

SOASTA

TouchTest™ Bamboo CI for iOS Tutorial

SOASTA TouchTest™ Bamboo CI for 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

CloudTest Continuous Integration Support	1
About This Tutorial	1
Bamboo iOS, Cocoa and Xcode Support Plugin	2
Mac and iOS Device Prerequisites	3
TouchTest Utilities and Plugins	5
Test Composition Prerequisites	5
Creating a New Plan	7
Configure Additional Tasks	10
Build the IPA.....	11
Download the TouchTest Utilities	16
Run MakeAppTouchTestable.....	20
Deploy the IPA Archive on iPhones and iPads	22
Create the Test Reports Folder	23
Using SCommand to Play One or More Compositions	24
Adding the JUnit Parser	26
Building the Project	28
Inspecting TouchTest Results in Bamboo	30

CloudTest Continuous Integration Support

You can display SOASTA's JUnit-friendly test results inside other JUnit-friendly applications. In Bamboo, just a few simple steps are necessary to integrate test composition results into the Report Summary dashboard. One, to create a folder for JUnitXML-compatible test output XML files, and then another to specify that folder in your Bamboo plan using a JUnit Parser task.

About This Tutorial

This tutorial provides guidance for two audiences:

- Users who would like to add iOS Testing using Xcode to a pre-existing Bamboo setup
- Users who are iOS Developers starting out with TouchTest who would also like to add TouchTest to a continuous integration setup using Bamboo

This tutorial guides the user through the process of using the Bamboo continuous integration (CI) tool with TouchTest by making a sample Xcode project touchtestable, as part of a small, but complete CI scenario that includes a successful test composition.

In order to do this, we'll first define a Bamboo project, a plan, ensure that we have the iOS, Cocoa, and Xcode plugin installed in Bamboo, and then add tasks to the plan's Default Job.

Once a project and plan are established, the following tasks are defined:

- A Source Code Repository task is added to retrieve the source project, Stockfish, using Bamboo's built-in git support

Note: A version of the Stockfish chess game, an open-source project available from GitHub, has been customized to include the Xcode project that is used as the example project. You can, of course, substitute your own mobile app's source and Xcode project for this example.

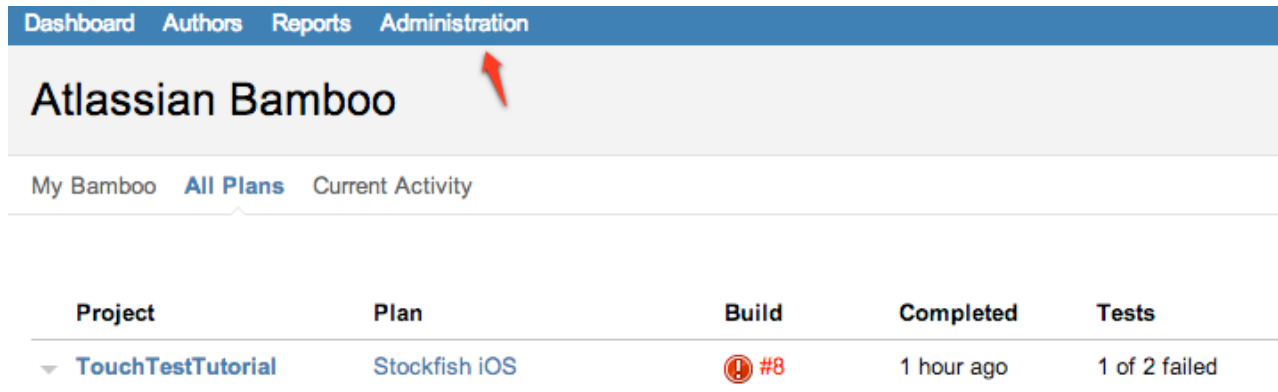
- A script task is added that will download the necessary TouchTest utilities (and always ensure the latest)
- A script task is added to run the MakeAppTouchTestable utility on the Xcode
- An Xcode task is added to build and deploy an IPA using Bamboo's Xcode support
- A script task is added to create a folder in the working directory that will receive TouchTest results (formatted as JUnitXML)
- A script task is added to run test compositions on specific mobile devices and examine the test results using Bamboo's JUnit reports support
- A JUnit task is added to process test results

TIP: If your organization is not already using Bamboo—refer to [the Atlassian Bamboo](#) site to register, download, and install it.

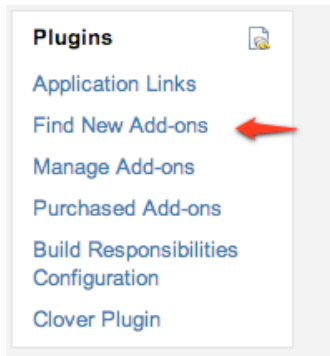
Bamboo iOS, Cocoa and Xcode Support Plugin

In addition to Bamboo itself, ensure that the Bamboo iOS, Cocoa, and Xcode plugin is installed before proceeding with the project creation steps. This is, of course, a requirement of all Bamboo iOS development and not just of TouchTest.

1. Login to your Bamboo server as Admin and then click Administration.



2. Scroll down to the Plugins section and click Find New Add-ons (Find New Plugins in earlier Bamboo versions).



Note: If Bamboo is running, use the alert box Pause button to pause the server.

3. Install the Bamboo iOS, Cocoa and Xcode Support plugin if it is not already installed (as shown below).


Atlassian Marketplace for Bamboo



Find and request powerful add-ons compatible with your Bamboo version on this streamlined Atlassian Marketplace. [Manage add-ons.](#)

The base URL configuration of your instance is inconsistent with the URL in your browser. This may prevent many operations on this page from working correctly. See [the UPM documentation](#) for more details about this error.

Search the Marketplace | Staff Picked | All Categories | Paid or Free



Bamboo iOS, Cocoa and Xcode Support
Atlassian Labs

BAMBOO TASKS **BUILD, RELEASE & DEPLOY**

Provides tasks for building Cocoa and iOS applications, recording OUnit/SenTestKit results and keychain management. This is an early preview and Atlassian appreciates any feedback you may have to help us improve this product.

★★★★☆ (5)
996 Downloads
Free

[Install](#)

[Audit Log](#) | [Bamboo Update Check](#) | [Settings](#)

The Universal Plugin Manager (v2.8.1) by Atlassian


4. Click Install. When you do so, the Install button is replaced by Manage.

Atlassian Marketplace for Bamboo



Find and request powerful add-ons compatible with your Bamboo version on this streamlined Atlassian Marketplace. [Manage add-ons.](#)

Search the Marketplace | Staff Picked | All Categories | Paid or Free



Bamboo iOS, Cocoa and Xcode Support
Atlassian Labs

BAMBOO TASKS **BUILD, RELEASE & DEPLOY**

Provides tasks for building Cocoa and iOS applications, recording OUnit/SenTestKit results and keychain management. This is an early preview and Atlassian appreciates any feedback you may have to help us improve this product.

★★★★☆ (5)
990 Downloads
Free

[Manage](#)

[Audit Log](#) | [Bamboo Update Check](#) | [Settings](#)

The Universal Plugin Manager (v2.8.1) by Atlassian

Mac and iOS Device Prerequisites

One of the key steps during an iOS automated build is deploying the app to your test device, *without requiring any human interaction*. Typical solutions (e.g. over-the-air distribution) require that the user accept a prompt. SOASTA's iOS App Installer Utility includes two tools that *silently* deploy either an IPA file or an APP file.

You will need the following hardware and configuration:

1. A dedicated machine running Mac OS X, with Xcode 4.2 or later. If you are using Bamboo, this can be either the server or a Mac running the remote agent.
2. One or more **tethered** devices. If you have more devices than USB inputs, you can use a USB hub. Note also that sufficient power to prevent the device from running down unexpectedly should be available via that USB input. Simulators can also be used.

A note on tethering: SOASTA TouchTest™ does **not** require tethering for recording or playback. However, you do need to tether the device for silent deployment of your app.

3. The physical iOS device(s) should have the iOS "Auto-Lock" setting set to "Never".
4. At runtime, whenever `scommand` is called upon to play a test composition from the command line, the TouchTest Agent must be running on the mobile device and connected to the correct server instance (e.g. the ones defined in the Bamboo tasks). TouchTest Agent registration steps are covered in the [TouchTest for iOS Tutorial](#).

TouchTest Utilities and Plugins

The following TouchTest software is downloaded in one of the three scripts tasks defined below. It's a good idea to become familiar with them in their own right. Also, note that all TouchTest utility software can be downloaded from the TouchTest, Welcome page, and of course, is not limited to use in CI projects.

- **MakeAppTouchTestable Utility** (this utility will be called at the appropriate time via a Bamboo job using an Execute Shell build step)

Note: This archive contains the necessary drivers upon which TouchTest relies. The CloudTest user specified to run the MakeAppTouchTestable utility must be a user with Mobile Device Administrator rights.

- **iOS App Installer Utility** (this utility contains two executables; the `ios_app_installer`, which is used to install IPA files to iOS physical devices, and the `ios_sim_launcher`, which is used to install compiled APP files to a simulator). This archive contains two executable files:
 - For deployment to Simulators, use the `ios_sim_launcher` executable found in the iOS App Installer Utility at the appropriate time(s) via a Bamboo job using an Execute Shell build step.
 - For deployment to iPhone and iPad devices, use the iOS App Installer Utility to deploy .ipa archives to the physical device(s). This executable can be called at the appropriate time(s) via a Bamboo job using an Execute Shell build step.

