



Automation Scripting in DeviceAnywhere Studio

DeviceAnywhere Enterprise Monitoring 6.0

DeviceAnywhere Enterprise Automation 6.1

Mobile App Monitoring 7.0

February 2014

Copyright Notice

Copyright © 1995-2014 Keynote Systems, Inc. All rights reserved

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT ANY EXPRESS REPRESENTATIONS OF WARRANTIES. IN ADDITION, KEYNOTE DISCLAIMS ALL IMPLIED REPRESENTATIONS AND WARRANTIES, INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

All text and figures included in this publication are the exclusive property of Keynote and may not be copied, reproduced, or used in any way without the express permission in writing of Keynote. Information in this document is subject to change without notice and does not represent a commitment on the part of Keynote. Keynote may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents except as expressly provided in any written license agreement from Keynote.

The trademarks or registered trademarks of Keynote Systems, Inc. in the United States and other countries include Keynote®, DataPulse®, CustomerScope®, Keynote Customer Experience Rankings®, Perspective®, Keynote Red Alert®, Keynote WebEffective®, The Internet Performance Authority®, MyKeynote®, SIGOS®, SITE®, keynote® The Mobile & Internet Performance Authority™, Keynote FlexUse®, Keynote DeviceAnywhere®, DeviceAnywhere®, Keynote DemoAnywhere®, Keynote MonitorAnywhere®. All related trademarks, trade names, logos, characters, design and trade dress are trademarks or registered trademarks of Keynote Systems, Inc. in the United States and other countries and may not be used without written permission. All other trademarks are the property of their respective owners.

Please forward any comments or suggestions regarding this document to Keynote Support.

Keynote Systems, Inc.
777 Mariners Island Blvd.
San Mateo, CA 94404

Contents

About This Document.....	9
Document Outline.....	9
Typographical Conventions.....	10
Contacting Support.....	10
Additional Documentation.....	10
1 Prerequisites and Overview.....	12
1.1 Prerequisites.....	12
1.1.1 System Requirements for DeviceAnywhere Studio.....	12
1.2 Product Components.....	13
1.2.1 DeviceAnywhere Enterprise Automation.....	13
1.2.2 DeviceAnywhere Enterprise Monitoring.....	17
1.2.3 Mobile App Monitoring.....	21
2 Concepts and Test Scripting Workflow.....	24
2.1 Concepts.....	24
2.2 Best Practice Scripting Workflow.....	26
3 Projects.....	28
3.1 Creating a Project.....	28
3.1.1 Project Properties.....	29
3.2 Opening a Project.....	33
3.3 Managing Projects.....	34
3.3.1 Project-Level Search.....	34
3.3.2 Other Functions.....	35
4 Begin Scripting—Actions.....	39
4.1 Creating an Action.....	39
4.1.1 Implementing an Action.....	40
4.2 Recording a Script.....	42
4.3 Action Properties Dialog Box.....	47
4.3.1 Overview Tab.....	48
4.3.2 Parameters Tab.....	48
5 Working with Commands.....	49
5.1 Command Overview.....	49
5.1.1 Command Categories and Descriptions.....	49
5.1.2 Command Properties Pane.....	52
5.1.3 Command Context Menu.....	54
5.2 Specifying Device Input.....	55
5.2.1 Text Entry Method.....	55

5.2.2	Text Format.....	56
5.2.3	Other Settings in Send Keys	57
5.2.4	Performing Device Hardware Operations	58
5.3	Reference Points	58
5.3.1	Image-Based Reference Points	59
5.3.2	Text-Based Reference Points	67
5.3.3	Audio Reference Points.....	77
5.4	Proofs.....	78
5.4.1	Checkpoint Control	78
5.4.2	Proofs in Execute Action.....	79
5.4.3	Commands in the Capture Category	80
5.5	Parameters and Variables	82
5.5.1	Parameters	83
5.5.2	Variables.....	86
5.5.3	Extract Text Command	93
5.5.4	Working with Data Sets	97
5.6	Error Definitions.....	102
5.6.1	Error Categories	103
5.6.2	Error Types	104
5.6.3	Implementing Error Messaging in Your Test Script	105
5.7	Web Elements in Commands	107
5.7.1	Requirements.....	108
5.7.2	Searching for Elements.....	110
5.7.3	Choosing an Action for a Selected Element	121
5.8	Native Objects in Commands.....	126
5.8.1	Requirements.....	127
5.8.2	Selecting a Native Object	127
5.8.3	Acting on a Selected Object	133
5.9	Script Logic	137
5.9.1	Branches	137
5.9.2	Loops	145
5.9.3	Script Termination.....	148
6	States.....	151
6.1	Creating a State.....	151
6.1.1	Implementing a State.....	152
6.2	State Properties Dialog Box	154
7	Automated Test Cases	155
7.1	Creating a Test Case	155
7.2	Test Case Properties Dialog Box	157
7.2.1	Overview Tab	157
7.2.2	Parameters Tab.....	157
7.2.3	Devices Tab.....	158
7.2.4	Run On Server Tab	158

8	Automated Test Cycles.....	160
8.1	Creating a Test Cycle.....	160
8.2	Test Cycle Properties Dialog Box.....	163
8.2.1	Overview Tab.....	163
8.2.2	Primary Devices Tab.....	163
8.2.3	Slot Assignments Tab.....	164
9	Managing Scripts and Schedules.....	166
9.1	Opening and Checking Out an Asset.....	166
9.2	Checking In and Closing an Asset.....	167
9.3	Merging and Splitting Implementations.....	168
9.4	Rolling Back to Previous Versions.....	170
9.5	Removing Assets.....	171
9.6	Removing Implementations.....	171
9.7	Searching for References.....	172
9.8	Other Functions.....	173
10	Executing Scripts and Viewing Results.....	174
10.1	Validation.....	174
10.1.1	Running Validation.....	174
10.1.2	Viewing Validation Results.....	175
10.2	Scheduling Script Runs.....	178
10.2.1	Test Cycles.....	178
10.2.2	Test Cases.....	183
10.3	Executing Scripts Ad Hoc.....	184
10.4	Execution Summaries.....	188
10.5	Viewing Results.....	190
10.5.1	Result Bar.....	190
10.5.2	Script Result Window.....	192
10.5.3	Uploading Results to the Web Portal.....	197
10.5.4	Viewing Results in the Web Portal.....	198
11	Examples.....	206
11.1	Action: Navigating to the Web.....	206
11.2	Action: Sending an SMS Message.....	209
11.3	Action: Adding a Contact.....	211
11.4	Action: Unpartitioned Web Action.....	214
11.5	State: Application Home Screen.....	215
11.6	State: Create Email Message Screen.....	216
11.7	Test Case: Requesting and Receiving the Weather Forecast.....	216
11.8	Multi-Device Test Case: Audio Validation.....	218
11.9	Test Cycle: Saving and Using Extracted Text.....	221

12	Command Reference	223
12.1	Find and Touch	223
12.1.1	Touching an Image	223
12.1.2	Touching a Text String	227
12.1.3	Touching a State	230
12.1.4	Swipe	232
12.2	Send Keys	234
12.3	Play Audio	234
12.4	Hardware Extension	236
12.5	Wait Event	237
12.5.1	Waiting for an Image	238
12.5.2	Waiting for Text	242
12.5.3	Waiting for Audio	244
12.5.4	Waiting for a State	246
12.5.5	Waiting for a Web Element	247
12.5.6	Waiting for an Object	250
12.6	Extract Text	252
12.7	Branch	255
12.8	Set Variable	257
12.9	Loop	257
12.9.1	Looping Based on Loop Conditions	257
12.9.2	Looping Over a Data Set	259
12.10	Wait	261
12.11	Success	261
12.12	Fail	262
12.13	Continue	263
12.14	Navigate To	264
12.14.1	Using an Image-Based Reference Point	266
12.14.2	Using a Text-Based Reference Point	268
12.14.3	Using a State	270
12.15	Execute Action	270
12.16	Execute Test Case	271
12.17	Execute Cleanup Action	273
12.18	Reset	274

12.19	Load Application	274
12.20	Browser Open	276
12.21	Close All Browser Sessions	276
12.22	Toggle Recording	277
12.23	Capture from Device	278
12.24	Toggle Extract Log	279
12.25	Web Element	280
12.26	Web Wait	284
12.27	Web Form	287
12.28	Web Touch	289
12.29	Launch App	291
12.30	Close App	292
12.31	Object Touch	293
12.32	Object Edit	294
12.33	Object Extract Text	295
12.34	Timer	297
Glossary		298

About This Document

This document describes how to write automated scripts in Keynote's DeviceAnywhere Studio client application. Keynote customers with licenses for DeviceAnywhere Enterprise Automation, DeviceAnywhere Enterprise Monitoring, and Mobile App Monitoring all use DeviceAnywhere Studio for writing automated scripts while interacting with real, live mobile devices.

- ◆ DeviceAnywhere Enterprise Automation is a component of DeviceAnywhere Enterprise, a shared service or dedicated infrastructure enabling you to record, script, schedule, and run automated tests on real, live mobile devices.
- ◆ DAE Monitoring is an enterprise-class service for monitoring mobile network and application availability and performance on smart devices. With DAE Monitoring, production support teams can easily write/record monitor scripts and schedule them at any frequency on real, live devices.
- ◆ Mobile App Monitoring is an enterprise-class monitoring service for mobile devices. Script scheduling takes place in the *Keynote Service Center* and results can be viewed in *MyKeynote*.

This document covers the automation scripting workflow in these products and explains in detail how you can use its powerful features to create, run, and view the results of test scripts that are portable across mobile devices.

NOTES This document does not cover the creation of transactions, monitors, or alerts for DAE Monitoring or timers for Mobile App Monitoring scripts.

Sections in this document that do not apply to specific products are called out.

Document Outline

In this document:

[Prerequisites and Overview](#) describes DeviceAnywhere Studio system requirements. It also introduces other product components of DAE Automation, DAE Monitoring, and Mobile App Monitoring.

[Concepts and Test Scripting Workflow](#) defines script concepts and building blocks and then describes how they fit into the scripting workflow.

[Projects](#) contains information on creating, opening, and managing projects.

[Actions](#) contains information on creating and implementing actions (basic units of scripting). Scripting actions by directly *recording* device interaction is covered.

[Working with Commands](#) contains information on core skills required to use commands in visual scripts:

- Using command properties
- Specifying device input
- Creating reference points
- Setting proofs for display in test results
- Working with parameters and variables
- Implementing error definitions
- Working with web elements

- Working with native objects
- Implementing script logic

[States](#) contains information on creating and implementing states (device-independent reference points).

[Automated Test Cases](#) describes how to create test cases consisting of actions and states

[Automated Test Cycles](#) describes how to create DAE Automation test cycles consisting of test cases selected to run on a subset of project devices.

[Managing Scripts and Schedules](#) describes common functions available for managing actions, states, test cases, DAE Automation test cycles, and DAE Automation schedules.

[Executing Scripts and Viewing Results](#) describes how to validate scripts, schedule DAE Automation test cycle and test case runs, execute ad hoc runs and view their results in DeviceAnywhere Studio and the DeviceAnywhere results and administration web portal for DAE Automation results.

[Examples](#) contains sample action, state, test case, and test cycle scripts.

[Command Reference](#) contains a field-by-field description of each visual scripting command.

[Glossary](#) contains brief definitions of key DeviceAnywhere Studio concepts and terminology.

Typographical Conventions

The table below describes the typographical conventions used in DeviceAnywhere documentation.

Style	Element	Example
Blue	Links and email addresses	http://www.keynote.com The Document Outline section describes the structure of this manual.
Bold	User interface elements such as menu items	Click My Devices in DeviceAnywhere Studio.
Monospace	Commands, code output, filenames, directories	Right-click the project's <code>test cases</code> directory.
Monospace bold	User input	In a command window, type adb devices .
<i>Italic</i>	Document titles and emphasis	Refer to the <i>DeviceAnywhere Enterprise Private System Installation Guide</i> for instructions on setting up server infrastructure.

Contacting Support

If you have any comments or suggestions regarding this document, contact Keynote Support. For inquiries about DeviceAnywhere product demonstrations and consulting services, contact your Keynote Solutions Consultant.

Customers can find additional support information at <http://support.keynote.com> or 1-888-KEY-SYST (539-7978).

Additional Documentation

You can find additional information at:

- ◆ DAE Automation: <http://www.keynotedevicewhere.com/testing-automation-documentation.html>
- ◆ DAE Monitoring: <http://www.keynotedevicewhere.com/monitoring-documentation.html>
- ◆ DeviceAnywhere Enterprise shared system with private devices:
<http://www.keynotedevicewhere.com/dae-pvt-devices-documentation.html>
- ◆ Mobile App Monitoring: <http://www.keynote.com/support/mobile-app-monitoring-documentation.html>

You can access documentation from the **Help** menu in DeviceAnywhere Studio.

1 Prerequisites and Overview

This chapter describes [prerequisites](#) for using DAE Automation, DAE Monitoring, and Mobile App Monitoring, and includes a brief tour of the [product components](#) of each of these products.

1.1 Prerequisites

This document assumes that you are familiar with the following operations in DeviceAnywhere Studio:

- ◆ Interacting with devices
- ◆ Creating projects

To use DAE Automation, DAE Monitoring, or Mobile App Monitoring, you require the following:

- ◆ DeviceAnywhere Studio client software, pointing to a fully operational and networked DAE or Mobile App Monitoring environment (including mobile devices, device servers, Access Server, SQL Server, and other components)—see [System Requirements for DeviceAnywhere Studio](#) below.
- ◆ User credentials in your customer account enabling you to log in to DeviceAnywhere Studio and the appropriate web portal (DeviceAnywhere results and administration portal, DAE Monitoring portal, or MyKeynote and Keynote Service Center, depending on your product), —if you require credentials, please contact your system administrator.
- ◆ A license to use DAE Automation, DAE Monitoring, or Mobile App Monitoring

NOTES In dedicated DAE Automation and Monitoring environments, the Local Device Server for locally attached devices and DeviceAnywhere Studio software may be installed on the same machine. The Local Device Server supports two devices concurrently attached to the server.

Components of a DeviceAnywhere test environment require a network connection to communicate with each other. DeviceAnywhere Studio must be able to communicate with other infrastructure components, whether hosted by Keynote or standalone behind your corporate firewall.

1.1.1 System Requirements for DeviceAnywhere Studio

Minimum system requirements are as follows:

- ◆ 2 GHz dual-core processor (Pentium 4)
- ◆ 2 GB RAM
- ◆ At least 10 GB available hard disk space
- ◆ Monitor with at least 1600 x 900 screen resolution
- ◆ Optional: Audio card for sound input/output
- ◆ Supported operating systems: Windows Vista, Windows 7, Windows Server 2008, Mac OS 10.2+

NOTES DeviceAnywhere Studio can be installed on Windows machines with 32-bit or 64-bit operating systems.

Linux is not supported for automated testing.

JDK v1.7+ must be installed for Mac OS.

1.2 Product Components

This section describes the components of [DAE Automation](#), DAE Monitoring, and Mobile App Monitoring.

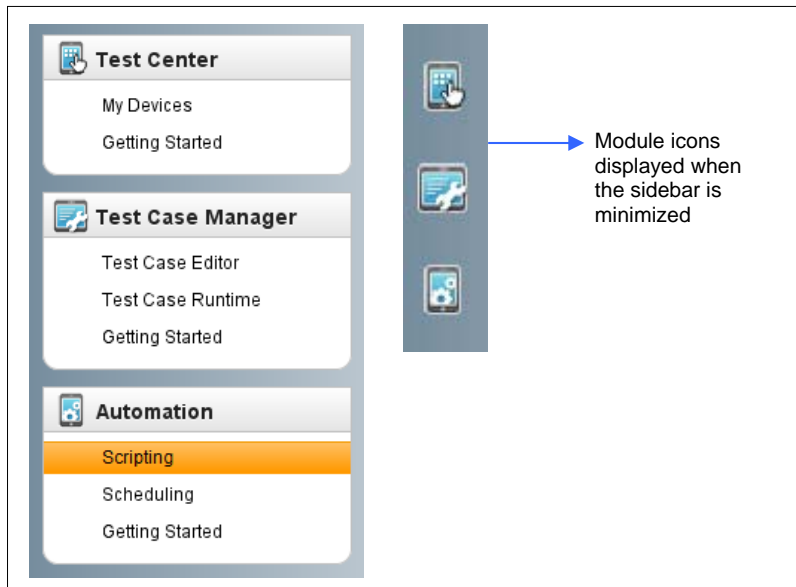
1.2.1 DeviceAnywhere Enterprise Automation

DAE Automation includes modules, or views, in [DeviceAnywhere Studio](#), a [web portal](#), and a [Java API](#).

1.2.1.1 DeviceAnywhere Studio Modules

The DeviceAnywhere Studio client application is the primary interface for interacting with devices and defining, managing, and running test scripts (manual as well as automated). DeviceAnywhere Enterprise includes the [Automation](#), [Test Center](#), and [Test Case Manager](#) modules, or views, which you can access from the application sidebar. Users will also see a module for **Links** for the portal and reporting an issue.

Figure 1-1 DeviceAnywhere Studio Sidebar—Module List and Icons



Automation



You can define automated test assets ([actions](#), [states](#), [test cases](#), [test cycles](#)) in this view. You can schedule or run scripts and debug them ad hoc, and view and upload results to the DeviceAnywhere results and administration web portal from here.

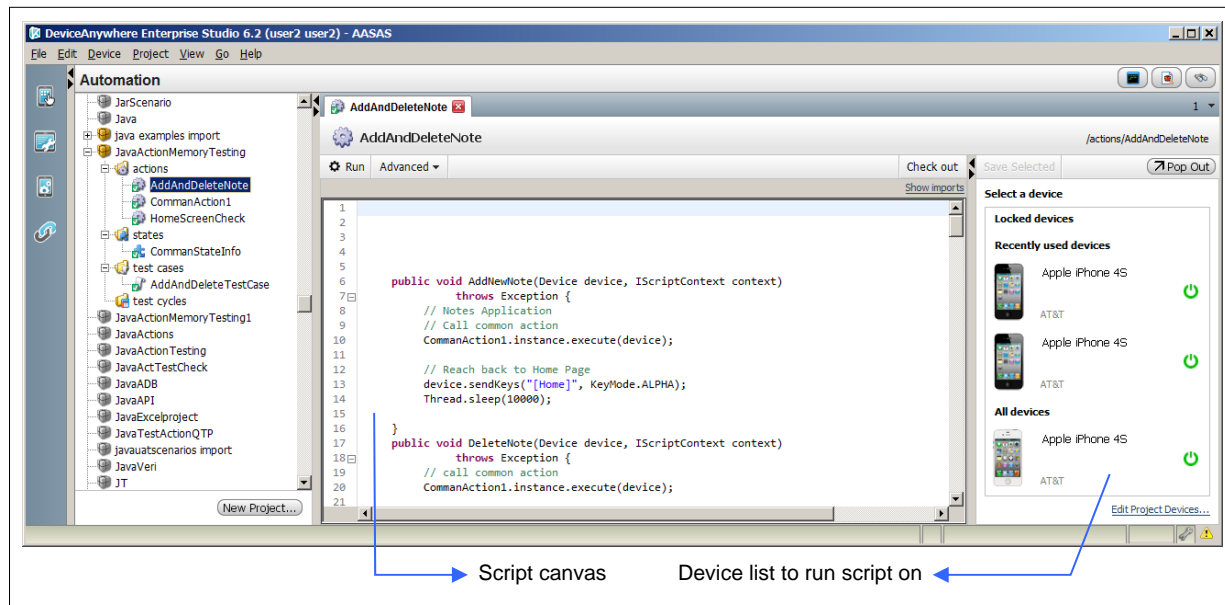
The Automation view shares the same library of [projects](#) as [Test Case Manager](#) but only lists automated test assets.

In the **Scripting** tab of the Automation view below, you can see the visual (Figure 1-2) and Java (Figure 1-3) scripting environments.

Figure 1-2 Automation Visual Scripting Environment

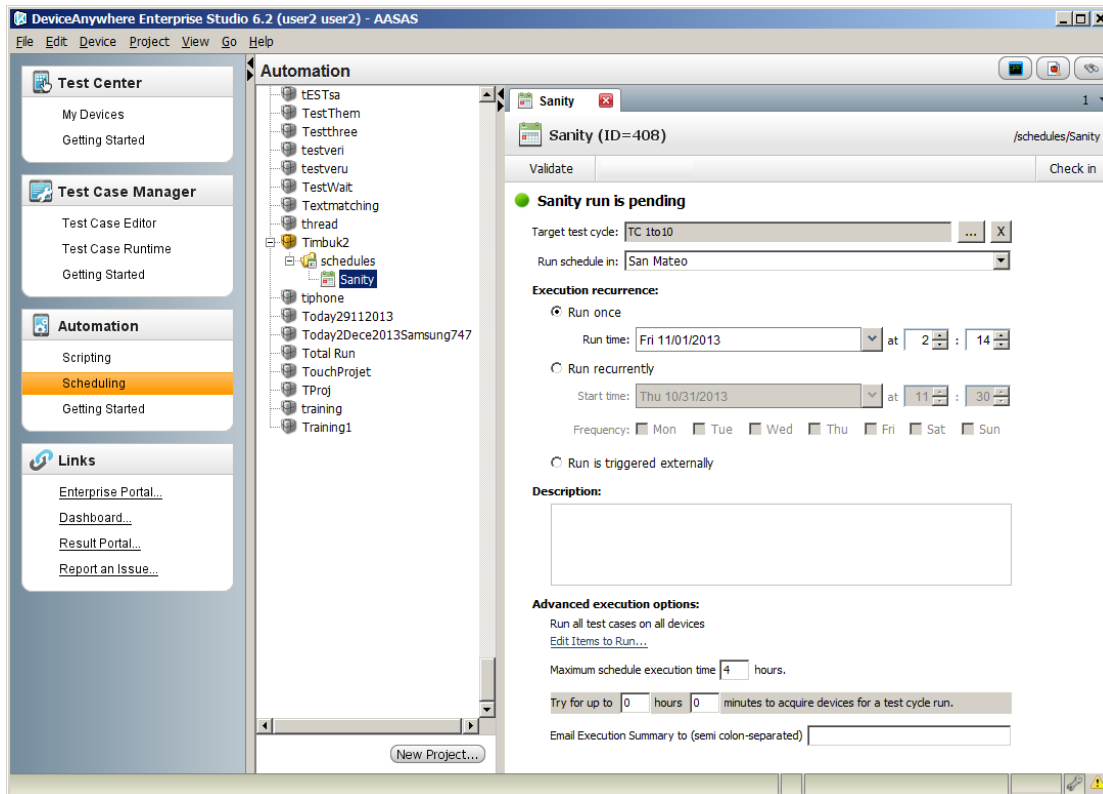


Figure 1-3 Automation Java Interface



The **Scheduling** tab in the Automation view enables you to create schedules for test cycle runs. You can select the test cases and the devices that you want to run them on and schedule them to run only on specific days of the week or hours of the day

Figure 1-4 Automation Scheduling Tab




Test Center

This view allows you to view devices and interact with them. You can also interact with devices in the Automation view.

Figure 1-5 Test Center View

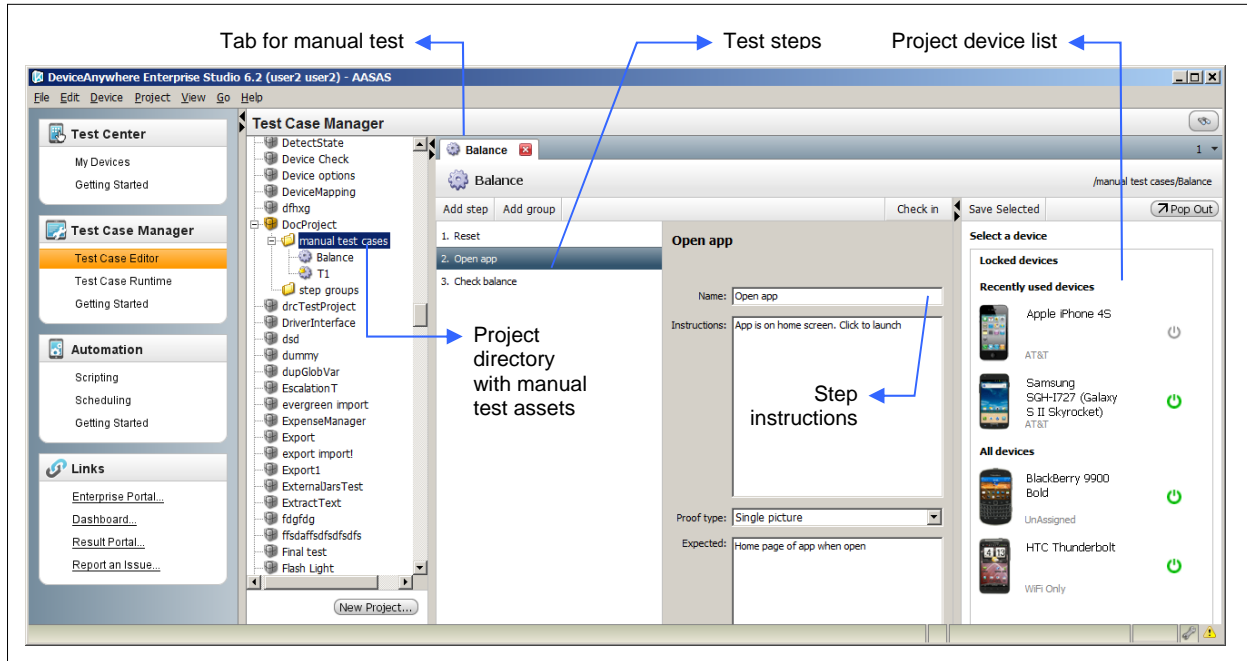


Test Case Manager

 You can define and run manual [test cases/cycles](#) in this view. It is often used in association with DeviceAnywhere Enterprise Automation to automate manual test cases, run mixed manual and automated test cases/ cycles, and view results. Test Case Manager consists of [Test Case Editor](#) and [Test Case Runtime](#).

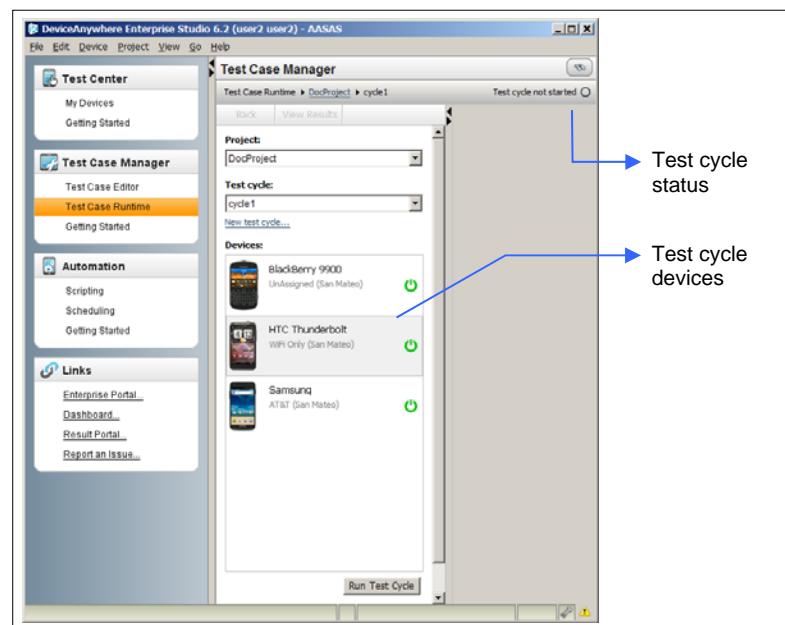
Test Case Editor is the interface for creating manual test cases. You can also link manual test case steps to automated scripts from here.

Figure 1-6 Test Case Editor



You can create manual test cycles run manual and partially automated test cases/cycles from Test Case Runtime. You can also track the progress of [test cycles](#) and view uploaded results from here.

Figure 1-7 Test Case Runtime



1.2.1.2 DeviceAnywhere Results and Administration Web Portal

After executing a test, you can view results in DeviceAnywhere Studio and/or the DeviceAnywhere results and administration web portal. The portal acts as a central repository for viewing and sharing test results. You can also manage your account from here. When you have logged in, click **Test Automation Results**.

NOTE Customers with a license for mobile test automation in the cloud-based DeviceAnywhere Enterprise service can access this page directly from DeviceAnywhere Enterprise Portal.

Figure 1-8 List of Test Results in the Portal

Update Time	Project	Test Unit	Test Unit Name	Primary Device	Result	Properties
10/18/2012 03:55:04 PM	DocProj3	TestCycle	Cycle1 - Execute Test Case	HTC Sensation 4G	Fail	
10/18/2012 03:09:58 PM	DocProject	Action	Bb	HTC Sensation 4G	Success	
10/18/2012 03:01:51 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 03:00:40 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:59:29 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:58:15 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:57:33 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:56:34 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:54:54 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:51:04 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:31:49 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 4	Fail	
10/18/2012 02:30:33 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 4	Success	
10/18/2012 02:30:19 PM	DocProject	Action	Bb	HTC Sensation 4G	Not Completed	
10/18/2012 02:29:02 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 4	Not Completed	
10/18/2012 02:16:51 PM	DocProject	Action	Bb	HTC Sensation 4G	Fail	
10/18/2012 12:44:51 PM	import and export project import12	Action	Sendsms	Apple iPhone 4	Success	

Click the **Success** or **Fail** link next to a script to see detailed results.

1.2.1.3 Java API

Use the Java API to create Java test and monitor scripts in DeviceAnywhere Studio or an IDE of your choice. The API also facilitates integrations with industry standard test automation and management tools from HP and IBM.

