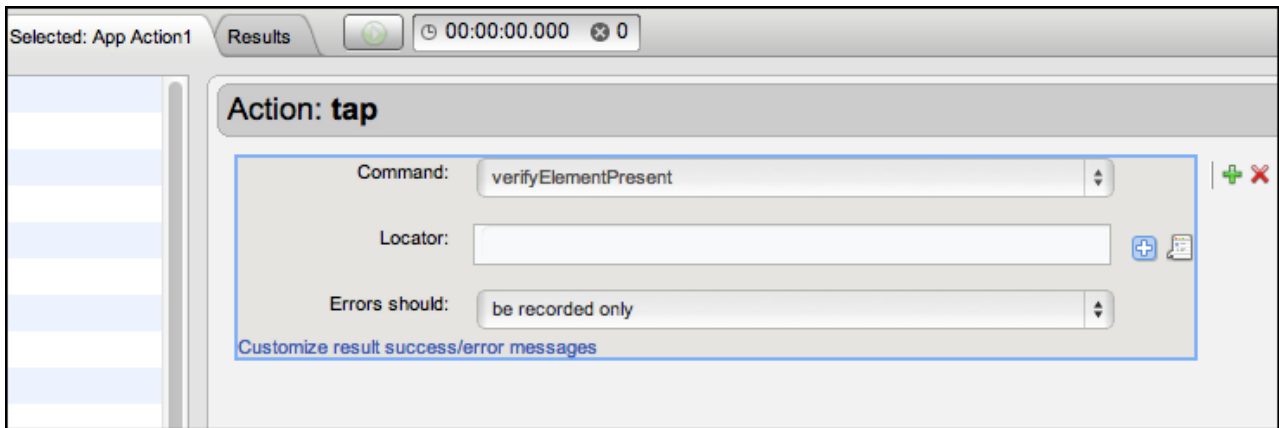
Note the content of the given app action. For example in AppAction1 of our sample clip, *classname=UIToolbarTextButton* had the text label, *Game*.

1. If it's not already open in the lower panel, open it now by double-clicking. The Input form appears.

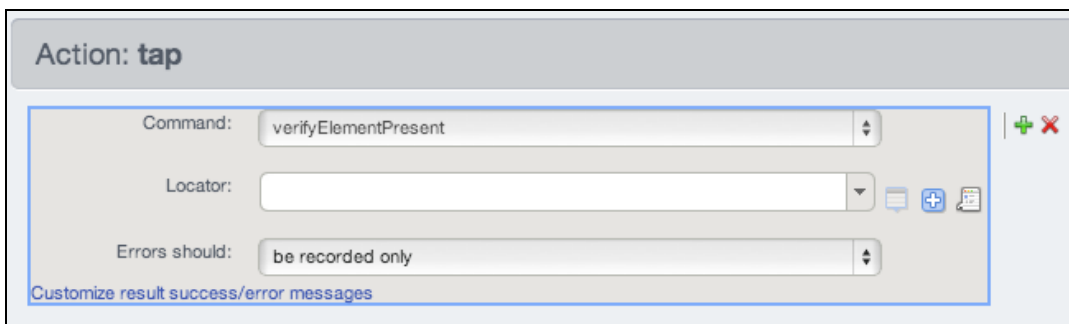


**TIP:** Optionally, in those cases where the locator value is not helpful and you also don't recall the action taken, you can re-enter recording mode. When you do so, the Locator field will display the Touch Locator icon as shown above. Ensure that the TouchTest Agent is still running on the device—and then click the icon to detect the element in the mobile app. For more about this important feature, refer to [Touch Locator for Mobile Apps](#).

2. Click Validations in the list (on the left) and then click the green Plus sign  in the Validations panel on the right.



A validation form is added to the right panel.



