SOASTA

TouchTest^{TM} Appcelerator Tutorial

Table of Contents

SOASTA

Why Mobile App Testing?

CloudTest 's new TouchTest™ technology delivers, for the first time, complete functional test automation for continuous multi-touch, gesture-based mobile applications. TouchTest™ technology delivers fast, precision functional testing while increasing the stability of automated tests across releases.

CloudTest® controls mobile devices through a lightweight software agent, SOASTA TouchTest Agent, and accesses them using their IP address. Devices can be dedicated to testing in the lab, used as part of a short external test, or crowd-sourced as part of a high volume, globally distributed test.

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

CloudTest® Basics

SOASTA provides fast, effective performance, load and functional testing of any modern Web application, Web service, or mobile application in a lab, staging or production environment using a unique multi-track user interface. 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.

The CloudTest® Central tab lists all primary features, organized by sections. Central's first section—highlighted in light blue—contains the Welcome page, and the primary test building tools.

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, as well as the dashboard where results are analyzed.

The **Clip** is the basic building block of a test as presented in the **Clip Editor** and has a **Target** such as HTTP traffic for a site, or a browser UI (web site); or in the case of TouchTest™, a mobile app. A clip can contain messages, actions, scripts, as well as delays and checkpoints—all of which can be organized into containers (chains, groups, pages, transactions, if-then-else, and swtich)—and parameterized as required.

TouchTest™ clips are recorded directly from the mobile app and added to the Clip Editor (as described in this tutorial) as you perform them on the mobile device. 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.).

What Does Touch Test Record?

TouchTest™ records the details of actual gestures and events that iOS invokes on th e 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 to uch 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 Actio n. 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.).

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

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 determined for a given mobile app it can be easily captured. The test designer can move quickly from recording to defining validations and other test details for those captured app actions. In the context of mobile testing, planning will also take 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, and multiple repeats—possibly in tandem with geographic location. Complex mobile app tests can be easily built utilizing one or all of these capabilities.

Adding TouchTest™ to an Appcelerator App

This section describes the steps necessary to make an exisiting Appcelerator project TouchTestable. This tutorial uses the Appcelerator sample app, KitchenSink. However, you can substitute any Appcelerator app and easily follow along.

Appcelerator Prerequisites

The following steps assume a minimal familiarity with Appcelerator and the following prerequisites:

- Titanium Studio is installed.
- An Appcelerator application or sample app, such as KitchenSink, is available to be made TouchTestable.

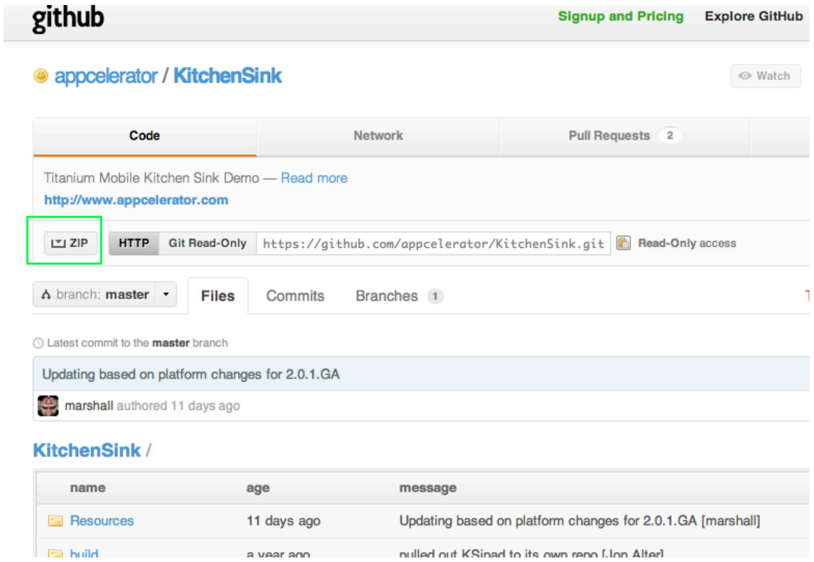
The remaining prerequisites are common to any Appcelerator project that utilizes the iOS Developer Program to install an iOS app on a device:

- The developer is enrolled in the iOS Developer Program (so that the Appcelerator app can be pushed to the iOS Device via iTunes).
- The Unique Device Identifier (UDID) of the iOS Device that will be used to test must be registered at the Apple Provisioning Portal.
- iTunes 10.x or greater must be installed on the desktop client that is running Titanium Studio.

Importing the Sample Project

If you haven't got an Appcelerator app yet, use the following steps to import the sample KitchenSink app used in this tutorial into Titanium Studo.

1. Download the KitchenSink project archive, and unzip it into your repository folder (for example, using the Github for Mac app).



Titanium Mobile Kitchen Sink Demo — Read more
http://www.appcelerator.com

https://github.com/appcelerator/KitchenSink.git

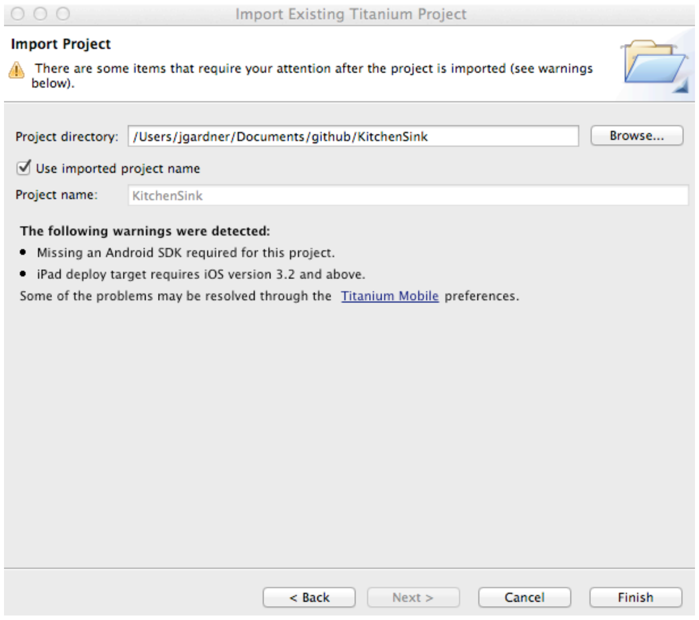
Latest commit to the master branch

Updating based on platform changes for 2.0.1.GA
marshall authored 11 days ago

KitchenSink /

name	age	message
Resources	11 days ago	Updating based on platform changes for 2.0.1.GA [marshall]
build	a year ago	pulled out KSipad to its own repo [Jon Alter]

2. Launch Titanium Studio and login.
3. Click File > Import. The Import Project box appears.
4. Click Next, and identify the folder to use as the Project directory. For example, `/Users/username/Documents/github/KitchenSink. Titanium Studio`).



Import Project
There are some items that require your attention after the project is imported (see warnings below).

Project directory: /Users/jgardner/Documents/github/KitchenSink

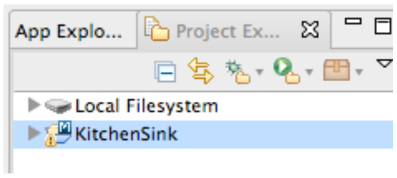
☑ Use imported project name

Project name: KitchenSink

The following warnings were detected:
- Missing an Android SDK required for this project.
- iPad deploy target requires iOS version 3.2 and above.

Some of the problems may be resolved through the Titanium Mobile preferences.

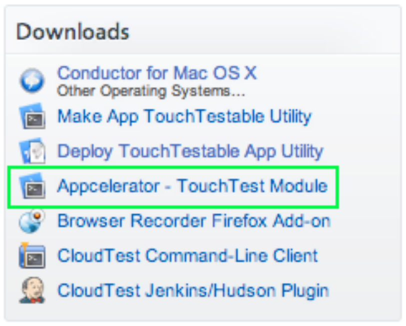
The KitchenSink project is created and appears in the Project Explorer list in Titanium Studio.



In our example, we downloaded the KitchenSink archive to /Documents/github/KitchenSink.

Install the TouchTest™ Module

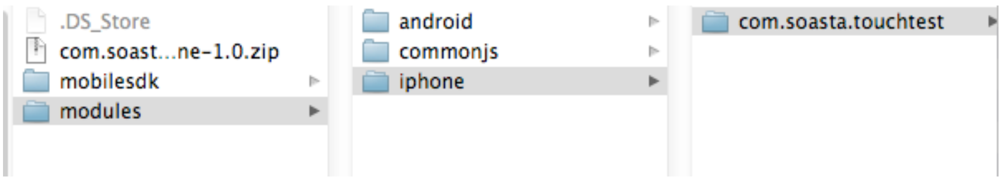
1. Download the TouchTest™ module from the CloudTest Welcome page's Download section.


Downloads
Conductor for Mac OS X
Other Operating Systems...
Make App TouchTestable Utility
Deploy TouchTestable App Utility
Appcelerator - TouchTest Module
Browser Recorder Firefox Add-on
CloudTest Command-Line Client
CloudTest Jenkins/Hudson Plugin

2. Copy the TouchTest™ module archive to /Library/ApplicationSupport/Titanium.
3. Unarchive the .zip file and you will get a "modules" folder.

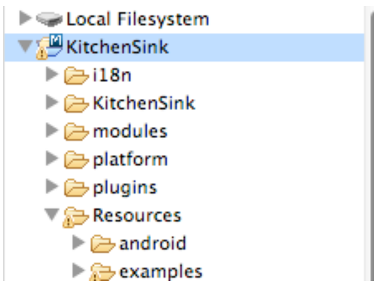
   If a modules folder doesn't already exist, then one is created.

This will result in creation of the /Library/ApplicationSupport/Titanium/modules folder for TouchTest.

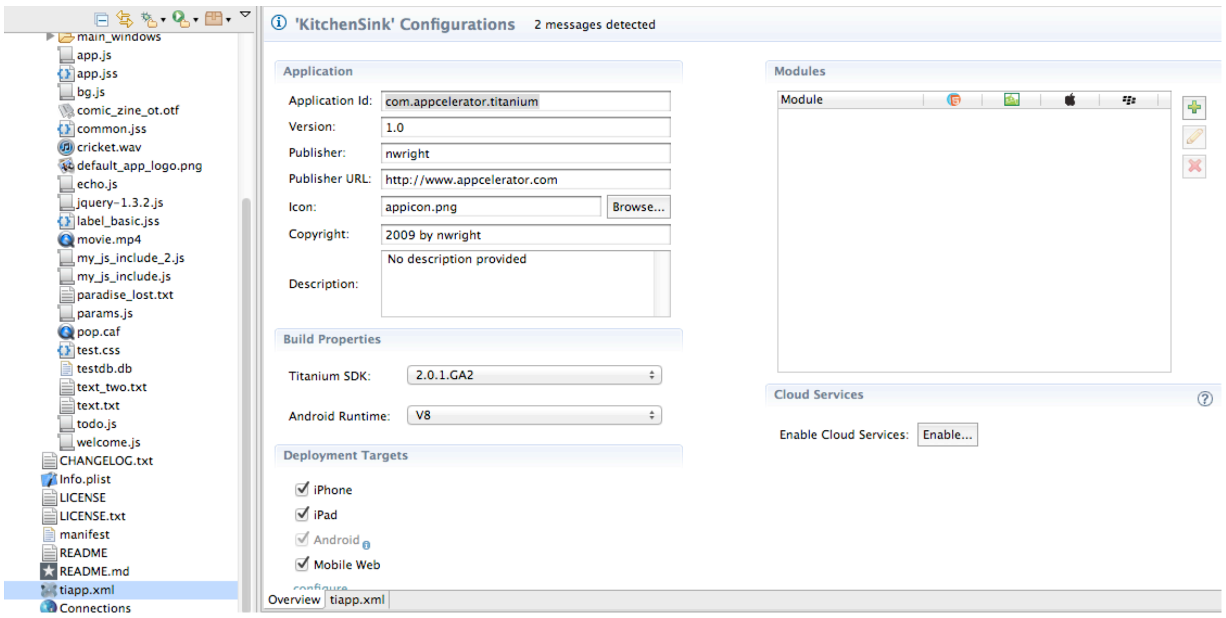


Add the TouchTest™ Module to a Project

4. Launch Titanium Studio and login.
5. Select the Appcelerator app to edit. For example, KitchenSink or your own app.



6. Locate and open *tiapp.xml* in your project (lower left below). The KitchenSink Configurations page displays in the workspace with the Overview tab selected.



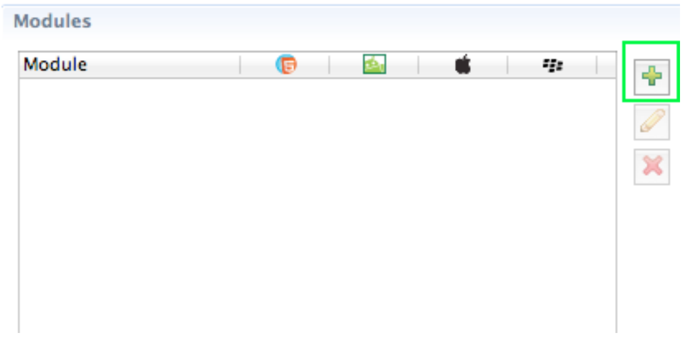
Application Id: com.appcelerator.titanium
Version: 1.0
Publisher: nwright
Publisher URL: http://www.appcelerator.com
Icon: appicon.png
Copyright: 2009 by nwright
Description: No description provided

Titanium SDK: 2.0.1.GA2
Android Runtime: V8

☑ iPhone
☑ iPad
☑ Android
☑ Mobile Web

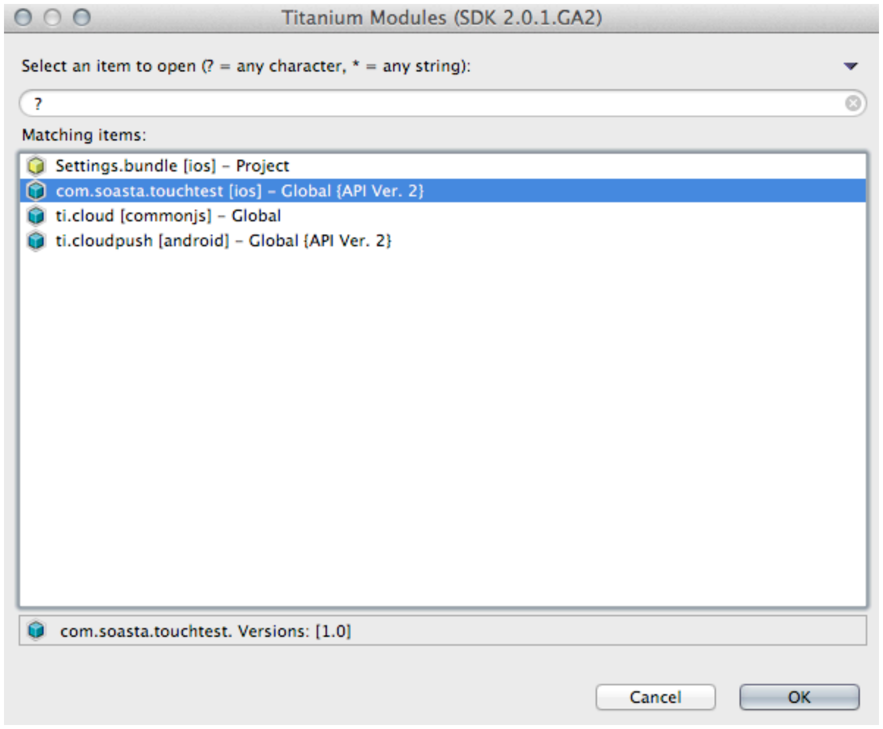
Enable Cloud Services:

7. In the Modules section on the right, click the Add button.



The Titanium Modules box appears.

8. Select the com.soasta.touchtest module and click OK.



Titanium Modules (SDK 2.0.1.GA2)

Select an item to open (? = any character, * = any string):

?

Matching items:

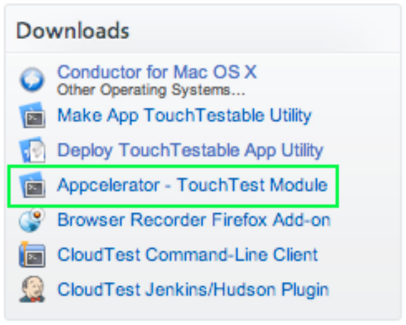
- Settings.bundle [ios] – Project
- com.soasta.touchtest [ios] – Global {API Ver. 2}
- ti.cloud [commonjs] – Global
- ti.cloudpush [android] – Global {API Ver. 2}

com.soasta.touchtest. Versions: [1.0]

Updating the TouchTest Module in an Existing Project

Occasionally, it will be necessary to update the Appcelerator Touch Test Module. Use the following steps to do so.

1. Exit Titanium Studio if it is running.
2. Return to the CloudTest Welcome page and download the TouchTest™ module again.