1.2.2 DeviceAnywhere Enterprise Monitoring

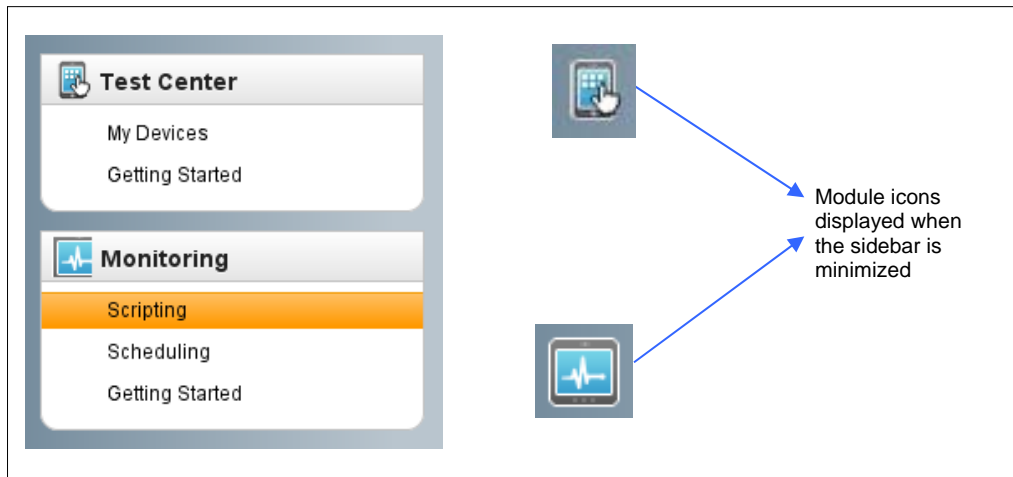
DAE Automation includes modules, or views, in [DeviceAnywhere Studio](#), a [web portal](#), and a [Java API](#) (described above).

1.2.2.1 DeviceAnywhere Studio

DAE Monitoring includes the [Monitoring](#) and [Test Center](#) modules, or views, which you can access from the application sidebar. The [Monitoring](#) module enables you to write and schedule monitor scripts. The [Test Center](#) module for device interaction is the same as seen by users of [DAE Automation](#).

You also have access to Links for the DAE Monitoring Portal and for reporting an issue to Keynote support.

Figure 1-9 DeviceAnywhere Studio Sidebar—Module List and Icons



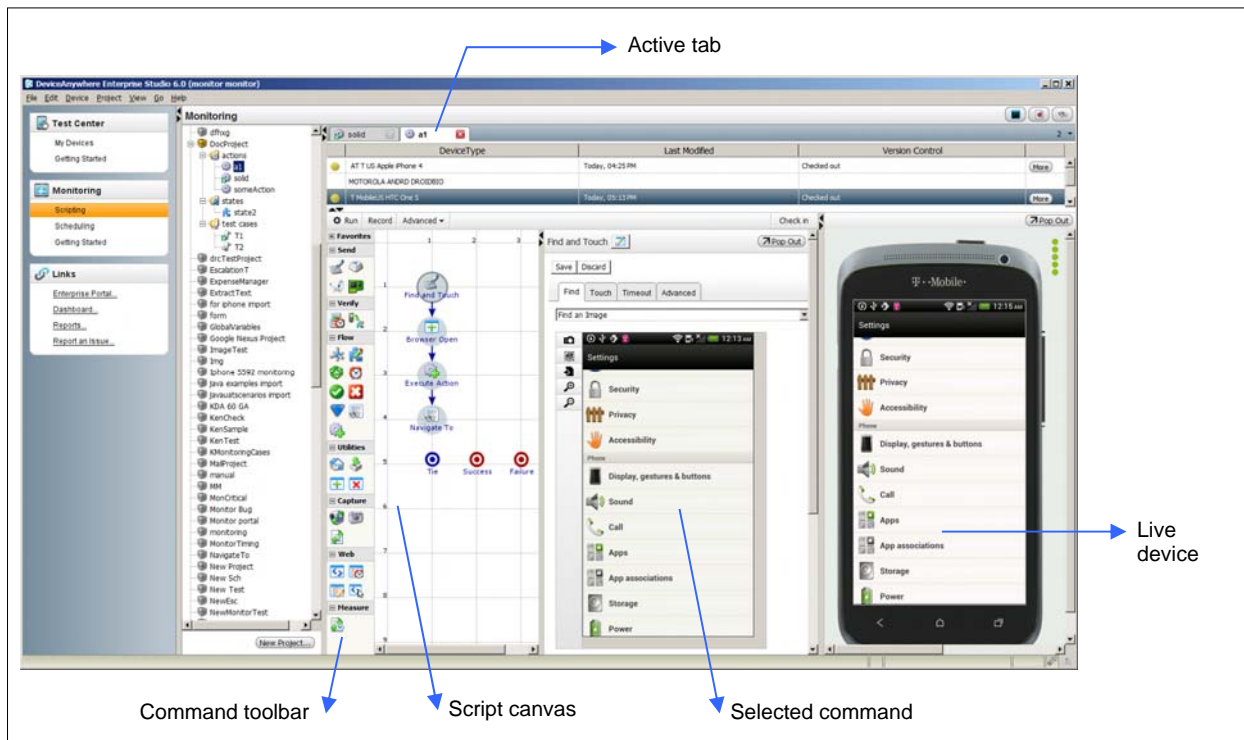
Monitoring



You can define automated monitor scripting ([actions](#), [states](#), [test cases](#)) and scheduling assets (transactions, monitors, QoS polices, alerts) in this view. You can schedule monitors or run and debug them ad hoc, and access the [DAE Monitoring Portal](#) from here.

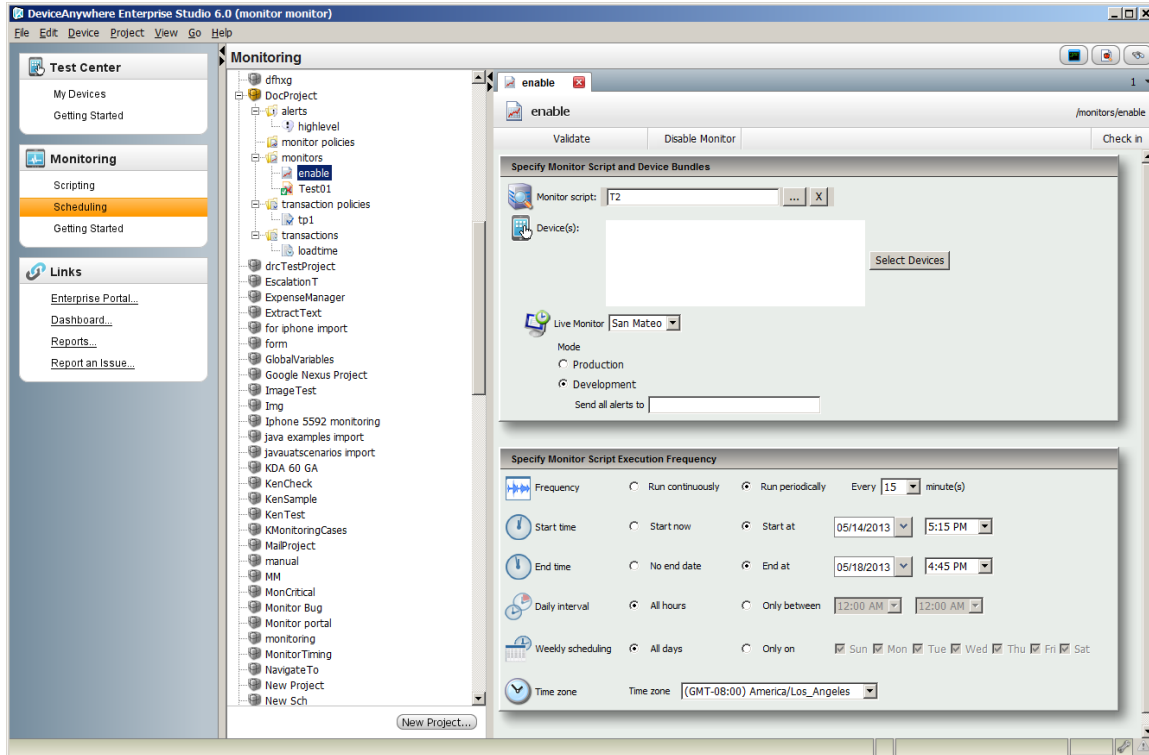
Figure 1-10 below shows the **Scripting** tab of the Monitoring view with an open script showing the visual scripting environment.

Figure 1-10 DAE Monitoring Visual Scripting Environment



The **Scheduling** tab in the Monitoring view enables you to set up transaction timers and policies, monitors and their schedules, monitor policies, and alerts triggered when policies are violated. Figure 1-11 below shows the interface for scheduling a monitor.

Figure 1-11 Scheduling Tab



Test Center

See the [Test Center](#) section above for a brief description of the Test Center module. You can also perform the same functions in the Monitoring module.

1.2.2.2 DeviceAnywhere Enterprise Monitoring Portal

The DAE Monitoring Portal provides a real-time dashboard view of currently running monitors and live device screens as scripts are executed on them. Users can also view historical monitor data such as a list of all monitor executions, success rates for individual monitors, error reports, trend charts, and detailed results for failed script runs containing device screenshots. All standard reports can be customized for display using filters and date ranges. Users can save report data or report criteria and schedule and generate reports from them at any time. You can also download aggregate or raw data used to generate a chart.

Figure 1-12 Portal Landing Page

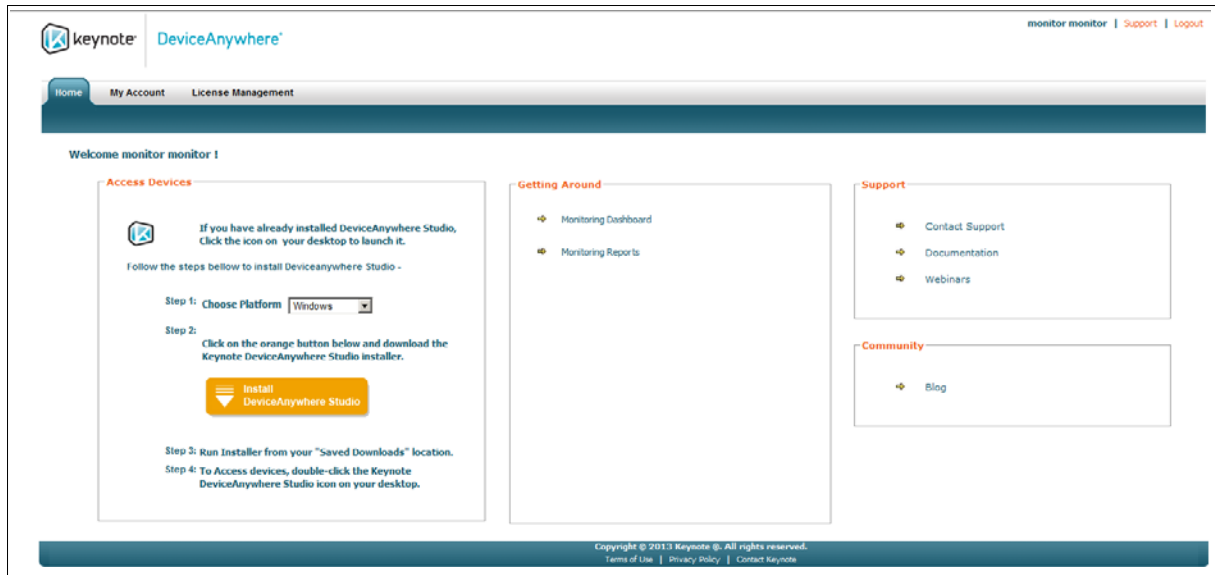


Figure 1-13 Chart — Transaction Performance (Run Times) on Different Devices

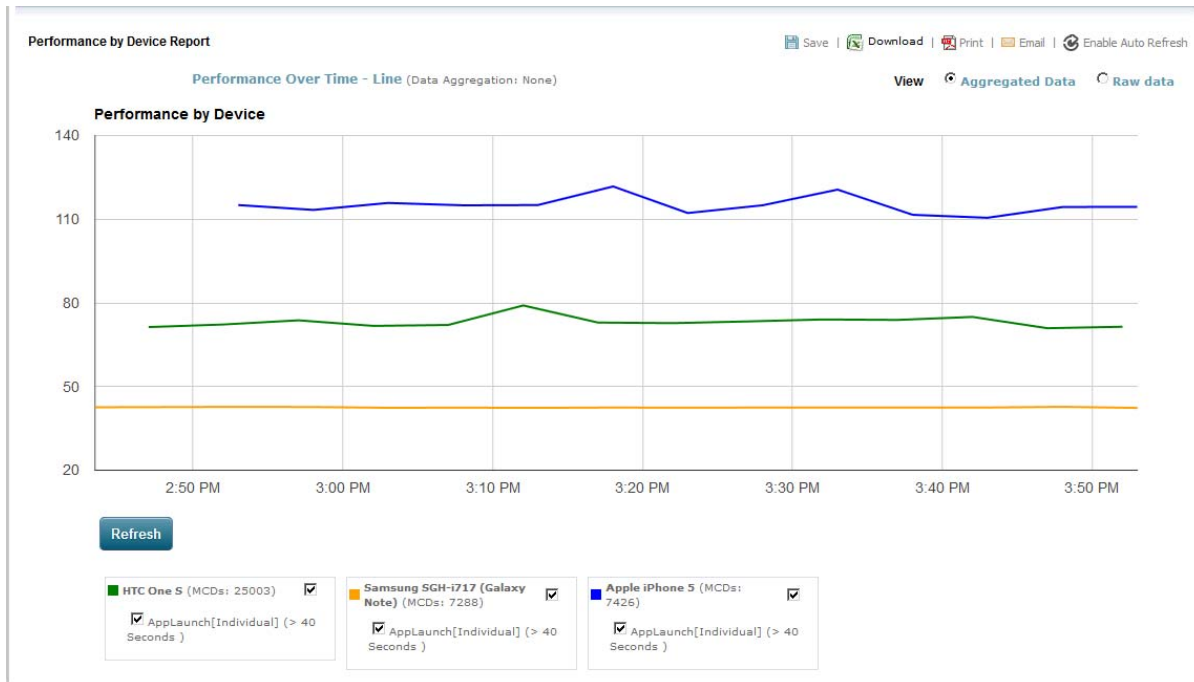


Figure 1-14 Dashboard

Project	Monitor Name	Device Name	Location	Last Status	Last Result	Next run	Trend	Attach
atest6newwebtest	atest6mon1	A_Apple iPhone 4S	San Jose	RUNNING	ERROR			Attach
TestingWaitimagechanges	Waitimage01	HTC One S	San Jose	RUNNING	SUCCESS			Attach
TestingWaitimagechanges	Waitimage02	Samsung SGH-I717 (Galaxy Not	San Jose	RECURRING	SUCCESS	5/15/2013 12:41:00		Attach

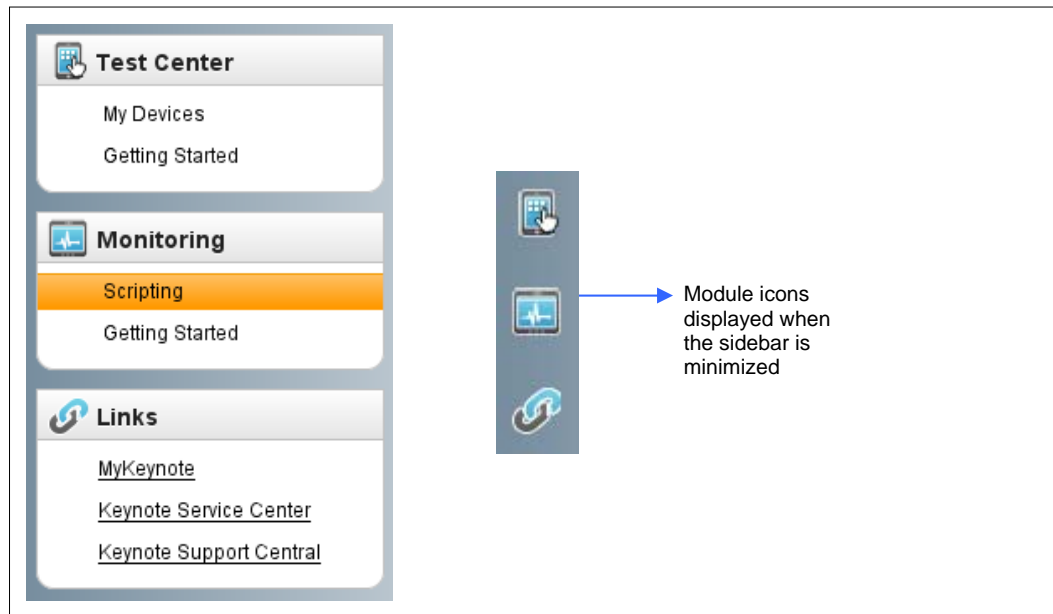
1.2.3 Mobile App Monitoring

Mobile Application Automation includes modules, or views, in [DeviceAnywhere Studio](#), the [MyKeynote](#) results portal, and the [Keynote Service Center](#) (KSC).

1.2.3.1 DeviceAnywhere Studio

Mobile App Monitoring includes the [Monitoring](#) and [Test Center](#) modules in DeviceAnywhere Studio. The [Monitoring](#) module enables you to write monitor scripts. The [Test Center](#) module for device interaction is the same as seen by users of [DAE Automation](#). You also have access to Links for MyKeynote, the Keynote Service Center, and for reporting an issue to Keynote support.

Figure 1-15 DeviceAnywhere Studio Sidebar—Module List and Icons



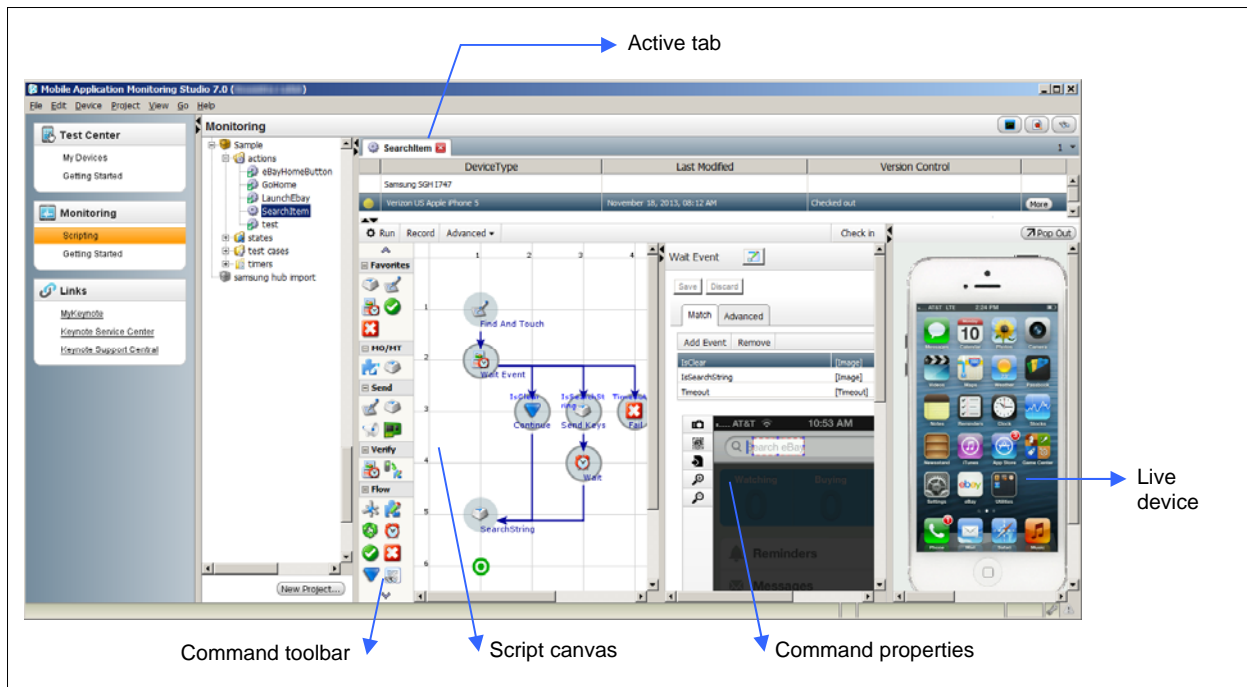
Monitoring



You can define automated monitor scripting assets ([actions](#), [states](#), [test cases](#), timers) in this view. You can run and debug scripts ad hoc from here. Figure 1-16 below shows the **Scripting** tab of the Monitoring view with an open script showing the visual scripting environment.

NOTE Mobile App Monitoring does not have a **Scheduling** tab as all provisioning is done in the [Keynote Service Center](#).

Figure 1-16 Mobile App Monitoring Visual Scripting Environment



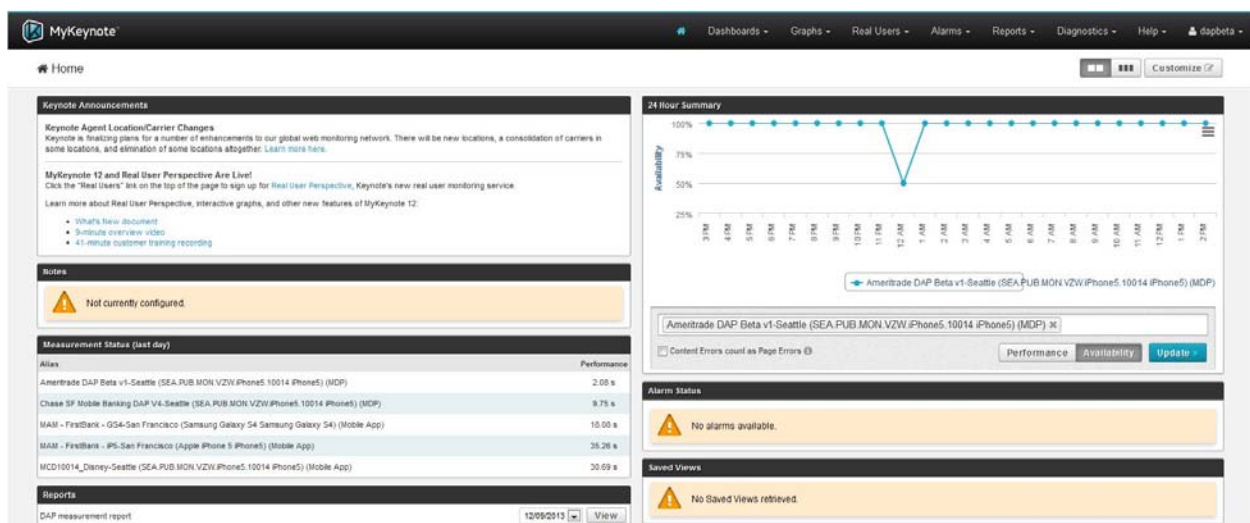
Test Center

See the [Test Center](#) section above for a brief description of the Test Center module. You can also perform the same functions in the Monitoring module.

1.2.3.2 MyKeynote

MyKeynote is a powerful web-based tool for visualization, analysis, and reporting the results of Mobile Application Monitoring scheduled script runs. You can also configure alarms from here. You can log in to MyKeynote at www.mykeynote.com or by clicking **MyKeynote** in the Links view of DeviceAnywhere Studio.

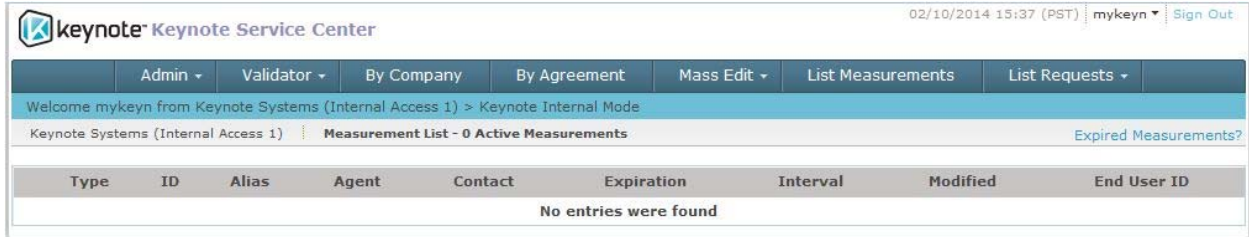
Figure 1-17 MyKeynote Landing Page



1.2.3.3 Keynote Service Center

Users with administrative rights can schedule scripts on real devices in the Keynote Service Center. You can also manage your account from here.

Figure 1-18 Keynote Service Center Landing Page



2 Concepts and Test Scripting Workflow

The test scripting workflow in DeviceAnywhere Studio is best understood with reference to the basic building blocks and hierarchy of scripting concepts. Although script units and the workflow can vary depending on the Keynote product you are using, the best practice approach to scripting in DeviceAnywhere Studio consistently leverages building blocks to define modular scripts. With DeviceAnywhere Studio's powerful tools for creating and organizing the reusable script components, you can minimize repetitive coding.

Scripts can be object-based, operating across device models, or consist of device-specific implementations to account for differences in device interfaces.

This chapter lays the foundation for sound scripting practices by defining script [concepts](#) and building blocks and then describing how they fit into the [scripting workflow](#).

2.1 Concepts

Project

A project is an organizational container for test scripts and test devices, generally specific to an application or functionality you wish to test.

Projects consist of test assets, ([actions](#), [states](#), [test cases](#), and [test cycles](#)) which are the main building blocks used to automate your mobile testing. Projects in Mobile App Monitoring also contain [timers](#), to mark and identify portions of a script to be measured. While schedules are created in MyKeynote for Mobile App Monitoring, DeviceAnywhere Studio projects contain [schedules](#) for executing DAE Automation and DAE Monitoring test cycles. Project metadata includes permissions, error definitions, project variables, and project dependency information.

See the chapter on [Projects](#) for information on creating and managing projects.

Test Case

A test case describes the broad process to accomplish a test or monitoring goal on your mobile application, device, or content. For example, a test case might consist of viewing and deleting call records from a device or searching for an item in a retail application. In DAE Monitoring and Mobile App Monitoring, test cases are deployed as monitor scripts.

Test cases are device-independent. They generally consist of calls to previously defined [actions](#) and [states](#) with additional commands to control script logic. See the chapters on [Test Cases](#) and [Working with Commands](#) for information on creating, managing, and adding commands to a test case; see [Examples](#) for sample test cases.

Action

Actions are the basic building blocks of a [test case](#) and units of scripting. You can think of actions as the discrete procedures that make up the larger process that is a test case—each broad step in a test case corresponds to an action and consists of a series of device interactions and verifications. While the size of actions is not limited, they are best thought of as smaller scripts that can be reliably reused across devices and test cases.

Actions are device-independent, that is, they are defined for all project devices. They can be unpartitioned, object-based scripts or consist of device-specific [implementations](#) to account for differences in interfaces. Examples of actions in a test case that tests a web site are resetting a device, opening a browser, and navigating to a URL.

State

States define known device conditions (in terms of text, an image, audio, or a web element from a device screen) that can be referenced to verify the result of an action or a sequence of device interactions. States identify and define a device screen, such as the device home screen or the browser home page, in order to specify an expected result and can be reused across scripts and devices.

States are device-independent, that is, they are defined for all project devices. They can be unpartitioned and object-based or consist of device-specific [implementations](#) to account for differences in interfaces. If an expected result changes over time, a state can be updated in one place, without having to update every script containing a reference to it.

See the chapters on [States](#) and [Working with Commands](#) for information on creating states and device-specific state implementations.

Implementation

If you are not creating object-based scripts that operate across all devices in your project, you will need to create [action](#) or [state](#) implementations to account for differences in device interfaces. An implementation contains the specific commands or device screens that make up an action or state on a particular device.

Implementations can be created from scratch or cloned and altered from other implementations for like devices. Refer to the chapters on [Actions](#) and [States](#) for information on creating implementations; see [Examples](#) for sample action and state implementations.

Test Cycle

A test cycle defines your test plan and consists of a group of [test cases](#) assigned to be run on all or a subset of project devices. Test cycles allow you to create device-test case groupings to focus testing on particular functional areas or test targets.

Test cycles can be run ad hoc or scheduled to run independently. Refer to the chapters on [Automated Test Cycles](#) and [Executing Scripts and Viewing Results](#) for information on creating and executing test cycles.

NOTE DAE Monitoring and Mobile App Monitoring users do not create test cycles.

Timer

Mobile App Monitoring only—creating timers and then applying them to portions of your script using the Timer command enables you to mark portions of your script to be tracked and measured. You can set up performance thresholds in MyKeynote for these script sections and trigger alarms based on the thresholds.

NOTE The corresponding concept of *transactions* for DAE Monitoring is discussed in detail in the [DAE Monitoring Best Practice Workflow](#).

Parameters and Variables

[Parameters](#) and [variables](#) enable you to implement complex script logic such as data variations, loops, and branches.

Parameters can be defined for [actions](#) or [test cases](#) and allow you to enter data dynamically into a device at the start of a script run. Parameters allow you to specify a different data value for each run. An action parameter, once created, can be used in any implementation. If you are testing multiple web sites on several devices, you can specify a different URL for each script run using a parameter.

Variables allow you to set/store values in your script. Variables can be associated with a single value or multiple values contained in a *data set*. Variables can be *global* or *script specific*:

- ◆ *Script variables* are script specific.
- ◆ *Global variables* are defined at the project level and can be used in any project script.

Refer to the chapter on [Working with Commands](#) for information on defining, calling, and setting parameter and variable values.