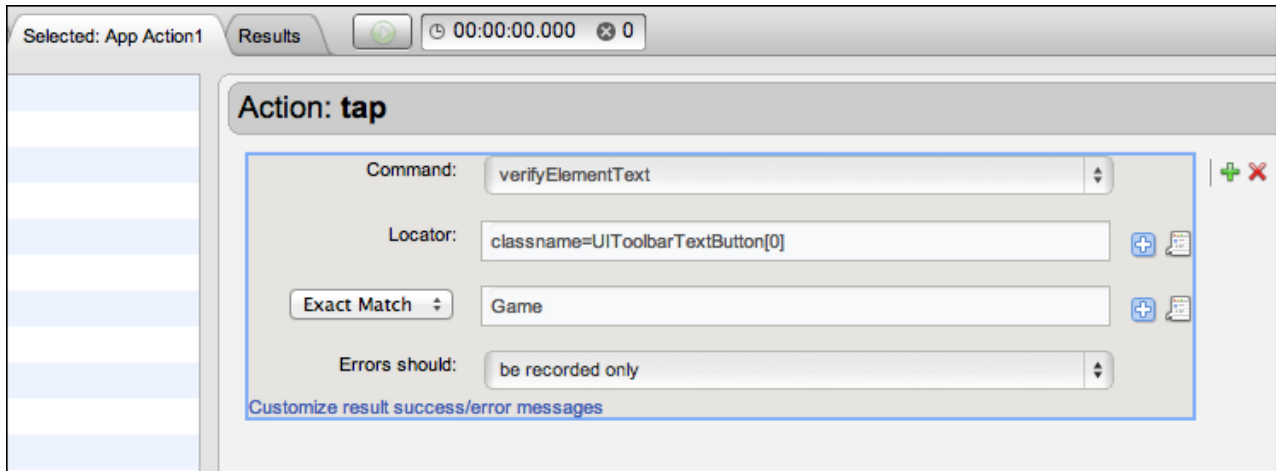
3. In the Command list, a list of commands is presented.

Click the drop-down and select *verifyElementText*. This Command verifies that the specified text is in the rendered element.

4. In the Locator field, enter the field name shown above. For example, *classname=UIToolbarTextButton*.

5. In the Match field, accept *Exact Match* and enter the button label text, *Game*.


Leave the default *be recorded only* set in the Failure action field.



6. Click Save on the Clip Editor toolbar.

### Adding `verifyElementPresent` on an App Action

Validations on app actions can verify that the recorded elements are present at composition runtime or verify details about that clip element. In the following steps, we'll use `verifyElementPresent` to set a simple validation that will fail, record only, or honor the App Action setting.

1. With App Action1 still open in the lower panel and the Validations node also still selected, click the green Plus  sign to the right of the text validation we added above.

A new validation form is added to the Action. If you added the `verifyElementText` validation in the section above, the new validation form appears below it.

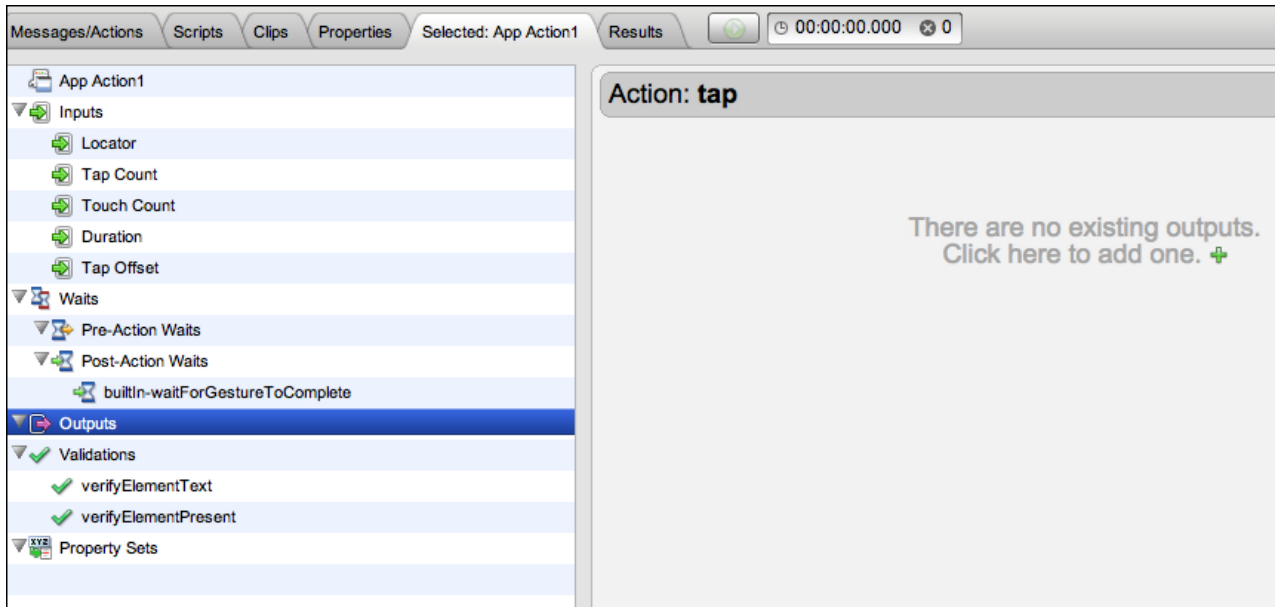
2. Leave `verifyElementPresent` selected in the form.
3. Click the Locator field. Enter the field name `classname=UIToolbarTextButton` exactly as it appeared when recorded.
4. Click Save on the Clip Editor toolbar.