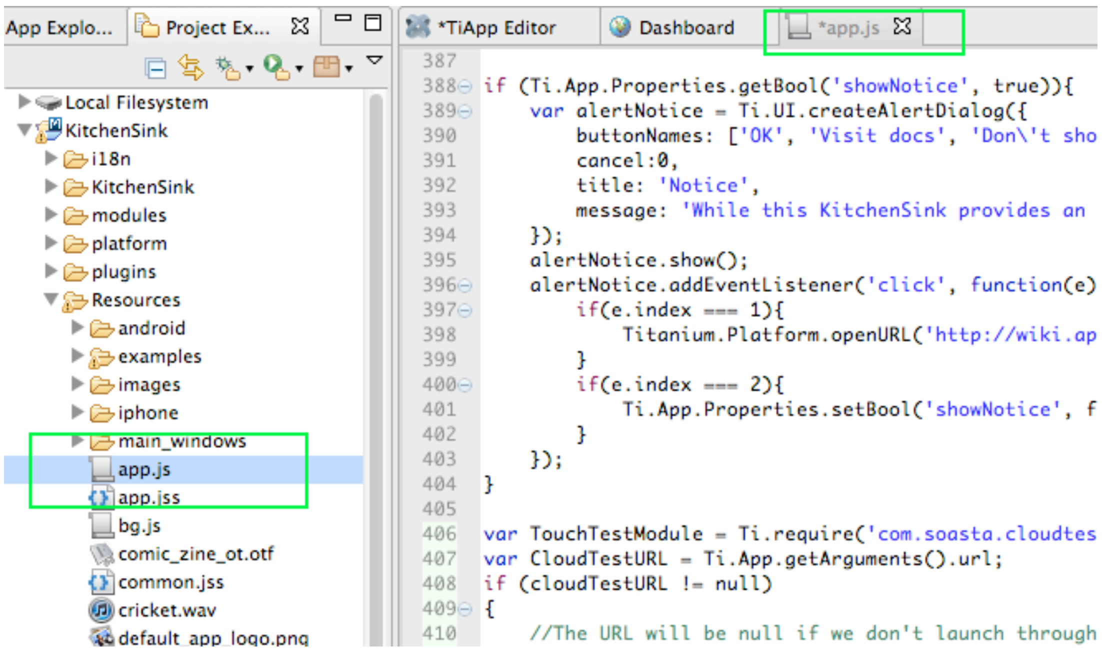
Downloads
Conductor for Mac OS X
Other Operating Systems…
Make App TouchTestable Utility
Deploy TouchTestable App Utility
Appcelerator - TouchTest Module
Browser Recorder Firefox Add-on
CloudTest Command-Line Client
CloudTest Jenkins/Hudson Plugin

3. Unarchive the module and copy its contents into the modules folder under /Library/Application Support/Titanium (replace the com.soasta.touchtest folder and its contents).
4. Launch Titantium Studio and rebuild your Titanium app.

Add TouchTest™ Initialization Code to the Project

Next, we will add the TouchTest™ initialization code to the project.

1. In the Project Explorer list, select and double-click to open the app.js file in its own tab.



```
if (Ti.App.Properties.getBool('showNotice', true)){
    var alertNotice = Ti.UI.createAlertDialog({
        buttonNames: ['OK', 'Visit docs', 'Don\'t sho
        cancel:0,
        title: 'Notice',
        message: 'While this KitchenSink provides an
    });
    alertNotice.show();
    alertNotice.addEventListener('click', function(e)
        if(e.index === 1){
            Titanium.Platform.openURL('http://wiki.ap
        }
        if(e.index === 2){
            Ti.App.Properties.setBool('showNotice', f
        }
    });
}

var TouchTestModule = Ti.require('com.soasta.cloudtes
var CloudTestURL = Ti.App.getArguments().url;
if (cloudTestURL != null)
{
    //The URL will be null if we don't launch through
```

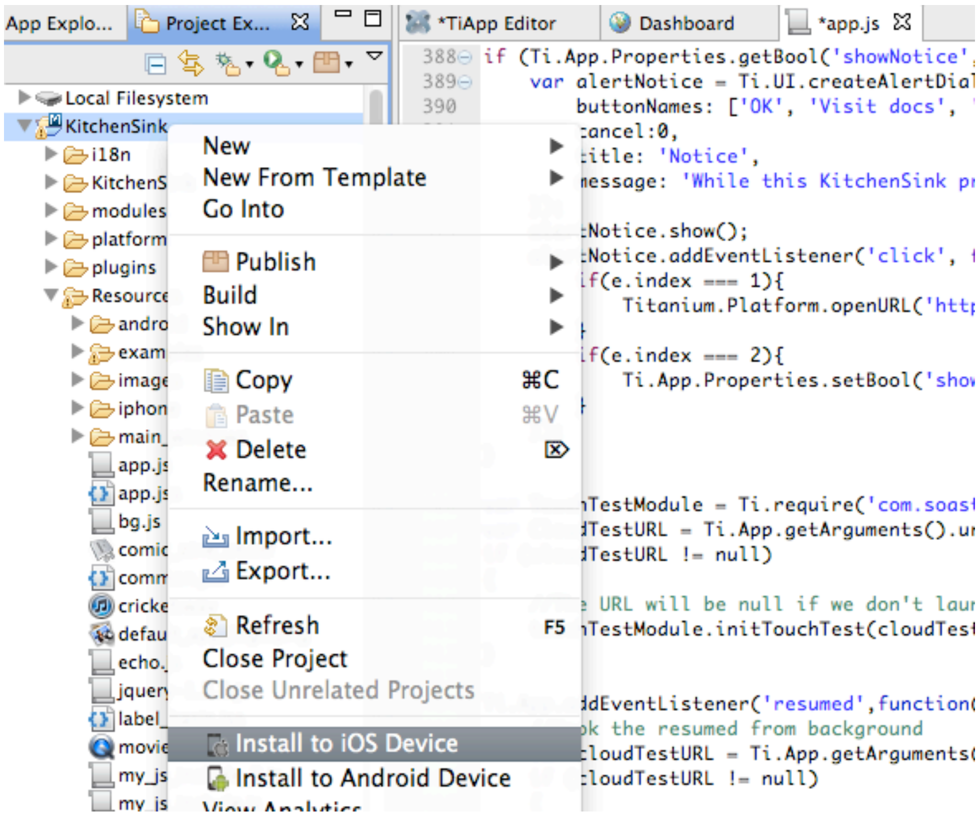
2. In the app.js tab, scroll to the end and paste or enter the following initialization code:

```
var touchTestModule = require("com.soasta.touchtest");
var cloudTestURL = Ti.App.getArguments().url;
if (cloudTestURL != null)
{
    // The URL will be null if we don't launch through
TouchTest.
    touchTestModule.initTouchTest(cloudTestURL);
}
Ti.App.addEventListener('resumed',function(e){
    // Hook the resumed from background
    var cloudTestURL = Ti.App.getArguments().url;
    if (cloudTestURL != null)
    {
      touchTestModule.initTouchTest(cloudTestURL);
    }
});
```

This completes the necessary steps to make the Appcelerator app TouchTestable.

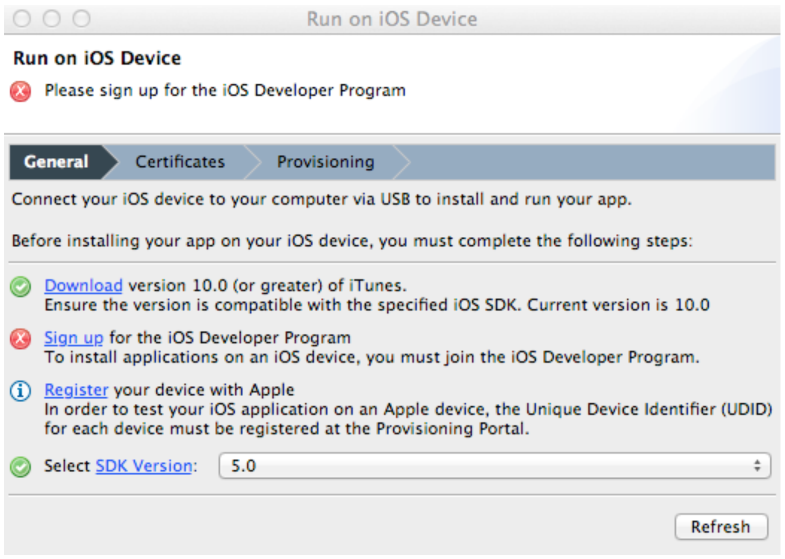
Install to the Device

1. Select the Project node for the app (i.e. KitchenSink) and right-click to select Install to iOS Device.



```
if (Ti.App.Properties.getBool('showNotice'
    var alertNotice = Ti.UI.createAlertDial
        buttonNames: ['OK', 'Visit docs', '
        :ancel:0,
        :itle: 'Notice',
        message: 'While this KitchenSink pr

    :Notice.show();
    :Notice.addEventListener('click', 
    if(e.index === 1){
        Titanium.Platform.openURL('htt
    }
    if(e.index === 2){
        Ti.App.Properties.setBool('sho
    }

    nTestModule = Ti.require('com.soas
    dTestURL = Ti.App.getArguments().u
    dTestURL != null)

    e URL will be null if we don't lau
    nTestModule.initTouchTest(cloudTes

    ddEventListener('resumed',function(
    ok the resumed from background
    cloudTestURL = Ti.App.getArguments(
    cloudTestURL != null)
```

2. The Run on iOS Device box appears.



Run on iOS Device

Please sign up for the iOS Developer Program

Connect your iOS device to your computer via USB to install and run your app.

Before installing your app on your iOS device, you must complete the following steps:

Download version 10.0 (or greater) of iTunes.
Ensure the version is compatible with the specified iOS SDK. Current version is 10.0

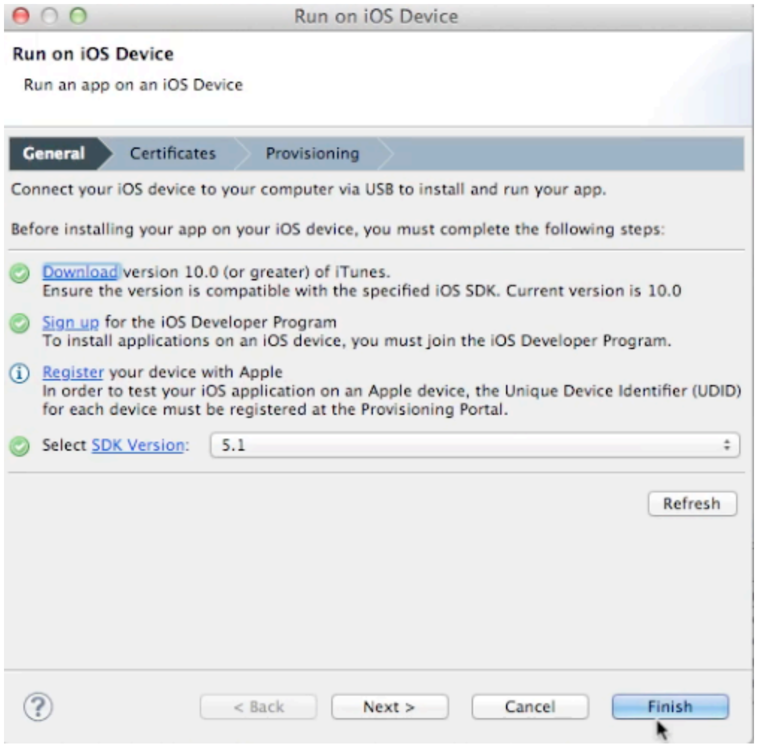
Sign up for the iOS Developer Program
To install applications on an iOS device, you must join the iOS Developer Program.

Register your device with Apple
In order to test your iOS application on an Apple device, the Unique Device Identifier (UDID) for each device must be registered at the Provisioning Portal.

Select SDK Version: 5.0

Note that if the logged in Titanium Studio user is not enrolled in the iOS Developer Program, this is indicated in the box.

3. Ensure that the iOS device is connected to the desktop client running Titanium Studio.



Run on iOS Device

Run an app on an iOS Device

Connect your iOS device to your computer via USB to install and run your app.

Before installing your app on your iOS device, you must complete the following steps:

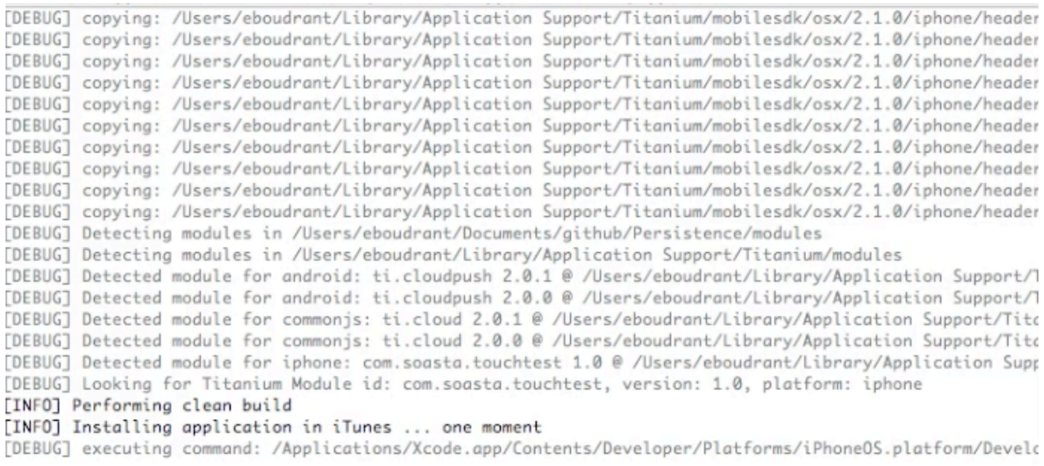
Download version 10.0 (or greater) of iTunes.
Ensure the version is compatible with the specified iOS SDK. Current version is 10.0

Sign up for the iOS Developer Program
To install applications on an iOS device, you must join the iOS Developer Program.

Register your device with Apple
In order to test your iOS application on an Apple device, the Unique Device Identifier (UDID) for each device must be registered at the Provisioning Portal.

Select SDK Version: 5.1

4. Click Finish to build the project and push it to the iOS Device.



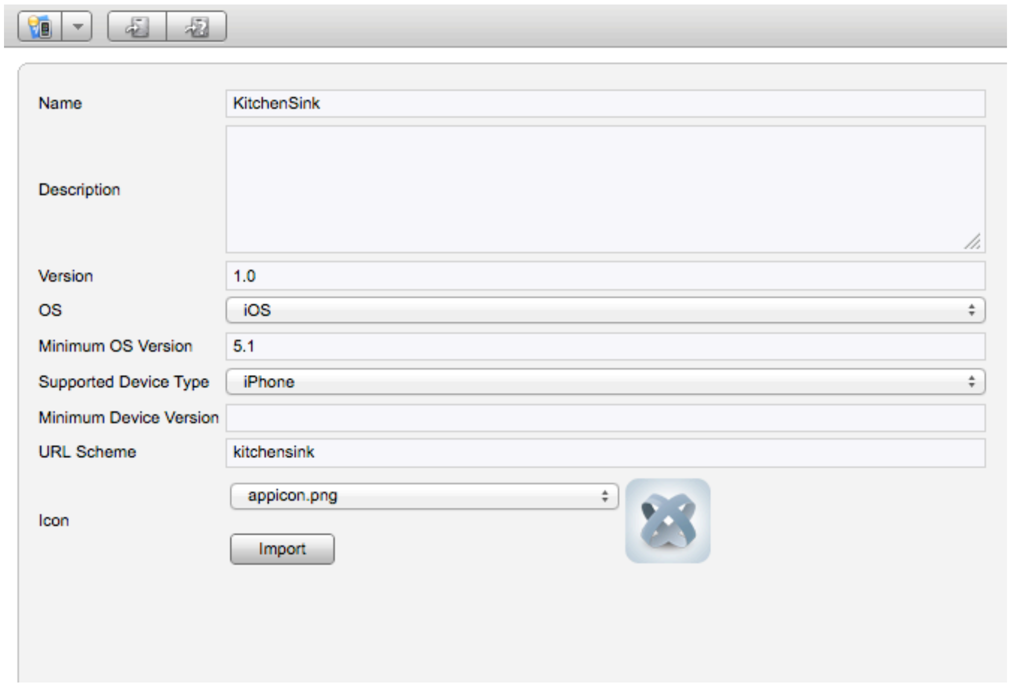
```
[DEBUG] copying: /Users/eboudrant/Library/Application Support/Titanium/mobilesdk/osx/2.1.0/iphone/header
[DEBUG] copying: /Users/eboudrant/Library/Application Support/Titanium/mobilesdk/osx/2.1.0/iphone/header
[DEBUG] copying: /Users/eboudrant/Library/Application Support/Titanium/mobilesdk/osx/2.1.0/iphone/header
[DEBUG] copying: /Users/eboudrant/Library/Application Support/Titanium/mobilesdk/osx/2.1.0/iphone/header
[DEBUG] copying: /Users/eboudrant/Library/Application Support/Titanium/mobilesdk/osx/2.1.0/iphone/header
[DEBUG] copying: /Users/eboudrant/Library/Application Support/Titanium/mobilesdk/osx/2.1.0/iphone/header
[DEBUG] copying: /Users/eboudrant/Library/Application Support/Titanium/mobilesdk/osx/2.1.0/iphone/header
[DEBUG] copying: /Users/eboudrant/Library/Application Support/Titanium/mobilesdk/osx/2.1.0/iphone/header
[DEBUG] copying: /Users/eboudrant/Library/Application Support/Titanium/mobilesdk/osx/2.1.0/iphone/header
[DEBUG] copying: /Users/eboudrant/Library/Application Support/Titanium/mobilesdk/osx/2.1.0/iphone/header
[DEBUG] Detecting modules in /Users/eboudrant/Documents/github/Persistence/modules
[DEBUG] Detecting modules in /Users/eboudrant/Library/Application Support/Titanium/modules
[DEBUG] Detected module for android: ti.cloudpush 2.0.1 @ /Users/eboudrant/Library/Application Support/T
[DEBUG] Detected module for android: ti.cloudpush 2.0.0 @ /Users/eboudrant/Library/Application Support/T
[DEBUG] Detected module for commonjs: ti.cloud 2.0.1 @ /Users/eboudrant/Library/Application Support/Tit
[DEBUG] Detected module for commonjs: ti.cloud 2.0.0 @ /Users/eboudrant/Library/Application Support/Tit
[DEBUG] Detected module for iphone: com.soasta.touchtest 1.0 @ /Users/eboudrant/Library/Application Sup
[DEBUG] Looking for Titanium Module id: com.soasta.touchtest, version: 1.0, platform: iphone
[INFO] Performing clean build
[INFO] Installing application in iTunes ... one moment
[DEBUG] executing command: /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Develo
```

When all of the conditions and steps above are completed, the app is pushed to iTunes and onto the iOS Device via the normal synchronization process.