The appropriate executable will be called at the appropriate time(s) via a Bamboo job using an Execute Shell build step.

- **CloudTest Command Line Client** (also known as **sCommand**, this command line interface utility will be called at the appropriate time via a Bamboo job using an Execute Shell build step)

Test Composition Prerequisites

This tutorial will call two compositions from the same Bamboo job that will "git" the project, make the app touchtestable, deploy the app to the specified devices, and finally, use sCommand to silently play the specified compositions.

If you're a new CloudTest Mobile user, refer to the following documentation before proceeding with this tutorial.

- Basic TouchTest recording is covered in the [TouchTest Tutorial](#).

TIP: Refer to the "Registering Your Device to Use TouchTest™" section of the TouchTest Tutorial to configure your mobile device for use within your CloudTest instance.

- Advanced TouchTest recording, including the use of validations and other accessors in the open source Stockfish mobile app, are covered in the [TouchTest Advanced Tutorial](#).

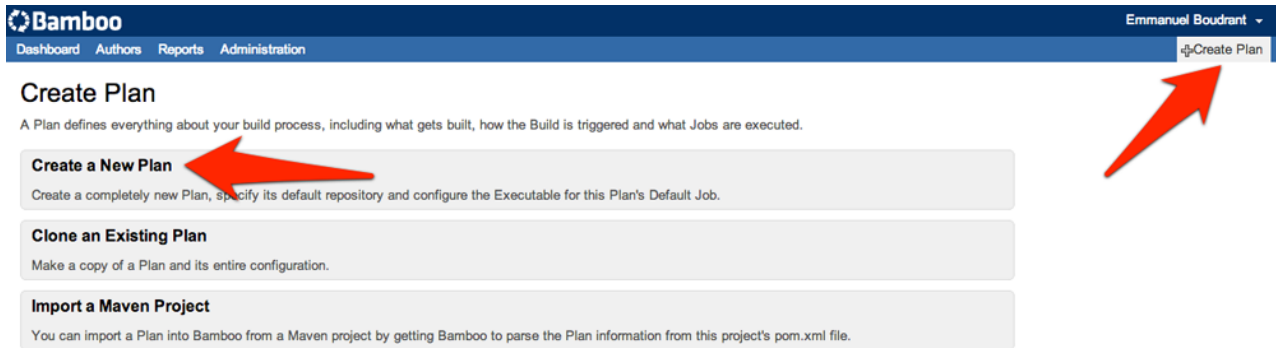
TIP: The test clips shown in the result dashboards at the end of this were created using the GitHub version of Stockfish used in this guide simply by following the steps presented in the following two Advanced Tutorial sections:

- Create a Simple TouchTest Clip – King Gambit Declined
- Advanced Clip Editing – Fool’s Mate

Creating a New Plan

The following steps require the Administrative privilege so be sure to login.

1. In top level Bamboo dashboard, click New Job.
2. In the top-right, click Create.



The screenshot shows the Bamboo web interface. At the top, there is a navigation bar with the Bamboo logo on the left and the user name 'Emmanuel Boudrant' on the right. Below the navigation bar, there are tabs for 'Dashboard', 'Authors', 'Reports', and 'Administration'. In the top right corner of the main content area, there is a button labeled 'Create Plan' with a plus icon. A red arrow points from this button towards the 'Create a New Plan' option in the main content area. The main content area is titled 'Create Plan' and contains a sub-header 'A Plan defines everything about your build process, including what gets built, how the Build is triggered and what Jobs are executed.' Below this, there are three options: 'Create a New Plan', 'Clone an Existing Plan', and 'Import a Maven Project'. A red arrow points from the 'Create a New Plan' option to the left.

Bamboo Emmanuel Boudrant

Dashboard Authors Reports Administration **Create Plan**

Create Plan

A Plan defines everything about your build process, including what gets built, how the Build is triggered and what Jobs are executed.

- Create a New Plan**
Create a completely new Plan, specify its default repository and configure the Executable for this Plan's Default Job.
- Clone an Existing Plan**
Make a copy of a Plan and its entire configuration.
- Import a Maven Project**
You can import a Plan into Bamboo from a Maven project by getting Bamboo to parse the Plan information from this project's pom.xml file.

6. Click Create a New Plan. The New Plan detail appears.
7. In the Project drop-down, select New Project (the plan can be part of a new or existing project).
8. Enter the following:
 - Project Name – *TouchTest Tutorial*
 - Project Key – *TTT* (plan keys and project keys are usually upper case)
 - Plan Name – *Stockfish iOS*
 - Plan Key – *STKIOS*

1 Create Plan > **2** Configure Tasks

Create a New Plan

On this page, you can create a new Plan, which defines everything about your build process, including what gets built, how the Build is triggered and what Jobs are created. More advanced configuration options (including those for plugins), and the ability to add more Jobs will be available to you after creating this Plan.

Plan Details

Project Select or add a Project that the new Plan will be created in.

Project Name* How do you want to call the Project within Bamboo? e.g. "Issue Tracking Application".

Project Key* This is the unique Project key to identify a Project. The key must contain only uppercase alphanumeric characters. e.g. "ITA".

Plan Name* How do you want to identify the new Plan?

Plan Key* This is the key for the plan which must be unique within a project. In conjunction with the project key, it is used to identify a build in URLs, trigger scripts and API calls. TH characters. e.g. "CORE"

Plan Description Choose a meaningful description for the new Plan. For example, "JIRA Release Plan".

Source Repositories

Source Repository Git support works best if the Git executable [capability](#) is defined for agents. If not defined, Bamboo will use JGit, which currently does not support submodules.

Repository URL* The URL of Git repository.

Branch The name of the branch (or tag) containing source code.

9. In the Source Repository drop-down, select Git to follow this tutorial or your own source control repository type.

10. In the Repository URL field, enter:

<https://github.com/elitecoder/stockfishchess-ios>

11. Click the Configure Tasks button.

Source Repositories

Source Repository

Git support works best if the Git executable `capability` is defined for agents. If not defined, Bamboo will use JGit, which currently does not support

Repository URL*

The URL of Git repository.

Branch

The name of the branch (or tag) containing source code.

Authentication Type

Use shallow clones

Fetches the shallowest commit history possible. Do not use if your build depends on full repository history.

Enable Repository Caching on Remote Agents

Cache repositories on remote agents to save bandwidth.

Trigger

Trigger type*

How should Bamboo trigger Builds for this Plan? (Dependent Builds are automatically triggered)

Polling Strategy Periodically

Scheduled

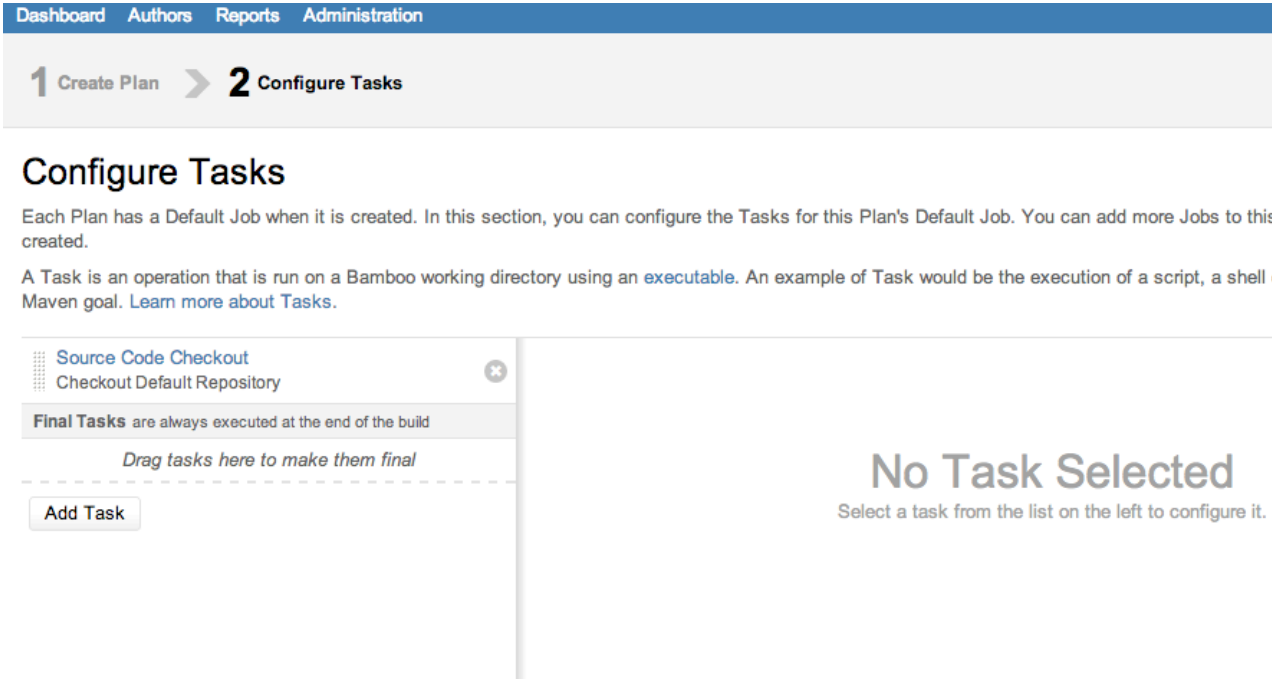
Please select a polling strategy

Polling Frequency

How often (in seconds) should Bamboo check the repository for changes?