2.2 Best Practice Scripting Workflow

The best practice test scripting workflow begins with creating a project to contain your test assets and selecting devices to test. After defining objectives in terms of high-level tests, you begin scripting by creating actions and implementations. You progressively define test building blocks till you are ready to construct modular test cases, which represent the high-level tests you want to execute, and test cycles.

In this workflow, *the actual process of scripting begins with creating actions and implementing them*. The greater part of scripting consists of creating action implementations, and the [visual scripting interface](#) provides the greatest number of commands for doing so. This manual follows the recommended order of creating test assets, which begins with actions, the basic unit of scripting, and leads up to test cycles.

The end-to-end scripting process consists of the following steps:

- 1 Create a [project](#) in the Automation or Monitoring view and add devices to it.
- 2 Conceptualize device-independent [test cases](#) as application-specific processes to accomplish a goal, e.g., check the weather forecast for a specified zip code.
- 3 Analyze your application to identify test case building blocks, or constituent procedures that can be used as the basis for creating reusable [actions](#).
- 4 Plan test cases by identifying smaller actions they consist of.
- 5 If writing monitoring scripts, identify key portions of your scripts that you would like to wrap timers around so you can track and measure them.
- 6 Begin scripting by creating actions to accomplish specific tasks within the larger test case objective, e.g., resetting the device, sending a message, or launching an application.
- 7 Create [states](#) to define known device conditions that can be referenced for verification in scripts and can be reused across scripts.
- 8 Create a [test case](#) consisting of calls to actions and states, and any script logic. Specify proofs (screenshots and video) and error definitions to be displayed in test results. You may also specify proofs in action scripts.

- 9 Validate and debug test scripts by running them and viewing results in DeviceAnywhere Studio. DAE Automation users can upload results to the [DeviceAnywhere results and administration web portal](#).
- 10 DAE Automation only—create [test cycles](#) consisting of subsets of project test cases and devices.
- 11 DAE Monitoring and DAE Automation—schedule your scripts in DeviceAnywhere Studio.
- 12 Publish your project; be sure to publish any dependent projects as well.
- 13 Mobile App Monitoring—provision your measurements in the Keynote Service Center.
- 14 Mobile App Monitoring—set up performance thresholds and alarms in MyKeynote.
- 15 View results in the web portal.

NOTE Please refer to the [DAE Monitoring Best Practice Workflow](#) for:

- ◆ An introduction to other script and scheduling assets contained in DAE Monitoring projects.
 - ◆ A more detailed two-stage workflow specific to DAE Monitoring concepts and building blocks
-

3 Projects

Projects are the organizational containers for all the test assets required to test your mobile application or service across a set of specific devices. Projects contain actions, states, test cases. Projects also contain these other entities based on the product you use:

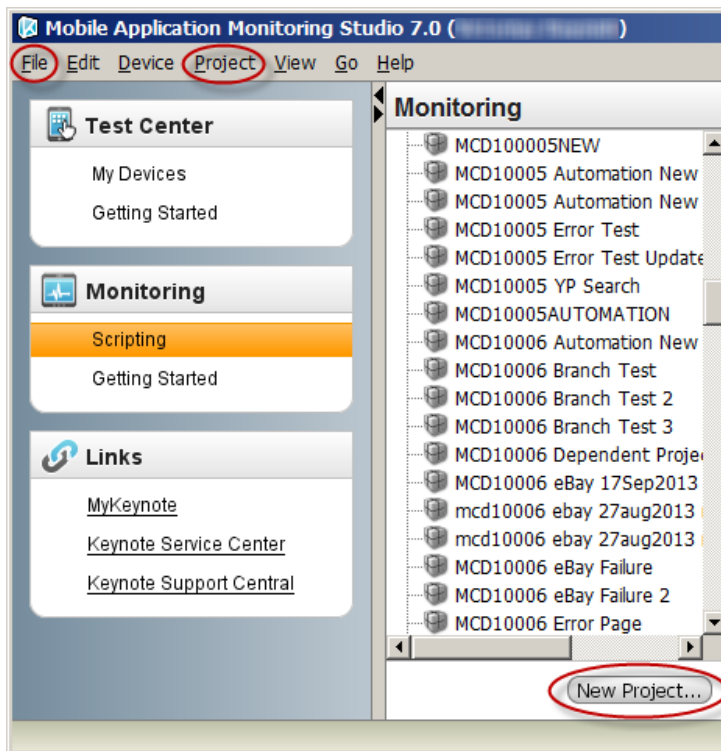
- ◆ *Schedules*—DAE Automation and DAE Monitoring
- ◆ *Timers*—Mobile App Monitoring
- ◆ *Test cycles*—DAE Automation

This chapter contains information on [creating, opening](#), and [managing projects](#). It also describes the folders and icons found in the project directory.

3.1 Creating a Project

You need to create a project to organize your test assets before you can begin scripting.

- 1 Make one of the following selections in the Automation/Monitoring view to create a project:
 - Select **New Project** from the bottom of the project list.
 - With an [open project](#) selected in the project list, select **File > New > New Project**.
 - Select **Project > New Project**.



- 2 Enter a project name in the first screen of the New Project wizard.
- 3 Click **More Options** to display additional controls:

- a Specify the **Version control** you want to use. The default is **DeviceAnywhere Version Control**—this stores revisions to test scripts on your local machine as well as the DeviceAnywhere database. Select **External Version Control System** to use a version control system of your choice.

CAUTION Once specified, the version control setting cannot be changed.

- b Specify project permissions for other users in the **Editable by** control. You can change these values later in the [Permissions Tab](#) of the [Project Properties](#) dialog box.
- 4 Click **Create Project**.
 - 5 This brings up the Project Properties dialog box where you specify devices, dependencies, variables, error types, and permissions. Click **Save** when done setting [properties](#).

Your project will appear in the project list with folders for actions, states, and test cases—this might take a few moments as the necessary files are prepared.

3.1.1 Project Properties

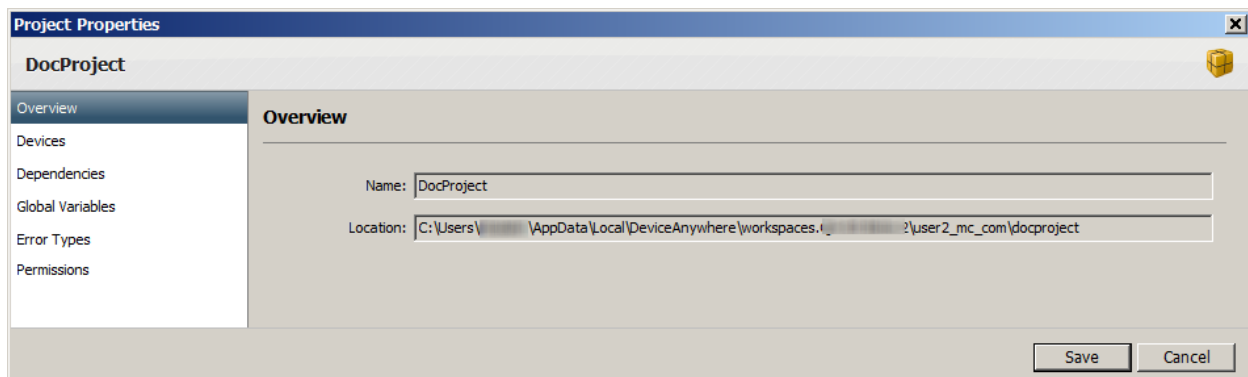
When creating a project, specify [devices](#), [dependencies](#), [variables](#), [error definitions](#), [sharing](#), and [permissions](#) in the Project Properties dialog box. Be sure to **Save** changes to project properties.

NOTE To change project properties at any time, right-click the project in the project list > **Properties**.

3.1.1.1 Overview Tab

View project name and default location on the local file system in the **Overview** tab.

Figure 3-1 Project Properties – Overview



3.1.1.2 Devices Tab

Specify project devices in the **Devices** tab of the Project Properties dialog box.

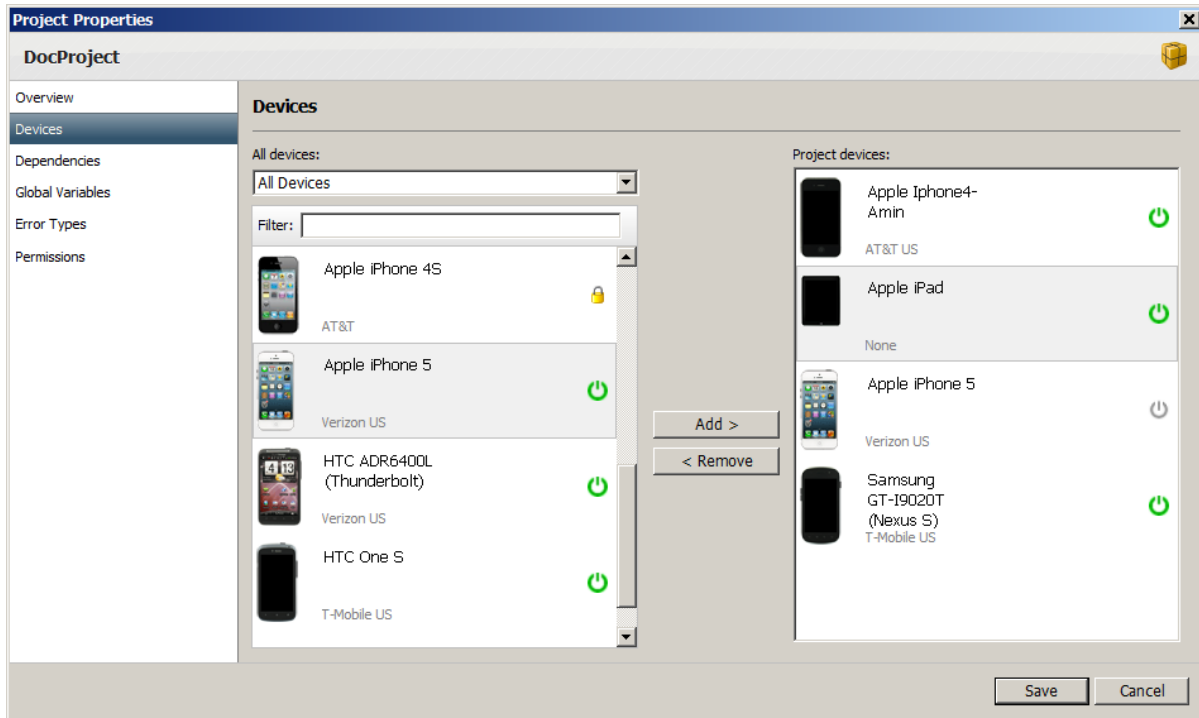
The **All devices** drop-down list displays the package containing your devices. Select a device from the list below and click **Add** to move it to the **Project devices** list; click **Remove** to remove it from the list. Use Ctrl-click to select multiple devices in the left pane; use Shift-click to select a device range.

Use the free-form **Filter** field to search for devices. The device list is dynamically filtered with each character you type. Filtering by the word “online” displays all online devices.

NOTES You may combine devices from different packages in your project.

If you remove a device on which you have executed tests and uploaded results, you will still be able to access the results in the web portal.

Figure 3-2 Devices Tab

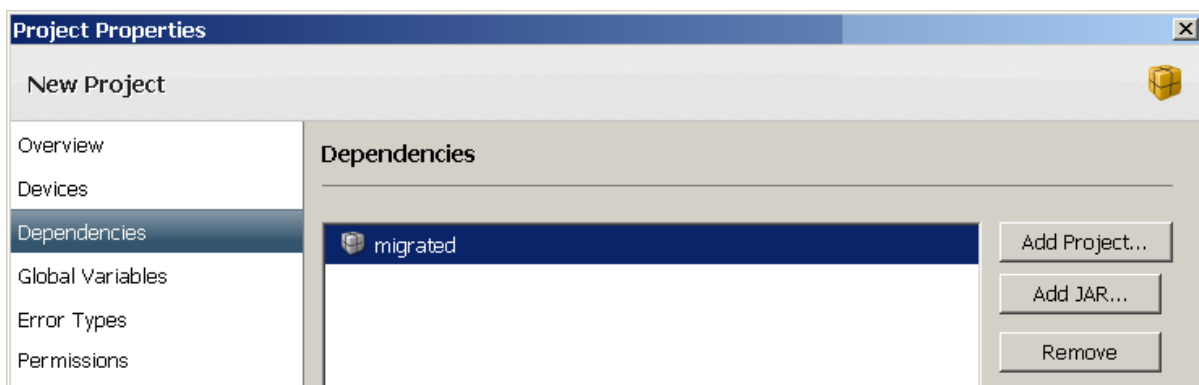


To edit project devices at any time, revisit project properties or click **Edit Project Devices** from an open script.

3.1.1.3 Dependencies Tab

Click the **Dependencies** tab of project properties to specify dependent projects and JAR files. This allows you to reuse the test assets of dependent projects.

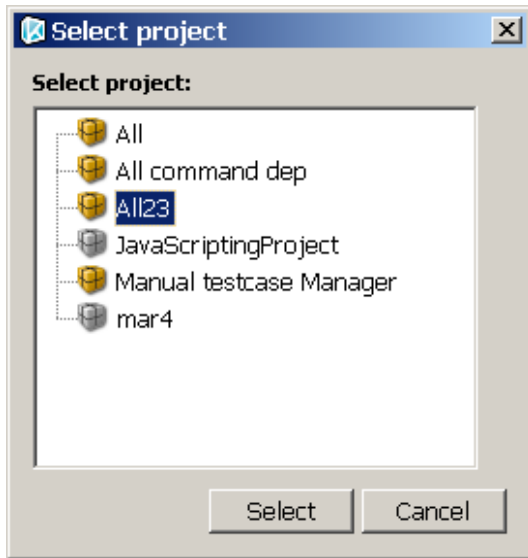
Figure 3-3 Project Properties – Dependencies



To specify a dependent project:

- 1 Click **Add Project**. (Click **Remove** to remove the project.)

- 2 Select a project from the list provided and click **Select**. Open projects and projects that you have loaded at least once are listed.



Your dependent project is now listed in the **Dependencies** tab.

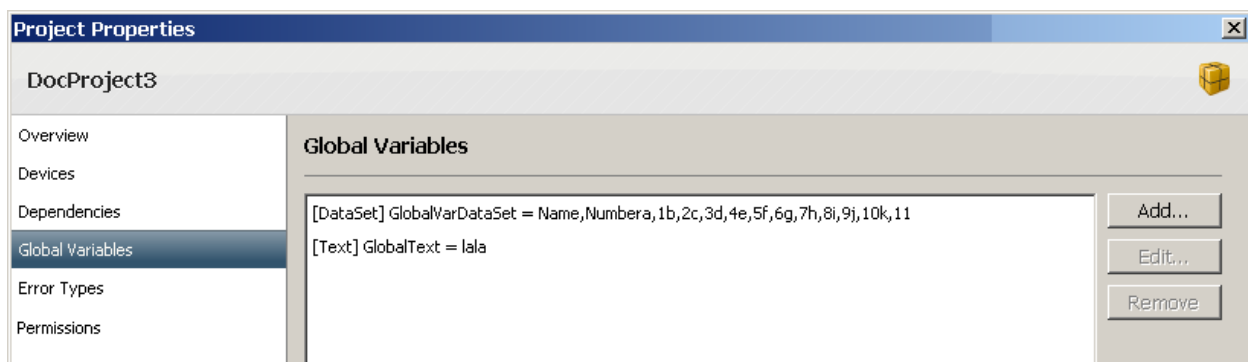
To specify a dependent JAR file:

- 1 Click **Add JAR**.
- 2 Select the file and click **OK**.

3.1.1.4 Global Variables Tab

Global variables can collect and store values from one script for use in another project script. For example, if an action consists of signing up to use a web site, you can store the credentials in a global variable and then call the variable from another action that logs in to and performs a transaction on that site. The **Global Variables** tab of the Project Properties dialog box displays a list of existing project/global variables. You can **Add**, **Remove**, or **Edit** global variables in the **Global Variables** tab.

Figure 3-4 Project Properties – Global Variables



Refer to the chapter on [Working with Commands](#) for detailed information on using [variables](#)—defining them, calling them from commands, and specifying values.

3.1.1.5 Error Types Tab

You can create error definitions and then implement them at various points in your test scripts to generate error messaging. The error management system consists of error categories containing error types. You can define error categories and their constituent error types in the **Error Types** tab of project properties.

When you select the **Error Types** tab, it displays the list of existing error types in a selected error category (see Figure 3-5 below). You can add, edit, and remove error types and default failure messages. You can also create error categories. Later during scripting, you can choose the error type triggered if different commands in your script fail—error types must be selected in command properties.

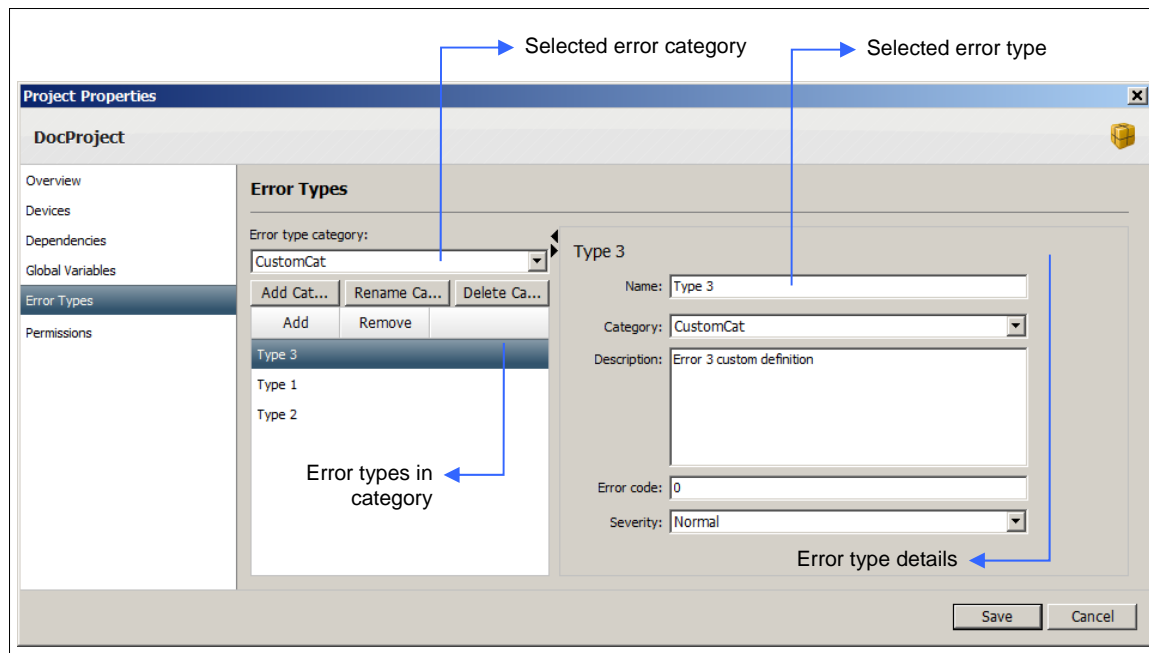
NOTES Mobile App Monitoring users may only create error types in the DAP error category.

The DeviceAnywhere error category that appears in DAE Automation and DAE Monitoring cannot be edited (i.e., you cannot add or remove error types). Do not use errors in this category as they are reserved for system issues.

Refer to [Error Definitions](#) in the chapter [Working with Commands](#) for detailed information on creating error definitions and then implementing error messaging in your test script.

The figure below shows error types in the **Error Types** tab. Select an error type from the list to view its details.

Figure 3-5 Project Properties—Error Types



3.1.1.6 Permissions Tab

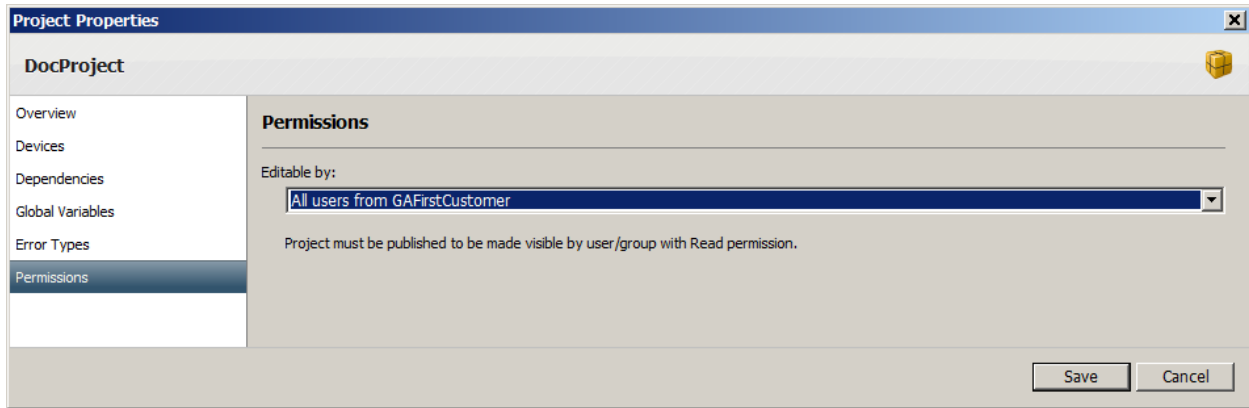
Specify project read and write permissions for other users in the **Permissions** tab.

NOTE You can also specify permissions in the opening screen of the [New Project](#) wizard.

From the **Editable by** drop-down list, select which users may edit your project. You can give edit permissions to:

- ◆ **All users from <account_name>**—All users in your account
- ◆ **Only me**—Yourself
- ◆ **<group_name> User Group**—Users from a specific user group

Figure 3-6 Project Properties – Permissions



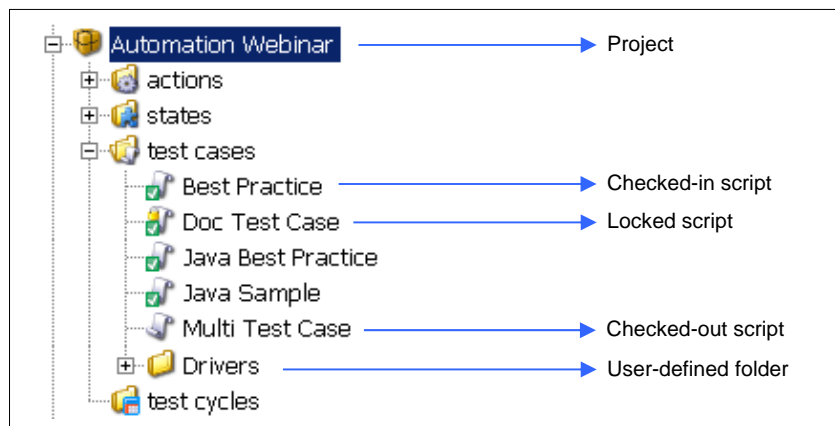
NOTE You must [publish](#) your project before it can be viewed by those with read-only permission.

3.2 Opening a Project


When you log in to DeviceAnywhere Studio, navigate to the Automation/Monitoring view and double-click a project from the project list to open it. (The project might take a few moments to be loaded as the version control system synchronizes local project files with those in the DeviceAnywhere database.)



This expands the project directory and displays automatically created folders for different types of test assets—[actions](#), [states](#), [test cases](#), and [test cycles](#). Note that each asset type has its own distinctive icon. You may create sub-folders for additional organization as you see fit.

Figure 3-7 Project Directory



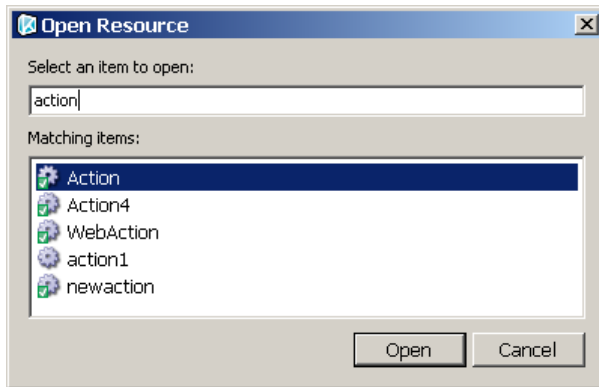
NOTE This image shows assets in the **Scripting** tab of the Automation view. In DAE Monitoring and DAE Automation, you can open a project and view its schedules in the **Scheduling** tab.

- ◆ The lock icon  near an asset indicates that it is locked for editing by another user.

- ◆ The check mark  near an asset indicates that it is checked in to the version control system.
- ◆ An asset without a check mark  indicates that you have checked it out for editing.

You can open test assets by double-clicking them in the project directory. You can also use the **View > Open Resource** command to open a dialog box that lists all the assets of your open projects. You can filter the list by typing a string in **Select an item to open**. The asset list is dynamically filtered with every keystroke entered into the field. Select an item from the list and click **Open**.

Figure 3-8 Open Resource



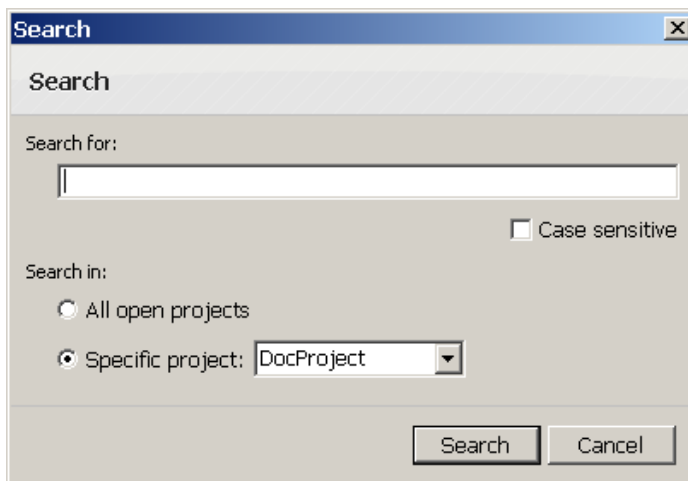
3.3 Managing Projects

This section describes how to use the powerful [project-level search](#) functionality and also describes [other project management functions](#).

3.3.1 Project-Level Search

You can search for any string, including the names of scripts, in a selected project or in all open projects.

- 1 Right-click your open project and select **Search**. This opens up the Search dialog box.



- 2 Enter a search string in the field provided.
- 3 Check **Case sensitive** if you wish to perform a case-sensitive search for the string as entered.
- 4 Choose a **Specific project** in which to search or select **All open projects**.

5 Click Search.

A status bar above the project list displays the outcome of the search and the number of occurrences found of the string, if any.



In the top-right corner of the Automation interface, the search results icon displays the number of occurrences found.

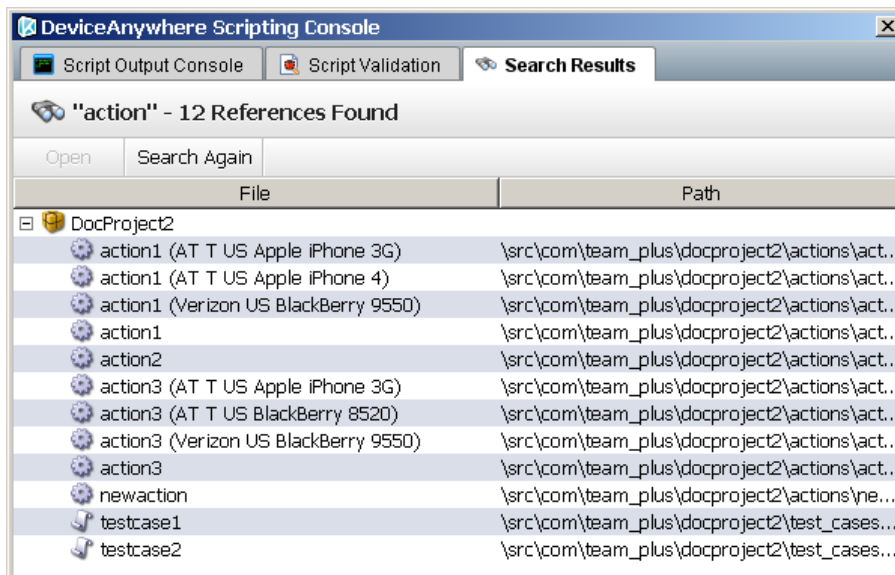


6 To open up search results in the DeviceAnywhere Scripting Console Window:

- Click **View Search Results** in the status bar or
- Click the search results icon in the top-left corner.

The **Search Results** tab of the DeviceAnywhere Scripting Console window displays the results of your search. The results of the most recent search persist until the next search.

Figure 3-9 Search Results



Each occurrence is listed with the name of the script containing it and its location in the local project directory.

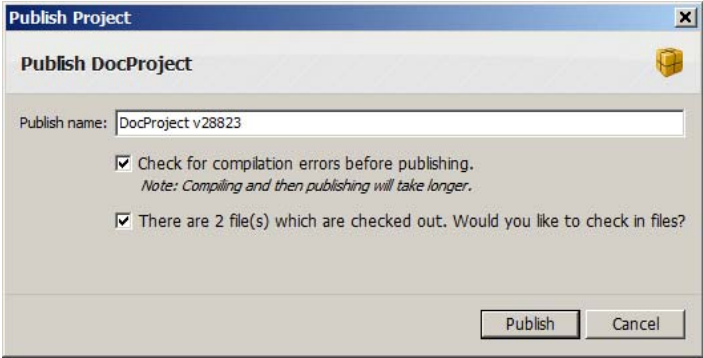
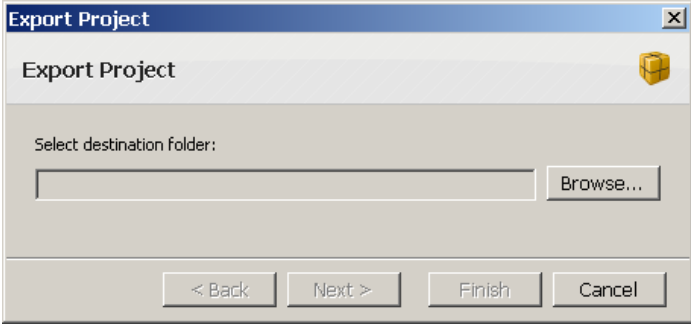

Double-click a search result to open up the script (the script must not be checked out by another user). You can also select the search result and click **Open**.