Adding a Mobile App to CloudTest® (Developer or Mobile Administrator)

The following steps describe how to manually add a mobile app to CloudTest®. This is necessary in order to make the Appcelerator app for selection during test clip setup.

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. For example, KitchenSink.



Name	KitchenSink
Description	
Version	1.0
OS	iOS
Minimum OS Version	5.1
Supported Device Type	iPhone
Minimum Device Version	
URL Scheme	kitchensink
Icon	appicon.png

3. Optionally, enter a description and an app version number. Version number will generally match Appcelerator 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. Provide the URL Scheme. Without doing that, 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.
8. Click Save to create this mobile app object in CloudTest® .

Registering Your Device to Use TouchTest™

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

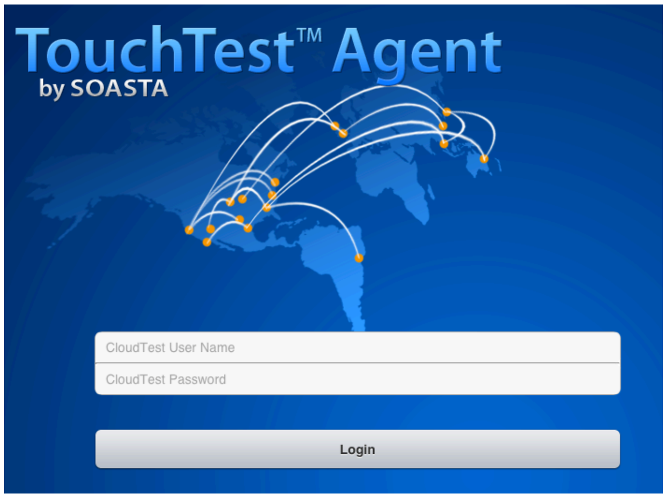
> **Note:** If you clear your cookies on the given mobile device after registration, you may need to register your device again so that TouchTest™ can recognize it. This does not consume an additional license.

1. On the mobile device, launch Safari and point it to:

   http://ctmobile.soasta.com/concerto/touchtest

   Note: For a Simulator, click the link provided.

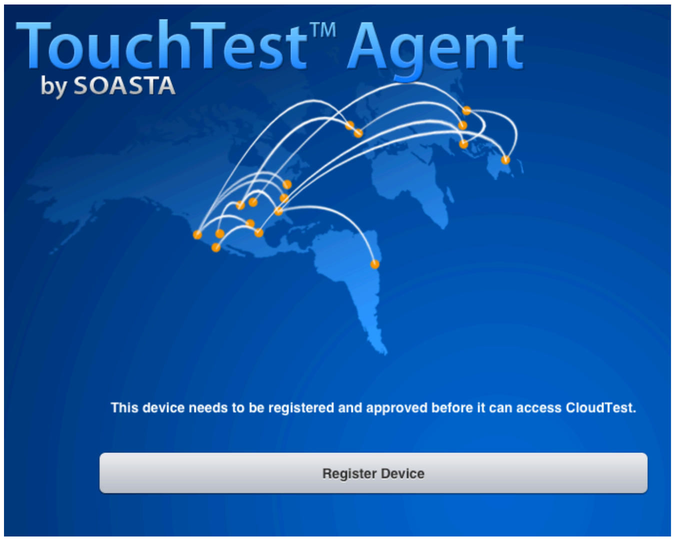
The following screen appears.



2. Login using your SOASTA CloudTest user name and password.

If the device is not registered, the Register Device page below appears.

Note: If you clear your cookies, you may need to register your device again so that TouchTest™ can recognize it. This does not consume an additional license.

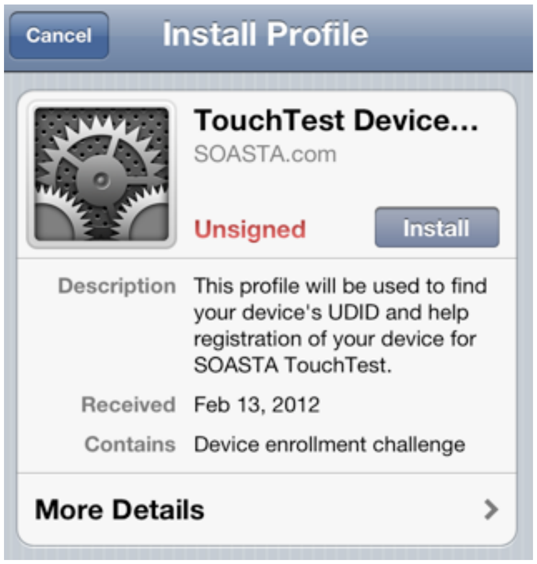


This device needs to be registered and approved before it can access CloudTest.

The Unique Device Identifier (UDID) will be used to register the mobile device for use with TouchTest™.

3. Click the Register Device button to continue.

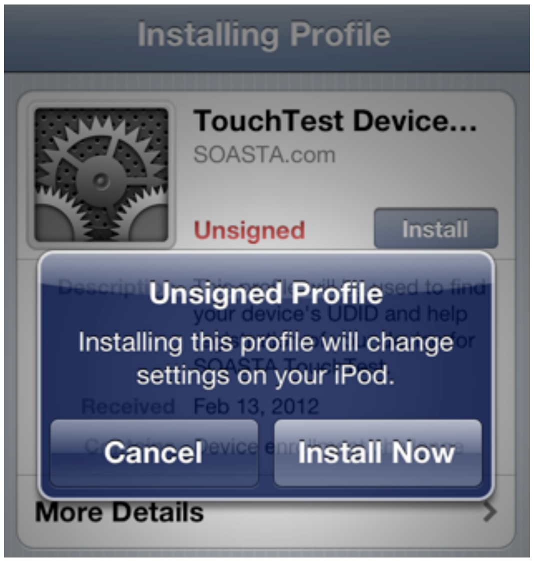
   a. First, the Install Profile screen appears. Click the Install button to proceed.



TouchTest Device...
SOASTA.com
Unsigned

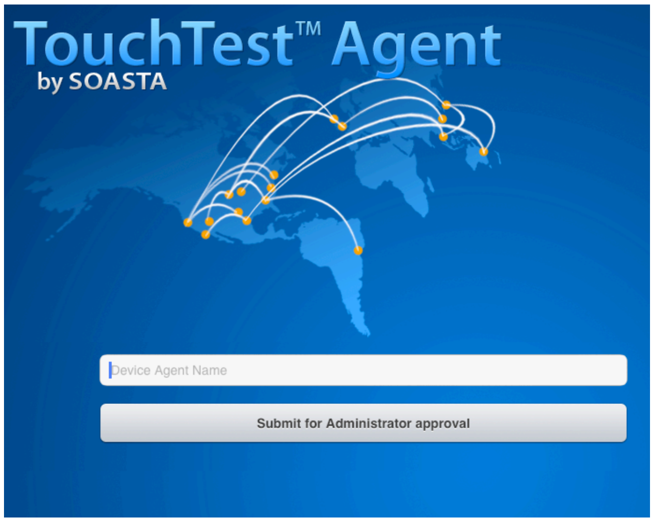
Description	This profile will be used to find your device's UDID and help registration of your device for SOASTA TouchTest.
Received	Feb 13, 2012
Contains	Device enrollment challenge

   b. The Unsigned 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 to authorize the profile installation.

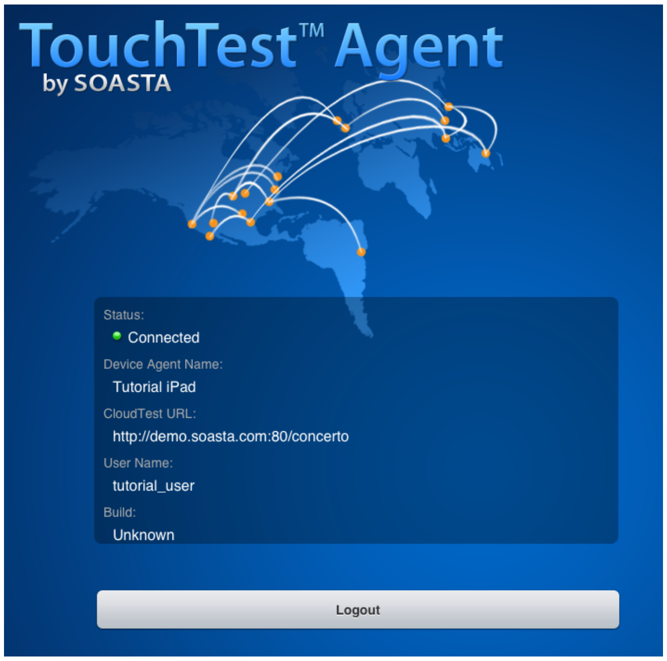
4. When prompted, give the TouchTest Agent a name. For example *Tester iPad*. Note that this name will be used throughout the product to refer to this device. Once entered the device name can only be changed by an Administrator.
5. Once this name is entered, click Submit for administrator approval.



Once the request for Administrator approval has been made, the TouchTest Agent will continue to poll CloudTest for approval.

Note: It is not necessary to keep the TouchTest Agent running while this approval is pending. The TouchTest Agent will resume polling for its approval once restarted.

If your device is approved by the Mobile Device Administrator, the Connected page will appear the first time TouchTest™ is launched in Safari on the approved device. On subsequent launches click Login to Connect and Logout to Disconnect.

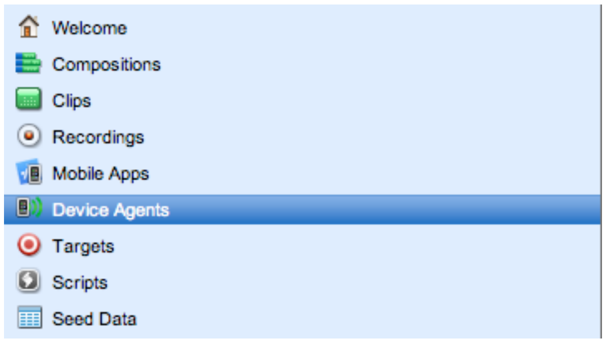


Status:
Connected
Device Agent Name:
Tutorial iPad
CloudTest URL:
http://demo.soasta.com:80/concerto
User Name:
tutorial_user
Build:
Unknown

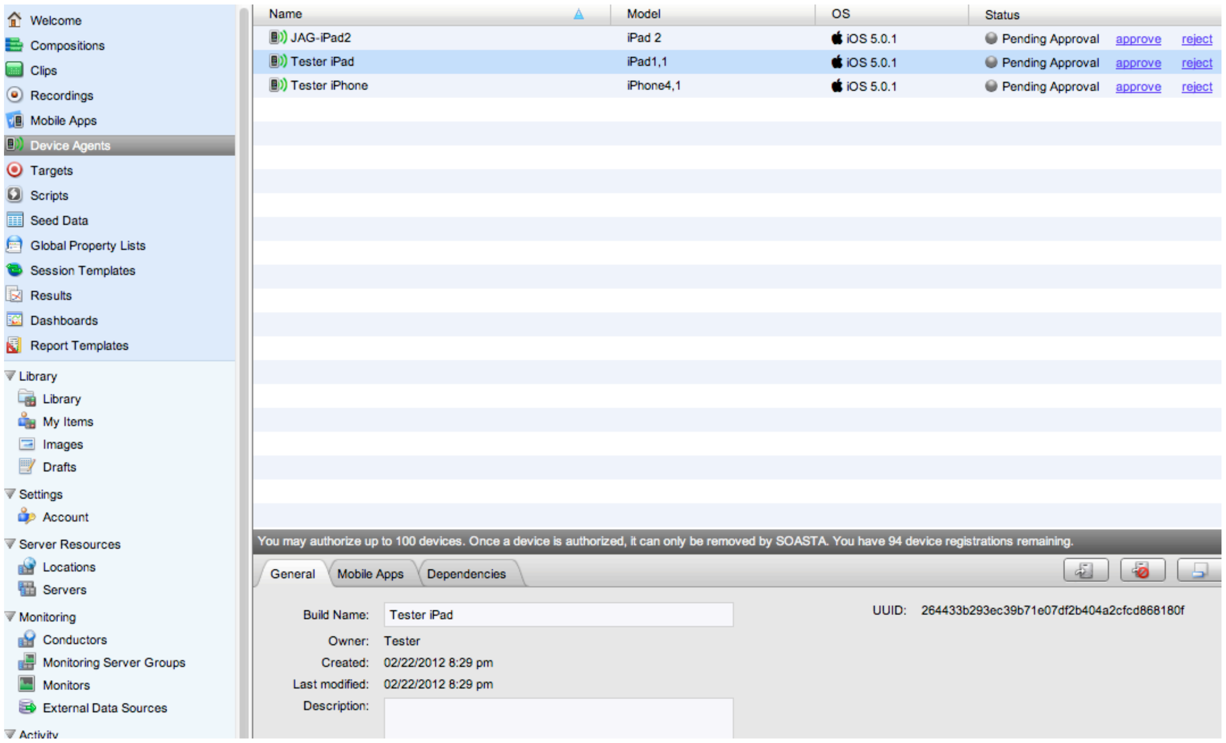
Approving a Mobile Device (Administrator Only)

The TouchTest™ Mobile Device Administrator has the responsibility to approve or reject the devices attempting to join testing. Administrators will use the following steps to approve/reject the devices attempting to join.

1. Login as the user with mobile device administrative rights.
2. Click Central > Device Agents



When you do so, the Device Agents list displays those devices in queue by name. Additionally, the model (iOS device type), OS (iOS version), and current status of the TouchTest™ agents are displayed in the list columns.



Name	Model	OS	Status		
JAG-iPad2	iPad 2	iOS 5.0.1	Pending Approval	approve	reject
Tester iPad	iPad1,1	iOS 5.0.1	Pending Approval	approve	reject
Tester iPhone	iPhone4,1	iOS 5.0.1	Pending Approval	approve	reject

You may authorize up to 100 devices. Once a device is authorized, it can only be removed by SOASTA. You have 94 device registrations remaining.

Build Name:	Tester iPad
Owner:	Tester
Created:	02/22/2012 8:29 pm
Last modified:	02/22/2012 8:29 pm
Description:	
UUID:	264433b293ec39b71e07df2b404a2cfcd868180f

Those devices that have the status Pending Approval need administrative attention:

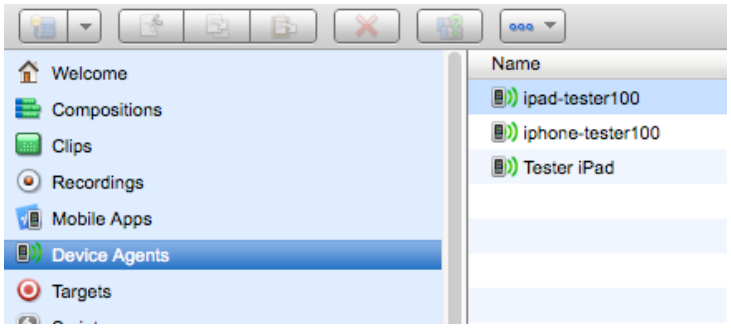
3. Click Approve to complete adding a device and Reject to deny its access.

Note: Once a device is approved it cannot be deleted without contacting SOASTA Support. CloudTest Lite users may approve their single device but will also need to contact SOASTA Support.

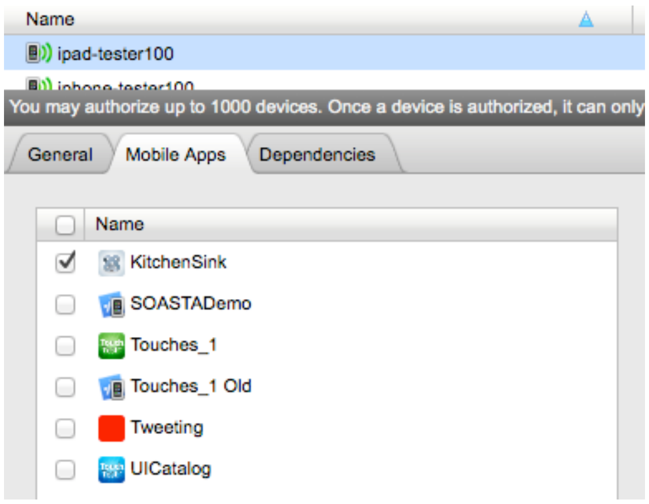
Associating Mobile Apps with a Device

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

1. In Central > Device Agents, 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 Moblile App(s) that you want to authorize this device to access. For example, KitchenSink.
4. Click Save on the lower panel toolbar.