Configure Additional Tasks

Clicking Configure Tasks on the New Plan detail page creates the Source Code Repository task show in the page below. This task will get the latest version of the sample project for each build.



Dashboard Authors Reports Administration

1 Create Plan > 2 Configure Tasks

Configure Tasks

Each Plan has a Default Job when it is created. In this section, you can configure the Tasks for this Plan's Default Job. You can add more Jobs to this created.

A Task is an operation that is run on a Bamboo working directory using an `executable`. An example of Task would be the execution of a script, a shell Maven goal. [Learn more about Tasks.](#)

Source Code Checkout
Checkout Default Repository

Final Tasks are always executed at the end of the build

Drag tasks here to make them final

Add Task

No Task Selected

Select a task from the list on the left to configure it.

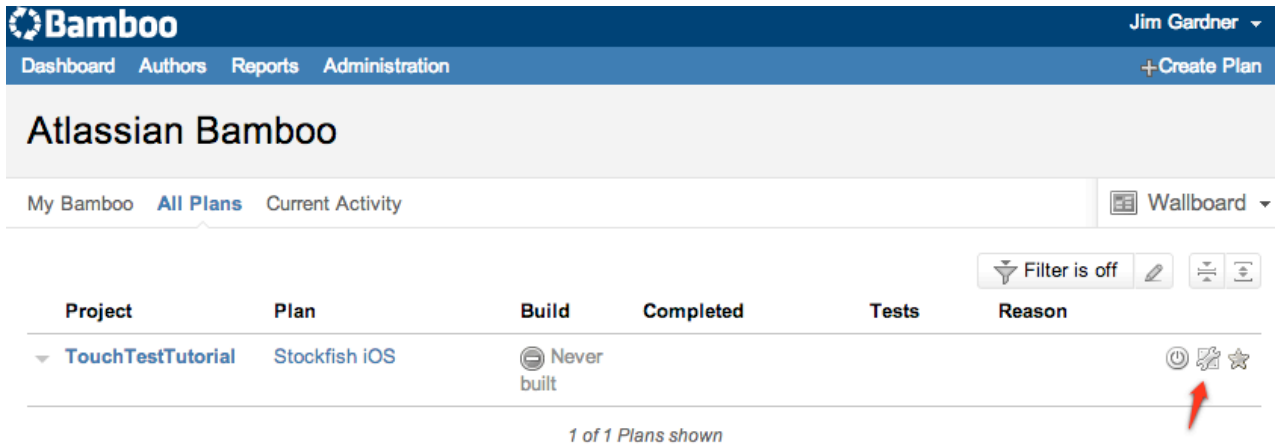
Since we'd like to know right away if our Xcode project is going to compile, we'll build the IPA file using Xcode first and run the build to demonstrate we're on the right track.

Build the IPA

In this section, we'll add an Xcode Task and set it up to compile the Stockfish.xcodeproj using a selected Apple SDK. Once this task has been added, we'll do our first build to ensure that we can build the IPA.

1. In the Tasks lists, click Add Task. The Configure Tasks box appears.

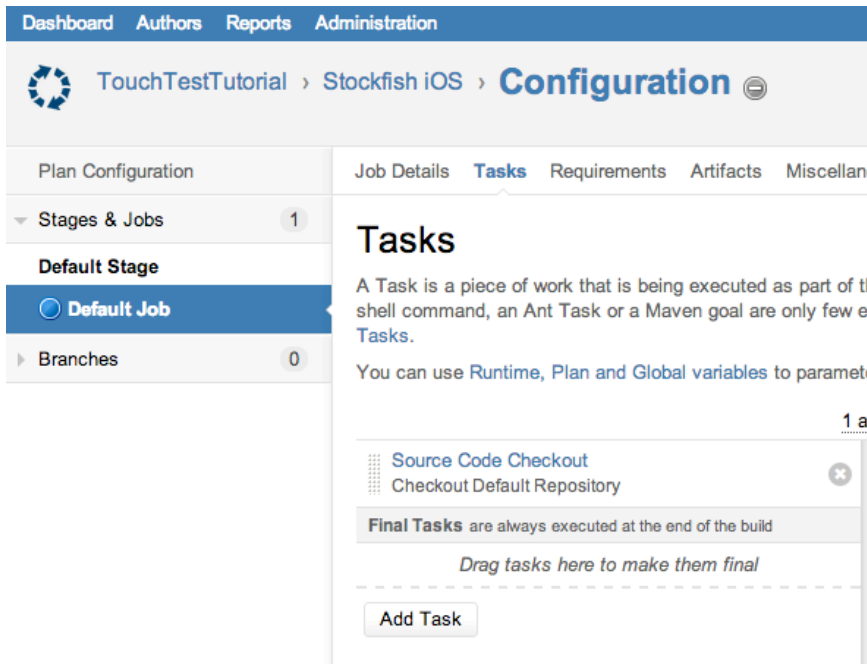
TIP: You can return to the Configure Tasks page by clicking Dashboard > Stockfish iOS, and then the Edit button.



The screenshot shows the Atlassian Bamboo dashboard. At the top, there's a navigation bar with 'Dashboard', 'Authors', 'Reports', and 'Administration'. Below that, the main heading is 'Atlassian Bamboo'. There are tabs for 'My Bamboo', 'All Plans', and 'Current Activity'. A 'Wallboard' button is on the right. Below the navigation, there's a table with columns: Project, Plan, Build, Completed, Tests, and Reason. The table contains one row: 'TouchTestTutorial' (Project), 'Stockfish iOS' (Plan), 'Never built' (Build), and '1 of 1 Plans shown' (Completed). In the Reason column, there are icons for power, edit, and star. A red arrow points to the power icon.

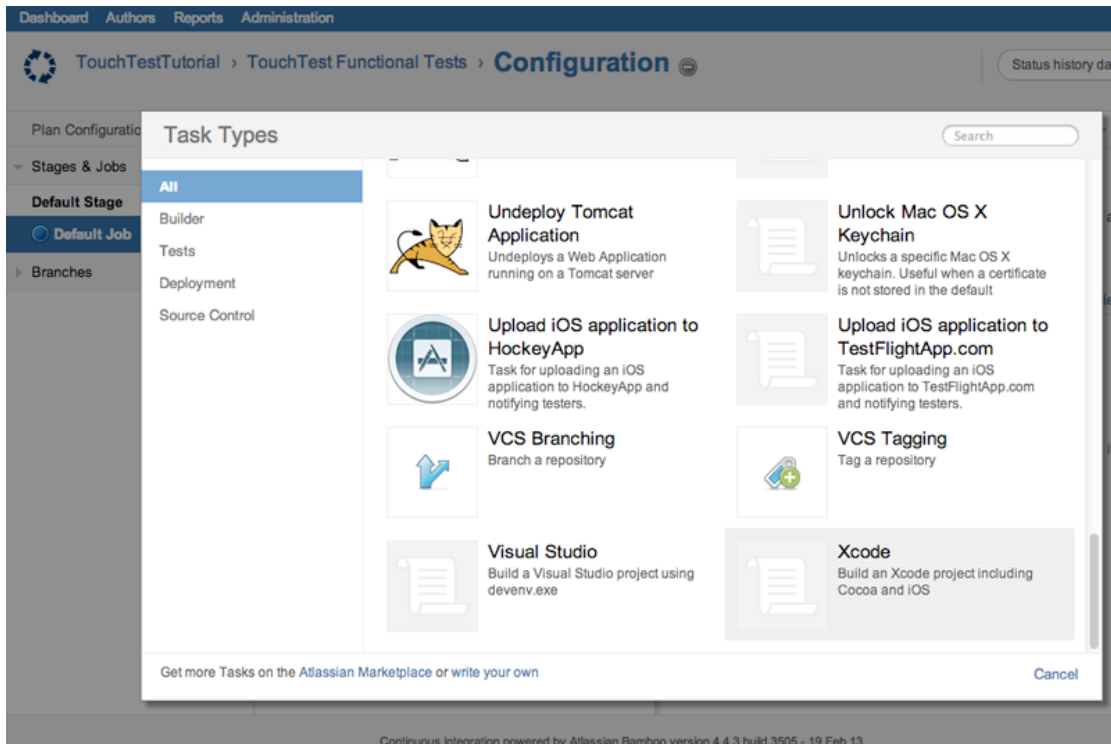
2. Select the Default Plan in the list and then the Tasks tab.