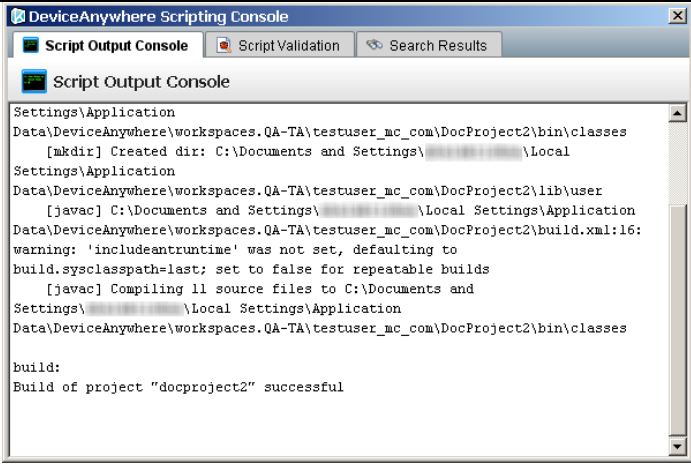
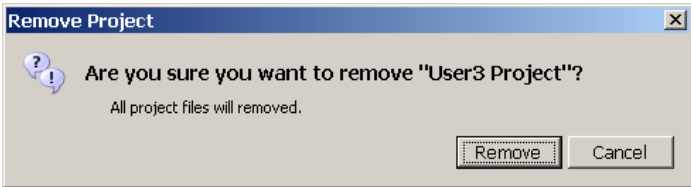
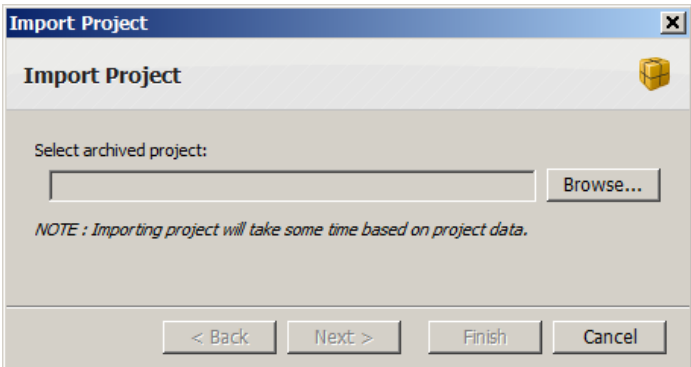
If you need to, click **Search Again** to search for the same string again, e.g., after making edits.

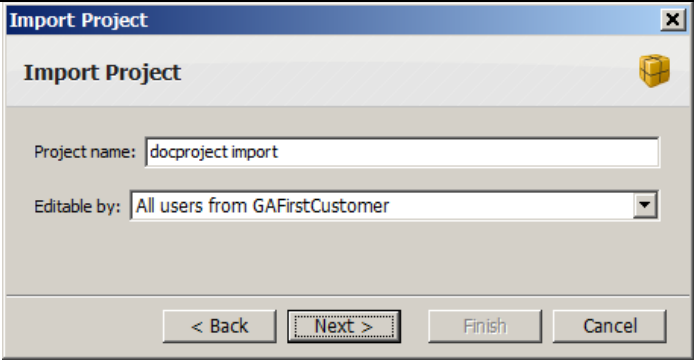
3.3.2 Other Functions

The table below describes the project management commands available from a combination of DeviceAnywhere Studio's **Project** menu and right-clicking a project in the project list.

Table 3-1 Project Management Commands

Command	Found in	Description
Close Project	Project menu Context menu	Closes project.
Open Project	Project menu Context menu	Opens the selected closed project.
Publish Project	Project menu Context menu	<p>A DAE Automation project must be published before its schedules can take effect. In the case of Mobile App Monitoring, a project and its dependent projects must be published so measurements can be provisioned in MyKeynote and be executed. Publishing also makes a project available for users with read-only permission.</p>  <p>You have the options to check in all files and compile the project before publishing.</p> <p>Publishing revises the project version in the version control system. When a project is republished, users with read-only permission receive a notification that they can view the new version.</p>
Export Project	Project menu Context menu	<p>Exports project as a ZIP file to a user-specified location.</p> 
Check In All	Project menu Context menu	Checks in all checked out assets.
Build Project	Project menu Context menu	<p>Compiles underlying Java code into classes. This occurs automatically when you run a script or detect a state (see Reference Points).</p> <p>When a project is built, the script output console icon in Studio's top-right corner displays a notification.</p>  <p>Click the icon to view console output in the DeviceAnywhere Scripting Console window. The contents persist until the next build.</p>

Command	Found in	Description
		
Validate All Files	Project menu Context menu	Flags errors in all project files – see Validation in Executing Scripts and Viewing Results .
Clean Project	Project menu Context menu	Deletes underlying compiled class files and rebuilds them from scratch. Use this command if your script/source code is not synchronized with the compiled class files, e.g., if you notice that your script run does not reflect recent edits.
Search	Project menu Context menu	Searches for a given text string in the selected project or all open projects.
Remove Project	Project menu Context menu	Removes project files. 
Properties	Context menu	Brings up the Project Properties dialog box .
New Project	Project menu	Creates a new project .
Import Project	Project menu	Imports an archived project ZIP file to create a project. Click Browse to select a project and click Next .  Specify a name and permissions for the imported project, as you would for a new project.

Command	Found in	Description
		
Enter Share Code	Project menu	Allows you to enter a share code so you can view a project created by another user.
Refresh Projects	Project menu	Refreshes the project list so you can see projects that have been created after you have logged in.

4 Begin Scripting—Actions

An action defines a specific procedure on all project devices, e.g., resetting the device, opening an application, or sending a text message.

An action can be object-based or consist of device-specific implementations on account of differences in device behavior, screen size, etc.

Actions are used as building blocks for test cases, which string together several actions to accomplish a larger test goal on a mobile device. Actions can be reused in any number of test cases.

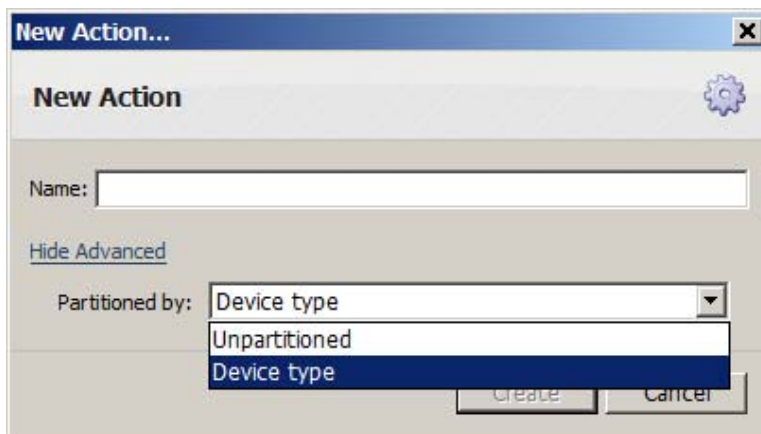
This chapter contains step-by-step instructions for [creating actions](#), [implementing them](#), and beginning scripting by [recording device interaction](#). It also describes the [action properties dialog box](#).

4.1 Creating an Action

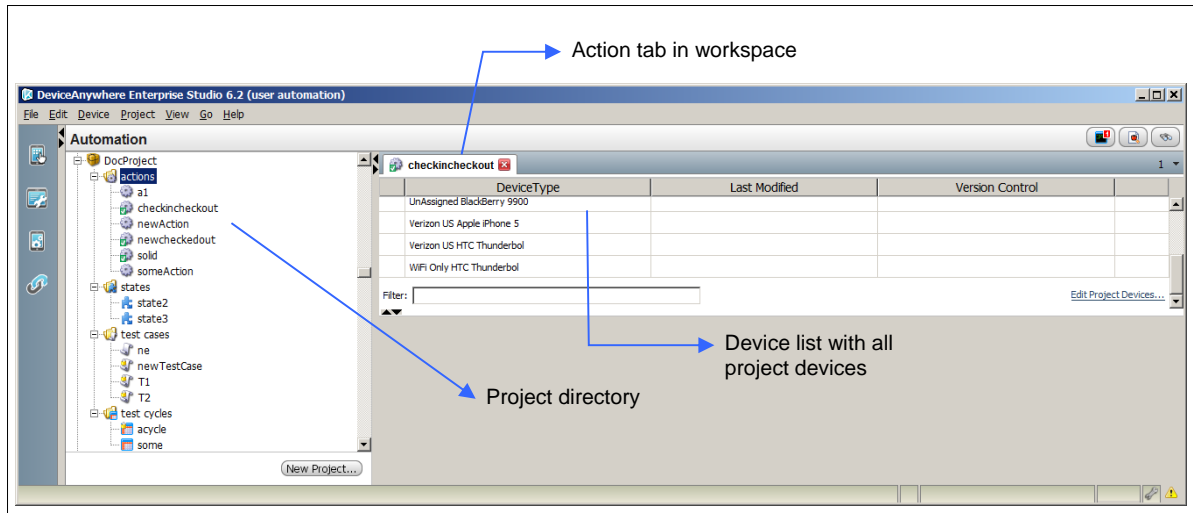
To create an automated action:

- 1 Make one of the following selections in the Automation/Monitoring view:
 - Right-click your project `actions` folder (or sub-folder) and select **New Action**.
 - While in your project, select **File > New > Action**.
- 2 Enter a name for your action in the New Action dialog box.
- 3 Optional: Click **Show Advanced** to display additional controls.

Choose to create an action that is **Partitioned** by device type or that is unpartitioned (as when creating object-based scripts). The default is a partitioned action with implementations for each device make/model. (If you have two instances of the same device in your project, they share one single implementation.)



- 4 Click **Create**. This opens a tab for the action in the workspace, where you can view the implementation status for each project device. (See [Implementing an Action](#) below.)



Use the scroll bar to scroll through the device list. You can also **Filter** the list. You can resize the pane by dragging the sizing handle that appears when you hover over the bottom edge of the device list.

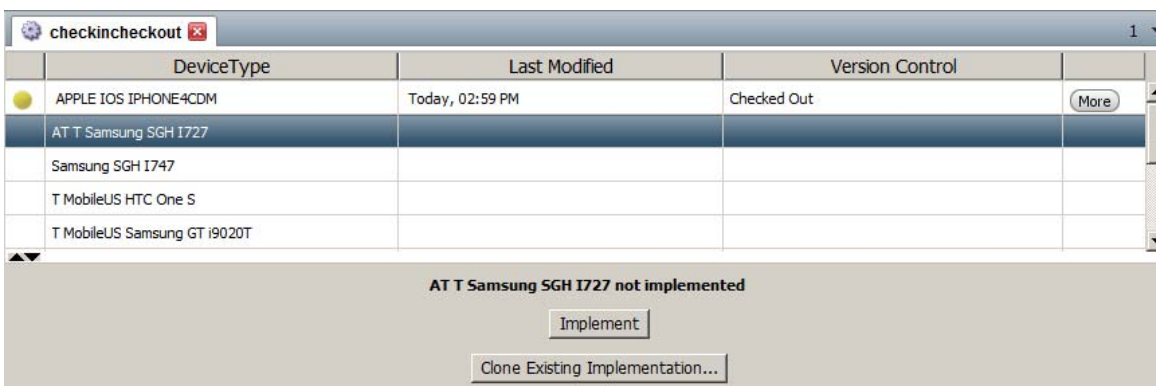
You can also use the buttons that appear at the bottom left   to collapse/expand the device list.

Click **Edit Project Devices** at the bottom of the device list to bring up the [Devices tab](#) of the [Project Properties](#) dialog box.

4.1.1 Implementing an Action


An implementation is a script with the specific commands (e.g., key presses, touchscreen events, hardware operations, script logic) that make up an action on a particular device. When you create a partitioned action, you must also implement it in order to specify the sequence of commands that carry out the action on each device. To implement an action:

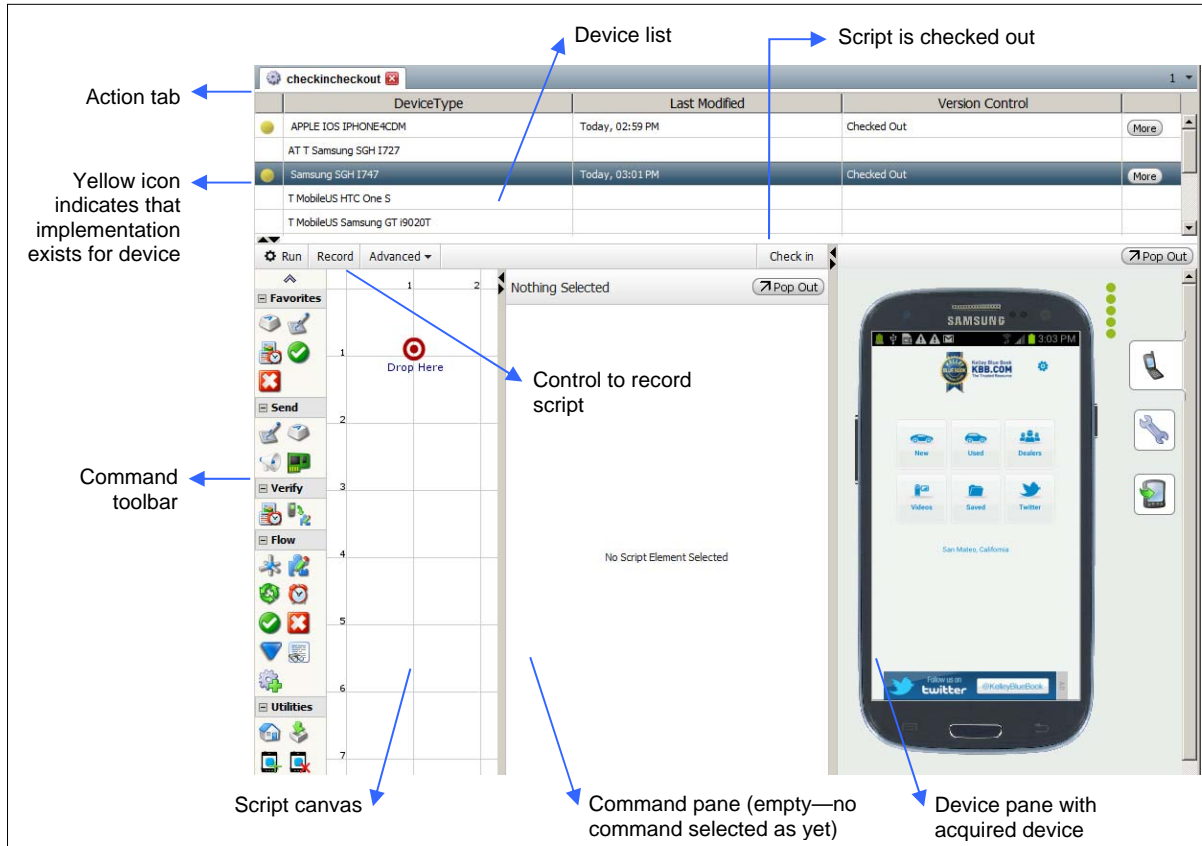
- 1 [Open the action](#) if isn't already. Each device is listed with its name in the **Device Type** column, date the implementation was **Last Modified**, and **Version Control** status of the script (checked in or checked out).
- 2 Select a device from the device list and click **Implement**. You can also click **Clone Existing Implementation** to copy and modify the implementation of another device.



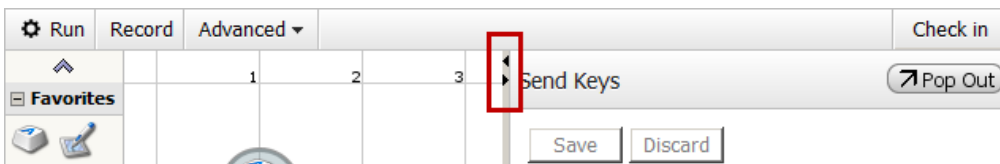
The action tab now displays the visual scripting environment below the device list, with the command toolbar, script canvas, and actual device.

NOTES If you do not see the device, hover over the right edge of the window and drag the sizing handle that appears.

The device list is updated to display a yellow icon  next to the device, indicating that an action implementation exists for it. The script is also listed as **Checked out** of version control for editing.



- 3 If your project contains two instances of the same device, select one from the device pane and click **Save Selected**.
- 4 Click **Acquire Device** in the device pane to interact with the device live.
- 5 Create a script by [recording device interaction](#) or by filling out command properties. Arrows connect the commands, indicating the order in which they will be performed. You can collapse/expand the command pane by using the arrows between the script canvas and the command pane.



See [Recording a Script](#) for information on directly capturing a script, including screen verifications, by interacting with a device.

See the chapter on [Working with Commands](#) for detailed instructions on filling out commands, including specifying device input, proofs, reference points, parameters and variables, error definitions, and script logic. You will also find information on scripting for web elements here.

- Optionally, for each command that you add to a script, right-click and select **Run from here** to verify the command and move your device into the correct state for defining the next command.

NOTES You cannot use **Run from here** when recording directly from a device.


Your device has to be in the correct state in order to run from a particular point. For example, if your command launches an application, ensure that the device displays the home screen or the application tray before clicking **Run from here**.

- [Check in](#) your implementation and release the device when done editing.

Check [Examples](#) for sample action implementations.

4.2 Recording a Script

Script capture by directly recording device interaction is available when creating action implementations:

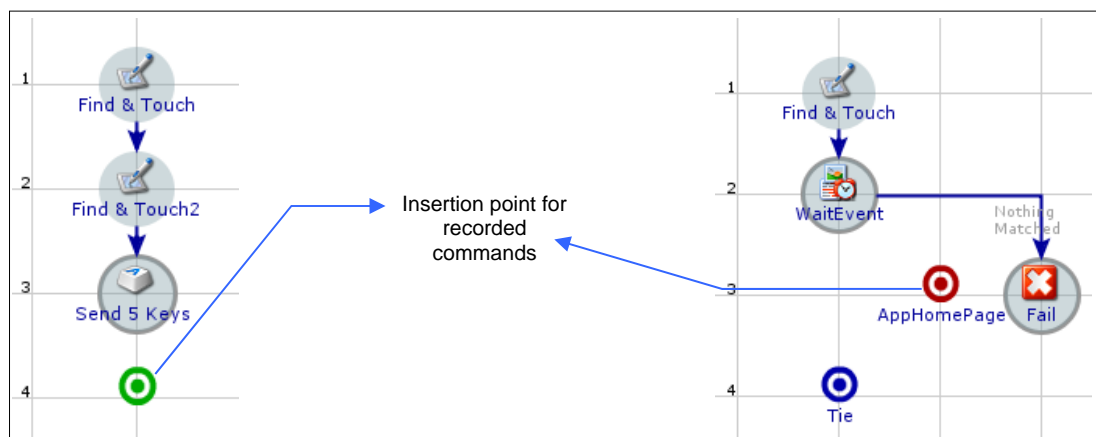
Click **Record** on top of the script canvas  and directly interact with the acquired device. In record mode, Send Keys and Find and Touch commands are inserted for device key presses and touchscreen taps/swipes respectively.

You can insert and define screen verifications (reference points) as you interact with a device in record mode. You can also insert and delete commands from the toolbar while in record mode.

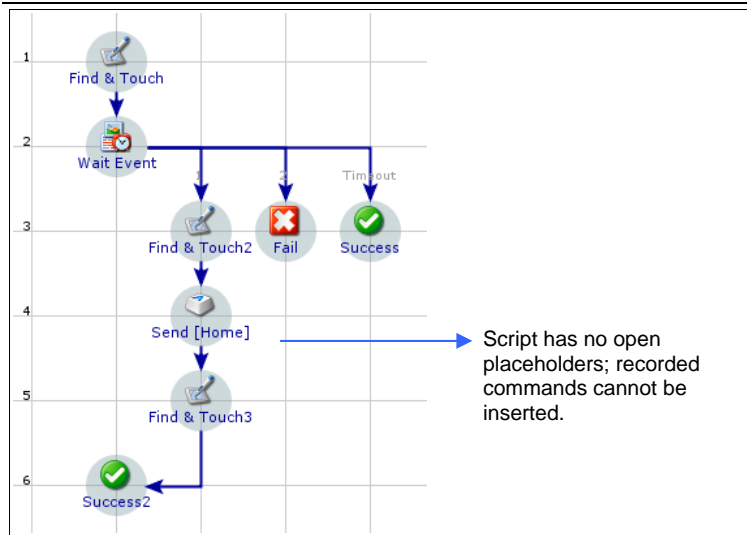
To record an action implementation:

- Open an action and load the implementation you wish to work on.
- If necessary, select a device from the device list and click [Implement](#).



NOTE If your script already has some commands, recorded commands will be inserted right below existing commands. If the last command in your existing script has several branches, e.g., a Wait Event command, clicking **Record** inserts commands into the first branch.

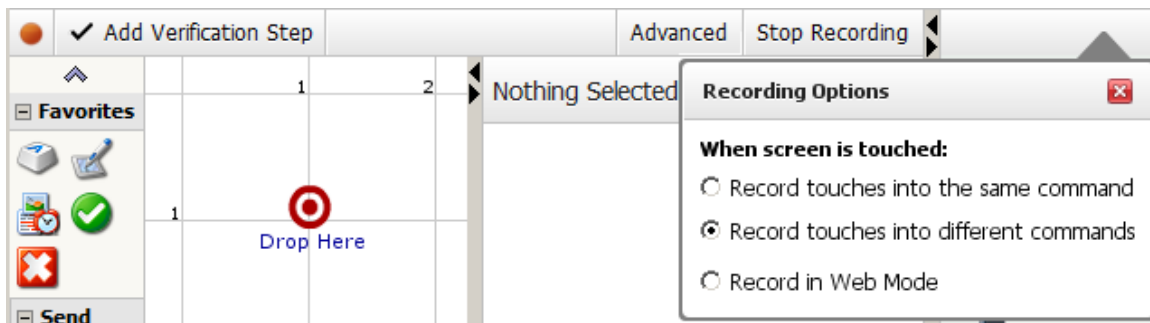


If your script has no open placeholders, i.e., all branches have termination points (Success or Fail commands), then you will not be able to insert recorded commands by clicking **Record** and interacting with the device.



See [Working with Commands](#) for information on branches created when inserting a [reference point](#) using the Wait Event command.

- 3 Acquire the device you wish to script on.
- 4 Click the **Record** button  above the script canvas. An blinking icon above the script canvas  indicates that recording is turned on.
- 5 Click **Advanced** and select recording options:



- **Record touches into the same command**—consecutive key presses or screen taps are recorded into a single Send Keys command. Touchscreen taps are defined in terms of the x and y coordinates, e.g., [Touch(51 , 146)]. If there is delay between presses/taps, they are recorded into separate Send Keys commands. Swipes (Ctrl key + drag mouse over device screen) are always captured in Find and Touch commands.
 - **Record touches into different commands**—each key press or screen tap is recorded into a separate command, Find and Touch for touchscreen taps and swipes, Send Keys for device key presses. Touchscreen taps and swipes are defined in terms of captured screen images.
 - **Record in Web Mode**—Select to work with web elements.
- 6 Click on your device to capture interaction as a test script. As each click is registered, you can see a corresponding command inserted in the script canvas. Command properties of the most recently

captured command are displayed in a pane next to the script canvas. Commands are automatically saved as they are captured.

NOTE When recording device interaction, you cannot use your computer keyboard for data entry on a device; you must press device keys or touchscreen.

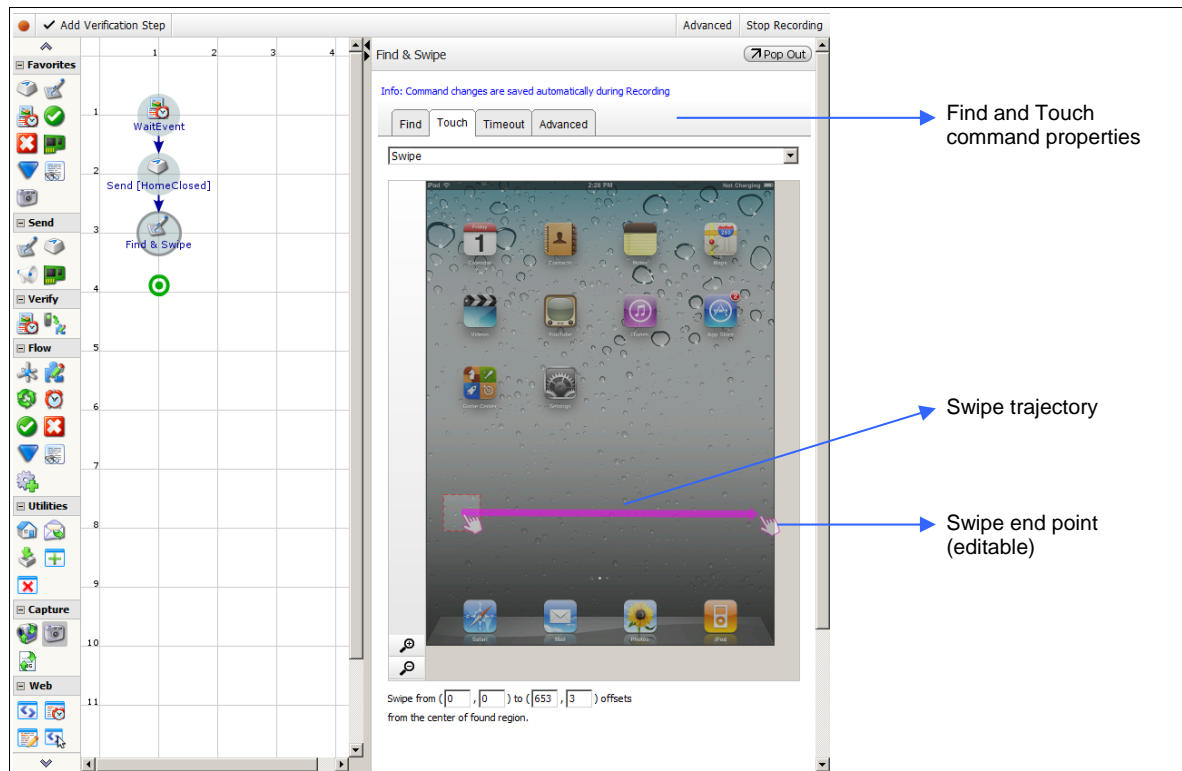


Commands might be automatically renamed to identify what they do, e.g., a Send Keys command to press the Home button might be renamed to Send [Home], and a Find and Touch command for a swipe might be renamed to Find & Swipe. You can always change command names as desired.

- 7 Insert a swipe by holding down the Ctrl key of your keyboard and dragging the mouse across the device screen. A green arrow on the device indicates the direction of the swipe.



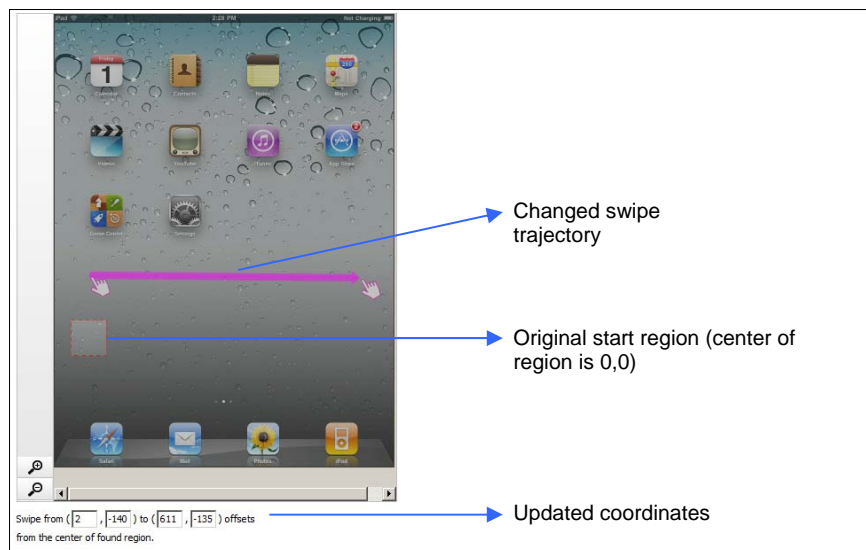
A Find and Touch command containing the recorded swipe is inserted into the script. The **Touch** tab of the command shows the start and end points and the trajectory of the swipe, which you can edit.



The swipe coordinates shown in the command indicate that the start point of the swipe is marked as 0,0 and the end point is defined in terms of positive or negative offsets from the start point.