Recording a TouchTest™ Scenario

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

- Create a new test clip within CloudTest®
- 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.

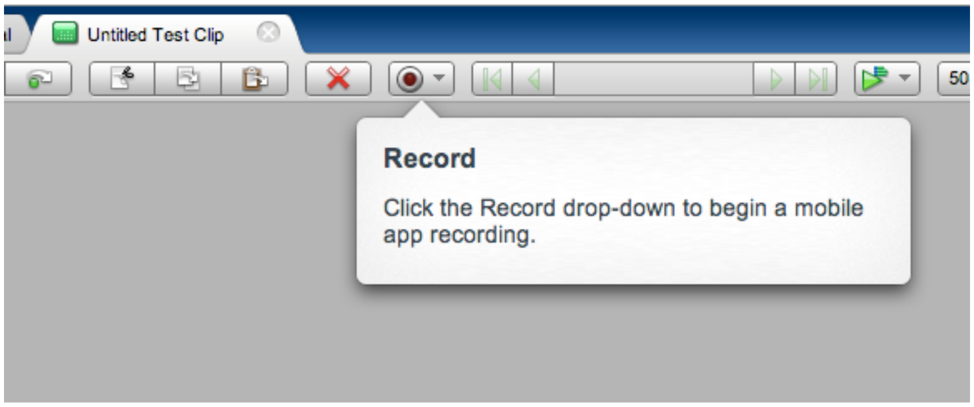
These and the following additional configuration steps are described in the remainder of this tutorial.

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

Create a Simple TouchTest™ Clip

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

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

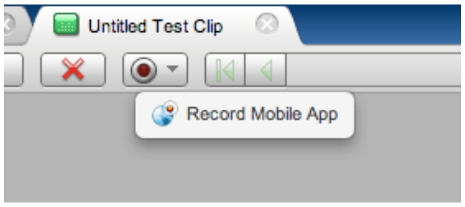


Record

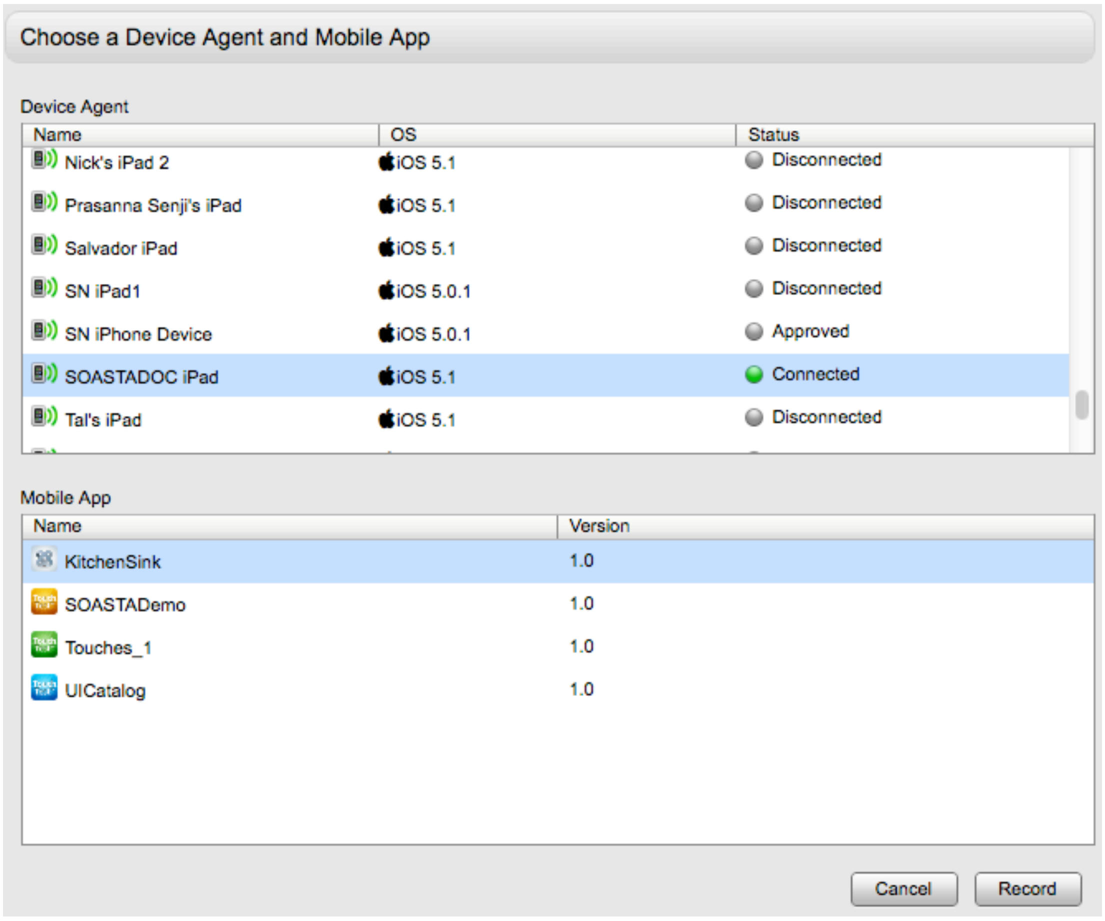
Click the Record drop-down to begin a mobile app recording.

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

2. Once ready, click the Record drop-down and then select Record Mobile App.



The Choose a Device Agent and Mobile App wizard appears.



Choose a Device Agent and Mobile App

Device Agent

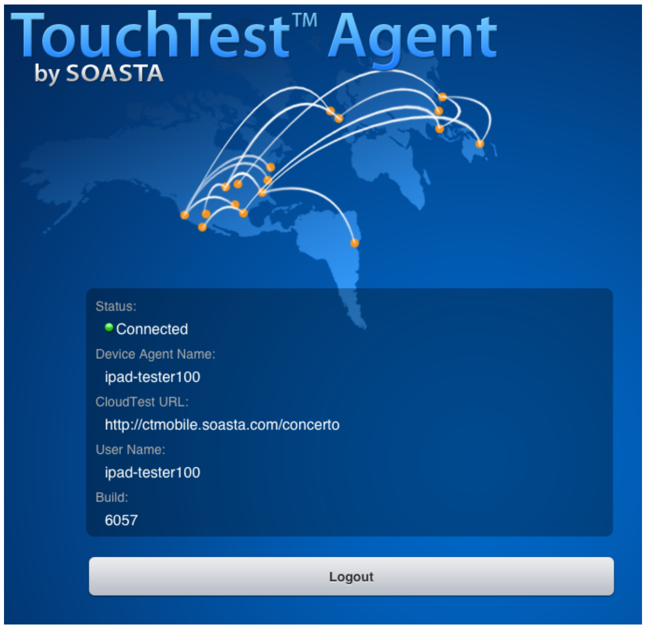
Name	OS	Status
Nick's iPad 2	iOS 5.1	Disconnected
Prasanna Senji's iPad	iOS 5.1	Disconnected
Salvador iPad	iOS 5.1	Disconnected
SN iPad1	iOS 5.0.1	Disconnected
SN iPhone Device	iOS 5.0.1	Approved
SOASTADOC iPad	iOS 5.1	Connected
Tal's iPad	iOS 5.1	Disconnected

Mobile App

Name	Version
KitchenSink	1.0
SOASTADemo	1.0
Touches_1	1.0
UICatalog	1.0

Note: If the Mobile Device Administrator has completed the steps above to associate one or more mobile apps with the device, those will appear in the Mobile App list whenever that device is selected.

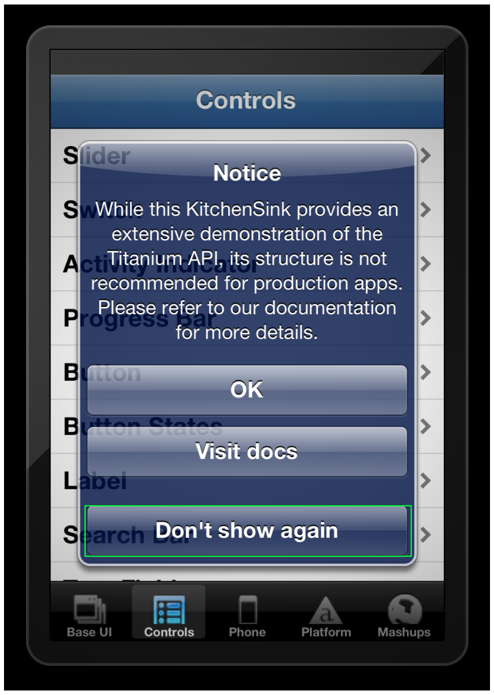
3. Select the TouchTest Agent that you created above and also select the mobile app.
4. Click the Record button in the wizard once your selection is made. TouchTest Agent will launch the selected app on the selected device.
5. The TouchTest Agent lauches in Safari on your mobile device.



Status:
Connected
Device Agent Name:
ipad-tester100
CloudTest URL:
http://ctmobile.soasta.com/concerto
User Name:
ipad-tester100
Build:
6057

Once successfully logged on, its Status will be *Connected*.

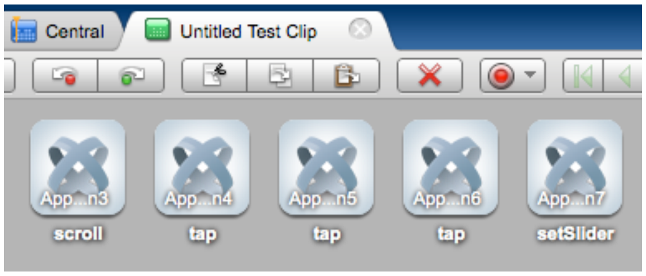
6. The mobile app launches to its initial screen. In our test, we pressed Don't show again knowing that since recording has already begun, this will cause an error on playback (unless the corresponding App Action is removed from the test clip during editing).



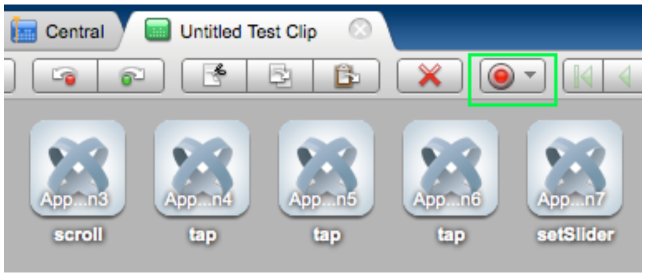
While this KitchenSink provides an extensive demonstration of the Titanium API, its structure is not recommended for production apps. Please refer to our documentation for more details.

7. Perform the planned mobile app user interactions on your mobile device.

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



8. Once the relevant interactions have been recorded, click the Record button again to stop the recording.



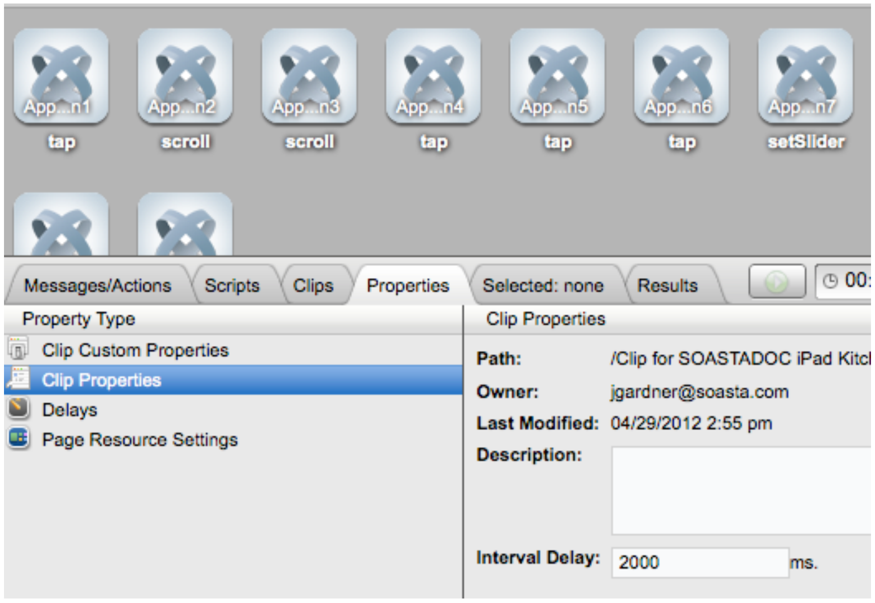
9. Click Save on the Clip Editor toolbar.

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 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.



Path: /Clip for SOASTADOC iPad Kitc

Owner: jgardner@soasta.com

Last Modified: 04/29/2012 2:55 pm

Description:

Interval Delay: 2000 ms.

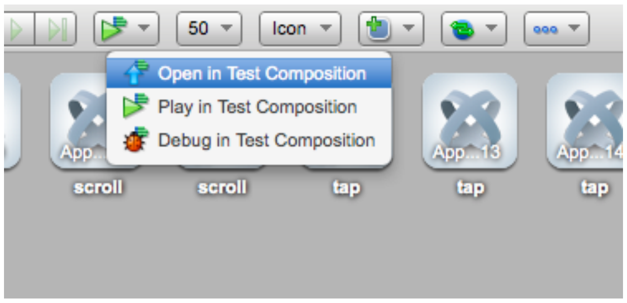
4. Click Save on the Clip Editor toolbar. When prompted, name the test clip.

TIP: It is possible to proceed to Create a Composition at any point after Recording is completed.

Create a Composition

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

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



- **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.

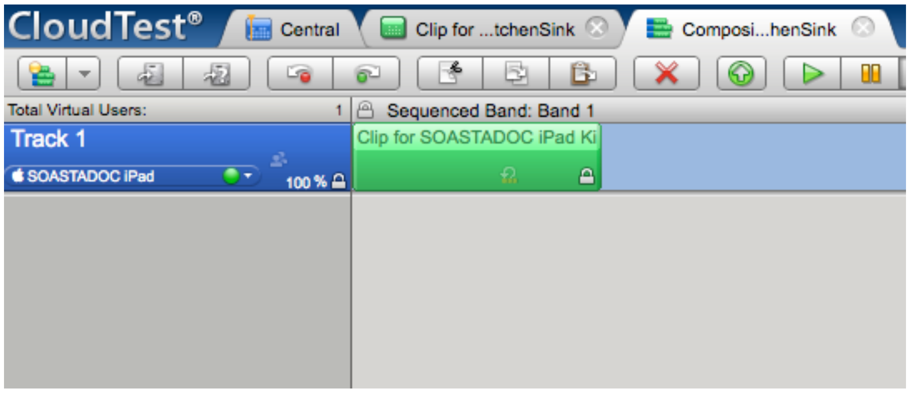
- **Play in Test Composition**

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**

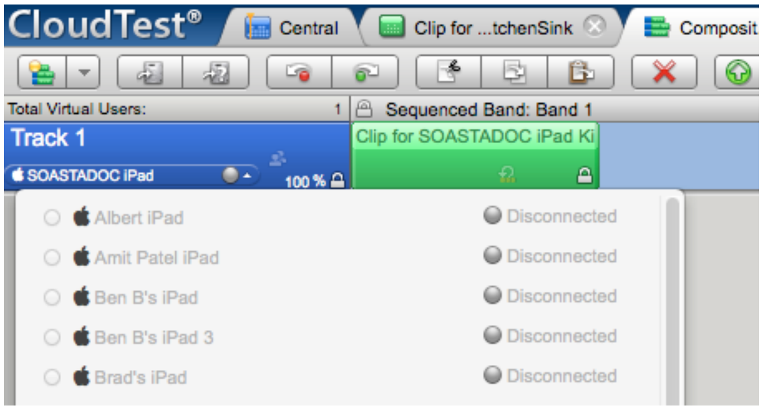
Choose Debug in Test Composition to add this clip to a new draft composition where it can be debugged in the Composition, Debugging tab before proceeding to edit parameters or play based on debug actions.

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



When opened from the Clip Editor, the device that was used to record the clip is automatically selected in the track (i.e. SOASTADOC iPad is selected in that field above). T

TIP: To playback this test composition on another device, click the drop-down and make another selection. This preference can also be set at the clip level.



2. Click Save on the Composition Editor toolbar and give the test composition a name or accept the default name.

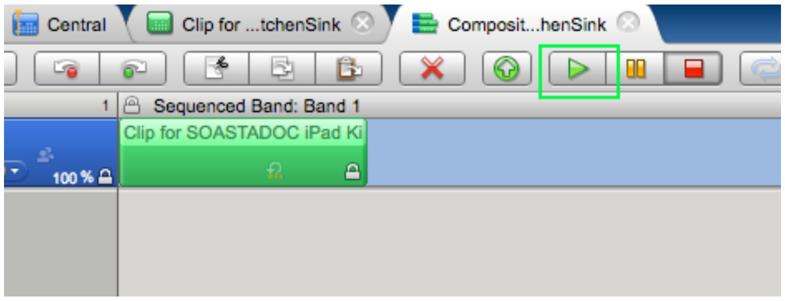
   For example, *Composition for KitchenSink*.