3. Click Add Task.

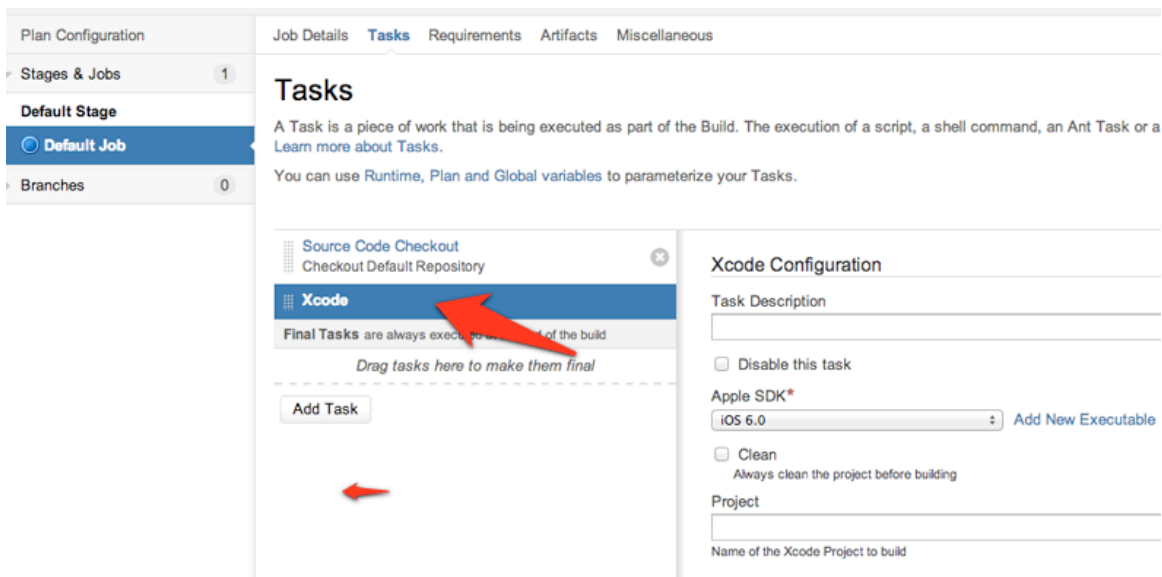


The screenshot shows the Bamboo Configuration page for the 'Stockfish iOS' plan. The 'Tasks' tab is selected. The page shows a list of tasks: 'Source Code Checkout' and 'Checkout Default Repository'. Below the list, there's a section for 'Final Tasks' with a dashed line and the text 'Drag tasks here to make them final'. An 'Add Task' button is at the bottom.

4. In the Task Types box, scroll down to select Xcode.

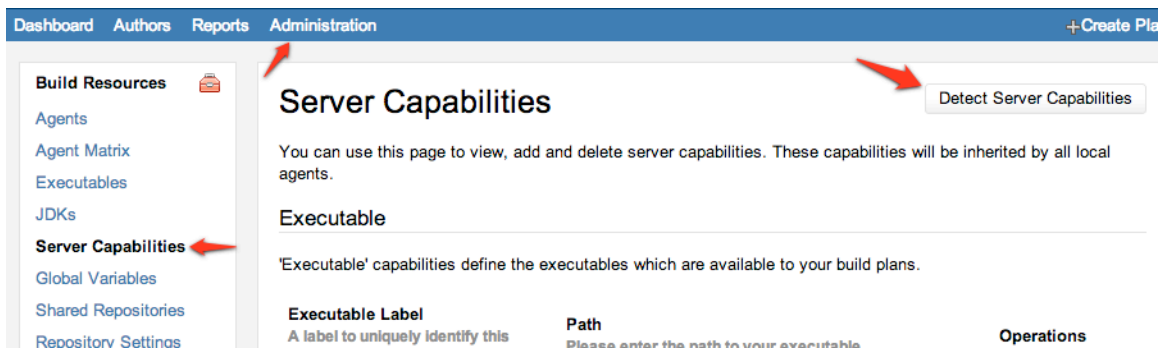


An Xcode Task is added to the Default Job (in the tree on the left).

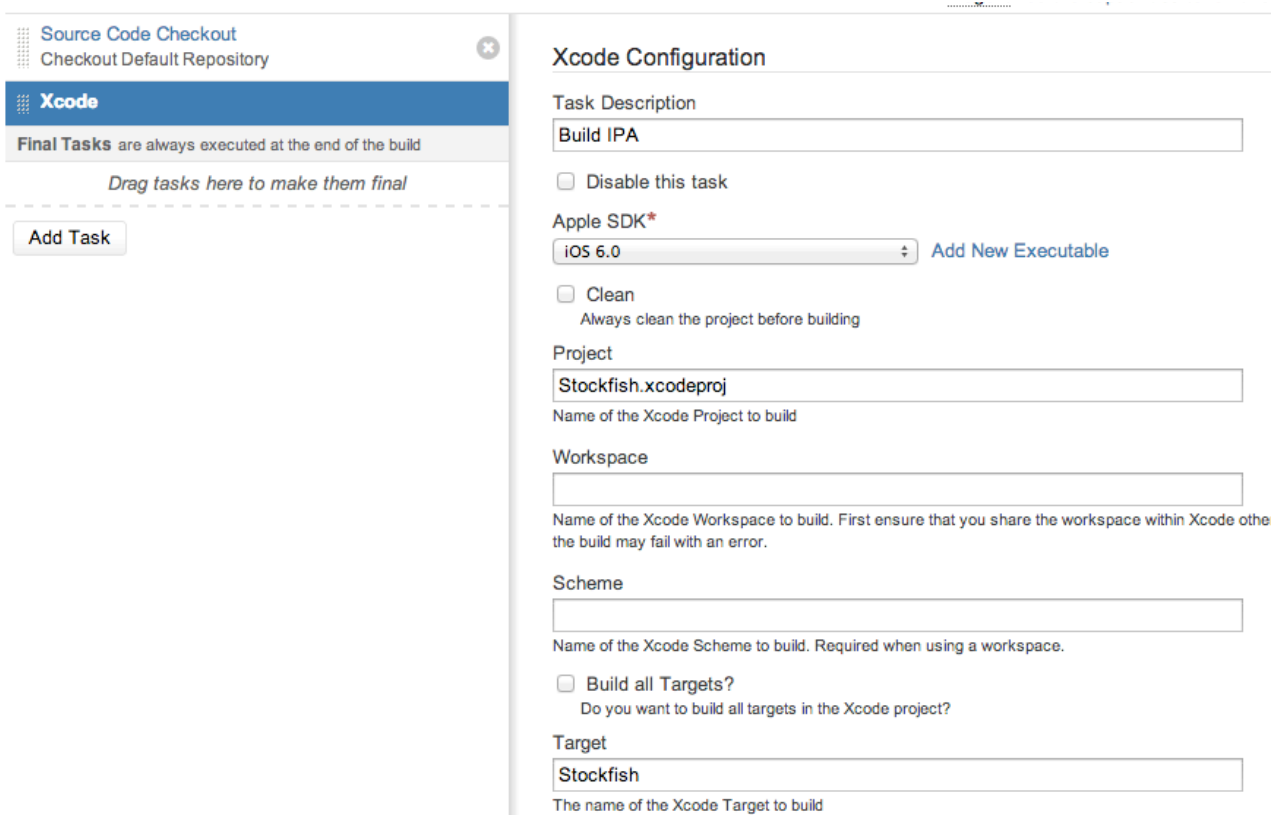


- In the Xcode Configuration form, enter the following:
 - Task Description – *Build IPA*
 - Apple SDK – *iOS 6.1* (in this example)
 - Project – *Stockfish.xcodeproj* (the real .xcodeproj name)
 - Target – *Stockfish*

Note: If the Apple SDK field has no entries, click Administration, Server Capabilities (in the first menu section on the left) and then on the Server Capabilities page click Detect Server Capabilities.



This will add a server capability per Apple SDK that you have installed.



- Continuing in the Xcode Configuration form, check the Build an .ipa for iOS Application Distribution box.
- In the iOS Application Path, enter the app path: *build/Release-iphoneos/Stockfish.app*.

Configuration

The name of the Xcode Configuration to build

Include OCUit/SenTestKit Test results
If you use SenTestKit or OCUit, checking this option will store the test results in Bamboo

Environment Variables

(Optional) Any extra environment variables you want to pass to your build. e.g. JAVA_OPTS="-Xmx256m -Xms128m". You can add multiple parameters

Working Sub Directory

(Optional) Specify an alternative sub-directory as working directory for the task.

Build an .ipa for iOS Application Distribution

iOS Application Path*

Path to the iOS application relative to the working directory.

Certificate Identity Name

Certificate Identity Name used to sign the application.

Embedded Profile

Path to the .mobileprovision profile to embed. Absolute paths are acceptable. Leave blank for no embedded profile.

- Click Save to complete the Xcode Task.

Plan Configuration

Stages & Jobs 1

Default Stage

Default Job

Branches 0

Job Details **Tasks** Requirements Artifacts Miscellaneous

Tasks

A Task is a piece of work that is being executed as part of the Build. The execution of a script, a shell command, an Ant Task or a Maven Task. [Learn more about Tasks.](#)

You can use [Runtime](#), [Plan](#) and [Global](#) variables to parameterize your Tasks.

- Source Code Checkout
 - Checkout Default Repository
- Xcode
 - Build IPA

Final Tasks are always executed at the end of the build

Drag tasks here to make them final

No Task Selected
Select a task from the list on the left

10. Before proceeding, enable the plan by checking Yes please!, and then Create.

Enable this Plan?

Yes please!
By selecting this option your Plan will be available for building and change detection straight away.
Do not select this option if you have advanced configuration changes to make after creation.

Alternately, you can check Plan Enabled on the Configuration page.

Plan Configuration | Plan Details | Source Repositories | Triggers | Branches | Stages | De

Stages & Jobs 1 | Variables | Miscellaneous

Default Stage

Default Job

Branches 0

Plan Details

You can update the names and description of the current project and plan.

Project Name*

Plan Name*

Plan Description
Choose a meaningful description for the new Plan. For exam

Plan Enabled

11. Click Run, Run Plan—either on the Plan Summary or Configuration page to perform the first plan build.

Run Plan

Run Customised...

st 25 builds

The Build Result Summary appears. If the IPA was built the success message appears.