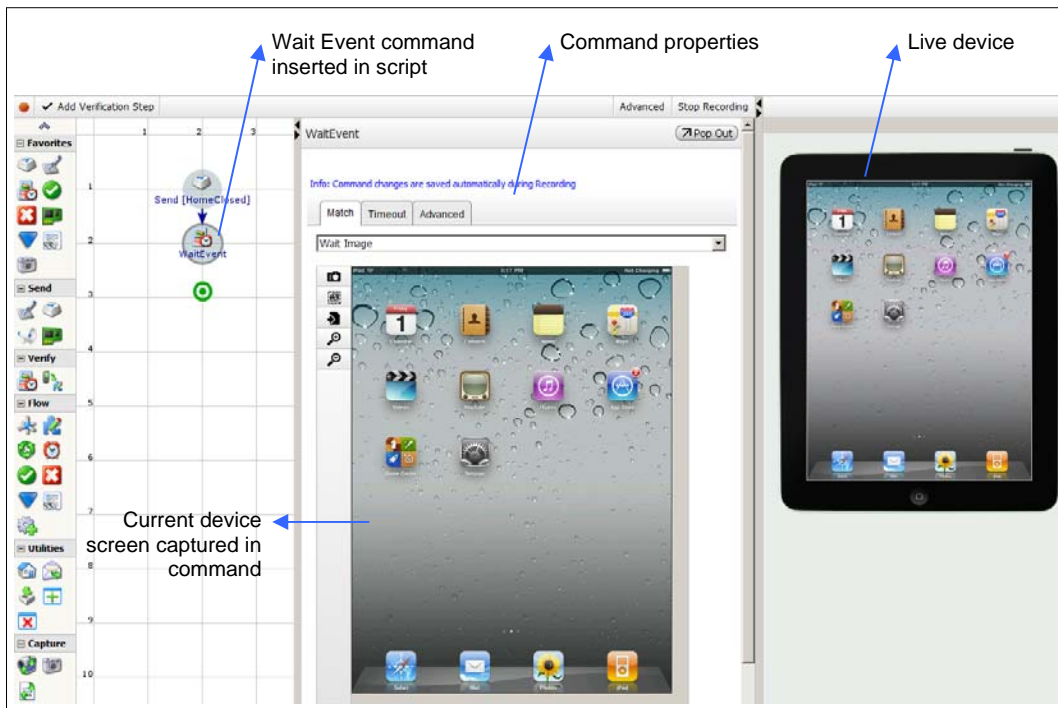
Swipe from (0, 0) to (-12, -712) offsets
from the center of found region.

You can move the hand icons to change the length and trajectory of the swipe. The updated coordinates show the offsets from the original start point (0,0).



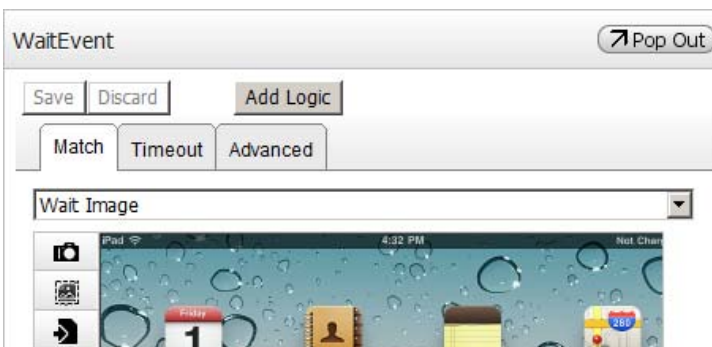
- 8 As you interact with a device, you might want to implement a verification for the current device screen between key presses or screen taps:
 - a With your device on the screen you wish to verify, click **Add Verification Step** above the script canvas. A Wait Event command is inserted in the script.

The current device screen is automatically captured in command properties, displayed in the center pane next to the script canvas. By default in record mode, a single image-based reference point is created. You can change the type and name of the default reference point.



NOTE The Wait Event command enables you to define multiple outcomes for a single device interaction. The reference points serve as bases for script branches. If you want to add branches, you must stop recording and click **Add Logic**.

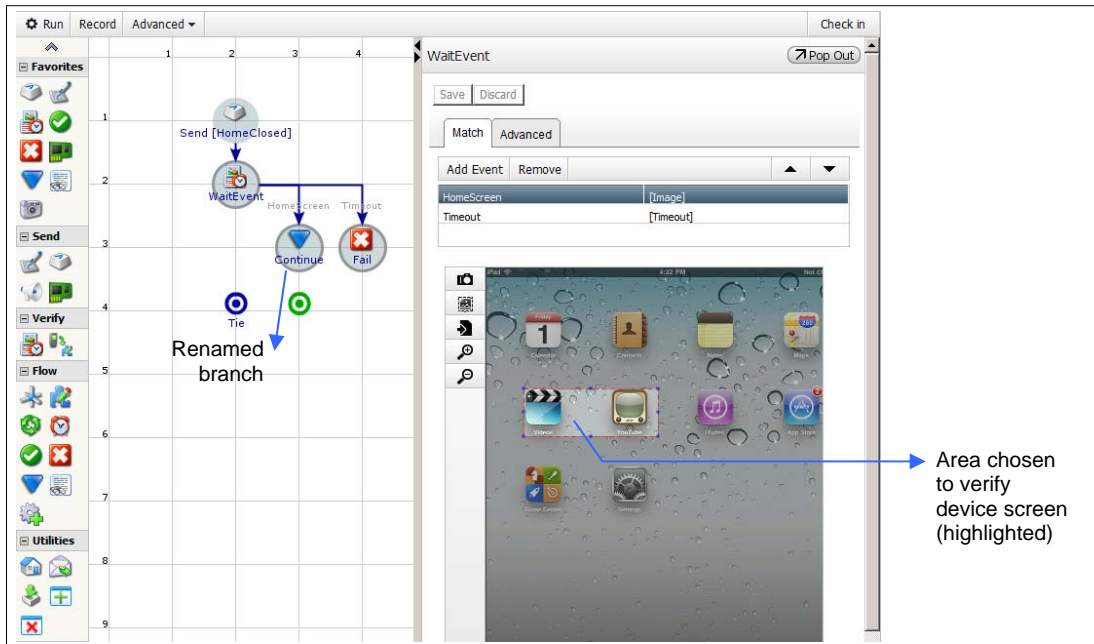
- b Choose the type of reference point desired (e.g., text-based or image-based) and define command properties in the center pane. Defining a reference point is discussed in detail in [Reference Points](#).
- c If you want to add branches, you must **Stop Recording** and click **Add Logic**.



This enables you to define additional reference points and also creates the Tie and Timeout branches. The Timeout branch defines script behavior when the image is not matched; the Tie

branch enables you to tie branches together and continue with the script after the branches have run their course. By default, the script is set to fail in the Timeout branch.

Save changes to Wait Event command properties in the center pane. The image below shows a recorded image-based verification with renamed branch and added logic showing the Tie and Timeout branches.



- 9 You can insert and define any command from the toolbar while in record mode:
 - a Drag a command to an insertion point—an open placeholder or between two existing commands. Command properties are displayed in the center pane.
 - b Fill out command properties—refer to [Working with Commands](#) for a detailed description of the various operations you can perform with commands. You can also consult the [Command Reference](#) for a field-by-field description of each command.

NOTE You can also construct a script by dragging in commands when you are not in record mode—see [Working with Commands](#) below.

- 10 When done interacting with the device, click the **Stop Recording** button  that appears above command properties. You can continue to insert other commands in edit mode by dragging commands to a placeholder and then filling out command properties—see [Working with Commands](#).
- 11 [Check in](#) your script to save changes.

See [Managing Scripts](#) for information on copying, checking in, and deleting actions.

4.3 Action Properties Dialog Box

The action properties dialog box displays action metadata and allows you to specify parameters that apply to all action implementations.

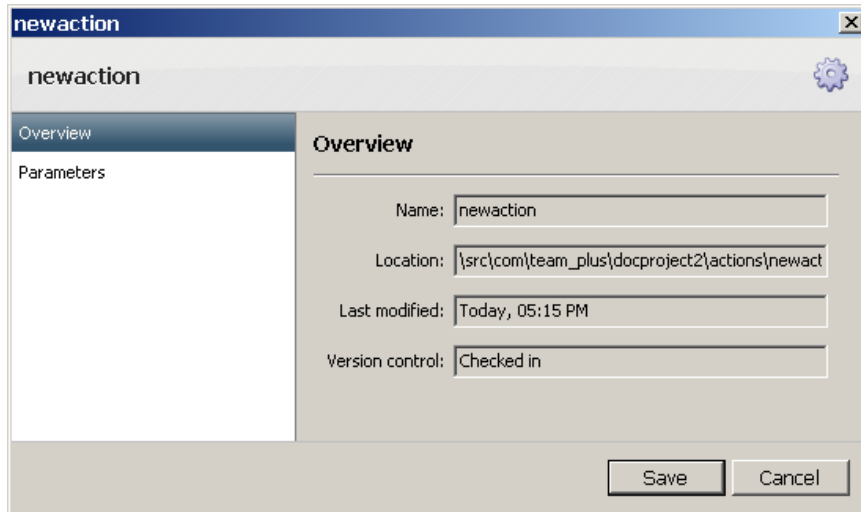
To open the action properties dialog box, right-click the action in the project directory and click **Properties**. Be sure to **Save** any changes to action properties.

4.3.1 Overview Tab

The **Overview** tab contains non-editable fields for action **Name** and **Location** within the project directory in the local file system, the date **Last modified**, and whether checked in or out of **Version control**.

The **Last modified** field is updated when an implementation is added/deleted or when action properties are edited and saved. Click **Cancel** or **Save** to close the dialog box.

Figure 4-1 Action Properties – Overview

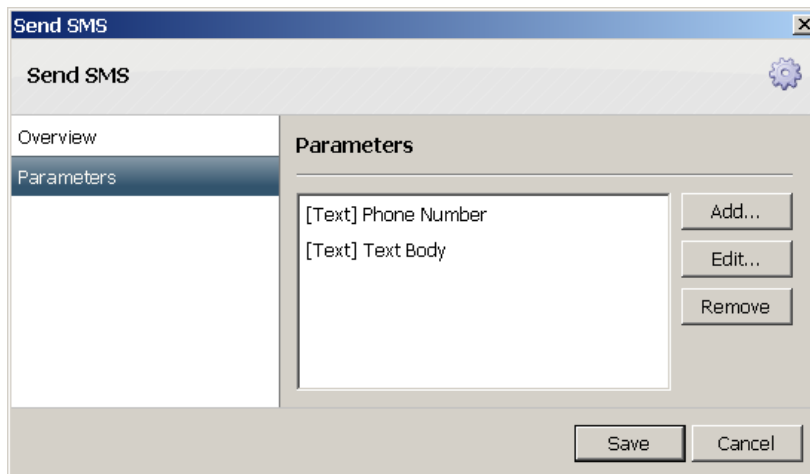


4.3.2 Parameters Tab

Defining parameters allows you to enter data dynamically into a device at the start of a script run, i.e., specify a different data value for each script run. Once created, an action parameter can be used in any implementation.

The **Parameters** tab of the action properties dialog box displays a list of existing parameters. You can **Add** or **Remove** script parameters and **Edit** parameter values.

Figure 4-2 Action Properties – Parameters



Refer to the chapter on [Working with Commands](#) for detailed information on using [parameters](#) – defining them, calling them from commands, and specifying values.

5 Working with Commands

Test scripts (action implementations, test cases, and test cycles) consist of a sequence of commands that accomplish key presses/touchscreen taps for navigation or data entry, hardware operations, and script logic. In addition, scripts also reference device conditions (defined by text, images, audio, or web elements) to verify the result of a command sequence.

This chapter provides an [overview](#) of DeviceAnywhere Studio commands and lays the foundation for working with commands by describing how to specify:

- ◆ [Device input](#)—device navigation and data entry in the Send Keys command and hardware operations in the Hardware Extension command
- ◆ [Reference points](#) for script verification in commands and states
- ◆ [Proofs](#) to be displayed in test results
- ◆ [Parameters and variables](#) that can be used within commands to implement complex script logic and set data values
- ◆ [Error Definitions](#) that can be inserted into commands in your test script for display in test results
- ◆ [Web Elements](#) in a web page that you can interact with directly
- ◆ [Script logic](#) such as loops, branches, and script termination

DeviceAnywhere Studio provides commands to execute a range of device interactions, verifications, and script logic. The available commands change based on the type of script you are building. You will find a greater number of commands in the action toolbar compared to the test case or test cycle toolbars (see **Error! Reference source not found.** below) as the greater part of your scripting consists of creating unpartitioned actions or device-specific action implementations.


Implementations are device-specific in that they consist of the specific key presses or device screens that make up an action on a particular device. For example, the key sequence for navigating to settings is different on different devices. Several commands for device-specific interactions are not available in the test case toolbar as test cases generally consist of calls to previously defined actions and states. The test cycle toolbar only contains commands to call test cases and to implement some script logic. (You can find field-by-field descriptions of all scripting commands in the [Command Reference](#).)

5.1 Command Overview

This section briefly describes individual commands and the broad [categories](#) into which commands are grouped. This section also describes the [command properties pane](#) and controls common to all commands. Finally, this section describes the [command context menu](#) accessible by right-clicking a command on the script canvas.

5.1.1 Command Categories and Descriptions

In the DeviceAnywhere Studio visual scripting environment, you can drag commands from the toolbar (see **Error! Reference source not found.**) onto the script canvas. Hover over a command icon to see a

tooltip with the command name .










Commands are grouped into categories, which can be maximized or minimized. The categories are:























- ◆ **Send**—Commands for directly interacting with the device, e.g., pressing keys, playing audio, performing a hardware operation, or touching a screen area
- ◆ **Verify**—Commands to define for script verification points and to extract text from device screens
- ◆ **Flow**—Commands for script logic and flow, e.g., loops, branches, calls to other scripts, wait times, termination points, placeholders, and variable values
- ◆ **Utilities**—Helper commands to upload test applications onto a device, power cycle the device, or open/close the native browser.
- ◆ **Capture**—Commands to capture device images, video, log information, or audio for test results
- ◆ **Web**—Commands to interact with web elements
- ◆ **Object**—Commands to interact with native objects in applications




The **Favorites** category places some frequently used commands at the top of each of the action, test case, and test cycle toolbars. You can change the selection at any time. To add a command to the category, right-click the command in the toolbar > **Mark Favorite**. To move a command out of favorites, right-click the command > **Unmark Favorite**.

The table below lists each command with a brief description. The sections that follow discuss the most important aspects of using commands such as specifying device input, reference points, parameters, error types, logic, and proofs. Detailed descriptions of commands can be found in the [Command Reference](#).

Table 5-1 Command Descriptions

Command	Description
 Find and Touch	On touchscreen devices, searches for and touches the center (or offset from the center) of a specified screen region. Also touches specified text from the device screen or the screen region from a selected state. Implements swipes.
 Send Keys	Enters a specified sequence of key presses and/or touchscreen events for device navigation or data entry.
 Play Audio	Plays an MP3 file or DTMF tones.
 Hardware Extension	Performs hardware operations such as disconnecting the data cable and turning on the camera light.
 Wait Event	Searches for a specified text string, device screen region, or element from a web page for script verification. Also waits to find a reference point defined in a state or to hear device audio for verification. Uses a combination of reference points as the basis for creating script branches.
 Extract Text	Stores a string of text from a device screen in a parameter or variable.
 Branch	Creates script branches based on the value of one or more variables or parameters.
 Set Variable	Sets the value of a variable or parameter .
 Loop	Creates a script loop that iterates over a set of commands for a fixed number of times or until certain conditions are met. Also creates a loop in which a set of commands iterate over specified values from a data set.

Command	Description
 Wait	Inserts a pause for a specified amount of time in the test script.
 Success	Terminates the script successfully at the specified point. Can include a customized termination message.
 Fail	Terminates the script with a “fail” result. Can include a customized error message.
 Continue	For use to tie a branch with no other commands to others or back to the main flow of the script
 Navigate To	Presses a single key or key sequence for a fixed number of times or until a reference point is found, whichever comes first.
 Execute Action	Calls a previously defined action.
 Execute Test Case	Calls a previously defined test case.
 Execute Cleanup Action	Calls a previously defined action to execute a cleanup script on the device. The cleanup script’s results are not added to test results.
 Reset	Disconnects and reconnects the battery, then powers on the device.
 Load Application	Uploads an application from the DeviceAnywhere application repository to an Android, BlackBerry, Brew, Brew MP, iOS, or Windows Phone 7 device (onto which applications can be side loaded).
 Browser Open	Opens native device browser and navigates to a specified URL.
 Close All Browser Sessions	Closes all open native browser sessions.
 Toggle Recording	Starts or stops capturing full-motion video or images for insertion into test results.
 Capture from Device	Captures video/snapshots of the current device screen for display in test results, as when streaming video.
 Toggle Extract Log	Starts and stops capturing device log information for display in test results.
 Web Element	Selects an element from the web page DOM tree and performs an appropriate action. Supported actions include extracting a value to a variable and setting a value (e.g., in a text box).
 Web Wait	Waits for a specified element to finish loading, come into focus or blur, become visible, or be hidden.
 Web Form	Selects/fills values in selected fields and submits a web form.
 Web Touch	Finds and clicks a web element, e.g., a button or hyperlink.
 Launch App	Opens a native application.
 Close App	Closes a native application.
 Object Touch	Finds an object by its text and touches it.

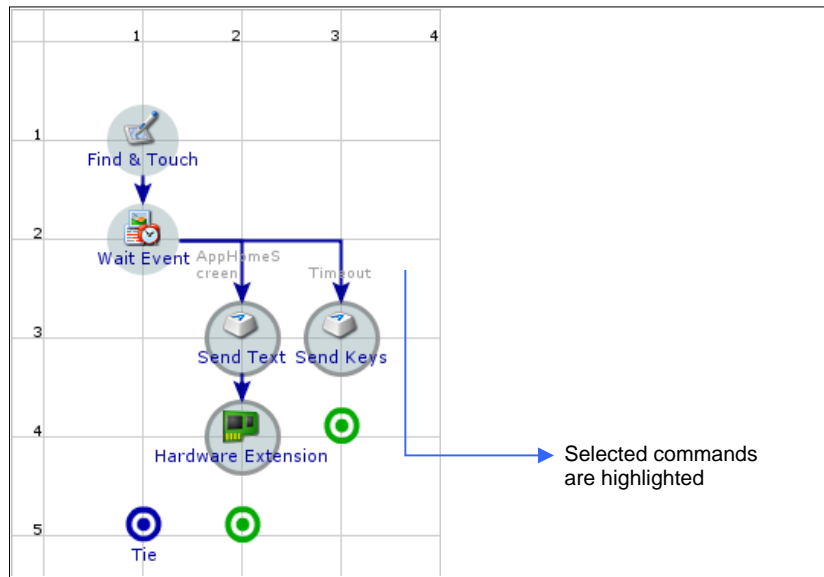
Command	Description
 Object Edit	Enters text in a native object after finding it and optionally, setting focus in it.
 Object Extract Text	Extracts text from a selected object and stores it in a variable for use elsewhere.
 Timer	Use a pair of these commands to apply a named timer to a portion of your Mobile App Monitoring script, which is then measured against performance thresholds.

5.1.2 Command Properties Pane

To use a command, drag it over from the toolbar to the desired location on the script canvas. The command properties pane is automatically displayed to the right of the script canvas.

The pane is also automatically displayed when you select any command on the script canvas. Click to select a command in the script canvas; use Ctrl-click or use your mouse to draw a box around and select multiple commands. Selected commands are highlighted in the script canvas.

Figure 5-1 Selecting Commands in a Script



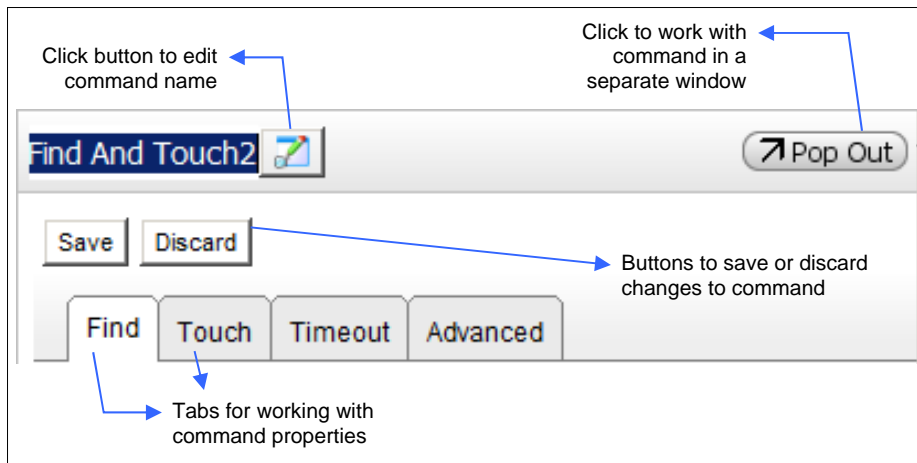
NOTE You need to have checked out the script in order to work with command properties.

Each command contains different settings and controls in its properties pane, depending on the function of the command. However, there are some settings common to all commands, discussed here.

Click the button at the top of the pane to edit command name. Both the command properties pane and the script canvas are updated to reflect the new name.

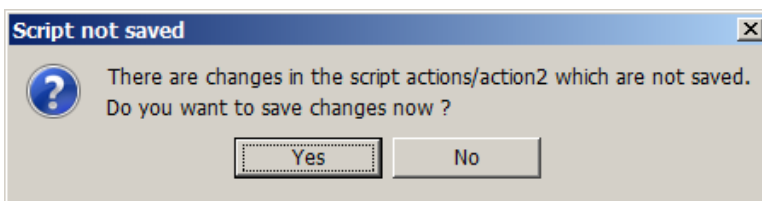
Command properties are organized in tabs, with main settings in the default tab. As you edit command properties, buttons to **Save** or **Discard** changes are enabled. The image below shows common properties in the Wait Event command.

Figure 5-2 Common Command Properties



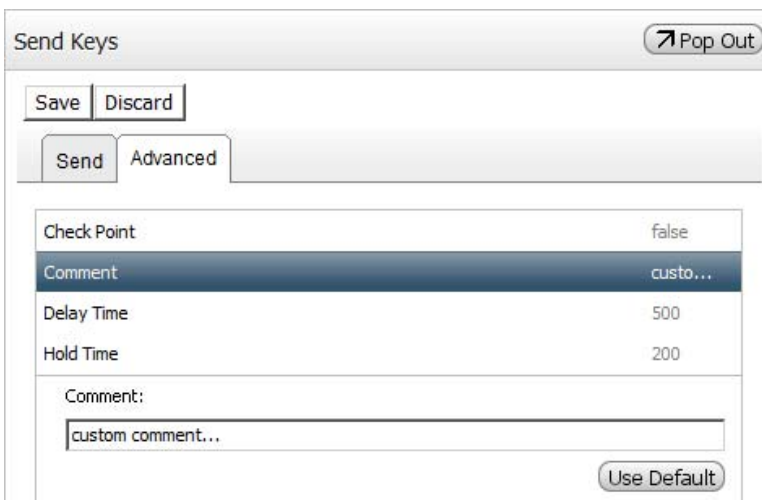
If you insert or select another command before saving, you are prompted to do so. Selecting **Yes** saves the entire script. You will lose changes if you select **No**.

Figure 5-3 Prompt to Save a Command



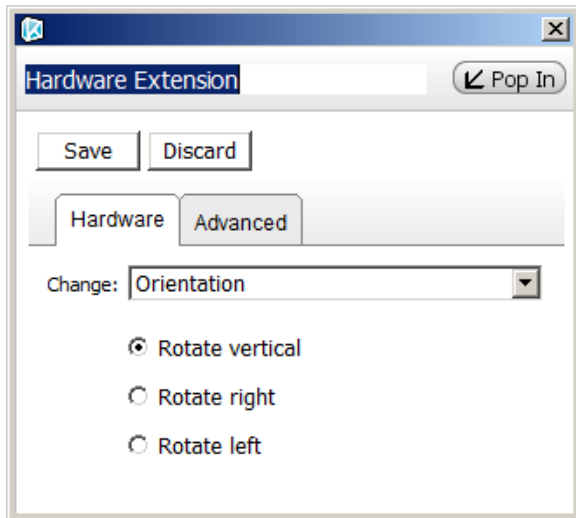
You can enter a **Comment** in the **Advanced** tab of a command. **Use Default** reverts to the default comment setting (blank).

Figure 5-4 Comment



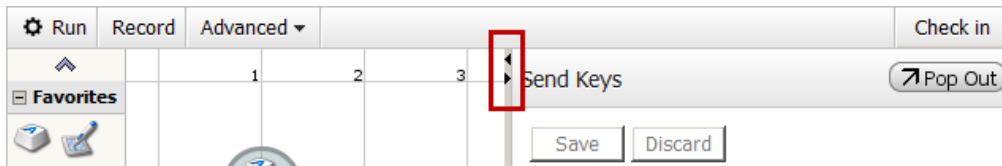
You can work with command properties in a separate window by clicking **Pop Out**; click **Pop In** to anchor command properties back to the DeviceAnywhere Studio window. The figure below shows Hardware Extension command properties in a separate window.

Figure 5-5 Command in a Separate Window



You can collapse/expand the command pane by using the arrows between the script canvas and the command pane.


Figure 5-6 Viewing/Hiding Command Properties



5.1.3 Command Context Menu

This section describes items in the context menu displayed when you right-click a command in the script canvas.

Table 5-2 Command Context Menu

Command	Description
 Run from here	Runs the script from the selected command. Useful when writing a test script to run through recently added commands. NOTE Ensure that your device is on the correct screen before running from a selected command.
Debug from here	Runs the script in debug mode from the selected command and opens embedded actions in separate tabs as they are being executed. NOTE Ensure that your device is on the correct screen before debugging from a selected command.
Undo	Reverses the most recent script edit (edit to command properties or command added/deleted).
Redo	Restores the script edit removed by the Undo command.
Select all nodes under this node	Selects the current command and commands in all branches under it for copying or cutting.
Copy/Copy all selected	Copies the selected command(s).
Paste	Pastes previously copied command(s).

Command	Description
Delete command/Delete all selected	Deletes the selected command(s).

5.2 Specifying Device Input

Using the Send Keys command as an example, this section describes how to specify key presses on a device in an Automation script. Send Keys allows you to specify keystrokes to be entered on device keyboards and touchscreens. You can use the information provided here to specify device input in the [Send SMS](#) and [Navigate To](#) command.

This section also briefly describes how to use the Hardware Extension command to perform [hardware operations](#).


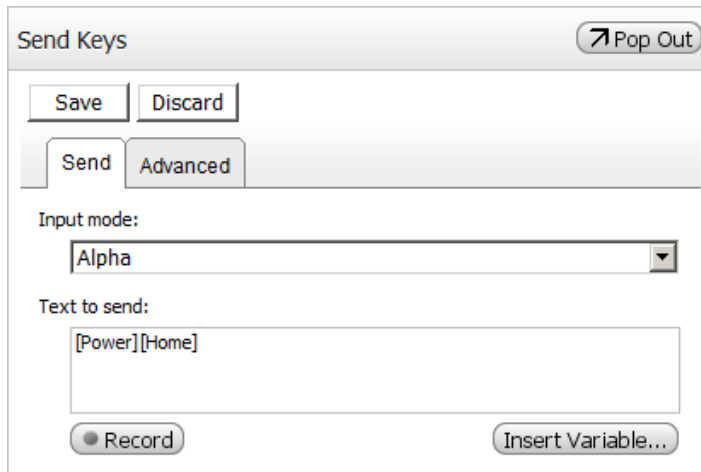
To work with Send Keys, drag and drop the command icon  onto the script canvas. Command properties are automatically displayed in a pane next to the script canvas.

Figure 5-7 Send Keys Command Properties



5.2.1 Text Entry Method

Enter a keyboard or touchscreen sequence in the **Text to Send** field. You can type or paste text into this field. (See [Text Format](#) below for information on formatting text entry in this field.) If you are not sure what a key is called, hover over the key in the device pane to view the name in a tooltip.

Figure 5-8 Finding the Name of a Key




You can also use the powerful recording feature of the Send Keys or Navigate To command to interact directly with the device and capture a key sequence:

NOTES This command-specific recording ability is available for Send Keys and Navigate To while in edit mode.

Directly capturing a script by interacting with a device (record mode) is discussed in [Recording a Script](#).

You must acquire the device in order to record a key sequence.

- 1 Select the Send Key or Navigate To command in the script canvas to display command properties.
- 2 Click **Record**.
- 3 Press device keys or touch device screen to save the key sequence in the Send Keys command. The captured key sequence is displayed in the **Text to send** field. The command automates entering this sequence into the device during script execution. When recording device interaction, you cannot use your computer keyboard for data entry on a device; you must press device keys or touchscreen.

NOTE You can pop the device out of the workspace  to interact with it in a separate window.

- 4 When done interacting with the device, select **Stop** in command properties.

5.2.2 Text Format

When typing or pasting text into the **Text to send** field, ensure that the name of each *navigation key*, e.g., Power, SoftL, or Left, is enclosed in square brackets. There should be no space before or after navigation keys, e.g., [Power] [Power] [Select] [Menu]. You can combine data entry with navigation keys, e.g., m.yahoo.com[SoftL].

You can also specify a *parameter* or *variable* in place of data entry (see [Other Settings in Send Keys](#) below). At runtime, the parameter or variable value is entered into the device. Parameter and variable names are enclosed within percent (%) signs in the **Text to send** field, e.g., %Website%. You can combine a parameter or variable with direct data entry or navigation keys, e.g., %Phone Number% [Down]. See

[Parameters and Variables](#) for detailed information on creating, calling, and setting parameter and variable values.

You can record or paste touchscreen events such as a *swipe* or *touch* in the **Text to Send** field.

The format for a touch is [Touch (x , y)], where x and y represent the coordinates, e.g., [Touch (265 , 285)].

The format for a swipe is [Swipe (x1 , y1) (x2 , y2) (z) (t)], e.g., [Swipe (60 , 427) (300 , 434) (2) (576)], where:

- ◆ (x1 , y1) represents the swipe start position.
- ◆ (x2 , y2) represents the swipe end position.
- ◆ (z) represents the number of intermediate steps between (x1 , y1) and (x2 , y2) to effect a swipe. The default value of (z) when you record a swipe is 10.
- ◆ (t) represents swipe time in milliseconds.