Playing a Composition

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

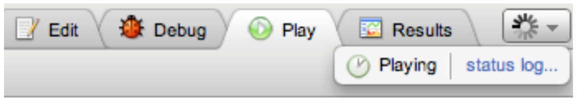
1. Ensure that the TouchTest Agent status is "Connected" on the mobile device via Safari.

2. In the Composition Editor, click Play to run the test composition. The mobile app actions performed when the clip was created are played back on the device.



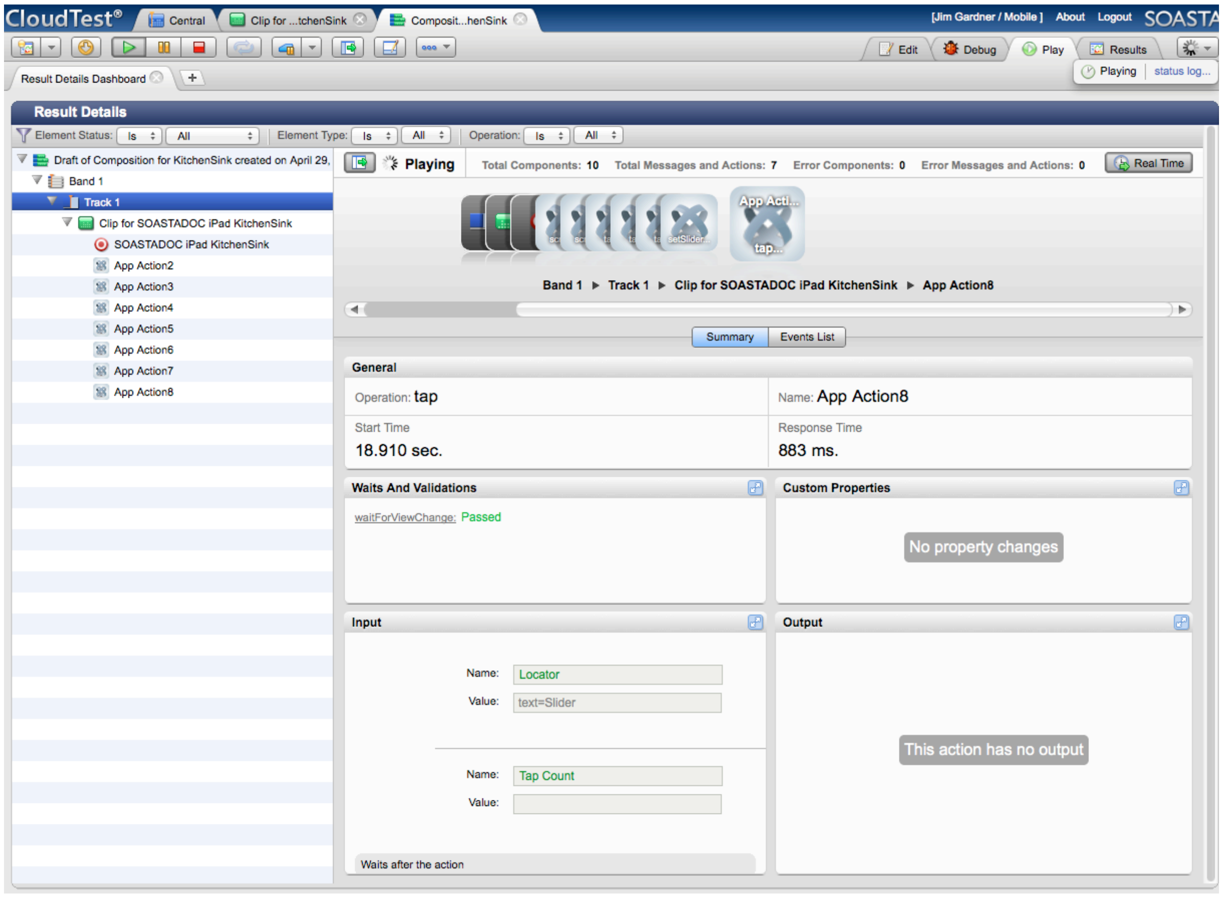
While the test runs, the Composition Editor automatically switches to the Play tab, and by default, the Result Details dashboard displays.

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



Result Details

The Result Details dashboard helps to discover the cause of errors in your test, if any.



Result Details

Element Status: Is All Element Type: Is All Operation: Is All

- Draft of Composition for KitchenSink created on April 29,
  - Band 1
    - Track 1
      - Clip for SOASTADOC iPad KitchenSink
        - SOASTADOC iPad KitchenSink
        - App Action2
        - App Action3
        - App Action4
        - App Action5
        - App Action6
        - App Action7
        - App Action8

Playing Total Components: 10 Total Messages and Actions: 7 Error Components: 0 Error Messages and Actions: 0 Real Time

Band 1 ▶ Track 1 ▶ Clip for SOASTADOC iPad KitchenSink ▶ App Action8

Summary | Events List

General

Operation: tap — Name: App Action8

Start Time 18.910 sec. — Response Time 883 ms.

Waits And Validations: waitForViewChange: Passed

Custom Properties: No property changes

Input

Name: Locator
Value: text=Slider

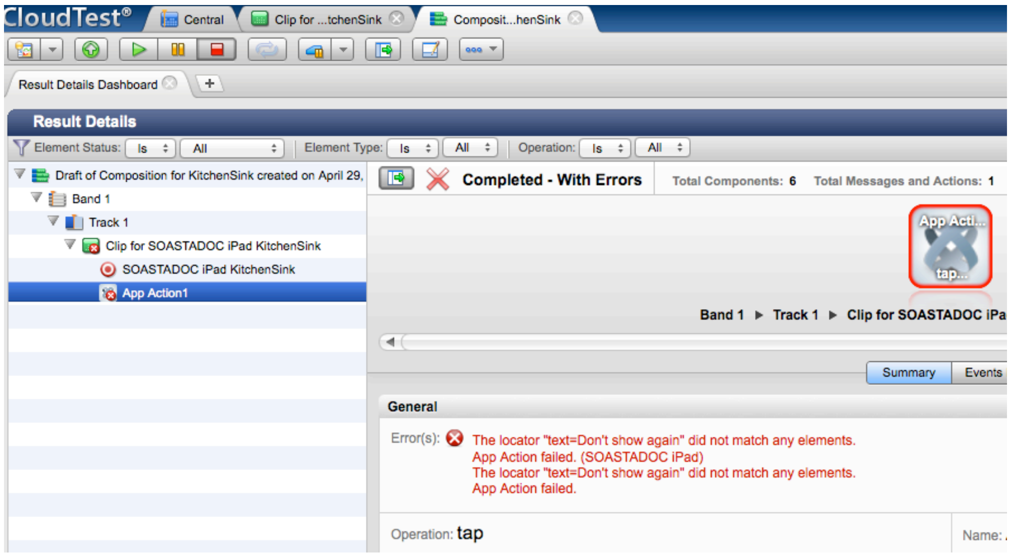
Name: Tap Count
Value:

Waits after the action

Output: This action has no output

While play continues results are posted in the Composition Editor, Play tab, Result Details widget.

If the composition fails for any reason, the Result Details dashboard clearly indicates that and presents details as to why.



Result Details

Element Status: Is All Element Type: Is All Operation: Is All

- Draft of Composition for KitchenSink created on April 29,
  - Band 1
    - Track 1
      - Clip for SOASTADOC iPad KitchenSink
        - SOASTADOC iPad KitchenSink
        - App Action1

Completed - With Errors Total Components: 6 Total Messages and Actions: 1

Band 1 ▶ Track 1 ▶ Clip for SOASTADOC iPa

Summary | Events

General

Error(s): The locator "text=Don't show again" did not match any elements.
App Action failed. (SOASTADOC iPad)
The locator "text=Don't show again" did not match any elements.
App Action failed.

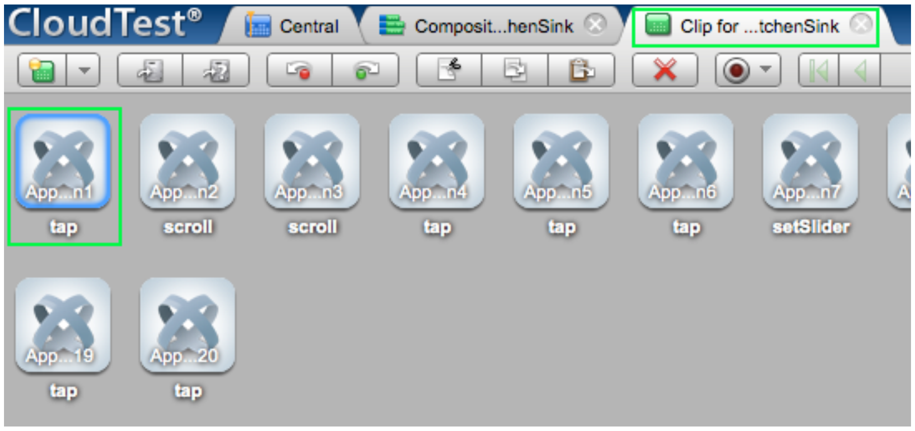
Operation: tap — Name:

The error above resulted because we pressed Don't show again in KitchenSink. And now, since the splash screen doesn't reappear on playback, an error results. In the next section, we'll do a quick fix to delete this item from the test clip.

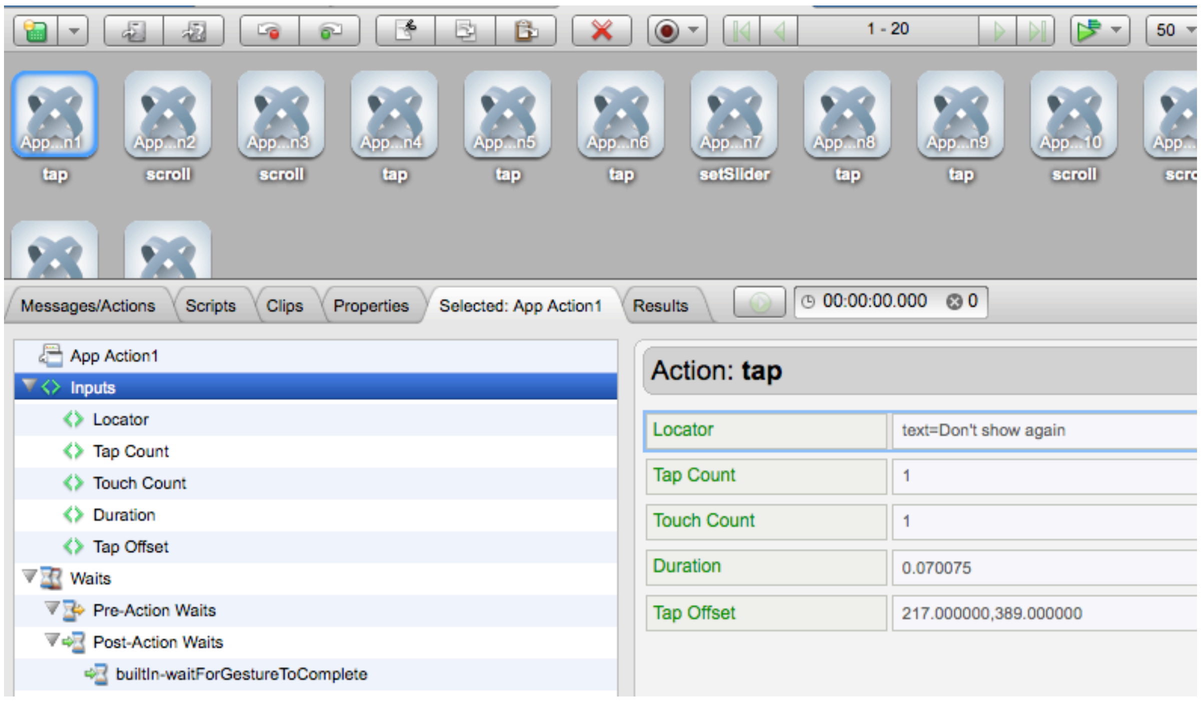
Double-Clicking Result Errors to Debug a Test

1. Double-click the App Action1 (or whichever composition element or clip element failed in a test).

The Clip Editor tab appears with the given clip element selected (in this case, the AppAction with the locator `text=Don't show again`).



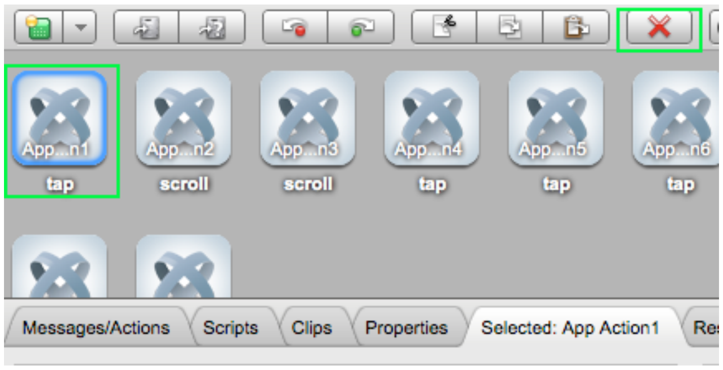
2. Double-click the selection in the Clip Editor to inspect its details.



Action: tap

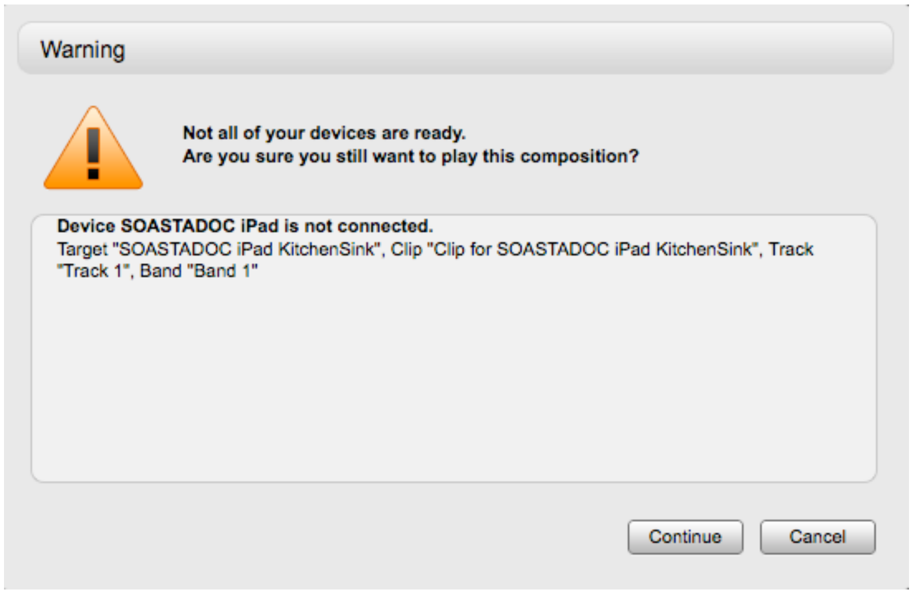
Locator	text=Don't show again
Tap Count	1
Touch Count	1
Duration	0.070075
Tap Offset	217.000000,389.000000

3. In this case, after verifying details about the app action in question, we now want to delete it (since the splash screen will no longer appear in the app). With the item still selected, click the Delete icon on the Clip Editor toolbar.



4. Save the test clip after making any changes.
5. Return to the test composition in the Composition Editor tab.
6. Click Play a second time.

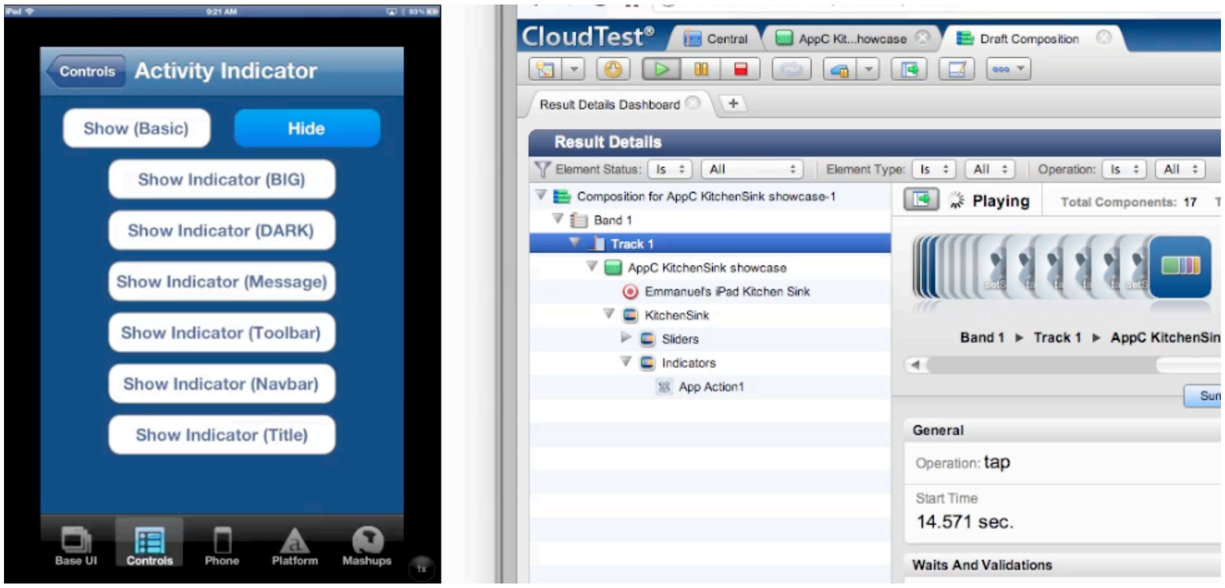
Note that if the iOS Device has gone to sleep during the above steps, a warning will appear to provide you an opportunity to resolve the device state.