#1 was successful – Manual build by Jim Gardner

Build Summary | Tests | Changes | Artifacts | Logs | Metadata

Build Result Summary

Details

Completed 23 Feb 2013, 12:31:36 AM – 4 minutes ago

Duration 37 seconds

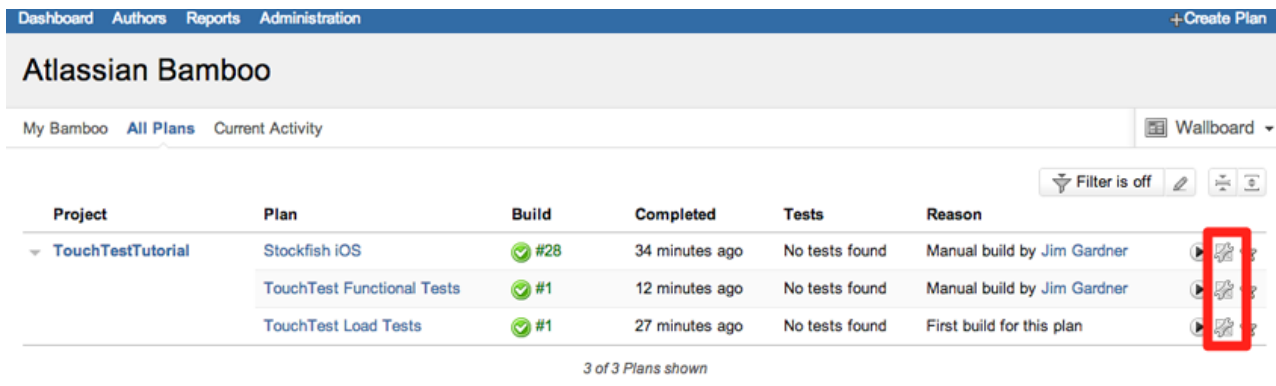
Labels None




Download the TouchTest Utilities

Next, we'll add a script task that will use curl to download all the necessary TouchTest utilities. The TouchTest utilities, including sCommand, should always be downloaded from the CloudTest server with which will be used in tandem.

Since this task will occur in every build our CI project will always use the latest SOASTA tools.

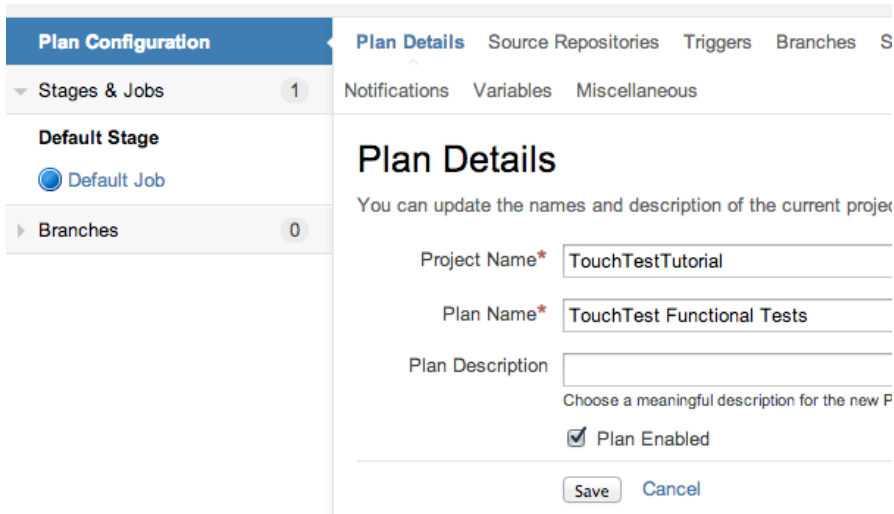
1. Click Dashboard to return to the list of projects/plans.
2. Click the Edit button in row of the plan to edit.



Project	Plan	Build	Completed	Tests	Reason	
TouchTestTutorial	Stockfish iOS	✔ #28	34 minutes ago	No tests found	Manual build by Jim Gardner	
	TouchTest Functional Tests	✔ #1	12 minutes ago	No tests found	Manual build by Jim Gardner	
	TouchTest Load Tests	✔ #1	27 minutes ago	No tests found	First build for this plan	

3 of 3 Plans shown

The Plan Details page appears.



Plan Configuration

- Stages & Jobs 1
- Default Stage
 - Default Job
- Branches 0

Plan Details Source Repositories Triggers Branches S

Notifications Variables Miscellaneous

You can update the names and description of the current project

Project Name* TouchTestTutorial

Plan Name* TouchTest Functional Tests

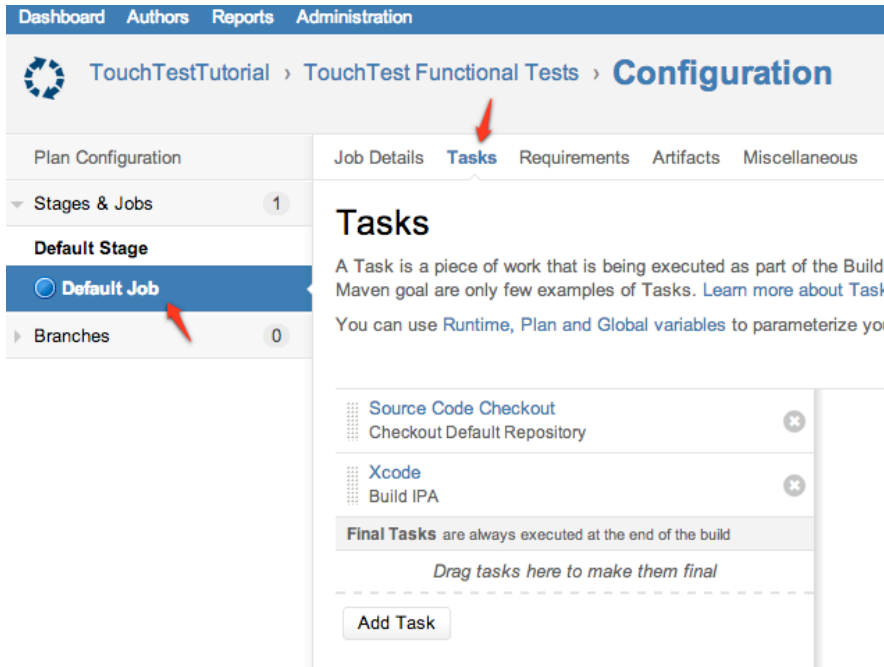
Plan Description

Choose a meaningful description for the new Plan

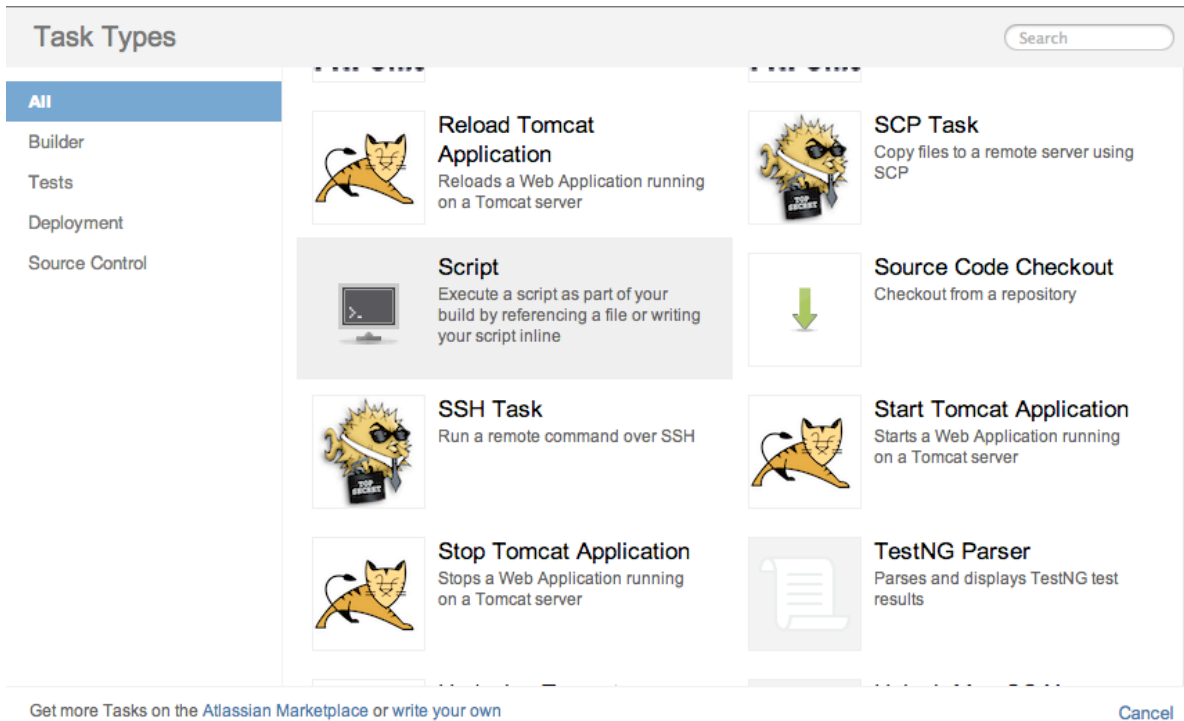
Plan Enabled

Save Cancel

3. Click Default Job in the list and then the Tasks tab to return to the Configuration page.



4. In the Tasks lists, click Add Task again.
5. Select the Script task type.



The new Script Task is added to the tasks list.