5.2.2.1 Some Special Keys and Characters

When specifying device input in the **Text to send** field, pressing the Enter key on your computer keyboard is not recognized. Instead, type [Enter] in the **Text to Send** field of Send Keys.

When sending data to a device, e.g., the content of a note, you might encounter an error when entering some special characters such as square brackets ([]), curly brackets ({ }), and semi-colons (;) using the Send Keys command. Alternatively, the characters are not displayed on the device as entered.

This issue does not impact the ability to send navigation keys presses to the device. For example, you can use the Send Keys or Navigate To commands to send the keys [Home] [Menu] to the device.

5.2.3 Other Settings in Send Keys

- ◆ You must specify an appropriate **Input mode** for all text entry into a device, depending on the type of field the text is being entered into—select **Alpha** (default) for an alphanumeric field such as the body of a note, **Numeric** for a numeric field such as a phone number, and **Web** for the URL field of a browser.

NOTE Alphanumeric fields support both letters (a, b, c...) and numerals (1, 2, 3...). Even if entering only numeric data into an alphanumeric field (such as a zip code into the body of a text message), you must set the input mode to **Alpha**.

- ◆ Select hold and delay times in the **Advanced** tab. **Hold Time** is the length of time a key is pressed for. **Delay Time** is the time between key presses. Hold and delay times are defined in milliseconds.

NOTE You must also specify key mode and hold and delay times in the [Navigate To](#) command.

- ◆ Click **Insert Variable** to call a variable or action parameter in the **Text to send** field. You would do this to pass the value contained in a variable to the **Text to send** field, or to call a parameter and dynamically provide its value at the start of a script run. For example, instead of directly entering a URL to navigate to, you can call a parameter and provide the URL at runtime. See [Parameters and Variables](#) below for detailed information on creating, calling, and specifying values for parameters and variables.

NOTE You need to have created a parameter or variable before you can call it in Send Keys.

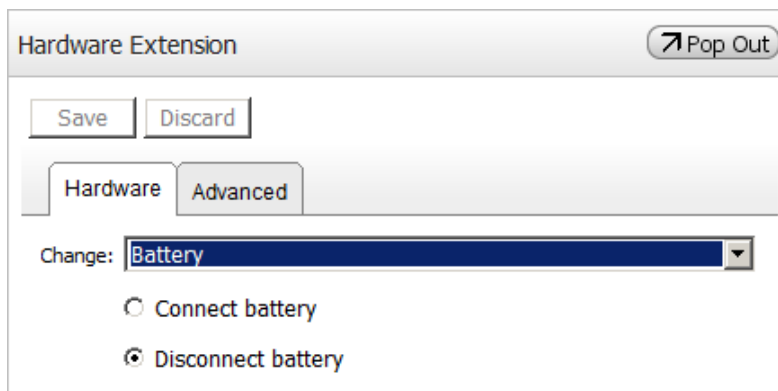
- ◆ In the **Advanced** tab, you can specify additional properties.
 - Select **Checkpoint** and then **true** from the drop-down list to capture the first and last screen of the device during command execution. Use **Default** reverts to the default setting (**false**). Refer to [Proofs](#) in this chapter for more information on collecting proofs for display in test results.
 - Select **Comment** and enter a comment in the field provided.

5.2.4 Performing Device Hardware Operations

In addition to key, touch, or gesture input, you can also program device hardware operations into your test script. You can use the [Reset](#) command to disconnect and reconnect the battery, then power on the device. Depending on the device you are working with, you can perform several hardware operations using the Hardware Extension command (see Figure 5-9 below). For example, on an iOS device, you can connect/activate or disconnect/deactivate the battery, camera light, or data cable. In addition, you can set device orientation to be vertical or to be horizontal by turning it left or right.

As in the Send Keys command, you can specify proofs and comments to be displayed in test results in the **Advanced** tab.

Figure 5-9 Hardware Extension Command Properties



5.3 Reference Points

Reference points are device conditions or device output that verify the result of a sequence of device interactions. A reference point serves as an expected result against which the outcome of a script can be verified. For instance, if your script navigates to the device idle screen, you can use text or an image from the idle screen to define an expected result.

Reference points can be based on text, an image, or audio. Reference points implemented in web applications or browser content can be based on identifying web elements in the markup—see [Web Elements](#) below. For information on native-object based reference points, see [Waiting for an Object](#).

Text-based reference points specify a text string from a device screen for script verification. *Image-based reference points* specify an area of a device screen to be used for script verification. You can also call for your script to detect device audio for verification (see [Image-Based Reference Points](#), [Text-Based Reference Points](#), and [Audio Reference Points](#) below).

Reference points can be device specific or device independent. You can create script- or *device-specific reference points* by using the [Wait Event](#) command in your action implementation. You can also create script-specific reference points in [Navigate To](#).

Device- or script-specific reference points are set in action implementations and are called so because they cannot be called from other scripts. The Wait Event command is not available in the test case toolbar (see **Error! Reference source not found.**) because test cases are device independent and generally consist of calls to other actions and states.

An audio reference point works a little differently from an image- or text-based reference point in that it does not define a device-specific sound; instead, it simply waits to hear any device audio (or a specified DTMF sequence) for script verification. You can safely copy and paste an audio reference point in Wait Event from one script to another in order to reuse it.

To specify a *device-independent reference point*, you must create a state that is defined for all project devices. States consist of device-specific implementations to account for differences in interfaces. (Refer to the chapter on [States](#) for information on managing states.) You can update a state in one place, without having to update every script that makes calls to it. Once you create a state, you can use the [Wait Event](#) command to call the appropriate state implementation in an action or test case.

You can also create a device-independent reference point by using a web element, usually in an unpartitioned script, for verification.

The mechanism for defining reference text, images, or audio in device-specific commands or device-independent states is the same, as for example, defining a reference image in Wait Event or in a state. The following sections describe how to create and define these types of reference points:

- ◆ [Image](#)
- ◆ [Text](#)
- ◆ [Audio](#)

5.3.1 Image-Based Reference Points

You can define an area of the device screen as a reference image. For instance, you can use an application icon as a reference image for the device home screen. At runtime, Keynote uses pixel-to-pixel image matching technology to match the actual device screen to the reference image for script verification.

You can create an image-based reference point in any of the following:

- ◆ In the [Wait Event command](#), the default type of reference point is image based. The current device screen is automatically captured in the command. You can then select a screen region as a reference image (see Figure 5-10 below). See [Script Logic](#) for a discussion of branches in Wait Event.
- ◆ In the [Navigate To command](#), select the **Condition** tab and select **Wait for image** from the drop-down list provided. The current device screen is automatically captured in the command. You can then select a screen region as a reference image (see [Script Logic](#)).
- ◆ [Create a new, device-independent state](#) and select the device for which you wish to create an implementation. Select **Wait image** from the **State Type** drop-down list (see Figure 5-11 below). You can then define a reference image.

Figure 5-10 Defining a Reference Image in the Wait Event Command

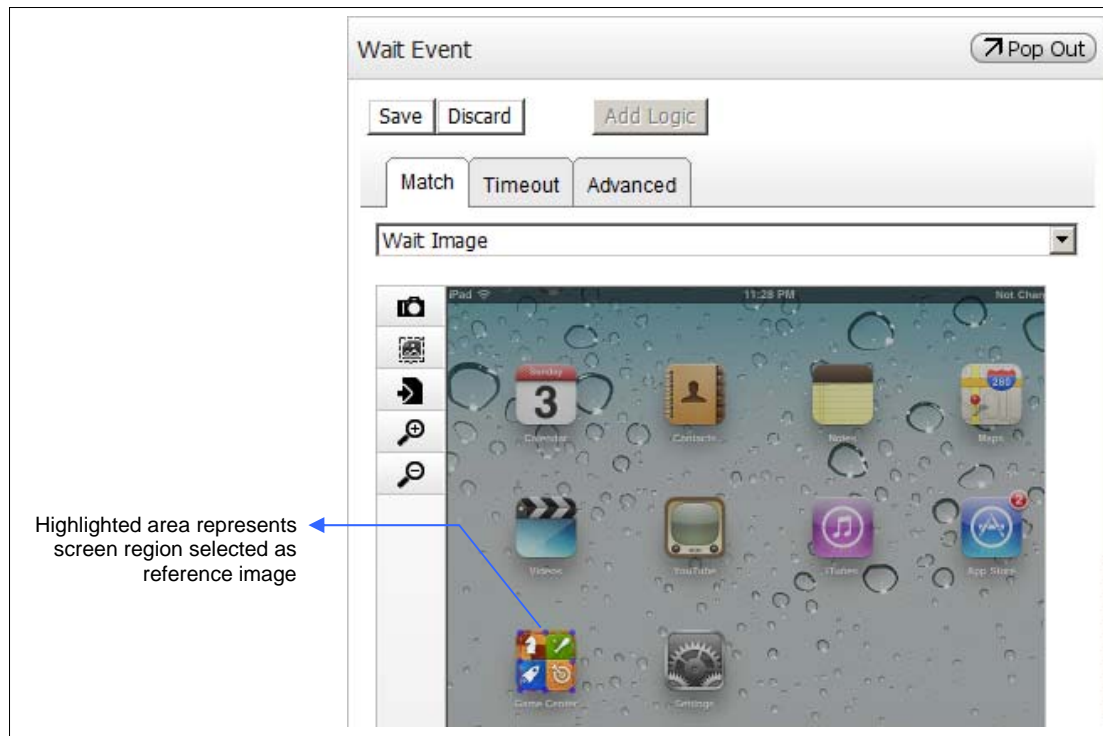
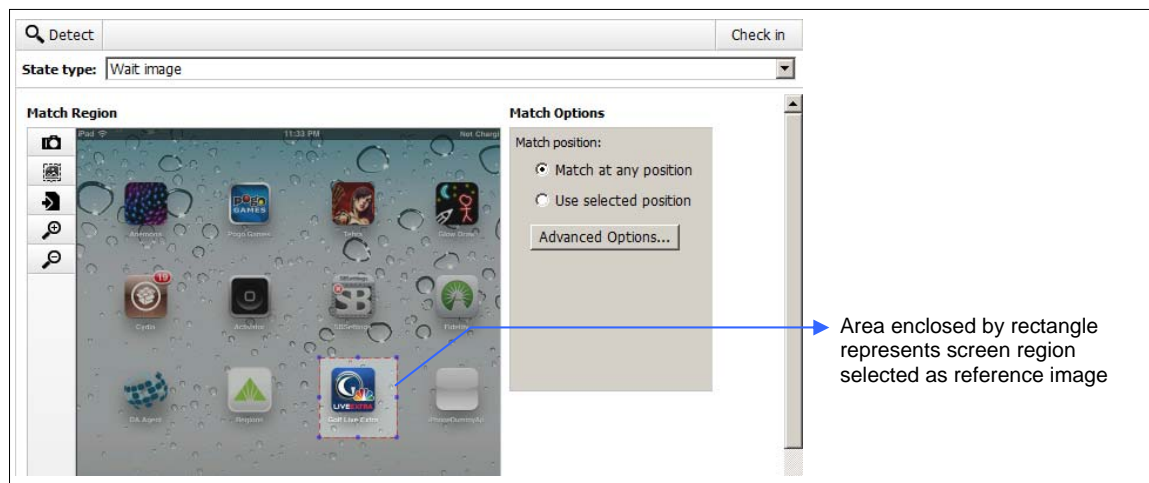


Figure 5-11 Defining a Reference Image in a State



The mechanism for defining an image-based reference point is by and large the same whether in a command (i.e., Wait Event, Navigate To) or in a state.

The sections below describe how to:

- ◆ [Define a reference image](#) in a state or a command.
- ◆ [Troubleshoot a reference image](#) or verify that you have correctly defined it.

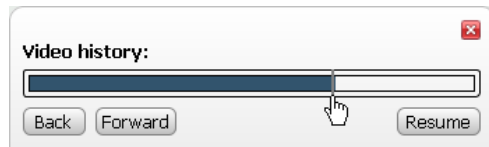
5.3.1.1 Defining a Reference Image

Be sure to have the device acquired and your state/action checked out so you can navigate through and capture device screens.


To define a reference image:

- 1 In the device pane, navigate to the screen you would like to use to define your reference image.

NOTE You can also use a screen from device history — right-click the device and select **Show Video History**. Click in the video history slider that appears to view a screen from device history.






- 2 In your command or state, take a snapshot  of the current device screen if necessary.

You can also choose to import an image  of the device screen to your command or state.

CAUTION Import only images from test results that were directly captured from the device and uploaded to the DeviceAnywhere results and administration web portal. (See [Viewing Results](#) for information on viewing and uploading results.) Images from any other source might not have the resolution required for the DeviceAnywhere pixel-to-pixel image matching technology, resulting in verification failures.

- 3 Using your mouse, select the screen region you would like to use as a reference image (see Figure 5-10 and Figure 5-11 for examples of screen region selection). You can select a region containing

graphics and/or text as a reference image. Use the zoom buttons   to enlarge or shrink device screen display. You can also clear your selection  and choose another area at any time.

- 4 Select the appropriate match position radio button:

- Select **Match at any position** to locate the reference image at any position on the device screen. Select this option, for example, if your reference image is an application icon whose location on a smartphone touchscreen might vary depending on other applications uploaded/deleted.
- Select **Use selected position** to match the reference image at exactly the same location as on the reference screen.

NOTE The match position radio buttons for a command can be found in advanced settings—click **Show Options Editor**. Advanced settings also allow you to select a range of images that can be considered matches as well as mismatches. You can specify *tolerance levels* for variations between the reference image(s) and the actual device screen (see [Advanced Image Match Settings](#)).

- 5 Enter the **Wait Time** or **Timeout** (in seconds) within which the reference image must be matched.

NOTE In Navigate To, you can specify a timeout in the **Logic** tab. In Wait Event, you use the **Timeout** tab. If you have chosen to **Add Logic** (i.e., additional default branches with additional possible outcomes, you will be able to set the command timeout in Wait Event's default **Match** branch.

- 6 To continue with the script if image verification fails, select **Continue with script**; select **Return failure immediately** to terminate the script with a "fail" result.
- 7 You can also opt to trigger an **Error Type**, and **Add Notes** to test results if you choose to fail your script.

Refer to [Timeout Tab—Implementing Error Messaging in Your Test Script](#) for detailed instructions on error messaging.

- 8 Select **Checkpoint** to specify [proofs](#) (screenshots) for display in test results.
- 9 In a state, you can click **Detect** to test that an area matching the reference image can be found on the device screen. This ensures that you have defined your state correctly.

DeviceAnywhere Studio displays the result of state detection above the script canvas.

- 10 Check in your state/script.

5.3.1.2 Advanced Image Match Settings

In commands as well as states, advanced settings allow you to compare your reference image to the current device screen or an imported image—DeviceAnywhere Studio highlights any mismatch. You can also define a range of images that can be considered matches as well as mismatches. You can then optimize tolerance levels for variation between the reference image(s) and the actual device.

Use advanced settings to troubleshoot areas of mismatch after a failed run. You can then reset tolerance levels to bring your reference image back into the match zone.

While it is not necessary to specify match and mismatch image libraries, this ability is useful:

- ◆ When your reference image seems to fail incorrectly due to small changes in device resolution or the device screen that are not visible to the human eye—you can define a range of images and adjust your tolerance settings so that any of them matches the live device screen.
- ◆ To delimit what can be considered a match by specifying mismatch image(s)—you can then adjust tolerance settings to ensure that the “mismatch” images always fail.
- ◆ If the device screen you have used for a reference image has dynamic elements, and you want to specify a range of match and mismatch possibilities—you can then adjust tolerance settings to include matches and exclude mismatches.

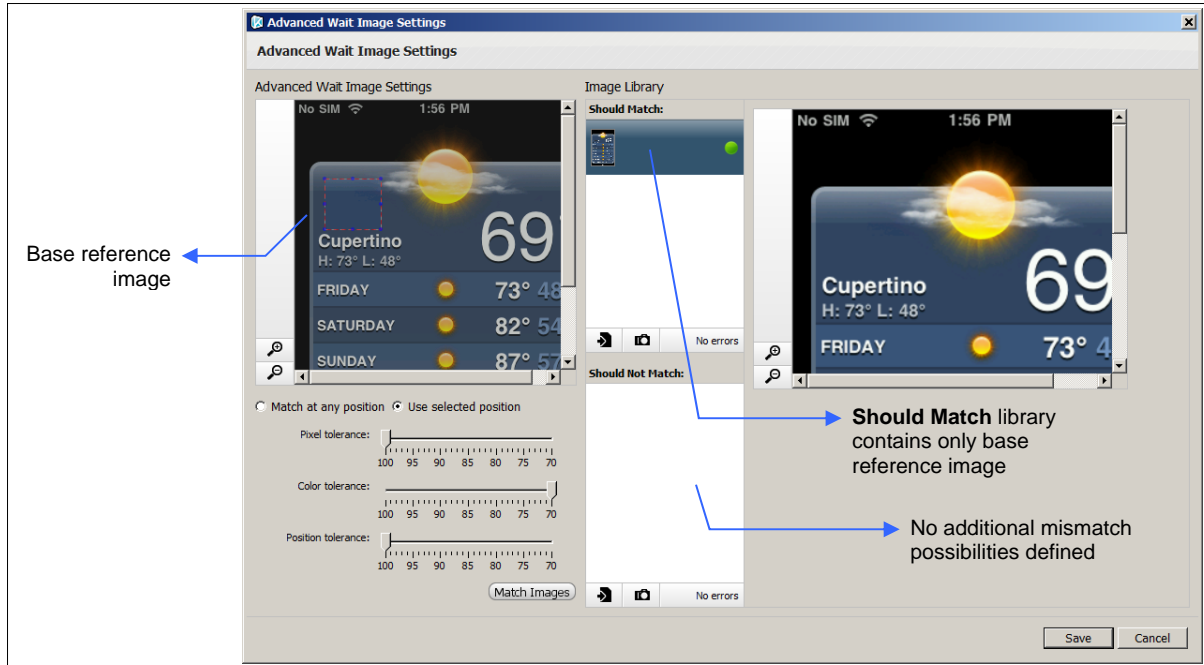
The purpose of setting match and mismatch ranges is to arrive at a tolerance setting that is flexible enough to accommodate minor differences in the target image.

NOTE When defining a range of image matches, be sure not to use widely different images or you will end up setting tolerance levels that produce false positive results.

To access the reference image wizard:


- 1 Open your command or state and view the screen region selected as a reference image.
- 2 Click **Advanced Options** in a state or **Show Options Editor** in a command.

The Advanced Wait Image Settings dialog box displays your initial (base) reference image and tolerance sliders at default levels. At this point, no additional match/mismatch possibilities are defined.





- 3 To compare the reference image to the live device screen or define another match possibility:
 - a On the device, navigate to the screen that can also be considered a match, e.g., another city with similar weather conditions in the iPhone weather application—the application has the same background color for similar weather conditions and times of day.
 - b In your command/state, capture the device screen in the **Should Match** section—click the **Take Snapshot** icon.

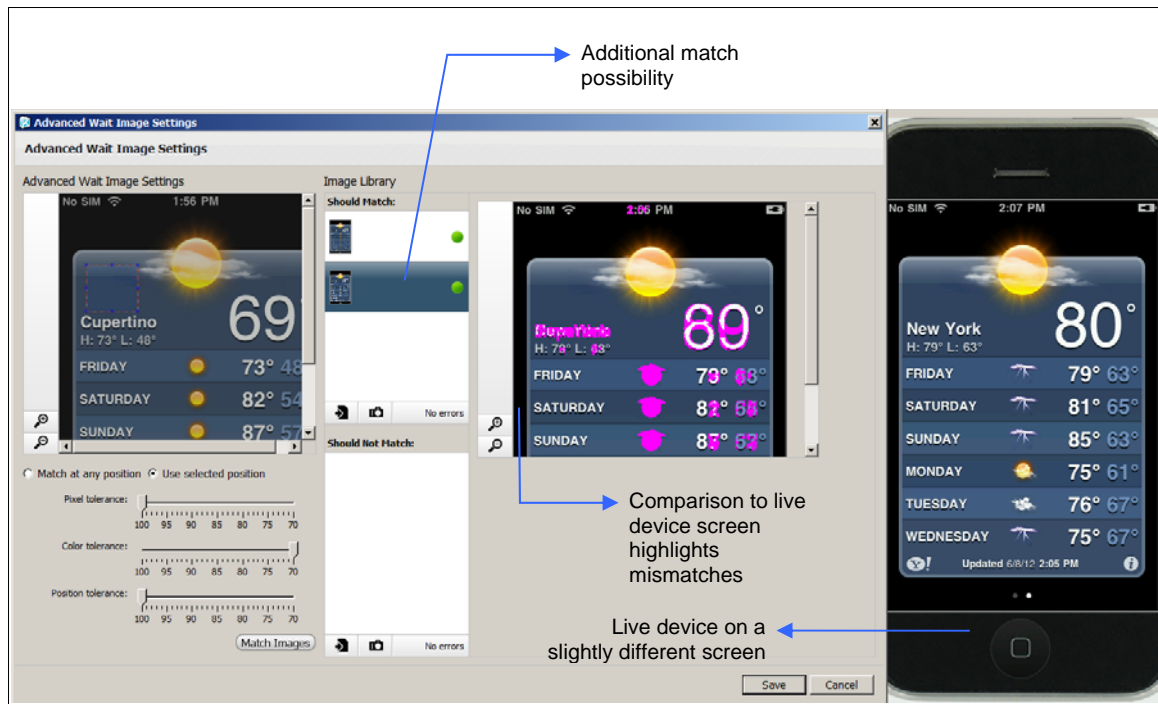


You can also import an image  into the match library, e.g., an image from test results of a screen that should have matched but did not. Be sure only to import images from test results that were uploaded to the web portal. Images from any other source might not have the resolution required to match the base image.

The base image is automatically compared to the captured device screen (or the imported image). The comparison is displayed in the right pane with any areas of mismatch highlighted in pink.

An additional match possibility is listed in the **Should Match** section—a green icon  indicates that the base reference image matches the corresponding area on the screen you just captured. A red icon  indicates that the base reference image does not match the corresponding area on the captured screen.

The **Use selected position** radio button is the default when you use advanced settings.





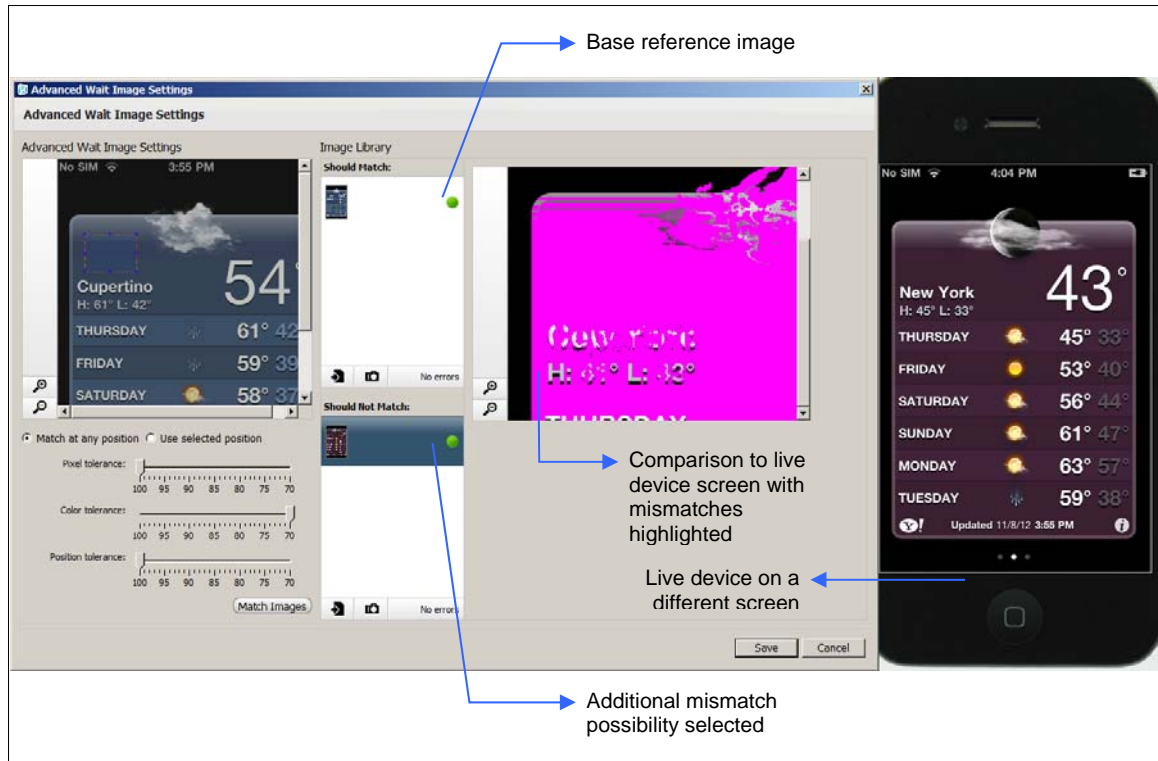
4 Optionally, define additional *mismatch* possibilities:

- a On the device, navigate to the screen that should be considered a mismatch, e.g., the same city with different weather conditions in the iPhone weather application—the application has a different background color for different weather conditions and times of day.
- b In your command/state, capture the device screen in the **Should Not Match** section—click the **Take Snapshot** icon.



The mismatch candidate is automatically compared to the base reference image. The comparison is displayed in the right pane with areas of mismatch highlighted in pink.

The mismatch candidate is listed in the **Should Not Match** section—a green icon  indicates that the base reference image *correctly does not match* the corresponding area on the screen you just captured/imported. A red icon  indicates that the base reference image *incorrectly matches* the corresponding area on the current device screen.



5 If any of the images in the **Should Match** or **Should Not Match** section have a *red icon*, you can:

- Reselect your base image to exclude portions of the device screen that cause mismatches.
- Change settings on your device to minimize the difference (and mismatches) between the reference image and the device screen.
- Adjust the tolerance sliders to bring your image back into the match zone.

Tolerance sliders control the variation allowed between the reference image(s) and the actual device screen during script execution. Tolerance settings can be optimized at the same time for the entire range of match/mismatch possibilities you have defined.

- Use the **Pixel tolerance** slider to set the percentage of pixels to match, with 100 being a strict match. Use this slider if the device screen has artifacts such as dots or lines, or if the reference image uses encoding (e.g., is taken from a JPEG or MPEG file).
- The **Color tolerance** slider sets the color matching required between the reference images and the device screen, with 100 being a strict match.
- The **Position tolerance** slider allows you to set the radius around each pixel's original position in which to look for the pixel, with 100 being a strict match (or the tightest radius).



6 Click **Match Images** to test your adjusted tolerance levels. Ensure that images in both the **Should Match** and **Should Not Match** sections have green icons.

NOTE Only move the sliders enough to display your images with green icons. Moving the sliders too far might result in false matches.

7 **Save** your advanced settings.

8 **Save** changes to your command (and remember to check your state or script in).

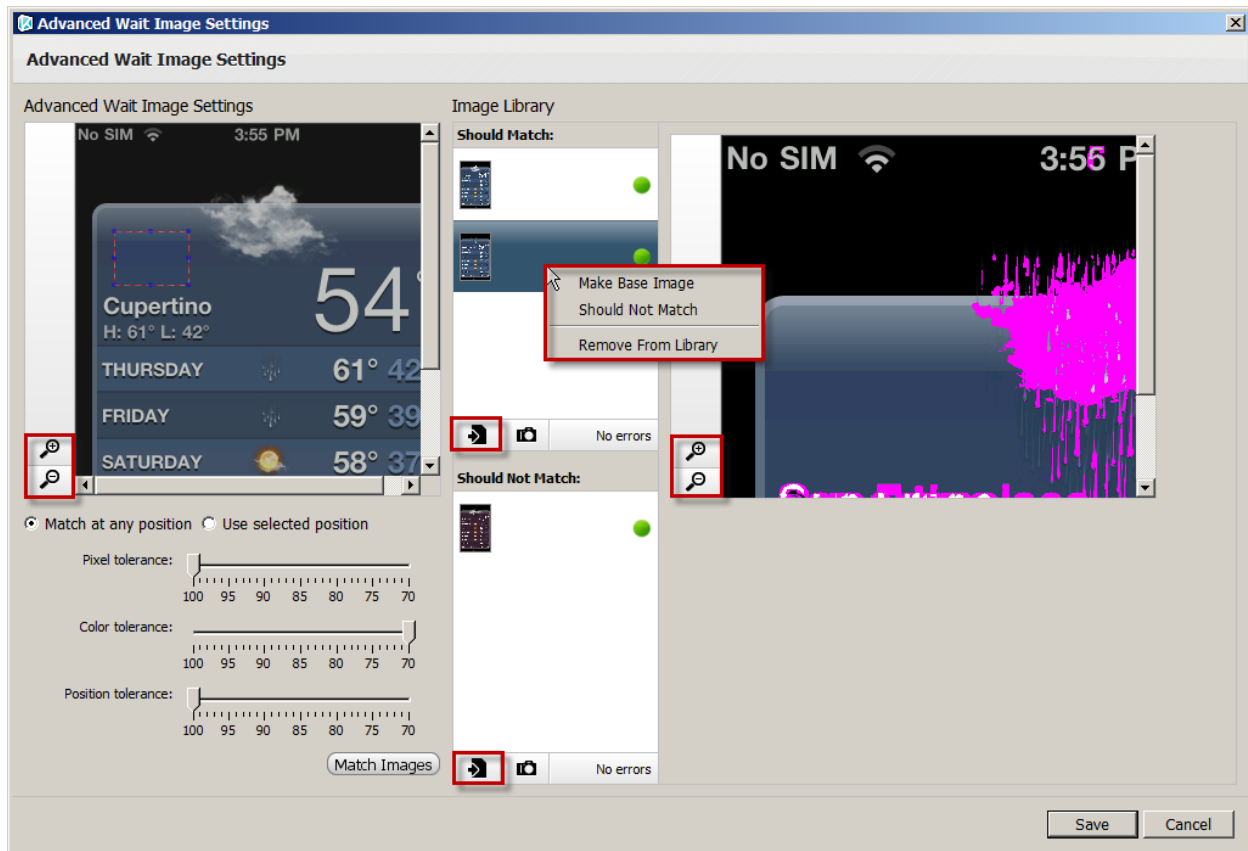
Additional controls are available in the advanced mode for reference images and are highlighted in Figure 5-12 below:

- ◆ Use the zoom buttons  to enlarge or shrink the device screen display.
- ◆ You can choose to import an image  to add to the **Should Match** or **Should Not Match** libraries.