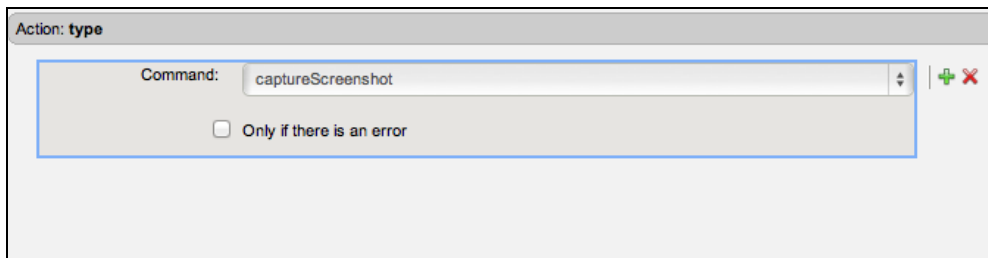
## Adding an Output

In the following steps, we will add an output to an App Action in the test clip we created above. This output will capture a screenshot of the test clip element as it is executed during runtime and this screenshot will be integrated into the test results.

1. With App Action1 still open in the lower panel, select Outputs.



1. In the right panel, click the green Plus sign. A new output is added to the Outputs list and the details are shown in the Element Info field.
2. Click the Command list drop-down, and then select `captureScreenshot` (if it is not already selected).



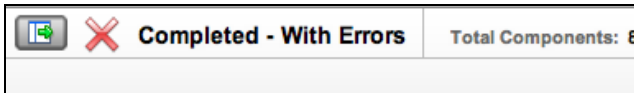
3. Check *Only if there is an error* if the output is desired only in the event this AppAction produces an error. Otherwise, leave it unchecked. For our example, we will leave it unchecked to ensure we get an output in every eventuality.

4. Click Save on the Clip Editor toolbar.
5. Return to the Composition Editor tab once again and click Play a second time.

In the following section, we'll inspect results for the validations and output that were set on App Action1 above.

## Identifying and Analyzing Common Errors

Despite the successful results above, in some cases your test may not succeed initially. As test advocates, we are often more interested in such failures. Your first task is to eliminate "false negatives." These are errors that result from network, environment, or from accessors, such as validations, that we will add later to make the test more useful.

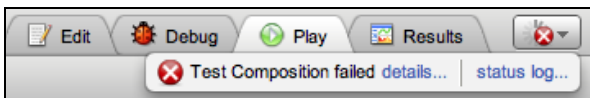


The way we approach them is first to identify where they occurred and then to analyze what occurred.

### Network or Communication Errors

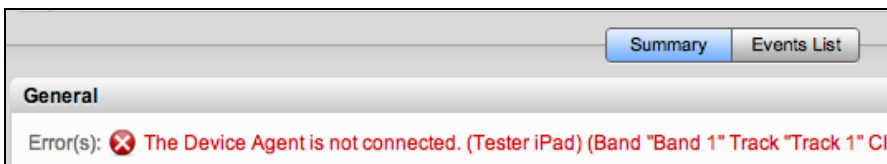
Initial errors in a simple test like the one above are most often only simple network or configuration errors having to do with test staging.

For example, if the Device Agent is not connected or is not responding the Composition Editor's Status Indicator will indicate "Test Composition failed" (shown below).



- Click Details to display additional information in a dialog box.

Composition-wide errors such as these are clearly indicated in the General section in the initial view of Result Details. They frequently are related the state of the Device Agent (e.g. if the device agent is not connected when you click Play).



In some cases, the TouchTest Agent may have been started but is no longer responding (or the device auto-lock may have been invoked). In such cases Logout and re-login, or wake up the device if it is in auto-lock mode (refer to the

Prerequisites section at the beginning of this tutorial for environment requirements including that Auto-Lock is off).

## App Action and Other Errors

The Result Details dashboard shows failures and marks test successful or unsuccessful by the Failure Actions that are set within a test composition and its children. Failure Actions are set stringently by default to fail the test for any error and show that failure in red.

Result Details clearly indicates the type of test failure that has occurred in a given case. The red "X" in the Navigation Tree easily distinguishes failures on specific app actions you recorded, and whenever the error item is selected in Result Details.

Once you have identified the error action (in this case App Action1), select it in the tree to drill down into the details.