Enter a Task Description. For example, *Download TouchTest Utilities*.

Tasks

A Task is a piece of work that is being executed as part of the Build. The execution of a script, a shell command, an Ant Task or a Maven goal are only few examples of Tasks. [Learn more about Tasks](#).

You can use [Runtime](#), [Plan](#) and [Global variables](#) to parameterize your Tasks.

1 agent has the capabilities

- Source Code Checkout
- Checkout Default Repository
- Xcode
- Build IPA
- Script**

Final Tasks are always executed at the end of the build

Drag tasks here to make them final

Add Task

Script Configuration

Task Description

Disable this task

Script location

Script body*

1	
---	--

(Windows: .bat file; Unix: /bin/sh compatible script)

Argument

(Optional) Argument you want to pass to the command. Arguments with spaces in them must be quoted

Environment Variables

- Next, we'll use the curl command to retrieve the utilities and unzip to extract them in the working directory. Enter the following script code using your own server's CloudTest URL (in this example, a CloudTest Lite instance with the IP address 10.0.1.6 is in use):

```
echo "Downloading iOSAppInstaller.zip from TouchTest server"  
curl http://<CloudTest URL>/concerto/downloads/mobile/iOSAppInstaller.zip > ./  
iOSAppInstaller.zip  
echo "Installing iOSAppInstaller.zip"  
unzip -o ./iOSAppInstaller.zip
```

```
echo "Downloading scommand.zip from TouchTest server "  
curl <CloudTest URL>/concerto/downloads/scommand/scommand.zip > ./scommand.zip  
echo "Installing scommand.zip"  
unzip -o ./scommand.zip
```

```
echo "Downloading MakeAppTouchTestable.zip from TouchTest server "  
curl http://<CloudTest URL>/concerto/downloads/mobile/MakeAppTouchTestable.zip  
> ./MakeAppTouchTestable.zip  
echo "Installing MakeAppTouchTestable.zip"  
unzip -o ./MakeAppTouchTestable.zip
```

- Click Save to complete the task.

The screenshot shows a web-based interface for configuring a build task. On the left, there is a sidebar with 'Source Code Checkout' and 'Checkout Default Repository' options. Below this is a 'Script' section with a blue header and a note that 'Final Tasks are always executed at the end of the build'. A dashed line separates this from a 'Drag tasks here to make them final' area, which contains an 'Add Task' button. The main area is titled 'Script Configuration' and contains the following fields:

- Task Description:** A text input field containing 'Download TouchTest utilities'.
- Disable this task:** An unchecked checkbox.
- Script location:** A dropdown menu set to 'Inline'.
- Script body*:** A code editor with 14 lines of shell script code, which is a copy of the code provided in the previous blocks.

Run MakeAppTouchable

TouchTest™ includes the MakeAppTouchable, which will automatically add the necessary components to an existing Xcode project to deploy TouchTest™, and additionally, the utility will also create the Mobile App entry in CloudTest® .

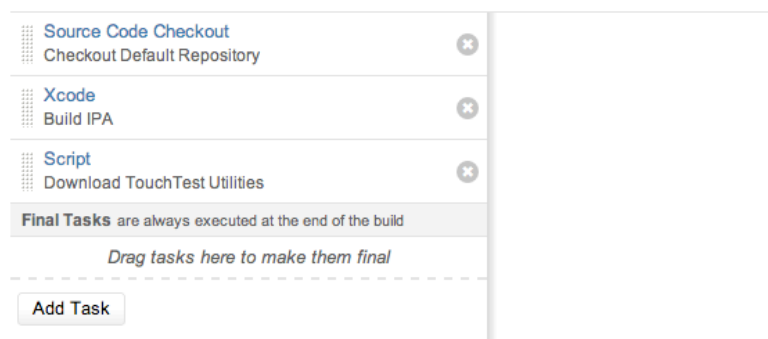
1. In the Tasks lists, click Add Task and select the Script task type a second time.

Job Details **Tasks** Requirements Artifacts Miscellaneous

Tasks

A Task is a piece of work that is being executed as part of the Build. The execution of a script, [Learn more about Tasks](#).

You can use [Runtime](#), [Plan](#) and [Global variables](#) to parameterize your Tasks.



2. In the Script Configuration form, enter *Run MATT* as the description.
3. Paste the MakeAppTouchable command into the script body. Do not use full paths (since the build folder changes +1 at each build. Bamboo ensures that the working directory is in use.

```
#Run MATT
```

```
sh MakeAppTouchable/bin/MakeAppTouchable -project  
Stockfish.xcodeproj -target "Stockfish" -url http://<CloudTest URL> -  
username SOASTA_DOC -password secret -overwriteapp
```

where:

- `<Xcode project file>` is the actual name of the ".xcodeproj" file representing your project (i.e. Stockfish.xcodeproj). In this case, only the project name is necessary.
- `<target name>` is the name of the Xcode target you would like to modify. In this case, *Stockfish* is the target name to modify.
- `<CloudTest URL>` is the server instance to connect. For example, a CloudTest Lite server on a network using an Apple router might be located at `http://10.0.1.6/concerto`.

4. Click Save to complete adding the task.

5. In the steps above, we opted to create the Xcode step early in order to verify it worked. This step actually needs to come after the Run MATT (since only after MakeAppTouchTestable is applied is the project touchtestable). Drag Build IPA into the fourth position in the list.

Tasks

A Task is a piece of work that is being executed as part of the Build. Th [Learn more about Tasks](#).

You can use [Runtime](#), [Plan](#) and [Global variables](#) to parameterize your T:

The screenshot shows a vertical list of build tasks. Each task has a three-dot menu icon on the left and a close icon on the right. The tasks are:

- Source Code Checkout (Checkout Default Repository)
- Script (Download TouchTest Utilities)
- Script
- Xcode (Build IPA) - This task is highlighted with a grey background.

Below the list is a section for Final Tasks, indicated by a dashed line. It contains the text: "Final Tasks are always executed at the end of the build" and "Drag tasks here to make them final". At the bottom of this section is an "Add Task" button.

Deploy the IPA Archive on iPhones and iPads

In this section, we'll create the script that will run the `ios_app_installer` to deploy the IPA archive created in the prior section. Before starting, note the path to the downloaded iOS App Installer Utility. The iOS App Installer Utility downloaded in the Download TouchTest Utilities section contains two executable files—`ios_sim_launcher` and `ios_app_installer`. From the working directory the executables were extracted to `/iOSAppInstaller/bin`.

Deployment is achieved for both Simulators and iPad/iPhone devices using these executables and the steps described below.

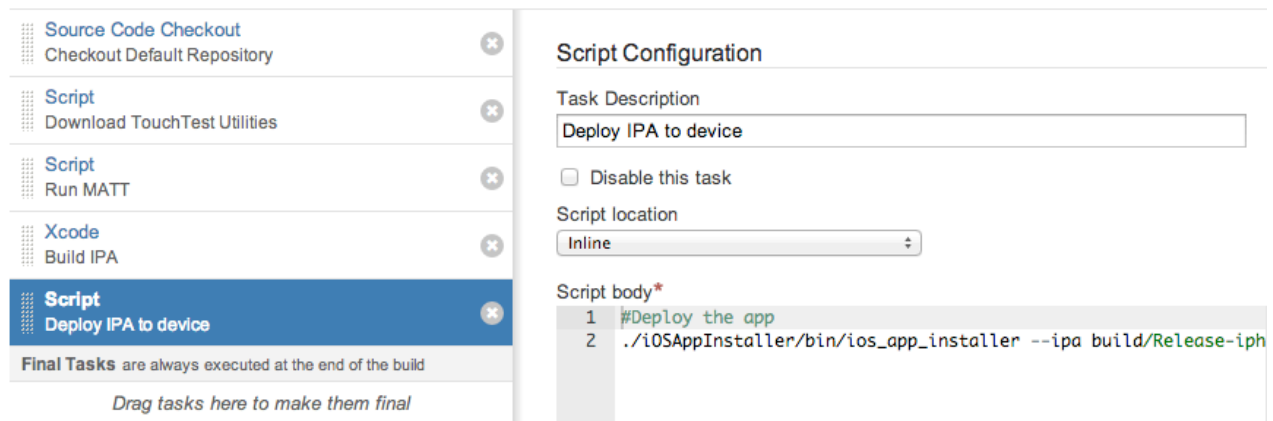
- For a Simulator, the `./iOSAppInstaller/bin/ios_sim_launcher -app` command is used
In order to deploy to a simulator, we must first build an APP file and then use that APP file with the `ios_sim_launcher`
 - For an iPhone or iPad, the `./iOSAppInstaller/bin/ios_app_installer -ipa`
In order to deploy to a physical device, we must first build an APP file, followed by building an IPA file, after which we can use the `ios_app_installer`.
1. In the Script Configuration form, enter the following (be sure to revise this example to use your own paths):

```
#Deploy the app
```

```
./iOSAppInstaller/bin/ios_app_installer --ipa build/Release-iphones/  
Stockfish.ipa
```

where:

- `--ipa <ipapath>` - is the path to the IPA archive built in the prior section.