CAUTION Only import images from test results that were directly captured from the device and uploaded to the web portal. Images from any other source might not have the resolution required for the DeviceAnywhere pixel-to-pixel image matching technology, resulting in verification failures.

- ◆ Right-click any item in the **Should Match** or **Should Not Match** library for additional controls:
 - **Make Base Image**—Replaces base image with selected image.
 - **Should Not Match**—Moves an image from the match to the mismatch library.
 - **Should Match**—Moves an image from the mismatch to the match library.
 - **Remove From Library**—Deletes image from library.

Figure 5-12 Wait Image Advanced Settings Additional Controls



5.3.2 Text-Based Reference Points

You can use a string of text from a device screen to define a text-based reference point. For instance, you can use static text from an application to verify that a device has opened it correctly. Instead of the pixel-

to-pixel matching technology used with reference images, Keynote uses optical character recognition (OCR) technology to verify that text on the device screen matches the reference string. OCR technology can match a string of text regardless of changes in font size or shape on the device.

You can create a text-based reference point in any the following:

- ◆ In the [Wait Event command](#), add a branch based on a text-based reference point by selecting **Wait for text** from the drop-down list.

The current device screen is automatically captured in the command. You can then set controls for text extraction.

See [Script Logic](#) for a discussion of branches in Wait Event.

- ◆ In the [Navigate To command](#), select the **Condition** tab and select **Wait for text** from the drop-down list provided. The current device screen is automatically captured in the command. You can then set controls for text extraction (see [Script Logic](#)).
- ◆ [Create a state](#) and select the device for which you wish to create an implementation. Select **Wait text** from the **State Type** drop-down list. You can then define reference text.

Figure 5-13 Defining Reference Text in a State

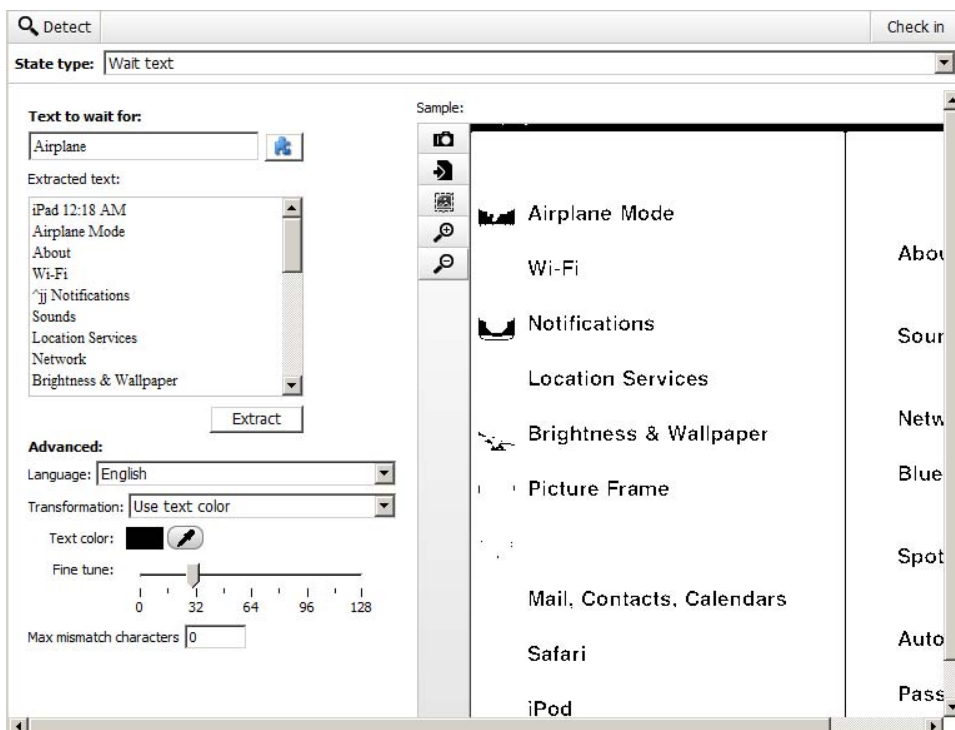
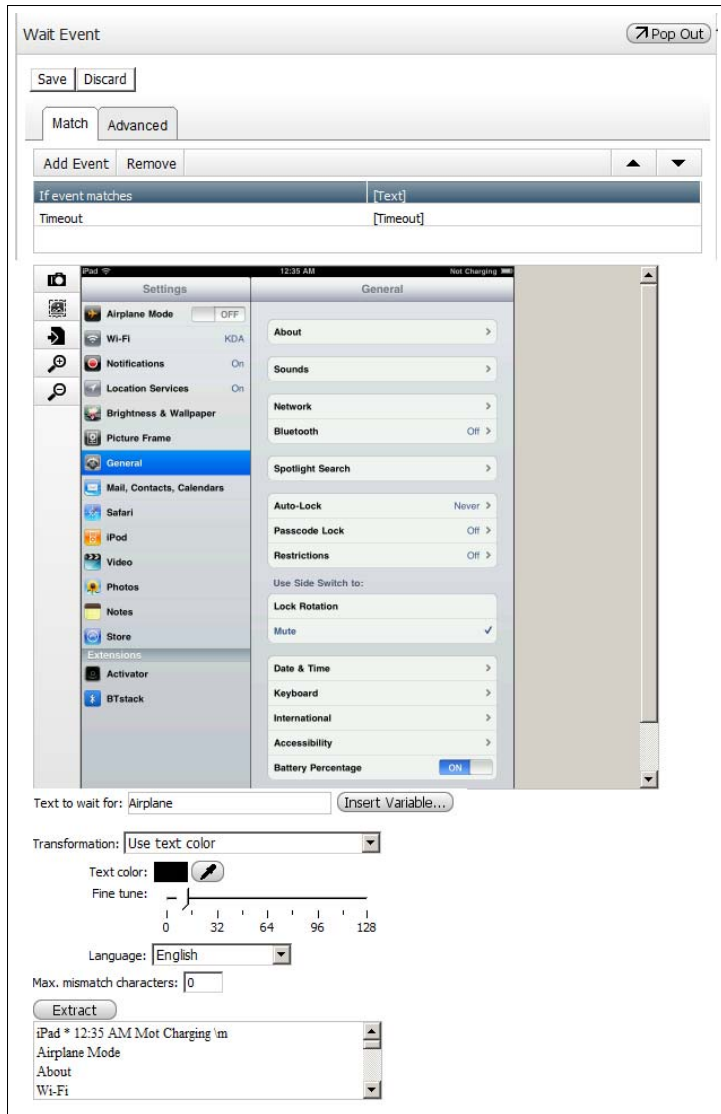


Figure 5-14 Defining Reference Text in the Wait Event Command



As mentioned in [Reference Points](#) above, the mechanism for defining a text-based reference point is by and large the same whether in a command (i.e., Wait Event, Navigate To) or in a state.

The sections below describe how to:


- ◆ [Define reference text](#) in a state or a command.
- ◆ [Transform text](#) when defining a string or reference text to aid extraction.

5.3.2.1 Defining Reference Text


Be sure to have the device acquired and your state/action checked out so you can capture and extract text from device screens.

To define reference text:

- 1 In the device pane, navigate to the device screen you would like to use to define your reference text.
- 2 Select text **Language** from the drop-down list provided (the default is English).

- 3 In your command or state, ensure that you can see an image of the current device screen. If not, take a snapshot .
- 4 Optionally in a command, limit text extraction to a screen region by selecting an area with your mouse.
- 5 *Transform* text from the device screen to facilitate extraction. Select a text **Transformation** type from the drop-down list. See [Text Transformations](#) below for detailed instructions.
- 6 Click **Extract** to view extracted text from the device screen (see Figure 5-13 and Figure 5-14 above).
- 7 Enter a subset of the extracted text in the **Text to wait for** field. This defines your reference text string. Text must first be extracted (and preferably [transformed](#)) before you enter a string in this field or your state will not be detected.

As verification is text based, you can pass in the value contained in a variable or parameter. To select a parameter or variable:

- In a state, click the puzzle piece icon .
- In a command, click **Insert Variable**.

The selected variable is displayed enclosed within percent (%) signs in the **Text to wait for** field, e.g., %Zip Code%. See [Parameters and Variables](#) for detailed information on creating and calling parameters and variables.

- 8 Specify the maximum number of characters in the string that can be mismatched (**Max mismatch characters**). For example, if you specify that any two characters can be mismatched in the string “California,” the term “Callforlna” will be considered a match.
- 9 Enter the **Wait Time** or **Timeout** (in seconds) within which the reference text must be matched.

NOTE In Navigate To, you can specify a timeout in the **Logic** tab. In Wait Event, you will be able to set the command timeout in Wait Event’s default **Match** branch.

- 10 Select **Checkpoint** to specify [proofs](#) (screenshots) for display in test results.
- 11 In a state, you can click **Detect** to test that an area matching the reference image can be found on the device screen. This ensures that you have defined your state correctly.

DeviceAnywhere Studio displays the result of state detection above the script canvas.



- 12 Check in your state/script.


5.3.2.2 Text Transformations

When using text from a device screen to [define reference text](#), you must facilitate text recognition by transforming the text. To transform text in DeviceAnywhere Studio:

- ◆ Select the [color of text](#) for the transformation engine to extract. Use this, for instance, to specify the color of text you would like to extract from a device screen when all the text you are looking for is a consistent color.
- ◆ Specify the [background color](#) of the area from which the transformation engine should extract text. Use this, for instance, to extract text from an area of the screen with a certain background color.
- ◆ Convert a colored device screen image to [black and white](#). Use this when selecting text on a graded background.
- ◆ Adjusting the [contrast](#) of the device screen. Use this method when text and background colors are not sufficiently differentiated, e.g., light green text on a dark green background.



Text Color

To transform text by selecting text color:

- 1 Select **Use text color** from the **Transformation** drop-down list.
- 2 Click the eyedropper icon  to select a *text color*.

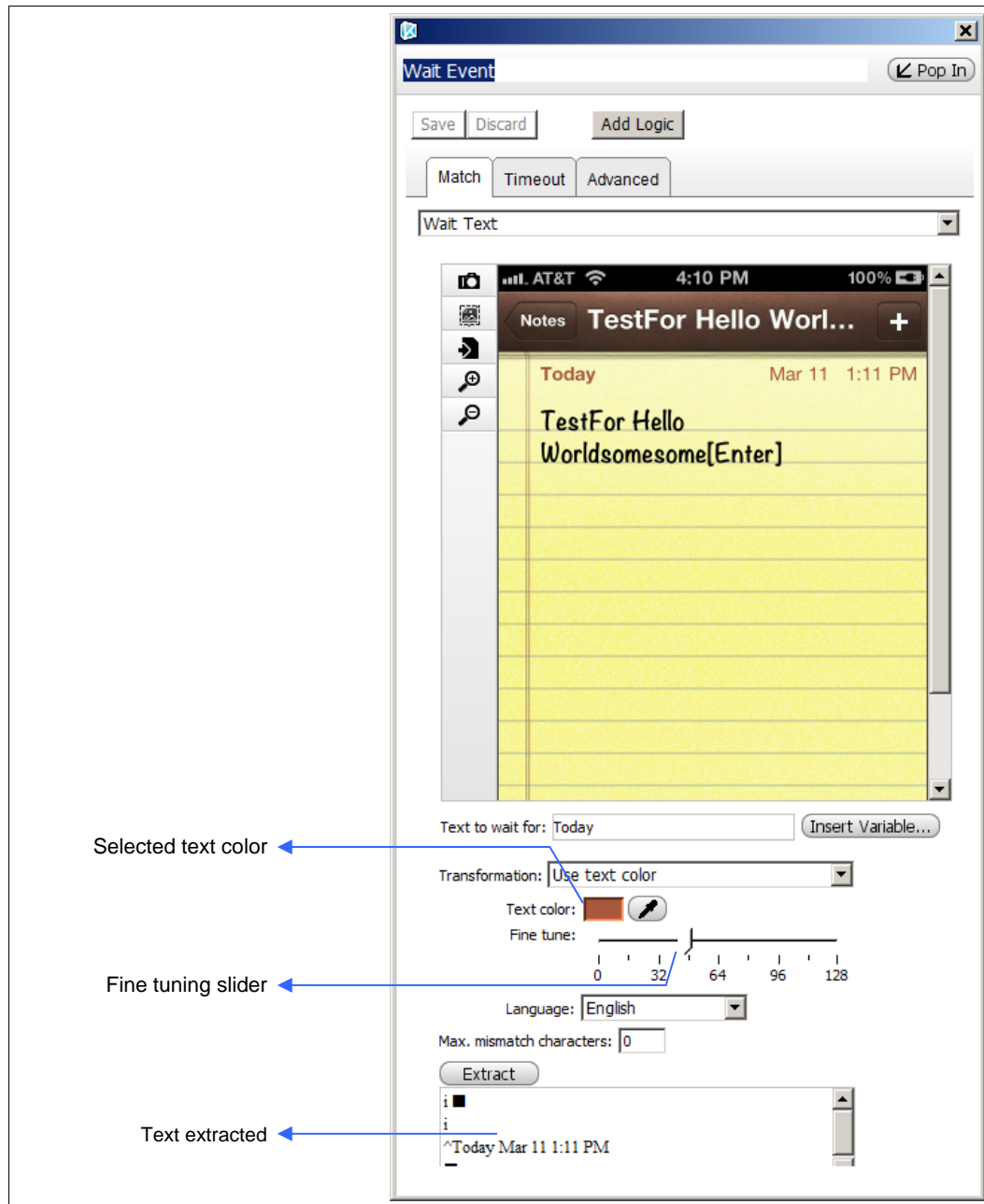
In a state, this opens up the device screen image in a pop-up **Pick Color** window.



- 3 Move your mouse over the device screen image. As you hover over the image, the color chip next to the eyedropper icon   dynamically displays color. Click to select a text color—this is displayed in the color chip. Hover over the chip to see the RGB component values of the selected color.
- 4 Use the slider to set threshold values for variations in the color of extracted text. Increase the slider to the extent required to reliably extract the text you want.
- 5 Click **Extract** and then select your reference text as described in [Defining Reference Text](#) above.

The figure below shows an example of specifying reference text by selecting text color.

Figure 5-15 Transforming Text Color (in Wait Event)



Text Background

To transform text by selecting text background color:




- 1 Select **Use text background** from the **Transformation** drop-down list.
- 2 Click the eyedropper icon  to select a *background color*.
In a state, this opens up the device screen image in a pop-up **Pick Color** window.
- 3 Move your mouse over the device screen image. As you hover over the image, the color chip next to the eyedropper icon   dynamically displays color. Click to select a background color—this is displayed in the color chip. Hover over the chip to see the RGB component values of the selected color (see Figure 5-16 below).
- 4 Use the slider to set threshold values for variations in the background color of extracted text. Increase the slider to the extent required to reliably extract the text you want.
- 5 Click **Extract** and then select your reference text as described in [Defining Reference Text](#) above.

Figure 5-16 below shows an example of specifying reference text by selecting text background color.

Converting to Black and White

To transform text by converting the device screen image to black and white:




- 1 Select **Convert to black and white** from the **Transformation** drop-down list.
- 2 Click the eyedropper icon  to select the *color that divides black from white*.
In a state, this opens up the device screen image in a pop-up **Pick Color** window.
- 3 Move your mouse over the device screen image. As you hover over the image, the color chip next to the eyedropper icon   dynamically displays color. Click to select a color that divides black from white—darker colors will be converted to black and lighter colors will be converted to white. The color chip displays the color. Hover over the chip to see the RGB component values of the selected color.
- 4 Use the slider to set threshold values for variations in the dividing color. A negative value makes the selected color lighter; a positive value makes it darker.
- 5 Click **Extract** and then select your reference text as described in [Defining Reference Text](#) above.

Figure 5-17 shows an example of specifying reference text by transforming the device screen image to black and white in a state.

Figure 5-16 Transforming Text Background (in Wait Event)

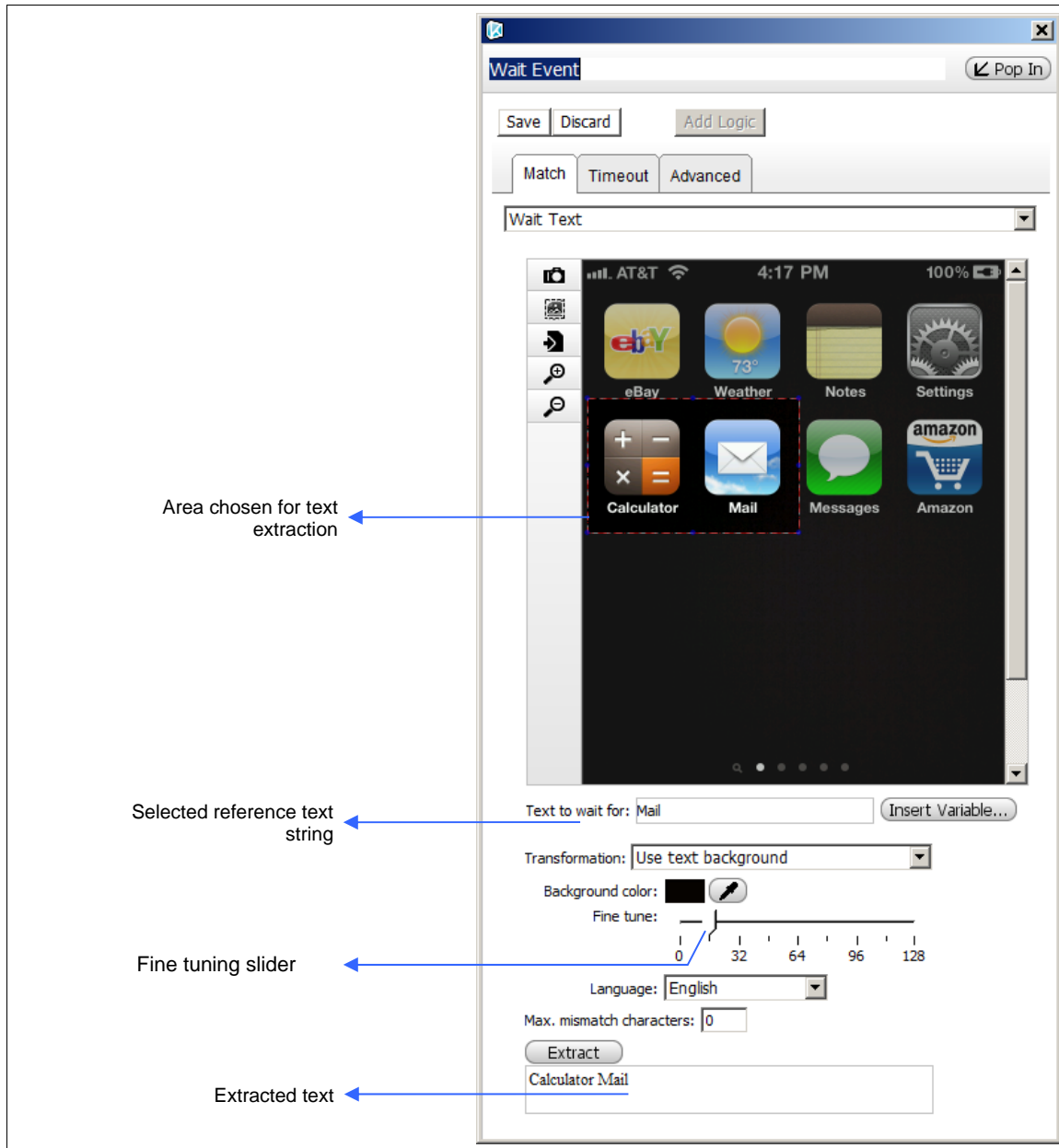
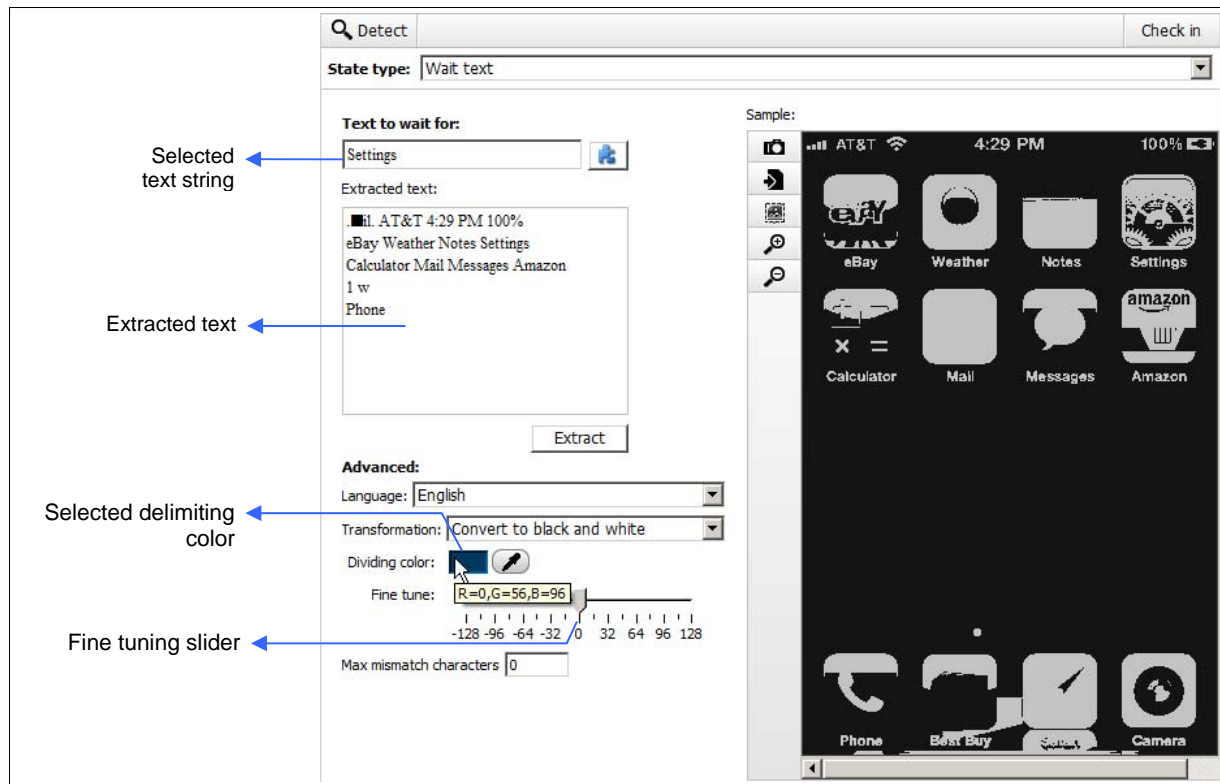





Figure 5-17 Transforming Device Image to Black and White (in a State)



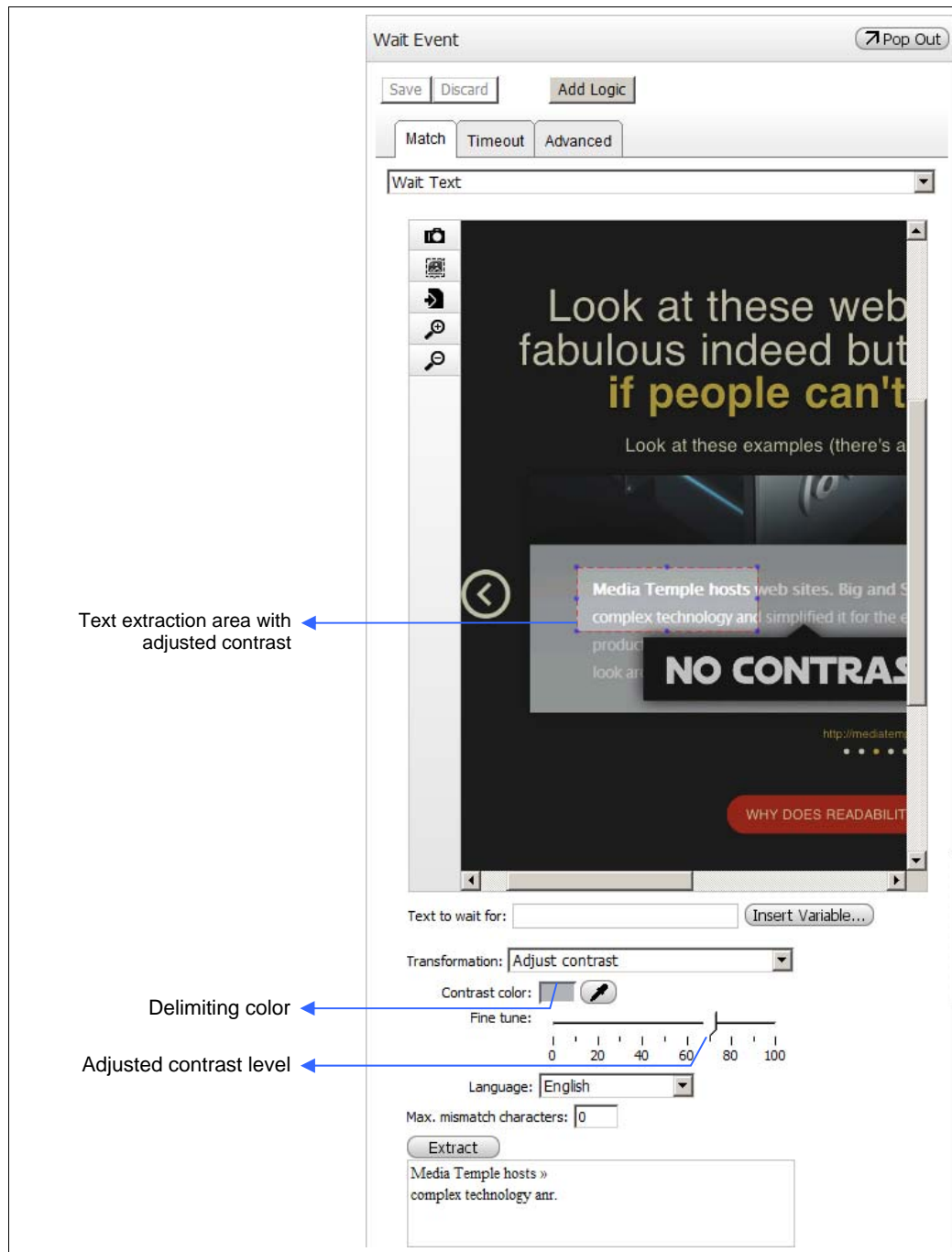
Adjusting Contrast

To transform text by adjusting the contrast of the device screen image:

- 1 Select **Adjust Contrast** from the **Transformation type** drop-down list.
- 2 Click the eyedropper icon  to select the *delimiting color*—*darker colors will be darkened while lighter colors will be lightened*.
In a state, this opens up the device screen image in a pop-up **Pick Color** window.
- 3 Move your mouse over the screen image. As you hover over the image, the color chip next to the eyedropper icon   dynamically displays color. Click to select a dividing color—this is displayed in the color chip. Hover over the chip to see the RGB component values of the selected color (see Figure 5-18 below).
- 4 Use the slider to adjust contrast.
- 5 Click **Extract** and then select your reference text as described in [Defining Reference Text](#) above.

The figure below shows an example of specifying reference text by adjusting contrast.

Figure 5-18 Adjusting Contrast of Device Image (in Wait Event)



5.3.3 Audio Reference Points

You can call for your script to detect device audio for verification. For instance, you can set an audio reference point to verify that a multimedia file from a web site has been played successfully.

You can create an audio reference point in one of the following ways:

- ◆ In the [Wait Event command](#), add a branch based on an audio reference point by clicking **Add Logic** and then **Add Event**. Select **Wait for audio** from the drop-down list.

NOTE Wait Event does not define a device-specific sound as a reference point. Instead, it simply waits to hear any device audio (or a specified DTMF sequence). While Wait Event is a script-specific command, audio reference points can be safely copied from one script to another.

- ◆ [Create a state](#) and select the device for which you wish to create an implementation. Select **Wait audio** from the **State Type** drop-down list.

Figure 5-19 Defining an Audio Reference Point in Wait Event

The screenshot shows the 'Wait Event' configuration window. At the top, there are 'Save' and 'Discard' buttons. Below them are 'Match' and 'Advanced' tabs. The 'Match' tab is active, showing a table with 'Add Event' and 'Remove' columns. The first row contains 'If event matches' and '[Audio]'. Below the table is a 'Timeout' field with '[Timeout]'. A dropdown menu is set to 'If audio is detected'. Under this dropdown, there are two radio button options: 'Detect any sound' (selected) and 'Detect DTMF tone'. The 'Detect any sound' option has a 'Volume sensitivity' slider between 'High' and 'Low'. A note below the slider says 'Note: use lower volume sensitivity to filter out ambient noise.' The 'Detect DTMF tone' option has a 'Tone Sequence' text field and a 'Play Tone' button.