The screenshot displays the 'Result Details' interface for a failed test. The left sidebar shows a navigation tree with 'App Action1' highlighted in red. The main content area shows the following details:

- General:** Error(s): Validation verifyElementText failed. Expected: "Game", observed: ""
- Name:** SOASTA Tutorial/SOASTATutorial/Composition for Stockfish King Gambit Declined/Draft of Composition for Stockfish King Gambit Declined created on October 24, 2013 11:46:50 AM PDT
- Status:** Completed
- Effective Duration:** 30 sec.
- Avg. Response Time:** 958 ms.
- Total Message Bytes:** Sent: 0, Received: 0
- Effective Message Throughput:** 0 msgs/sec.
- Custom Properties:** No property changes
- Monitored Device(s):** CPU, Memory, Battery, Bytes Sent & Received charts.

You can get an events view of the selection by clicking the Summary tab (shown above).

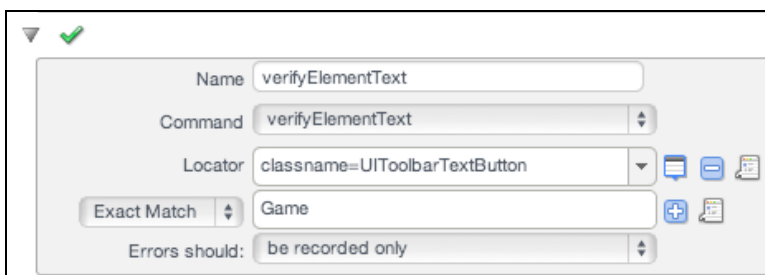
Event(s)				
Event	Time	Level	Event Code	Description
11	2623	Info	App Action: send	<b>Performing App Action.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"
12	2624	Verbose	Transport: appbeg	<b>Performing App Action "App Action1" for Destination "SOASTA Demo iPad Stockfish copy-1", operation "tap".</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"
13	9673	Verbose	Transport: append	<b>App Action "App Action1" completed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"
14	9673	Info	Validation: vstart	<b>Starting validation "verifyElementText".</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"
15	9678	<b>Error</b>	Validation: vcfail	<b>Validation of response body did not pass.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1" ▼ Details: Glob expression match failed: Expression: "Game" Value:
16	9679	<b>Error</b>	Validation: vfail	<b>Validation verifyElementText failed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"
17	9679	Info	Validation: vstart	<b>Starting validation "verifyElementPresent".</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"
18	9679	Info	Validation: vpass	<b>Validation verifyElementPresent passed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"
19	9679	Info	App Action: sent	<b>App Action completed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"

## Correcting Validation Errors using Touch Locator

The error in App Action1 in our example clip shown above resulted because the locator recorded for `text=UIToolbarTextButton` failed. In such a case, double-click the result item, App Action1, in order to return to that same action in the Clip Editor.

While inspecting the failed action in the Clip Editor, List view, click the Locator drop-down (first icon to the right of the Locator entry field) to see if any additional locators were recorded for this action. Try the next one in the order listed and replay the test.

In our test clip, there were no additional locators captured for App Action1. As a result, we decided to use the Touch Locator tool to find additional locators to use.

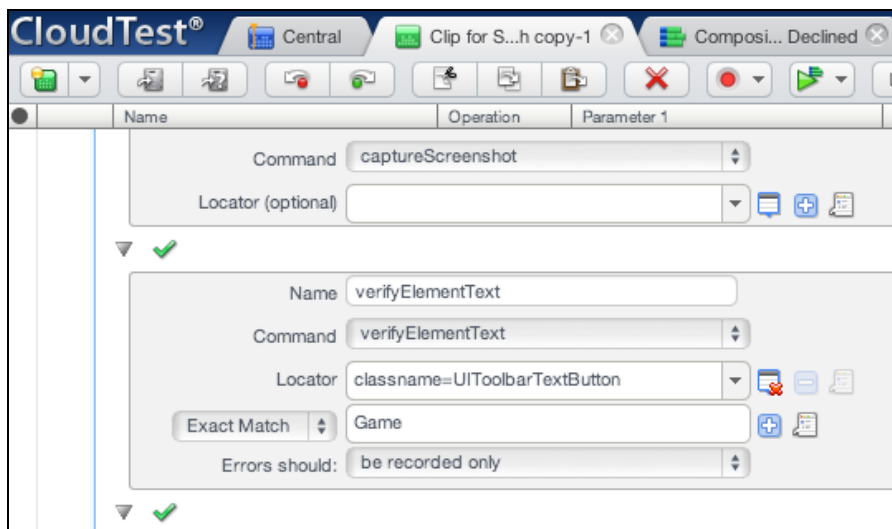


We used the following steps to quickly repair App Action1 (with TouchTest Agent app on top on the iOS device):