The screenshot shows the Xcode Script Configuration window. On the left, a list of tasks is shown, with 'Deploy IPA to device' selected. The main area displays the configuration for this task: 'Task Description' is 'Deploy IPA to device', 'Script location' is 'Inline', and 'Script body' contains the following code:

```
1 #Deploy the app
2 ./iOSAppInstaller/bin/ios_app_installer --ipa build/Release-iph
```

The iOS App Installer Utility will deploy to all the tethered provisioned devices by default. If you'd like to limit the deployment to specific devices use the following optional parameters:

- `--udid <list>` - One or more device UDID in a comma-separated list, if unspecified it install on all the connected iOS devices

- `--device <list>` - device name list comma-separated, if unspecified it will install on all the connected iOS devices

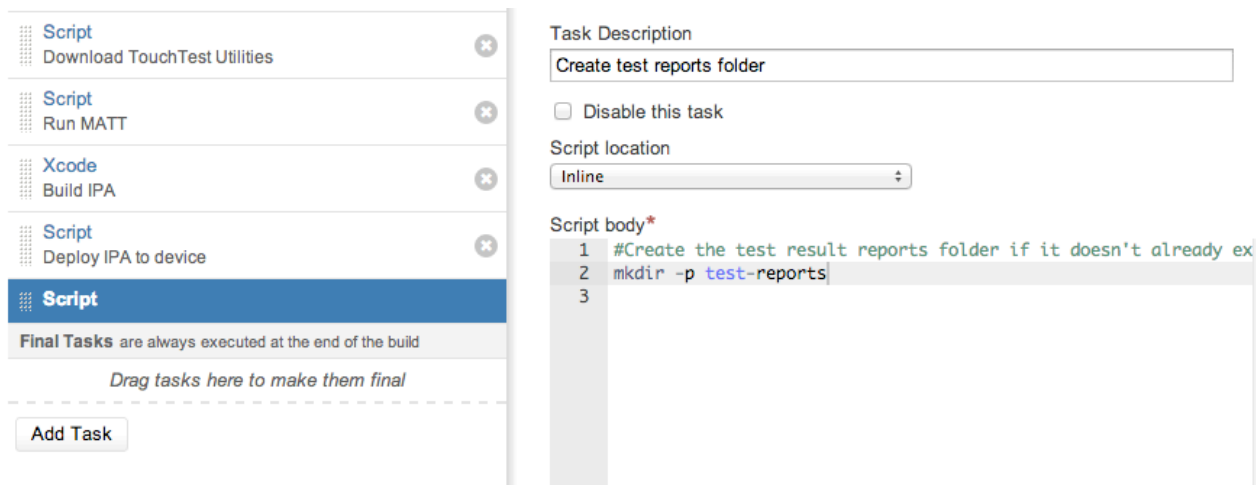
2. Click Save to complete adding this script task.

Create the Test Reports Folder

Next, we'll create another script task where we'll direct sCommand to output JUnitXML-ready test results. This location must match the field you specify in the JUnit Parser section (below).

1. Click Add Task and once again select the Script task type.
2. In the description field, enter "Create test reports folder."
3. Add a line to create the test result reports folder if doesn't already exist:

```
#Create the test result reports folder if it doesn't already exist  
mkdir -p test-reports
```



4. Click Save to complete adding this script task.

Tasks

A Task is a piece of work that is being executed as part of the Build. The execution of a script, a shell command, an Ant Task examples of Tasks. [Learn more about Tasks.](#)

You can use [Runtime](#), [Plan](#) and [Global variables](#) to parameterize your Tasks.

One agent

Source Code Checkout
Checkout Default Repository

Script
Download TouchTest Utilities

Script
Run MATT

Xcode
Build IPA

Script
Deploy IPA to device

Script
Create test reports folder

Final Tasks are always executed at the end of the build

No Task Selected
Select a task from the list on the left to

Using SCommand to Play One or More Compositions

Next, we will add SCommand lines that will silently play the specified test compositions on the specific CloudTest instance. Additionally, we will add arguments that will output junitxml-compatible XML code that will appear in Bamboo for each test result.

1. In the description field, enter *Play test compositions*.
2. In the Script Configuration form, enter the following:

```
# Run the first composition.
# The result will be stored in the "test-reports/foolsmate.xml" file.
# Bamboo will use this file to render the test report.
./scommand/bin/scommand \
  cmd=play \
  name="/SOASTA Tutorial/Advanced/Composition for Fools Mate Clip" \
  wait \
  format=junitxml \
  url=http://10.0.1.6/concerto \
  username=SOASTA_DOC \
  password=secret >test-reports/foolsmate.xml
```

3. Enter any additional compositions for the task:

```
# Run the second composition.
# The result will be stored in the "test-reports/kingsgambit.xml" file.
# Bamboo will use this file to render the test report.
./scommand/bin/scommand \
  cmd=play \
  name="/SOASTA Tutorial/Advanced/Composition for King Gambit Clip" \
  wait \
  format=junitxml \
  url=http://10.0.1.6/concerto \
  username=SOASTA_DOC \
  password=secret >test-reports/kinggambit.xml
```

The screenshot displays the Bamboo build system interface. On the left, a list of tasks is shown, with the 'Script' task selected. Below the list, there is a section for 'Final Tasks' and an 'Add Task' button. On the right, the 'Task Description' is 'Play test compositions'. The 'Script location' is set to 'Inline'. The 'Script body' contains the following code:

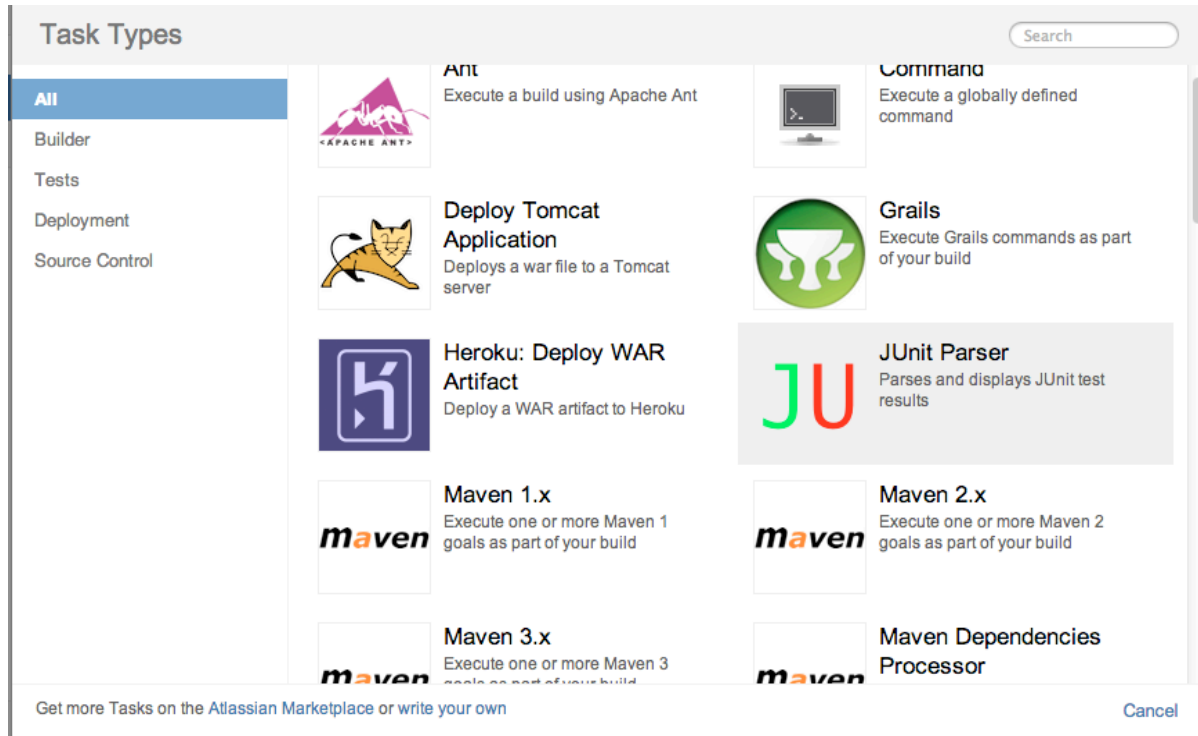
```
7  wait \
8  format=junitxml \
9  url=http://ctmobile.soasta.com/concerto \
10 username=SOASTA_DOC \
11 password=magma >test-reports/foolsmate.xml
12
13 # Run the second composition.
14 # The result will be stored in the "test-reports/kingsgambit.x
15 # Bamboo will use this file to render the test report.
16 ./scommand/bin/scommand \
17   cmd=play \
18   name="/SOASTA Tutorial/Advanced/Composition for King's Gambi
19   wait \
20   format=junitxml \
21   url=http://ctmobile.soasta.com/concerto \
22   username=SOASTA_DOC \
23   password=magma >test-reports/kinggambit.xml
24
```

4. Click Save to complete adding this script task.

Adding the JUnit Parser

Finally, we will add the Bamboo JUnit Parser task, which will utilize the sCommand JUnit output to incorporate test composition results into the Report Summary page.

1. In the Tasks lists, click Add Task.
2. In the Task Types box, choose JUnit Parser.



3. In the description field, enter *Parse TouchTest results*.

- Specify the default custom results directory. For example, entering `**/test-reports/*.xml` will create a "test-reports" folder in the working directory.