Warning

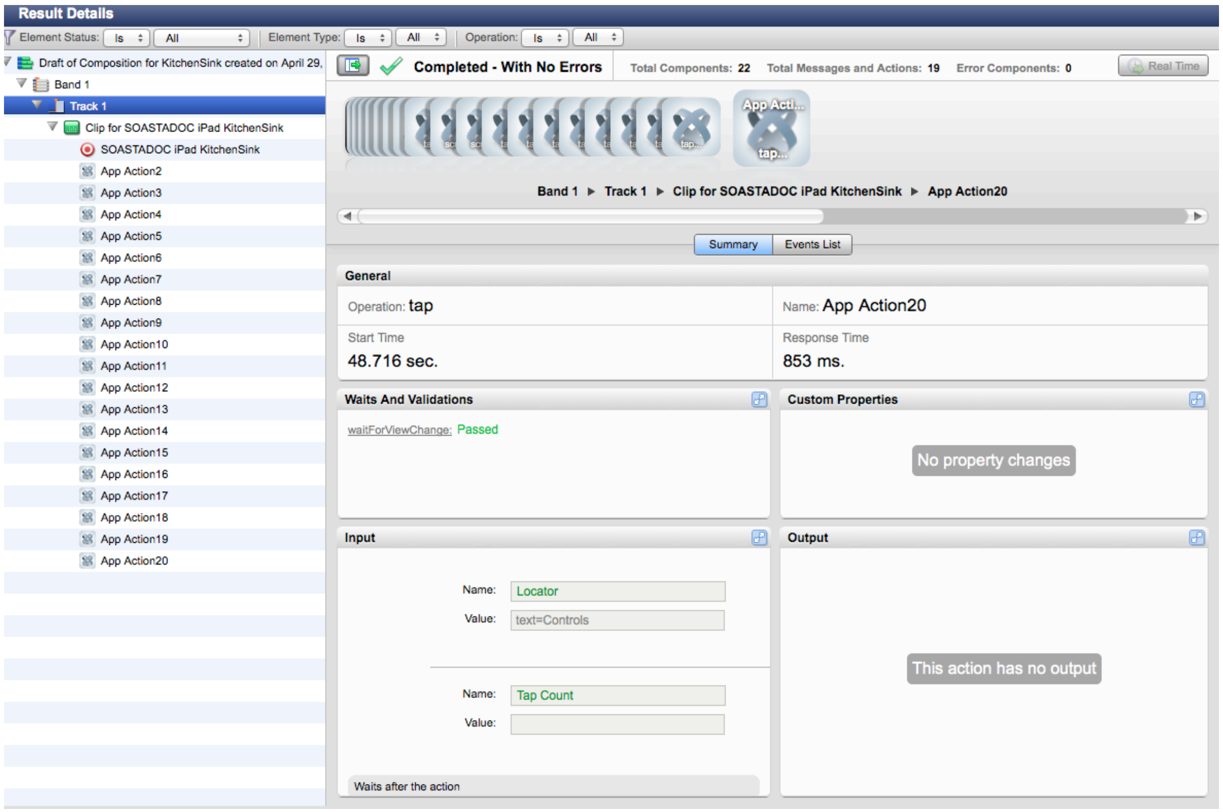
Not all of your devices are ready.
Are you sure you still want to play this composition?

Device SOASTADOC iPad is not connected.
Target "SOASTADOC iPad KitchenSink", Clip "Clip for SOASTADOC iPad KitchenSink", Track "Track 1", Band "Band 1"

The composition plays a second time; the screenshot below shows the runtime correlation between the device and the CloudTest desktop client.



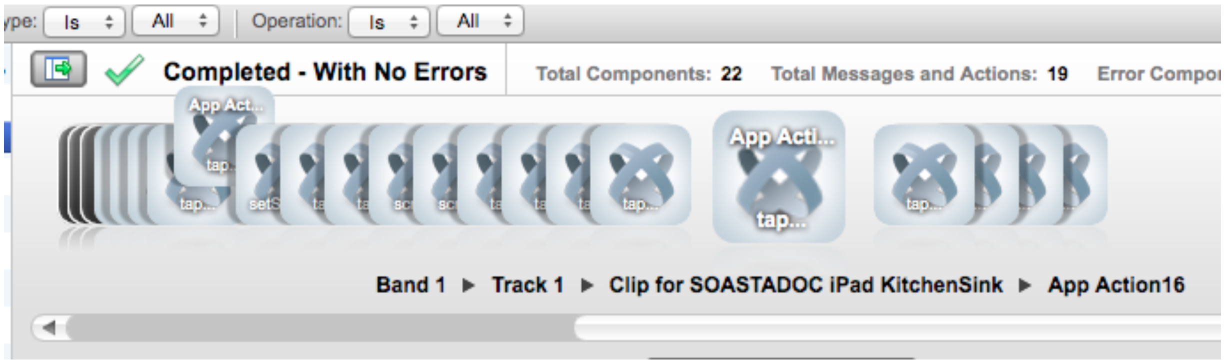
This time, play completes successfully and the final results are posted in the Results tab (also in the Result Details widget). If the test passed on all points, the status "Completed – With No Errors" is clearly posted in the Result Details dashboard.



Navigating Result Details

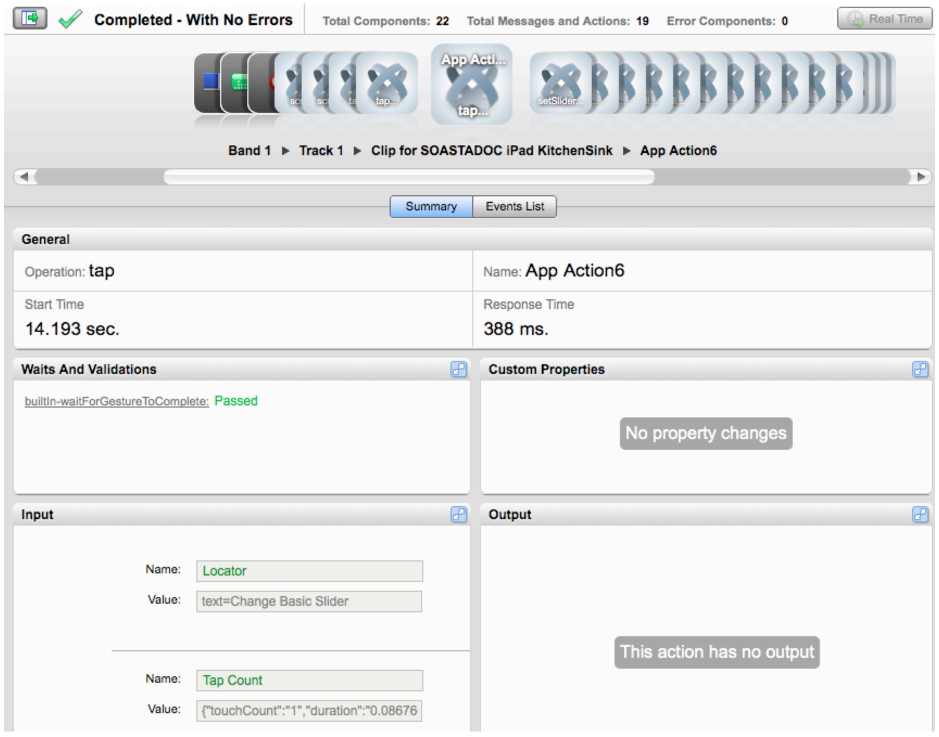
Click to expand the nodes in the Navigation Tree on the left as they appear.

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


ype: Is All Operation: Is All
Completed - With No Errors
Total Components: 22
Total Messages and Actions: 19
Error Compo
App Act...
tap...
App Acti...
Band 1 ▶ Track 1 ▶ Clip for SOASTADOC iPad KitchenSink ▶ App Action16

This stream is also shown in the tree as elements are executed during play. As play continues, the focus is set to the last executed element. 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.

- 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.


Completed - With No Errors
Total Components: 22
Total Messages and Actions: 19
Error Components: 0
Real Time
App Acti...
tap...
setSlider
Band 1 ▶ Track 1 ▶ Clip for SOASTADOC iPad KitchenSink ▶ App Action6
Summary
Events List

General

Operation: tap
Name: App Action6
Start Time
14.193 sec.
Response Time
388 ms.

Waits And Validations

builtIn-waitForGestureToComplete: Passed

Custom Properties

No property changes

Input

Name: Locator
Value: text=Change Basic Slider

Name: Tap Count
Value: {"touchCount":"1","duration":"0.08676

Output

This action has no output

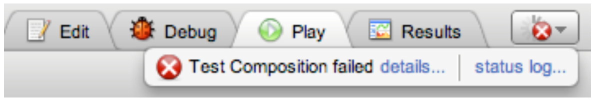
Identifying and Analyzing Common Errors

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

Network or Communication Errors

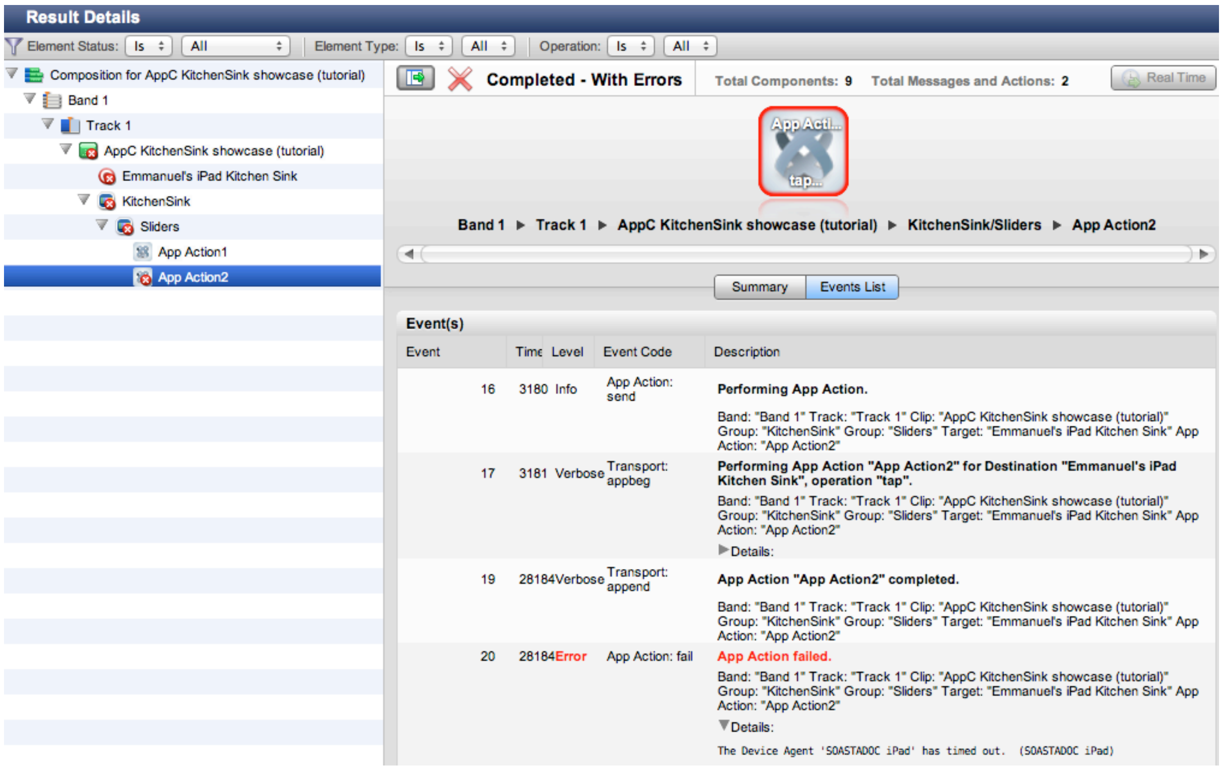
Initial errors in a simple test like the one above are also often simple network or configuration errors having to do with test staging. They frequently are related the state of the Device Agent (e.g. if the device agent is not connected when you click Play).

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.

In some cases, the TouchTest Agent may have been started but is no longer responding (such as was the case with the Device Agent timeout error shown beow). In such cases Logout and re-login to the TouchTest Agent.


Result Details
Element Status: Is All
Element Type: Is All
Operation: Is All
Composition for AppC KitchenSink showcase (tutorial)
Band 1
Track 1
AppC KitchenSink showcase (tutorial)
Emmanuel's iPad Kitchen Sink
KitchenSink
Sliders
App Action1
App Action2
Completed - With Errors
Total Components: 9
Total Messages and Actions: 2
Real Time
App Acti...
tap...
Band 1 ▶ Track 1 ▶ AppC KitchenSink showcase (tutorial) ▶ KitchenSink/Sliders ▶ App Action2
Summary
Events List
Event(s)

Event	Time	Level	Event Code	Description
16	3180	Info	App Action: send	**Performing App Action.** Band: "Band 1" Track: "Track 1" Clip: "AppC KitchenSink showcase (tutorial)" Group: "KitchenSink" Group: "Sliders" Target: "Emmanuel's iPad Kitchen Sink" App Action: "App Action2"
17	3181	Verbose	Transport: appbeg	**Performing App Action "App Action2" for Destination "Emmanuel's iPad Kitchen Sink", operation "tap".** Band: "Band 1" Track: "Track 1" Clip: "AppC KitchenSink showcase (tutorial)" Group: "KitchenSink" Group: "Sliders" Target: "Emmanuel's iPad Kitchen Sink" App Action: "App Action2" ▶Details:
19	28184	Verbose	Transport: append	**App Action "App Action2" completed.** Band: "Band 1" Track: "Track 1" Clip: "AppC KitchenSink showcase (tutorial)" Group: "KitchenSink" Group: "Sliders" Target: "Emmanuel's iPad Kitchen Sink" App Action: "App Action2"
20	28184	Error	App Action: fail	**App Action failed.** Band: "Band 1" Track: "Track 1" Clip: "AppC KitchenSink showcase (tutorial)" Group: "KitchenSink" Group: "Sliders" Target: "Emmanuel's iPad Kitchen Sink" App Action: "App Action2" ▼Details: The Device Agent 'SOASTADOC iPad' has timed out. (SOASTADOC iPad)

App Action and Other Errors

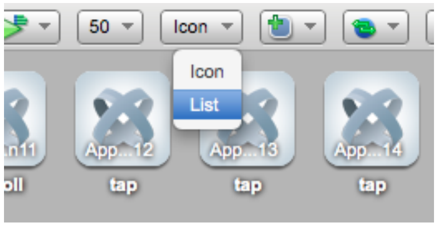
CloudTest reports all failures and marks test successful or unsuccessful by the Failure Actions set within it. Failure Actions are set stringently by default to fail the test for any error and show that failure in red.

Result Details clearly indicates the type of test failure that has occurred in a given case. Failures on specific app actions you recorded are easily distinguished by the red “X” in the Navigation Tree and when the error item is selected in Result Details.

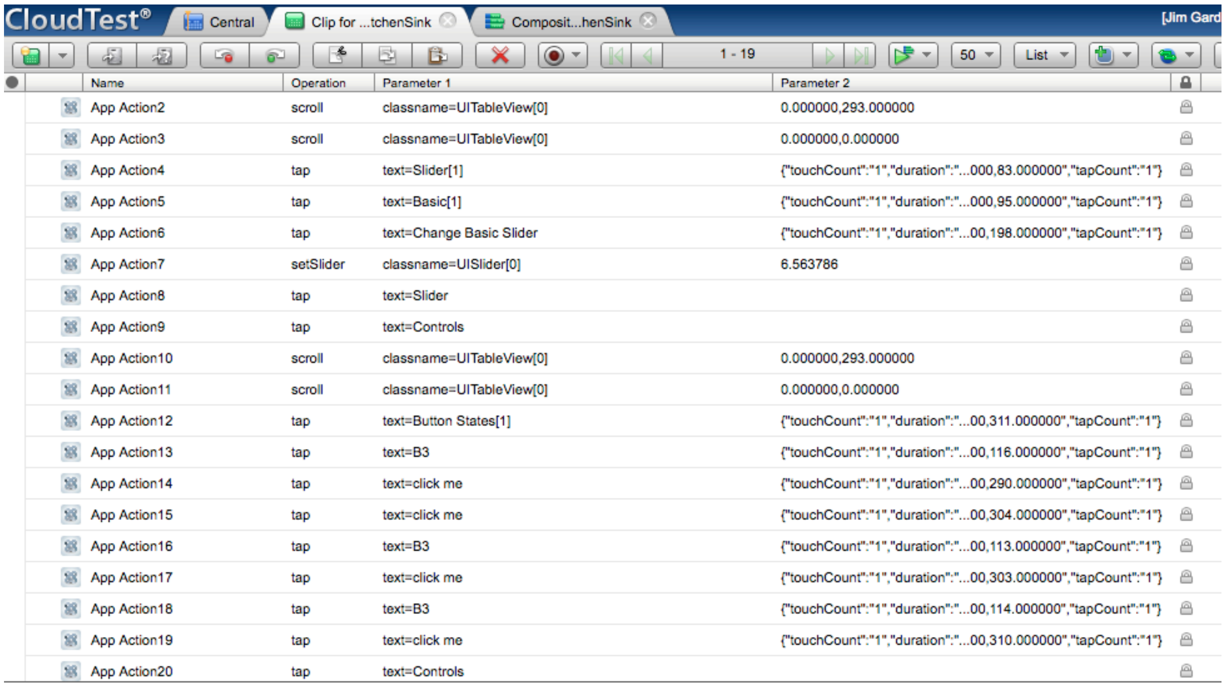
Advanced Clip Editing

Now that we’ve played this simple test composition successfully, and even fixed a simple error while learning how CloudTest checks 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.