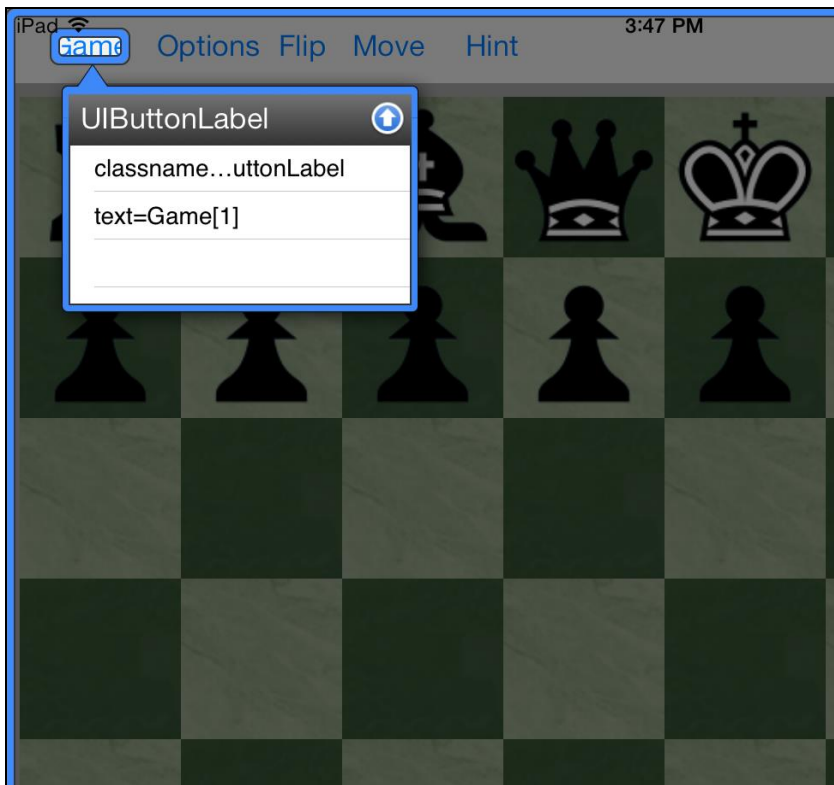
1. Ensure TouchTest Agent is running on top on the device or simulator.
2. Leave App Action1 selected and expanded in the Clip Editor, List view.
3. Drop down the Locator list and select *<add entry>*. Omitting this step will replace the Locator rather than adding it. We like to keep our options open by adding instead of replacing.
4. Invoke record mode a second time (e.g. using the Clip Editor toolbar, Record button).
5. Once on the correct view, in Stockfish there is only one, click the Touch Locator icon. Note that in more complicated apps with many views you'll need to be on the right view before clicking the Touch Locator icon.

A blue border surrounds the page on the device and the in the Clip Editor the Touch Locator icon is active.

**TIP:** If you find that the mode is stuck, try toggling the Touch Locator icon for the given action off and then on again. For more about the Touch Locator feature, see [Touch Locator for Mobile Apps](#).

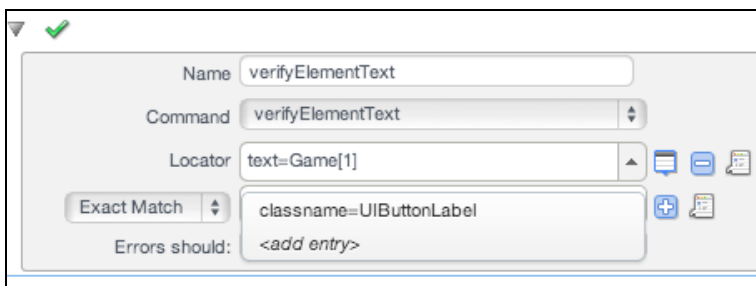


6. Long press the word "Game" on the top menu until the blue border collapses around it and the list of locators shown below appears.



7. Tap and hold the Up Arrow icon (in the Locators box above) to include this list of locators into App Action1.

The blue border will disappear and the expanded action will be populated with the additional locator(s).



8. Select the *new* first locator (e.g. other than *classname=UIButton*). In this case, that is *text=Game[1]*.

9. Repeat the Touch Locator procedure for any additional failures on validations in your test clip.

**TIP:** While working in Touch Locator mode, **delete any new app actions** that are inadvertently recorded. In some cases, if you are navigating between views to get to the right element, more than one new action will be recorded. You can recognize these unwanted actions by their out of order action numbers (e.g. their names are not sequential to the action you selected before entering Touch Locator mode).