The screenshot shows the Xcode build system configuration interface. On the left, a list of tasks is displayed, with 'JUnit Parser' and its sub-task 'Parse TouchTest results' selected. The right pane shows the configuration for 'JUnit Parser Configuration'. The 'Task Description' is 'Parse TouchTest results'. There is a checkbox for 'Disable this task' which is unchecked. The 'Specify custom results directories' field contains the path `**/test-reports/*.xml`. Below this, a note asks 'Where does the build place generated test results?' and explains that the field is a comma-separated list of test result directories. At the bottom of the configuration pane are 'Save' and 'Cancel' buttons.

- Click Save to complete adding the JUnit Parser task. This completes creation of the TouchTest CI scenario tasks. Now, we're ready to do a full build.

Building the Project

Before you click Run, Run Plan, lets review ensure that the TouchTest Agent is running on each device you want to test.

1. To build the project, click the "Run, Run Plan" link on any page where it appears.

The build will start and the Build Result Summary page appears.

#6 is building - 2 mins remaining – Manual build by Jim Gardner

Build Summary Changes Artifacts Logs

Build Result Summary

Details

Started 25 Feb 2013, 5:05:18 PM – 10 seconds ago

Labels None

Agent Default Agent

Revision ceb8eb1...

Write a comment...

Live activity log for Default Job

```
25-Feb-2013 17:05:18 Build TTT-STKIOS-J081-6 started building on agent Default Agent
25-Feb-2013 17:05:18 Build working directory is /Users/jgardner/bamboo-home/xml-data/build-dir/TTT-STKIOS-J081
25-Feb-2013 17:05:18 Executing build TTT-STKIOS-J081-6
25-Feb-2013 17:05:18 Starting task 'Checkout Default Repository' of type 'com.atlassian.bamboo.plugins.vcs:task.vcs.checkout'
25-Feb-2013 17:05:18 Updating source code to revision: ceb8eb1b3e56c3846d9ce6b64727142fadf246d0
25-Feb-2013 17:05:18 Fetching 'refs/heads/master' from 'https://github.com/elitecoder/stockfishchess-ios'.
25-Feb-2013 17:05:20 Checking out revision ceb8eb1b3e56c3846d9ce6b64727142fadf246d0.
25-Feb-2013 17:05:20 Already on 'master'
25-Feb-2013 17:05:20 Updated source code to revision: ceb8eb1b3e56c3846d9ce6b64727142fadf246d0
25-Feb-2013 17:05:20 Finished task 'Checkout Default Repository'
25-Feb-2013 17:05:20 Running pre-build action: Build Number Stamper
25-Feb-2013 17:05:20 Running pre-build action: Clover Grails PreBuild Action
25-Feb-2013 17:05:20 Running pre-build action: VCS Version Collector
25-Feb-2013 17:05:20 Running pre-build action: Repository Isolation Enabler Action
25-Feb-2013 17:05:20 Running pre-build action: Maven Settings Prebuild Action
25-Feb-2013 17:05:20 Starting task 'Download TouchTest Utilities' of type 'com.atlassian.bamboo.plugins.scripttask:task.builder.script'
25-Feb-2013 17:05:20 Beginning to execute external process for build 'TouchTestTutorial - Stockfish iOS - Default Job'
... running command line:
/bin/sh /var/folders/2n/sqf9zyps4v1052lhx5_q891r000qp/T/TTT-STKIOS-J081-6-ScriptBuildTask-1708737267256484028.sh
```

After a short delay, you should see a progress bar appear on the left side of the page. Click this progress bar to watch the build process "live" in the Console view.

You should see the following happen in the Live activity log for Default Job:

- a. Bamboo checks out the source code from Git.
- b. Bamboo downloads the TouchTest utilities extracts them into the working directory.
- c. Bamboo runs the MakeAppTouchTestable utility, which makes the Stockfish app touchtestable.
- d. Bamboo runs the build IPA.
- e. Bamboo runs the Deploy IPA to device script. On the build node, you should see Xcode immediately launch and deploy the Stockfish app to the tethered device. **Do not interact with Xcode while this is happening.** Once the app has been deployed, Xcode will automatically exit.

- f. On the tethered device, you should see the Stockfish app briefly appear. It will immediately switch over to the SOASTA TouchTest Agent web page (in Safari), with the Status showing "Connected." Note that the tethering requirement originates with Xcode and is not a requirement of touch testing.
- g. Bamboo plays the CloudTest compositions using SCommand. On the tethered device, you should see the Stockfish app launch and run through the test steps for each composition you included. When the test finishes, Stockfish will exit, and the SOASTA TouchTest Agent page will reappear on the device. Finally, the build results are posted incorporating the JUnitXML output received.
- h. Test results are presented by one of two categories: New Failures (these can include test errors posted in results by sCommand) and Fixed Tests (tests that failed in one or more prior builds that are fixed in the current build).

Inspecting TouchTest Results in Bamboo

For a successful test with no failures, the Bamboo Test Report Summary page merely lists the All Tests section with the given package (i.e. in this case the package equates to a CloudTest repository folder).

The screenshot shows the Bamboo Test Report Summary page for a successful build. The breadcrumb navigation is "TouchTestTutorial > Stockfish iOS > #5". A green banner at the top states "#5 was successful – Manual build by Jim Gardner". Below this, there are tabs for "Build Summary", "Tests", "Changes", "Artifacts", "Logs", and "Metadata". The main heading is "Build Result Summary" with a sub-heading "Details". The build status is "Completed" on "25 Feb 2013, 4:57:52 PM – 18 minutes ago", with a "Duration" of "4 minutes" and "Labels" set to "None". A "Show More" link is present. A summary card displays three metrics: "0 New Failures", "0 Existing Failures", and "1 Fixed". Below this is a comment section with a "Write a comment..." input field. The "Tests" section shows "Fixed Tests 1". A table lists the test results:

Test	Failing Since	View Job	Durati
Advanced Composition for Fools Mate Clip 🔗	#4 (Manual build by Jim Gardner)	Default Job	30 se

In the tests sections (near the bottom), test compositions will be listed if they are new failures (which doesn't apply in a result marked "successful (shown in green above)", or if they failed in a prior build and passed in this build (fixed tests).

- Clicking the test composition link (e.g. Composition for Fools Mate Clip) opens the Test Summary page, which gives greater detail about the performance record of this item.

Dashboard Authors Reports Administration + Create Plan

TouchTestTutorial > **Stockfish iOS**

✔ ❌ ❌ ❌ ✔ ✔ ✔

Job Summary Recent Failures History **Tests** Files Logs

▶ Run ⚙️ Actions

Test Summary

This page summarises information for **Composition for Fools Mate Clip** in test class **Advanced**.

Details

Test Class **SOASTA Tutorial.Advanced**

Test **Composition for fools mate clip**

Last ran ✔️ #7 (Manual build by [Jim Gardner](#) – 10 minutes ago)

First ran #4

Failures **1**

Successes **3**

Passing since #5 (Manual build by [Jim Gardner](#) – 30 minutes ago)

Test Statistics

75%
Successful

Successful Jobs: 3 / 4

Average Duration: 28 seconds

Recent Failures

Failed In	Fixed In	Time Taken To Fix
❌ #4 Manual build by Jim Gardner <small>51 minutes ago</small>	✔️ #5 Manual build by Jim Gardner <small>30 minutes ago</small>	1 build 18 minutes

The following failure information is for last 25 builds:

- Average time to fix test when failed: **18 minutes**
- Average number of builds between fixes: **1 build(s)**

Things get more interesting when an error in the test composition occurs. The subsequent SCommand output is displayed (in text) on the Build Result Summary page (as discussed above).

#4 failed – Manual build by Jim Gardner

Build Summary Tests Changes Artifacts Logs Metadata Run Actions

Build Result Summary

Details

Completed 25 Feb 2013, 4:37:17 PM – 54 minutes ago
 Duration 3 minutes
 Labels None

Responsible

This build has been failing since #2
 No one has taken responsibility

Assign responsibility Claim full responsibility

1
New Failures

0
Existing Failures

0
Fixed

Show More

Write a comment...

Tests

New Test Failures 1

Test	View Job	Duration	
Advanced Composition for Fools Mate Clip	Default Job	20 secs	Quarantine

Completed: Composition completed. App Action failed. (Band "Band 1" Track "Track 1" Clip "Fools Mate Clip")

In this case, the New Test Failures section is added (under Tests) with the name of the test composition listed with a link to more of the SCommand details.

Clicking the link under the Test Name section where the composition is named displays an Error detail page (for the given error).

Job: Default Job failed

Job Summary Tests Changes Artifacts Logs Metadata Run Actions

Composition for fools mate clip: Test Case Result

The below summarizes the result of the test "Composition for fools mate clip" in build 4 of TouchTestTutorial - Stockfish iOS - Default Job.

Description	Composition for fools mate clip	Duration	20 secs
Test Class	SOASTA Tutorial.Advanced	Status	Failed (New Failure)
Method	Composition for Fools Mate Clip		

Error Log


```
Completed: Composition completed. App Action failed. (Band "Band 1" Track "Track 1" Clip "Fools Mate Clip")
```

The Error Log presents the main error message, which is also found in the summary section on the Result Details dashboard (in the SOASTA UI).

In the error below a new failure, an image validation in the Composition for Fools Mate has failed.

Tests

New Test Failures 1

Test	View Job	Duration
⌵  Advanced Composition for Fools Mate Clip ⌵	Default Job	32 secs
<pre>Completed: Composition completed. Validation of response body did not pass. (Band "Band 1" Track "Track 1" Clip "Fools Mate Clip")</pre>		

Refer to [Result Details Dashboard](#) for a quick review of Result Details features.

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