1. 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.
2. Once the Clip Editor tab is in view, click the drop down Icon/List button on the toolbar and then select List.



The List view is useful while clip editing, because it shows all the parameters (8th column to the right) and their corresponding inputs in one tabular view.



Name	Operation	Parameter 1	Parameter 2
App Action2	scroll	classname=UITableView[0]	0.000000,293.000000
App Action3	scroll	classname=UITableView[0]	0.000000,0.000000
App Action4	tap	text=Slider[1]	{"touchCount":"1","duration":"...000,83.000000","tapCount":"1"}
App Action5	tap	text=Basic[1]	{"touchCount":"1","duration":"...000,95.000000","tapCount":"1"}
App Action6	tap	text=Change Basic Slider	{"touchCount":"1","duration":"...00,198.000000","tapCount":"1"}
App Action7	setSlider	classname=UISlider[0]	6.563786
App Action8	tap	text=Slider	
App Action9	tap	text=Controls	
App Action10	scroll	classname=UITableView[0]	0.000000,293.000000
App Action11	scroll	classname=UITableView[0]	0.000000,0.000000
App Action12	tap	text=Button States[1]	{"touchCount":"1","duration":"...00,311.000000","tapCount":"1"}
App Action13	tap	text=B3	{"touchCount":"1","duration":"...00,116.000000","tapCount":"1"}
App Action14	tap	text=click me	{"touchCount":"1","duration":"...00,290.000000","tapCount":"1"}
App Action15	tap	text=click me	{"touchCount":"1","duration":"...00,304.000000","tapCount":"1"}
App Action16	tap	text=B3	{"touchCount":"1","duration":"...00,113.000000","tapCount":"1"}
App Action17	tap	text=click me	{"touchCount":"1","duration":"...00,303.000000","tapCount":"1"}
App Action18	tap	text=B3	{"touchCount":"1","duration":"...00,114.000000","tapCount":"1"}
App Action19	tap	text=click me	{"touchCount":"1","duration":"...00,310.000000","tapCount":"1"}
App Action20	tap	text=Controls	

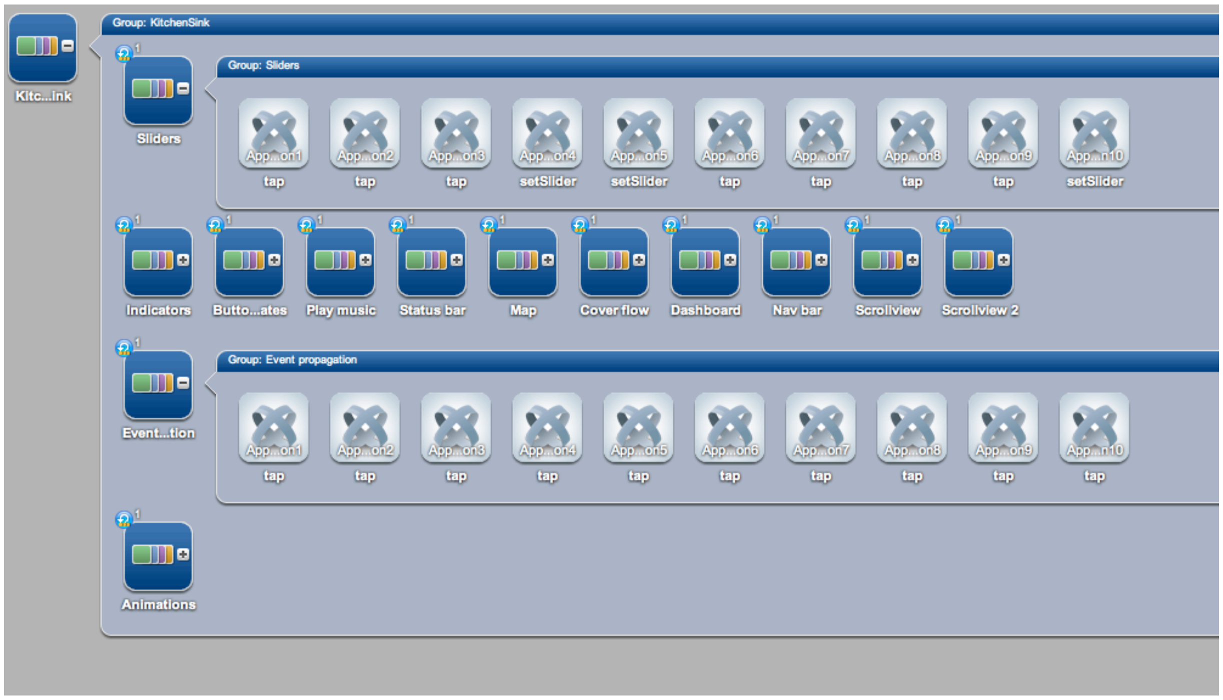
When additional paremeters are present they are displayed to the right of the Parameter 1 column.

Planning a Complex Test in KitchenSink

Your first clip using the sample app (or your own app) was probably relatively simple. CloudTest clips of great complexity can also be devised.

For example, in the sample clip shown below, we recorded actions from all of the following KitchenSink sections. After which, we used the Clip Editor to group all the like actions together using the right-click Add to Group command. These “like” actions correspond to the sections in the sample app.

- Sliders
- Indicators
- Button States
- Play music
- Status bar
- Map
- Cover flow

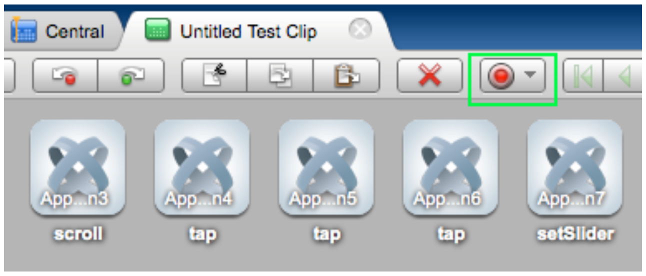


- Dashboard
- Nav bar
- Scroll view
- Event propagation and Animations

After all of the groups above were created they were then reselected and repeats were added at the group-level (this can be done on the clip surface in Icon view or by using the right-click Add Repeat command). Finally, they were placed into a master group called KitchenSink (once again by selecting them all and choosing the Add to Group command).

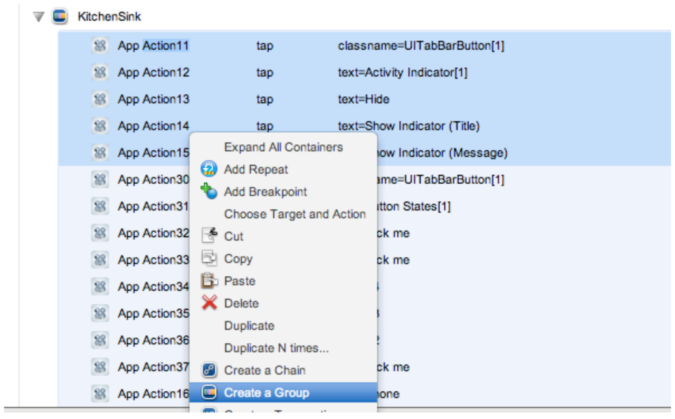
Increasing Test Clip Complexity

1. Perform the planned mobile app user interactions on your mobile device as before.
2. Once the relevant interactions have been recorded, click the Record button again to stop the recording.



3. Click Save on the Clip Editor toolbar.
4. To add groups or other containers, first identify and select the relevant actions, then right-click and choose the relevant command (i.e., Add to Group). The advantage of using collections such as groups is primarily to make results easier to follow. Additionally, items in a group can share properties such as repeats.

   In the example below, we identified and selected all the app actions created via the KitchenSink, Indicators section, then using the right-click Action menu we chose the Add to Group command.


KitchenSink

App Action11	tap	classname=UITabBarButton[1]
App Action12	tap	text=Activity Indicator[1]
App Action13	tap	text=Hide
App Action14	tap	text=Show Indicator (Title)
App Action15		ow Indicator (Message)
App Action30		ame=UITabBarButton[1]
App Action31		tton States[1]
App Action32		ck me
App Action33		ck me
App Action34		
App Action35		
App Action36		
App Action37		ck me
App Action16		one

Repeat this command for each selection. It is also possible to record each section, halt recording, organize those app actions into a group, and then restart recording from that point.

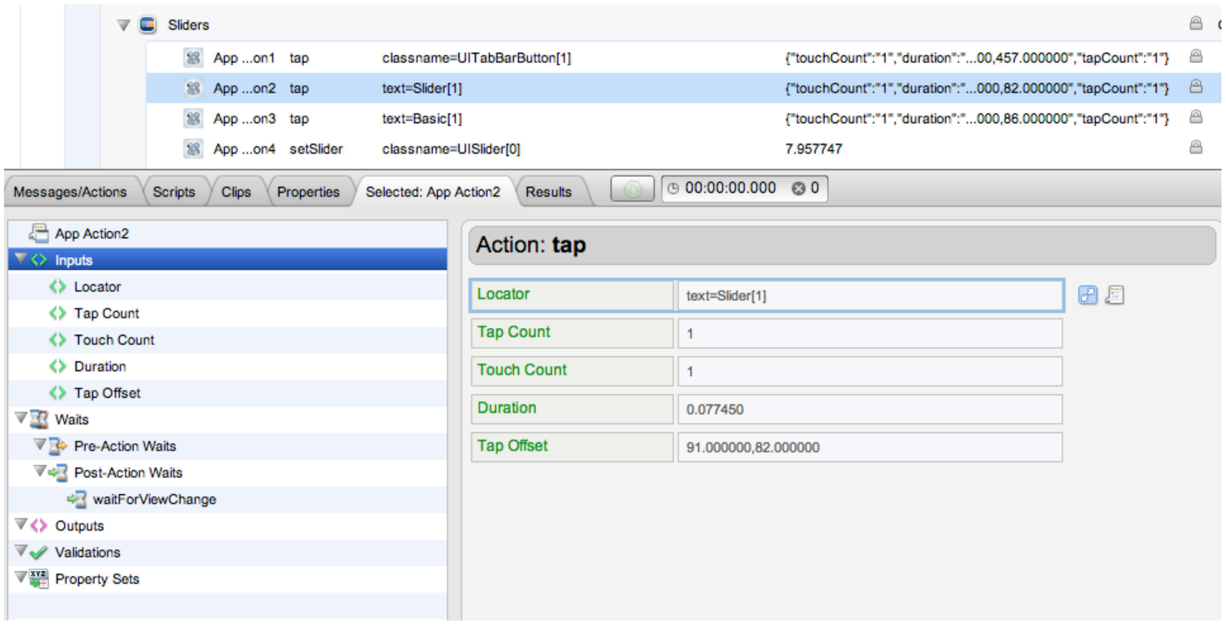
Inspecting App Action Details

Examine elements and properties for any App Action by selecting it in the workspace above and then double-clicking it. When you do so, the <*Selected: Action Name*> tab displays the contents of the selection in its own pane.

In the test clip below the recorded AppAction2 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 double-click the first app action in the clip that has a text parameter.

   In our sample clip, this is *Selected: AppAction2*. This app action has five Inputs: Locator, Tap Count, Touch Count, Duration, and Tap Offset.



Sliders

App ...on1	tap	classname=UITabBarButton[1]	{"touchCount":"1","duration":"...00,457.000000","tapCount":"1"}
App ...on2	tap	text=Slider[1]	{"touchCount":"1","duration":"...000,82.000000","tapCount":"1"}
App ...on3	tap	text=Basic[1]	{"touchCount":"1","duration":"...000,86.000000","tapCount":"1"}
App ...on4	setSlider	classname=UISlider[0]	7.957747

Action: **tap**

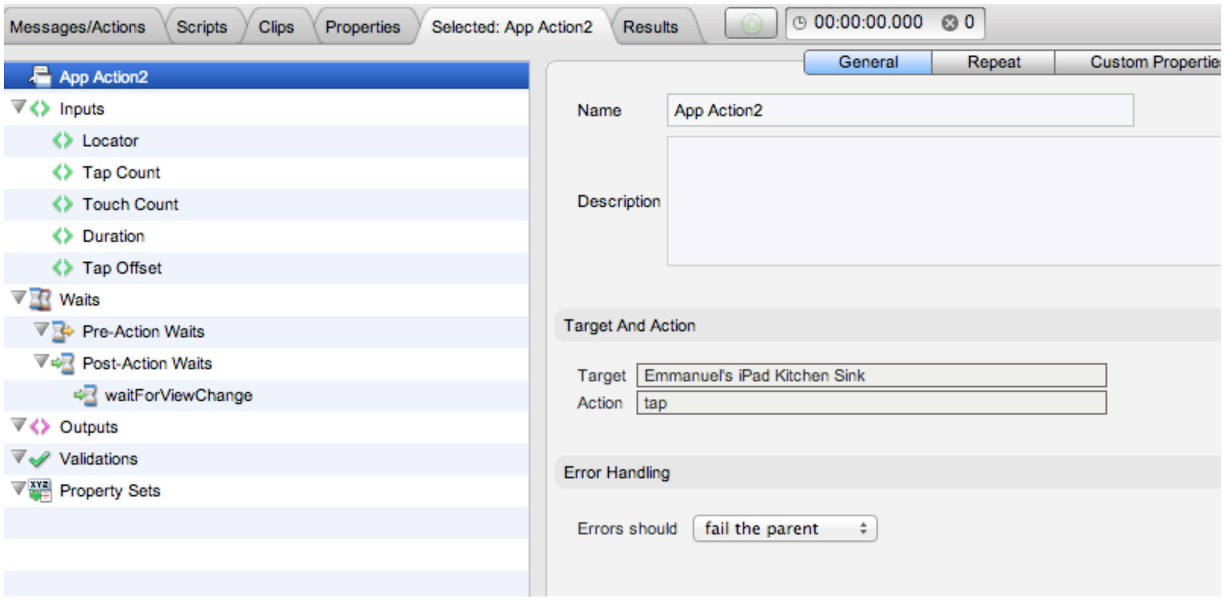
Locator	text=Slider[1]
Tap Count	1
Touch Count	1
Duration	0.077450
Tap Offset	91.000000,82.000000

App Action Elements and Properties

While the *Selected: Action* tab active, the Action level properties are shown in the tree on the left.

1. 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.



App Action2
- Inputs
  - Locator
  - Tap Count
  - Touch Count
  - Duration
  - Tap Offset
- Waits
  - Pre-Action Waits
  - Post-Action Waits
    - waitForViewChange
- Outputs
- Validations
- Property Sets

Name: App Action2
Description:

Target And Action
Target: Emmanuel's iPad Kitchen Sink
Action: tap

Error Handling
Errors should: fail the parent

•Other settings, including Waits, Inputs, Outputs, Validations, and Property Sets can be set by clicking that node in the tree and then performing the desired action on the right.

1.In the *Selected: AppAction2* 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 some content or event occurred as expected and have a corresponding Failure Action. App Action validations can range from simple true/false conditions to more complex conditions. 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*

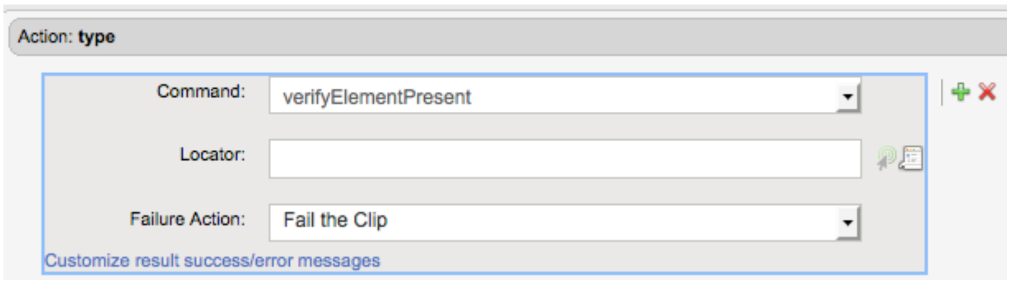
    Global properties are defined within the Central > Global Properties List and are “global” within the entire SOASTA CloudTest environment—and can be used across compositions.

Adding a Text Validation

Next, we will add a validation on AppAction2 (or the app action whose operation is *type* and whose locator is *placeholder=Username* in the clip you recorded from the SOASTA Demo app). 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.

1. Note the content of the given app action. For example in AppAction2 of our sample clip, Locator had the value *text=Slider[1].*
2. If it's not already open in the lower panel, open it now by double-clicking.
3. Click Validations in the list and then click the green Plus sign ➕ in the Validations panel on the right.