10. Save the test clip and play the composition again.

## Support for Accessibility Locators

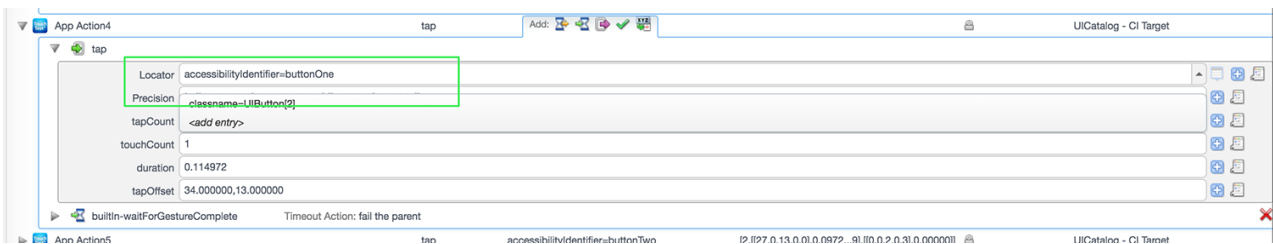
As of SOASTA 57.07, TouchTest supports iOS Accessibility Locators. Accessibility Identifiers and Accessibility Labels are now available as locating strategies, finding UI elements on-screen in addition to text, classname, etc.

### Accessibility Identifiers

**NOTE:** These instructions assume that Accessibility Identifiers have been implemented in your Xcode.

To reach the Accessibility Identifier option:

1. Log into your CloudTest instance.
2. Select an existing clip or create a new one.
3. In your Clip, expand **App Action > Tap**.
4. In the Locator dropdown menu, choose the `accessibilityidentifier=` option.



For general information on the locator tool, please visit [Using the Locator Tool](#).

### Accessibility Labels

**NOTE:** These instructions assume that Accessibility Identifiers have been implemented in your Xcode.



To reach the Accessibility Label option:

1. Log into your CloudTest instance.
2. Select an existing clip or create a new one.
3. In your Clip, expand **App Action > Tap**.
4. In the Locator dropdown menu, choose the accessibilityidentifier= option.

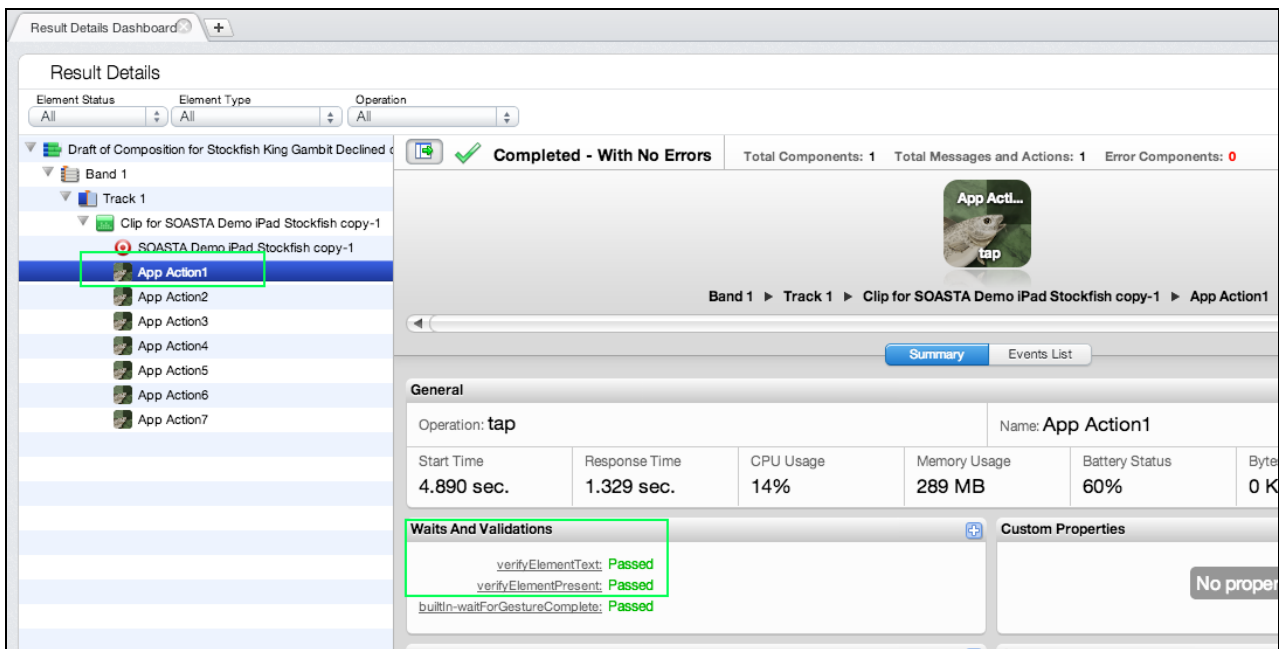


For general information on the locator tool, please visit [Using the Locator Tool](#).

## Analyzing Results

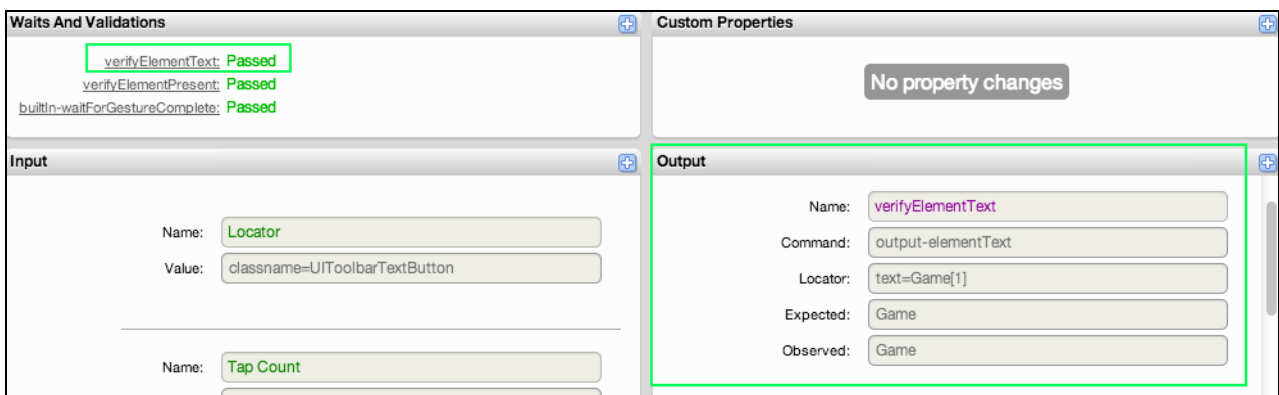
Result Details has several methods for navigating through the test results. Click the checkboxes in the cover flow to quickly display actions) only (within a single band, track, test clip, or chain.

1. Expand the Navigation Tree until App Action1 is in display and then select it as shown below.



**TIP:** Minimize the Monitored Device(s) section to create more visual space on the dashboard.

2. In the Navigation Tree, select the clip element that had the validation. For example, App Action1 (as shown above).
3. Inspect the information on the Summary tab for the selection. In the result above, both the validations on App Action1 passed (using the replacement locator `text=Game[1]`).
4. Click the first validation in the Waits and Validations section and then click the Plus icon to expand the Output panel. Note that the replacement locator has succeeded and the expected text, *Game*, is now validated.



5. Scroll down to view the *captureScreenshot* output defined for App Action1. Note the Add Validation button below the captured shot. You can use this to quickly add image validation using *verifyScreenshot*.



Note: Since we didn't check *Only if there is an error* in the Output form so a shot of the success is included in this result for the given app action.

6. Click the second validation in the Waits and Validations section and then click the Plus icon to expand the Output panel. Note that the original locator has succeeded and the element is validated as present using it.
7. Click the Events List tab for the given selection to view action-related events, including validations. Click the Details arrow to inspect any event's details.

Event(s)				
Event	Time	Level	Event Code	Description
				copy-1" ▶Details:
14	9895	Info	Validation: vstart	<b>Starting validation "verifyElementText".</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"
15	9901	Verbose	Validation: vcpass	<b>Validation of response body passed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1" ▼Details: Exact match: Value matched: Same
16	9901	Info	Validation: vpass	<b>Validation verifyElementText passed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"
17	9901	Info	Validation: vstart	<b>Starting validation "verifyElementPresent".</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"
18	9901	Info	Validation: vpass	<b>Validation verifyElementPresent passed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"
19	9901	Info	App Action: sent	<b>App Action completed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1"
20	9901	Statistics	App Action: stats	<b>App Action statistics.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for SOASTA Demo iPad Stockfish copy-1" Target: "SOASTA Demo iPad Stockfish copy-1" ▶Details:

## Appendix I: Using TouchTestIDs in Your Project Source Code

For developers, SOASTA TouchTest also provides the ability to define explicit mobile app locators, known as TouchTestIDs (TTIDs) as an integral part of touch-testing. TouchTestIDs (TTIDs) are provided as a means to make your tests more user-friendly and readable. Once implemented, TouchTest gives preference to recording the TouchTestID as the default locator.