   An Action: type form is added to the right panel.



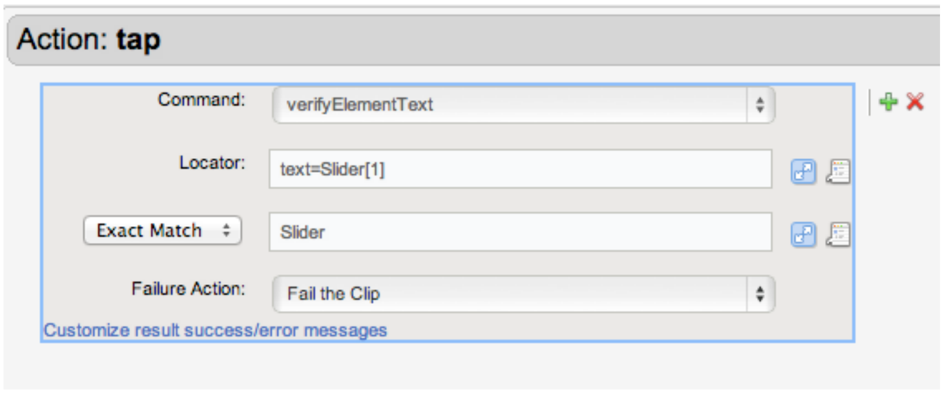
Action: **type**

Command:	verifyElementPresent
Locator:	
Failure Action:	Fail the Clip

Customize result success/error messages

4. 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 field.
5. In the Locator field, enter the field name shown above. For example, *text=Slider[1]*.
6. In the Match field, accept *Exact Match* and enter the username. For example, *soastadoc*.
7. Leave the default *Fail the Clip* set in the Failure action field.



Action: **tap**

Command:	verifyElementText
Locator:	text=Slider[1]
Exact Match	Slider
Failure Action:	Fail the Clip

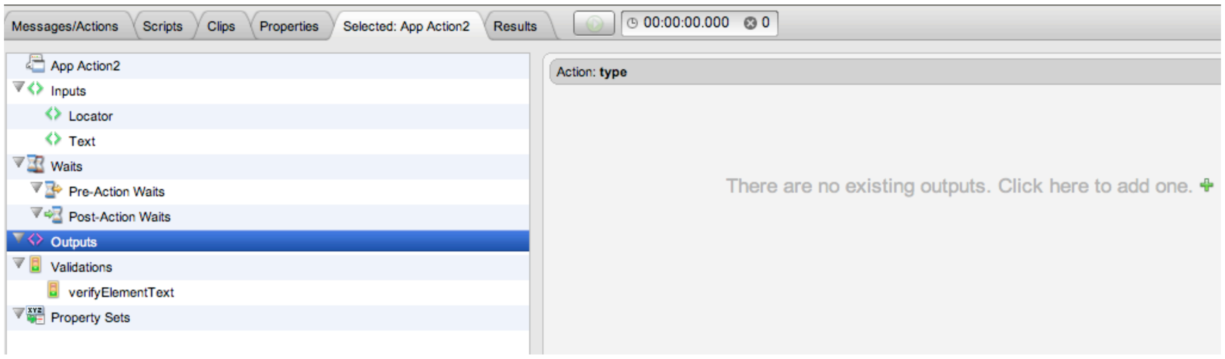
Customize result success/error messages

8. 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.

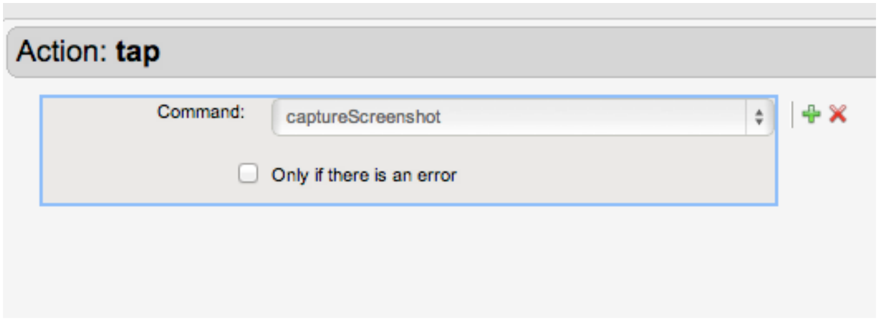
1. Once again, select the Action to display its properties in the sub-pane below.



There are no existing outputs. Click here to add one.

1. Click Outputs in the list on the left, and then in the *Action: type* field just to the right, click the green Plus sign again. 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 scroll to the top of the list and select *captureScreenshot*.


Action: tap

Command: captureScreenshot

Only if there is an error

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.

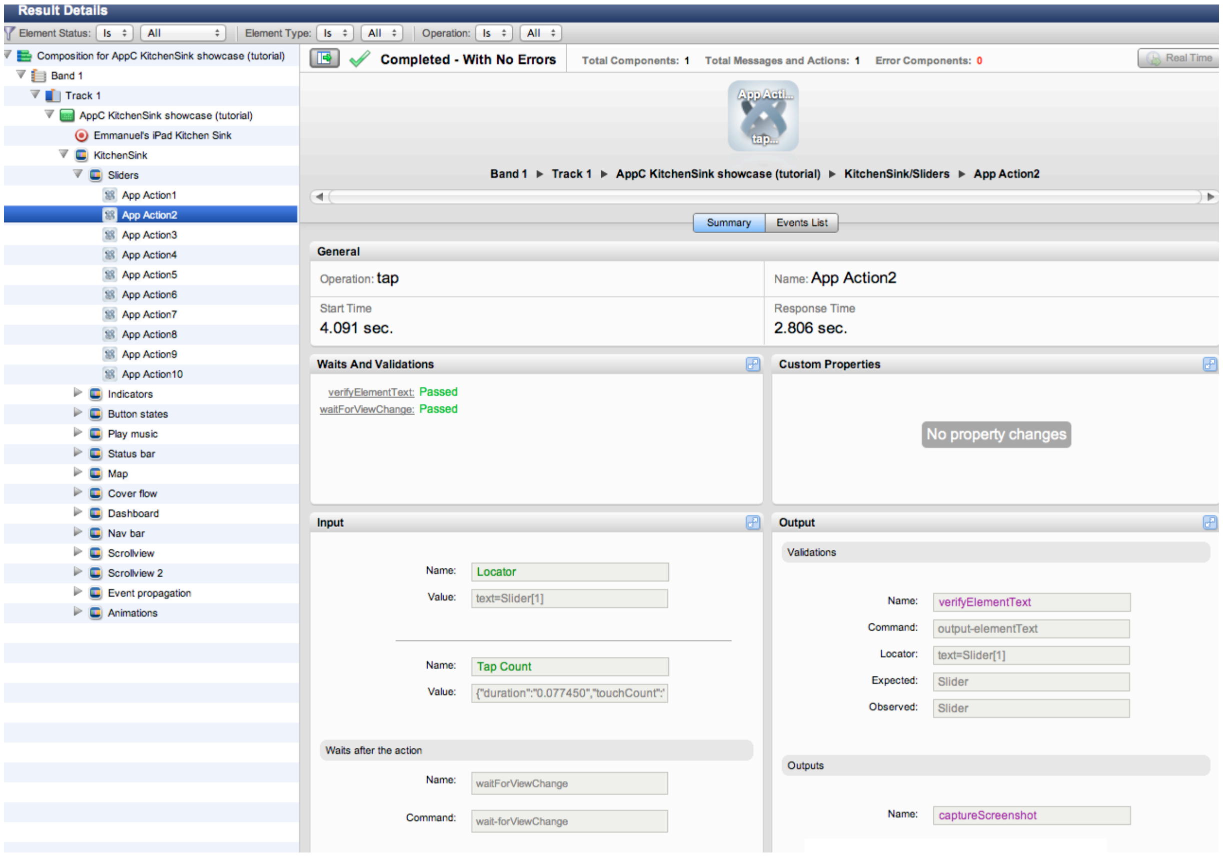
   In the following section, we'll inspect results for the validations and output that were set on AppAction2 above.

Analyzing Results

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

In the best case scenario, the parameters added in Advanced Clip Editing work without a hitch. We can easily verify the status of our parameteres in Result Details.

- Expand the Navigation Tree until AppAction2 is in display and then select it as shown below.


Result Details

Element Status: Is All | Element Type: Is All | Operation: Is All

- Composition for AppC KitchenSink showcase (tutorial)
  - Band 1
    - Track 1
      - AppC KitchenSink showcase (tutorial)
        - Emmanuel's iPad Kitchen Sink
        - KitchenSink
          - Sliders
            - App Action1
            - App Action2
            - App Action3
            - App Action4
            - App Action5
            - App Action6
            - App Action7
            - App Action8
            - App Action9
            - App Action10
          - Indicators
          - Button states
          - Play music
          - Status bar
          - Map
          - Cover flow
          - Dashboard
          - Nav bar
          - Scrollview
          - Scrollview 2
          - Event propagation
          - Animations

Completed - With No Errors | Total Components: 1 | Total Messages and Actions: 1 | Error Components: 0

Band 1 ▶ Track 1 ▶ AppC KitchenSink showcase (tutorial) ▶ KitchenSink/Sliders ▶ App Action2

General

Operation: tap	Name: App Action2
Start Time 4.091 sec.	Response Time 2.806 sec.

Waits And Validations

verifyElementText: Passed
waitForViewChange: Passed

Custom Properties

No property changes

Input

Name: Locator
Value: text=Slider[1]

Name: Tap Count
Value: {"duration":"0.077450","touchCount":

Waits after the action

Name: waitForViewChange
Command: wait-forViewChange

Output

Validations

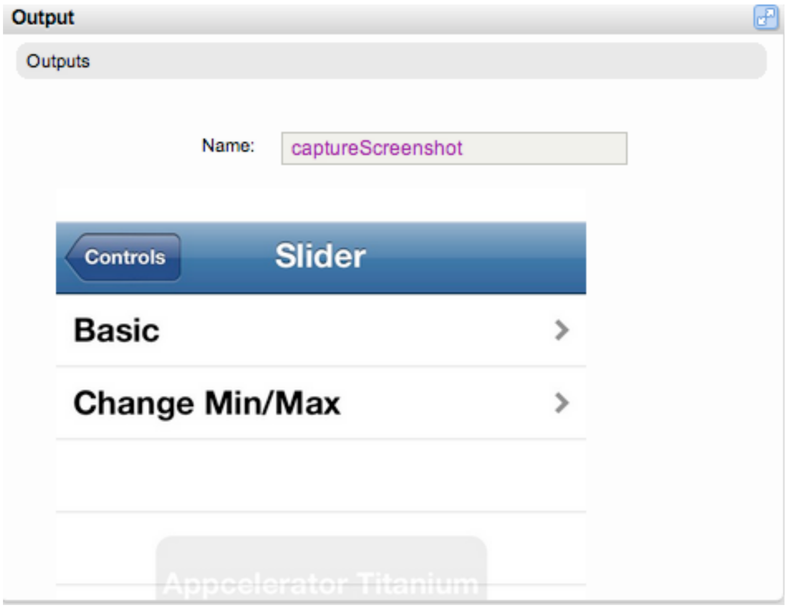
Name: verifyElementText
Command: output-elementText
Locator: text=Slider[1]
Expected: Slider
Observed: Slider

Outputs

Name: captureScreenshot

1. To inspect the latest result for the test, open the result in the Composition Editor > Results tab (if it is not already open.
2. Select the clip element that had the validation. For example, AppAction2 (as shown above).
3. Inspect the information on the Summary tab for the selection. In the result above, both the validations on AppAction2 passed in the result shown.

4. Scroll down, if necessary, to view the *captureScreenshot* output defined for AppAction2 in our example.



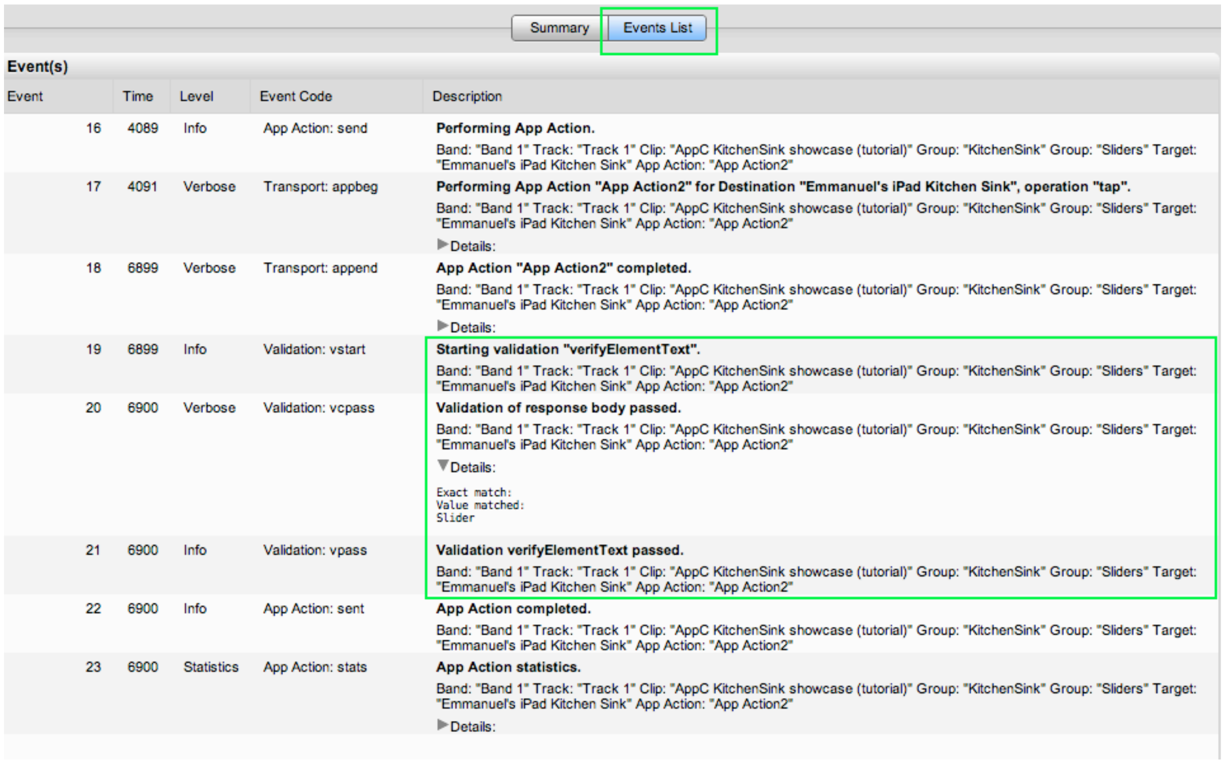
Output

Outputs

Name: captureScreenshot

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.

5. 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
16	4089	Info	App Action: send	**Performing App Action.** Band: "Band 1" Track: "Track 1" Clip: "AppC KitchenSink showcase (tutorial)" Group: "KitchenSink" Group: "Sliders" Target: "Emmanuel's iPad Kitchen Sink" App Action: "App Action2"
17	4091	Verbose	Transport: appbeg	**Performing App Action "App Action2" for Destination "Emmanuel's iPad Kitchen Sink", operation "tap".** Band: "Band 1" Track: "Track 1" Clip: "AppC KitchenSink showcase (tutorial)" Group: "KitchenSink" Group: "Sliders" Target: "Emmanuel's iPad Kitchen Sink" App Action: "App Action2" ▶Details:
18	6899	Verbose	Transport: append	**App Action "App Action2" completed.** Band: "Band 1" Track: "Track 1" Clip: "AppC KitchenSink showcase (tutorial)" Group: "KitchenSink" Group: "Sliders" Target: "Emmanuel's iPad Kitchen Sink" App Action: "App Action2" ▶Details:
19	6899	Info	Validation: vstart	**Starting validation "verifyElementText".** Band: "Band 1" Track: "Track 1" Clip: "AppC KitchenSink showcase (tutorial)" Group: "KitchenSink" Group: "Sliders" Target: "Emmanuel's iPad Kitchen Sink" App Action: "App Action2"
20	6900	Verbose	Validation: vcpass	**Validation of response body passed.** Band: "Band 1" Track: "Track 1" Clip: "AppC KitchenSink showcase (tutorial)" Group: "KitchenSink" Group: "Sliders" Target: "Emmanuel's iPad Kitchen Sink" App Action: "App Action2" ▼Details: `Exact match:` `Value matched:` `Slider`
21	6900	Info	Validation: vpass	**Validation verifyElementText passed.** Band: "Band 1" Track: "Track 1" Clip: "AppC KitchenSink showcase (tutorial)" Group: "KitchenSink" Group: "Sliders" Target: "Emmanuel's iPad Kitchen Sink" App Action: "App Action2"
22	6900	Info	App Action: sent	**App Action completed.** Band: "Band 1" Track: "Track 1" Clip: "AppC KitchenSink showcase (tutorial)" Group: "KitchenSink" Group: "Sliders" Target: "Emmanuel's iPad Kitchen Sink" App Action: "App Action2"
23	6900	Statistics	App Action: stats	**App Action statistics.** Band: "Band 1" Track: "Track 1" Clip: "AppC KitchenSink showcase (tutorial)" Group: "KitchenSink" Group: "Sliders" Target: "Emmanuel's iPad Kitchen Sink" App Action: "App Action2" ▶Details:

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