Minimally, using TouchTestIDs requires the following source code modifications for both iOS and Android projects:

1. An import statement in the view where TTIDs will be used (e.g. using the OS-specific syntax required).
2. In any given view, call `setTouchTestId` and assign a string parameter to each view that needs one.

Once TTIDs are added TouchTest gives preference to recording the TouchTestID as the default locator. The use of `touchTestId` in Locators is described below.

### Adding TouchTest™ IDs to An iOS App

For developers, SOASTA TouchTest™ provides TouchTest Driver to enhance mobile app locators as an integral part of touch-testing. The use of `touchTestId` in Locators is described below. For a much more advanced scenario, refer to the [TouchTest Advanced Tutorial](#).

**Note:** Use the `touchTestId` conditionally in a manner that guarantees it is not part of code that gets submitted to the App Store. This can be done in iOS using Conditional Compilation such as the `#ifdef/endif` convention shown below.

1. Identify the source file where the view is initialized. One example of where views can be initialized is in the Tocuhes app's method, `awakeFromNib`.

2. Include the TouchTest™ header file in the source file with the initialization of the view by using:

```
#ifndef TOUCHTESTDRIVER
#import "TouchTest Driver.h"
#endif
```

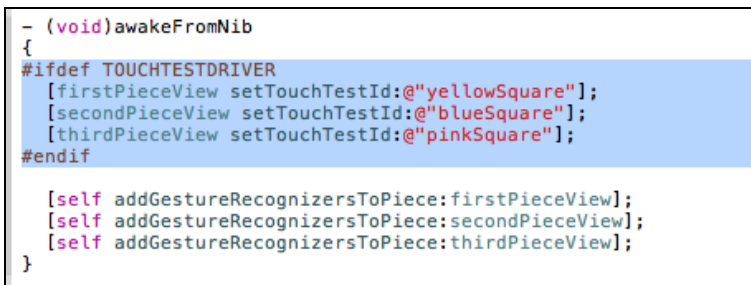
3. Next, call the `setTouchTestId` method on each view that will be located.

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

For example,

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

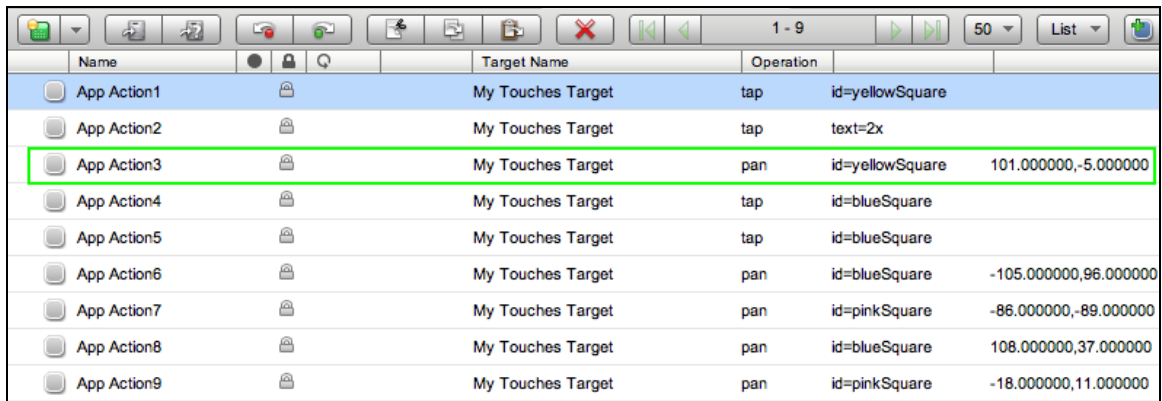
The selected (blue) lines in the screenshot below were added to the given source file.

A screenshot of a code editor showing a method `-(void)awakeFromNib`. The code is enclosed in curly braces. A blue highlight covers the following lines: `#ifndef TOUCHTESTDRIVER`, `[firstPieceView setTouchTestId:@"yellowSquare"];`, `[secondPieceView setTouchTestId:@"blueSquare"];`, `[thirdPieceView setTouchTestId:@"pinkSquare"];`, and `#endif`. The rest of the code, including `[self addGestureRecognizerToPiece:firstPieceView];`, `[self addGestureRecognizerToPiece:secondPieceView];`, `[self addGestureRecognizerToPiece:thirdPieceView];`, and the closing brace, is not highlighted.

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

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

Note that in the example test clip below the `touchTestDriverId` is shown where present.



Name	Target Name	Operation	
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





SOASTA, Inc.

444 Castro St.

Mountain View, CA 94041

866.344.8766

<http://www.soasta.com>