Figure 5-20 Defining an Audio Reference Point in a State

Set your audio reference point as follows:

- 1 To detect any device audio, select the **Wait Any** radio button in a state and **Detect any sound** in Wait Event. To detect a sequence of DTMF tones, select **Wait DTMF Sequence** in a state and **Detect DTMF Tone** in Wait Event. Use the latter option, for instance, if your script navigates through and receives confirmation DTMF tones from an IVR system.

You can also use this option if pairing the Play Audio and Wait Audio commands in a two-device test case to concurrently play and wait for DTMF tones from one device to the other.

- 2 Enter a number in the **Audio detect threshold** field in a state to minimize the possibility of mistaking underlying noise for device audio. In a command, move the **Volume sensitivity** slider—move the slider to the **Low** end if you expect device audio to be compromised by interference or static.
- 3 If you have chosen to wait for DTMF tones, enter the sequence of numbers, e.g., 1233, in the DTMF tone sequence field.
- 4 Click **Play** to test the DTMF tones associated with the number sequence.
- 5 Check in your state/script.

5.4 Proofs

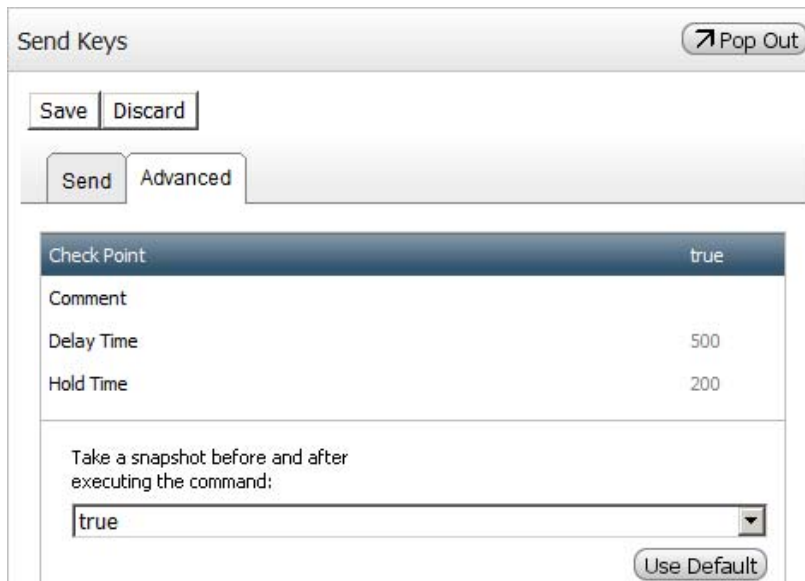
Proofs are device screenshots or video captured during script runs and displayed in [test results](#). Proofs are specified, gathered, and displayed for each scripting command and can be useful in troubleshooting failed steps.

You can specify proofs in the [Checkpoint control](#) of any command, in the [Execute Action](#) command, or by using scripting commands in the [Capture category](#) (Toggle Recording or Capture From Device):

5.4.1 Checkpoint Control

The **Advanced** tab of most commands enables you to capture the first and last screen of the device during command execution. Select **Checkpoint** and then **true** from the drop-down list that appears. **Use Default** reverts to the default setting (**false**).

Figure 5-21 Checkpoint Tab in Send Keys



5.4.2 Proofs in Execute Action

In the Execute Action **Advanced** tab, select the appropriate **Proof Type** from the drop-down list (see Figure 5-22).

NOTE Proof settings for Execute Action are separate from proof settings for individual commands in the action being called. In test results, results for individual commands are displayed nested under those for Execute Action.

- ◆ **Do not capture anything** for no proofs
- ◆ **Capture last image** to capture the last device screen of the command
- ◆ **Capture first and last images** to capture the first and last screen of the command
- ◆ **Capture images at fixed interval** to capture images at regular intervals (specified in seconds in the **Image Interval** field)
- ◆ **Capture whole page** to capture a scrolling image (e.g., of a Web page that is longer than the device screen)
- ◆ **Capture video** to capture device video in MPEG format

You can opt to **Include Audio** information in your video proof.

Figure 5-22 Specifying Proofs in Execute Action



5.4.3 Commands in the Capture Category

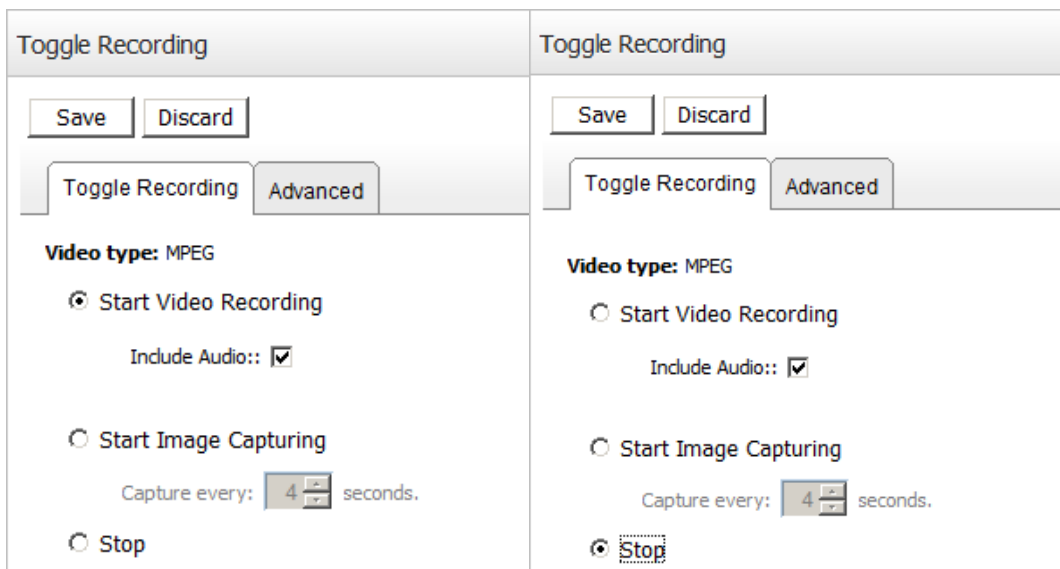
The Capture category contains visual scripting commands that you can drag onto a script canvas to capture device images, video, and log files for insertion into test results.

In the Toggle Recording command (below), you can start and stop capturing:

- ◆ MPEG video (**Start Video Recording/Stop**)—you can opt to **Include Audio** information.
- ◆ Images at regular intervals (**Start Image Capturing/Stop**)—enter the interval in seconds.

In your script, use a pair of these commands to indicate start and stop points for capturing device video/snapshots.

Figure 5-23 A Pair of Toggle Recording Commands



Capture From Device is a single command for capturing snapshots or video of the current device screen. The script does not proceed to the next command until capture using this command is completed. Use this command to capture automatic and dynamic changes to the device screen as when streaming video. You can capture:

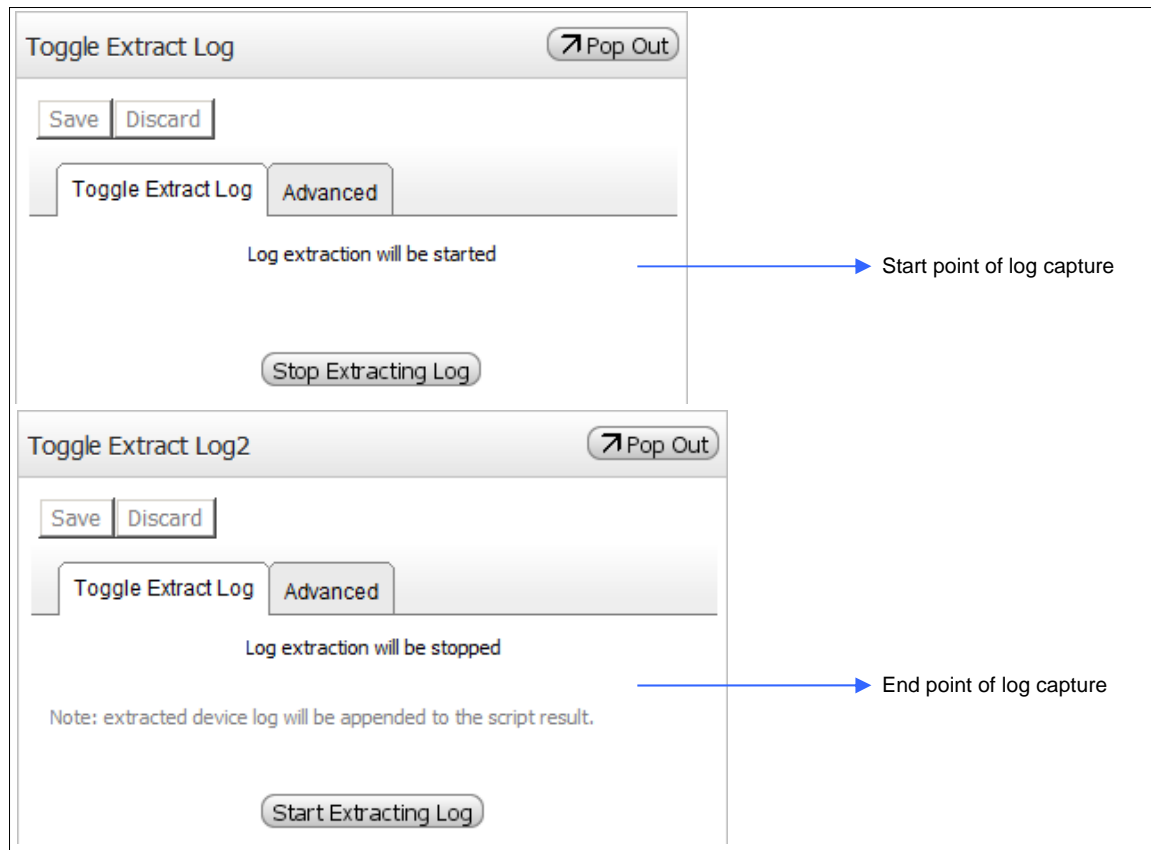
- ◆ A single image
- ◆ Images of a scrolling screen—several screenshots are taken of a scrolling screen.
- ◆ Images at fixed intervals for a set length of time—enter the interval and the length of time in the fields provided.
- ◆ Video

Figure 5-24 Capture From Device—Image and Video Options

The figure displays two screenshots of the 'Capture From Device' dialog box. The top screenshot shows the 'Image' type selected in the 'Type' dropdown. Under 'Image', three radio button options are visible: 'Take a single snapshot' (selected), 'Images of scrolling screen' (with a sub-note: 'A scrolling screen is captured in multiple screenshots taken till the screen stops scrolling. Resulting screenshots are series of image files.'), and 'Start Image Capturing' (with sub-fields for 'Capture every: 2 seconds' and 'Capture for: 30 seconds'). The bottom screenshot shows the 'Video' type selected. It features a 'Capture for' field set to '60 seconds' with a note: 'Note: execution will be paused while the video is captured.' Below this, 'Video type: MPEG' is displayed, and the 'Include Audio' checkbox is checked.

Toggle Extract Log indicates start and end points in your script for capturing the device log, which is displayed in test results. In your script, use a pair of these commands to indicate start and stop points for capturing the device log. Insert the first command where you want to start capturing the log. Insert the second command where you want to end capturing device log and click **Stop Extracting Log**.

Figure 5-25 A Pair of Toggle Extract Log Commands



5.5 Parameters and Variables

This section discusses [parameters](#) and [variables](#), which help implement complex script logic (data variations, loops, branches). This section also discusses the [Extract Text](#) command, which extracts a text string from a device screen and allows you to store it in a parameter or variable. Finally, this section discusses variables that refer to entire [data sets](#) instead of a single value. The data set can either be created/entered in DeviceAnywhere Studio or imported from an external source.

Parameters allow you to enter data dynamically into a device at the start of an ad hoc script run, i.e., specify a different data value for each run. For example, if you are testing multiple web sites on several devices, you can specify a different URL for each script run using a parameter. Each update of the parameter is automatically valid for all implementations. Parameter values are used as the basis for script logic, to specify device input, or to define a string of reference text.

You can define parameters for actions or test cases. An action parameter, once created, can be used for all implementations. Test case parameters are *not* available for use in constituent actions.

Variables allow you to set/store values in your script to be used as the basis for script logic, to specify device input, or to define a string of reference text. You can set variable values in various ways, including passing in text extracted from a device at run time. Unlike parameters, variable values cannot be specified at the start of a script run. (You can, however, use a parameter in place of the variable value and then provide the parameter value at runtime.) Variables can be associated with multiple values contained in a *data set*. Variables can be *global* or *script specific*:

- ◆ *Script variables* are script specific, i.e., they are not available for use outside the test case or action in which they are created.
- ◆ *Global variables* are defined at the project level and can be used in any project script. The value stored in a global variable in one script can be used in another project script that is part of the same run session.

5.5.1 Parameters

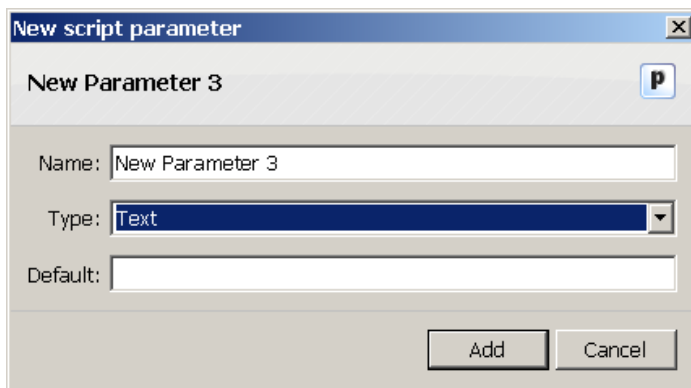
This section describes how to [create parameters and set default values](#), [call parameters](#) in commands, and [set parameter values](#).

5.5.1.1 Creating Parameters and Setting Default Values

Action and test case parameters are created in **Parameters** tab found either in the [action properties dialog box](#) or the [test case properties dialog box](#). You can also set default parameter values and view a list of existing parameters in the **Parameters** tab. Parameters can be text, numeric, or Boolean—text parameters can have alphanumeric values, numeric parameters can have only numeric values, and Boolean parameters have either a “true” or “false” value.

To *create a parameter*:

- 1 Check out your action implementation or test case.
- 2 Right-click the script in the project directory > **Properties** to open action or test case properties.
- 3 Select the **Parameters** tab (for example, see Figure 4-2 Action Properties – Parameters).
- 4 Click **Add** (or **Remove** to remove a parameter). This opens up the New script parameter dialog box.

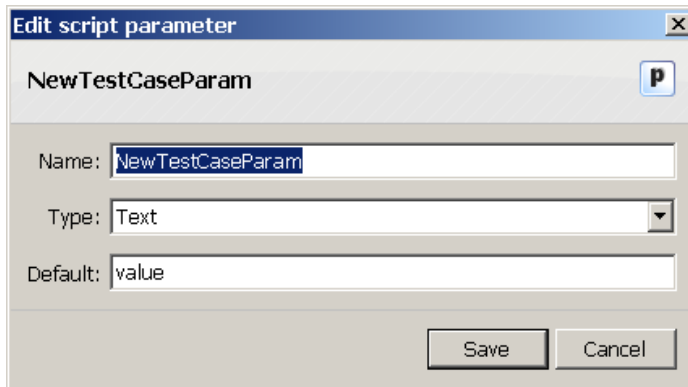


- 5 Enter a **Name** for the new parameter.
- 6 Choose a parameter **Type**. You can choose **Text**, **Number**, or **Boolean**.
- 7 If you wish, enter a **Default** value for the parameter.
- 8 Click **Add**. The parameter is now listed in the **Parameters** tab (see Figure 4-2 Action Properties – Parameters and Figure 7-2 Test Case Properties – Parameters).
- 9 Click **Save** to save your changes to action/test case properties.

To *edit default parameter values* in action/test case properties:

- 1 Check out your action implementation or test case.
- 2 Open the action or test case properties dialog box and select the **Parameters** tab.

- 3 Select a parameter from the list and click **Edit**. This brings up the Edit script parameter dialog box.




- 4 Change the value as desired. You can also change the parameter **Name** and **Type** and click **Save**.
- 5 **Save** your changes to action/test case properties.

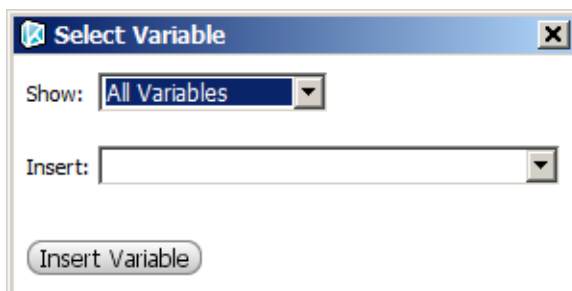
5.5.1.2 Calling Parameters

In commands such as [Send Keys](#) or [Navigate To](#) that allow you to enter keystrokes on a device, you can use a parameter in place of data entry and provide the parameter value at the start of a script run. Parameters are also used in the [Branch](#) and [Loop](#) commands as the basis for script logic. This is discussed in detail in [Script Logic](#).

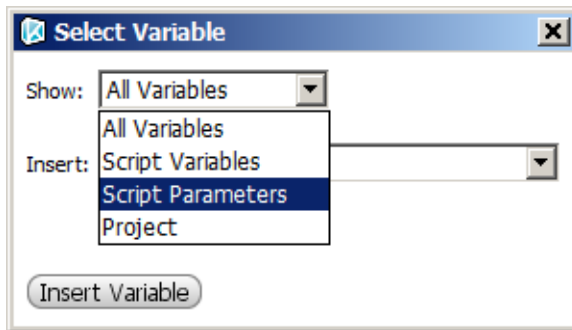
NOTE You must [create a parameter](#) before you call it in a command.

Several commands, (e.g., Send Keys, Execute Action, Navigate To, Extract Text) allow you to insert a parameter into a field by clicking the **Insert Variable**  icon or button. Using Send Keys as an example, the procedure below describes how to call a parameter in a command:

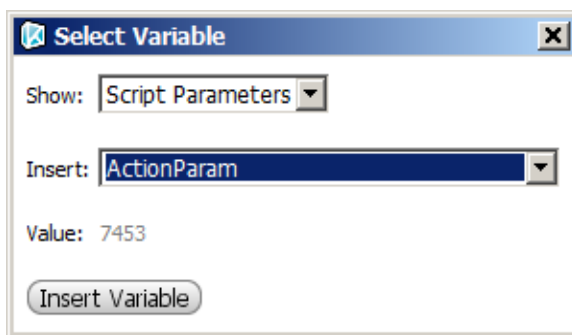
- 1 Drag the Send Keys command onto your script canvas and double-click to open it.
- 2 Click **Insert Variable** next to **Text to Send**. This brings up the Select Variable dialog box.



- 3 Select **Script Parameters** from the **Show** drop-down list.

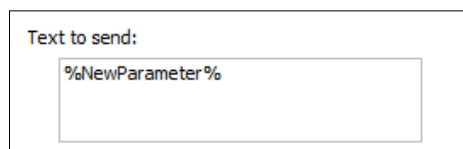


- 4 Select a parameter from the **Insert** drop-down list. The default value of the selected parameter is displayed.



CAUTION Be sure that the value of the parameter is appropriate for the device field that text is being entered into. For example, if your parameter passes a text string into a numeric field on your device, your script will fail.

- 5 Click **Insert Variable**. This displays the parameter (enclosed within percent signs) in **Text to Send**.



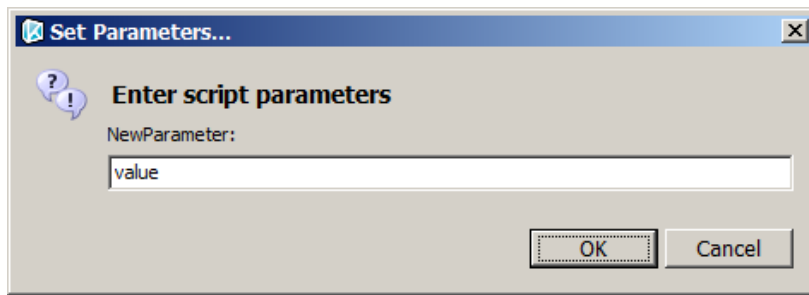
You can also type in the parameter name in fields where it is appropriate to use a variable or parameter. Type parameter names enclosed within percent (%) signs, e.g., `%Website%`. Type the parameter name correctly to ensure that your script does not fail. Be sure that the value of the parameter is appropriate for the **Key Mode** of the device field that text is being entered into: An alphanumeric field supports both text and numeric parameter values; however, a numeric field supports only numeric parameter values (see [Specifying Device Input](#)).

5.5.1.3 Setting Parameter Values

You can set parameter values when you start a [script run](#). For instance, if your script calls for sending a text message, you can create a parameter for the destination phone number as well as the message body, and specify parameter values at runtime.

When you run your action or test case from the [Automation](#) view (click **Run** or **Advanced** > **Run in Debug Mode** above the script canvas), you must specify any parameter values in the Set Parameters dialog box that appears. The dialog box lists script parameters with default values, if any.

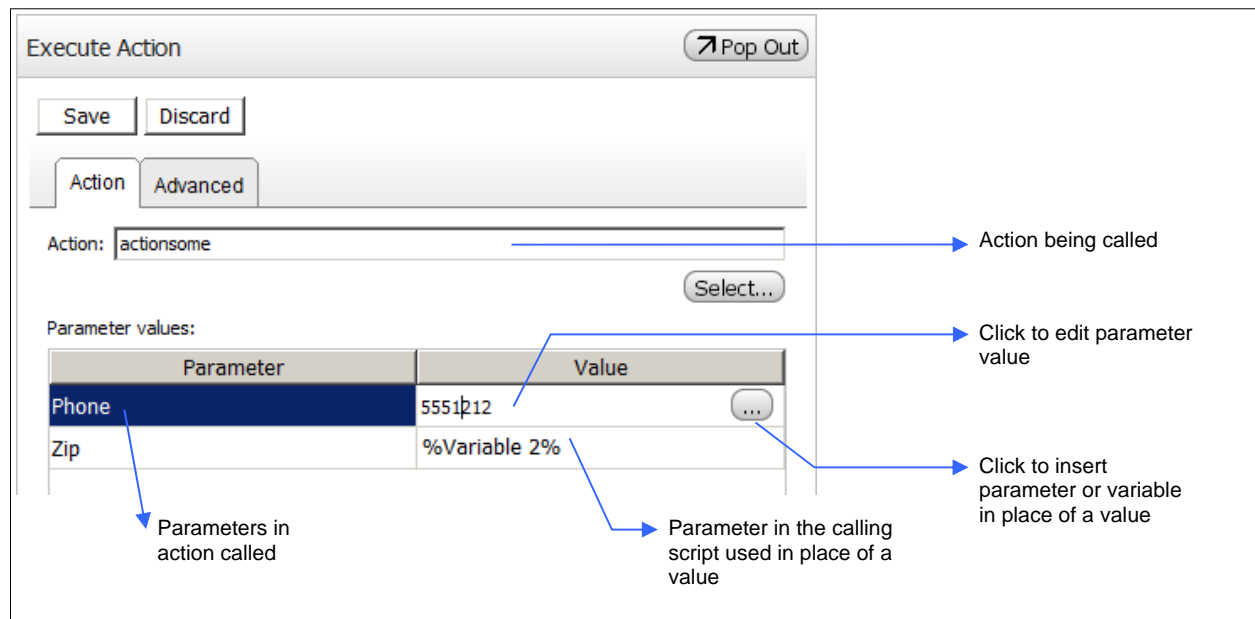
Figure 5-26 Setting Parameter Values



Before clicking **Run** or **Advanced > Run in Debug Mode** in DeviceAnywhere Studio to run a script that calls an action using the [Execute Action command](#), any parameters for the action being called must be set in Execute Action command properties:

- 1 Select Execute Action to view the command properties pane.
Parameters for the selected action called are listed with default values (see Figure 5-27 below).
- 2 Click a value to edit it. Click the button next to it to insert a parameter or variable from the calling script in place of a value.

Figure 5-27 Editing Parameter Values in Execute Action



When running a test case in a manual test cycle from [Test Case Runtime](#), you must specify values for parameters in constituent actions in the Set Parameters dialog box.

5.5.2 Variables

This section describes how to [create global variables](#), [create script variables](#), [specify variable values using the Set Variable command](#), and [use variables in commands](#).

Variables can be text, numeric, or Boolean—text variables can have alphanumeric values, numeric variables can have only numeric values, and Boolean variables have either a “true” or “false” value. Additionally, variables can be associated with an entire data set instead of just a single value—see

[Working with Data Sets](#). Once the variables have been created, you can set their values in the script, and then call them from commands (e.g., Branch, Loop).

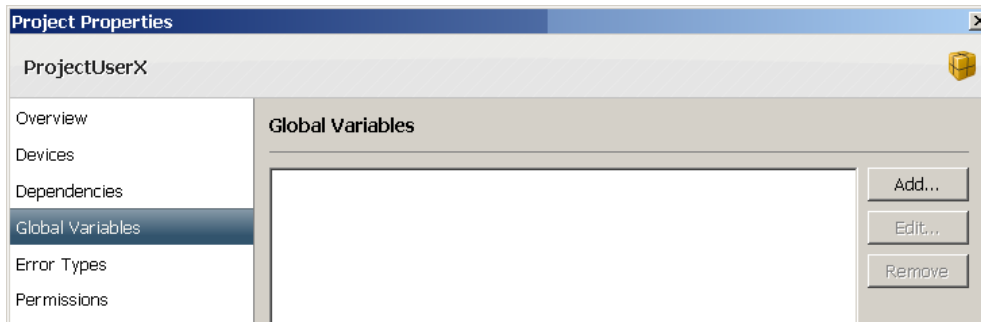
NOTE If you have created a variable that is associated with a data set, you can only call it using the Loop command—see [Working with Data Sets](#).

5.5.2.1 Creating Global Variables and Setting Default Values

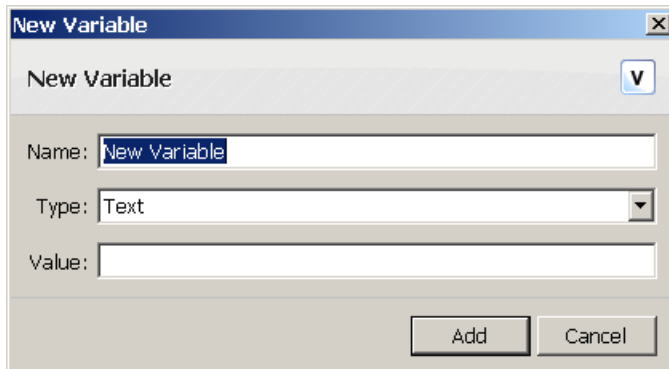
Global variables can store values from one project script and use them in another. For example, if an action consists of signing up to use a web site, you can store the credentials in a global variable and then call the variable from another action that logs in to and performs a transaction on that site. Both actions must be part of the same run session.

You can create global, or project, variables in the [Global Variables tab](#) of the [Project Properties dialog box](#). To create a global variable:

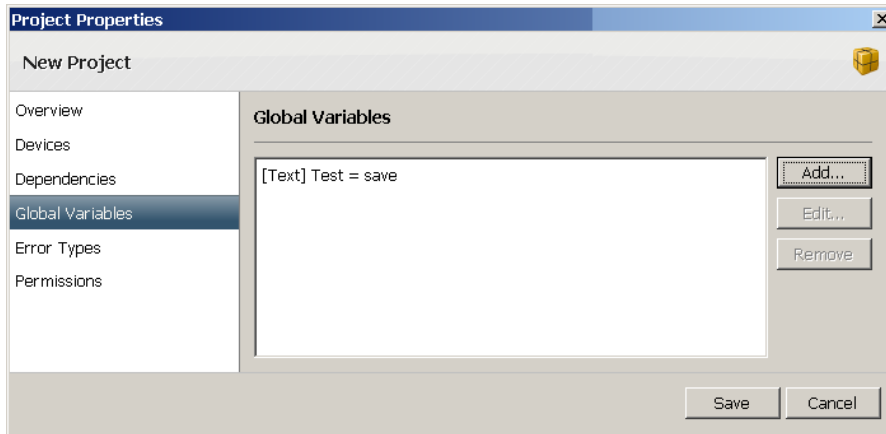
- 1 Right-click your open project in the Automation project list and select **Properties**.
- 2 Select the **Global Variables** tab.



- 3 Click **Add**. (Click **Remove** to remove a variable.) This brings up the New Variable dialog box.



- 4 Enter a **Name** for the new variable.
- 5 Choose a variable **Type**. You can choose **Text**, **Number**, **Boolean**, or **DataSet**. (Global variables can be associated with a data set—see [Working with Data Sets](#).)
- 6 If you wish, enter a **Default Value** for the variable.
- 7 Click **Add**. The variable is now listed in the **Global Variables** tab of the Project Properties dialog box.



8 **Save** your changes.

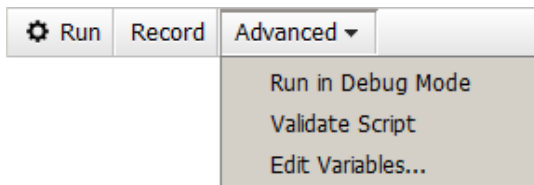
5.5.2.2 Creating Script Variables and Setting Default Values

You can create script variables by clicking **Advanced > Edit Variables** above the script canvas of an action implementation, test case, or test cycle.

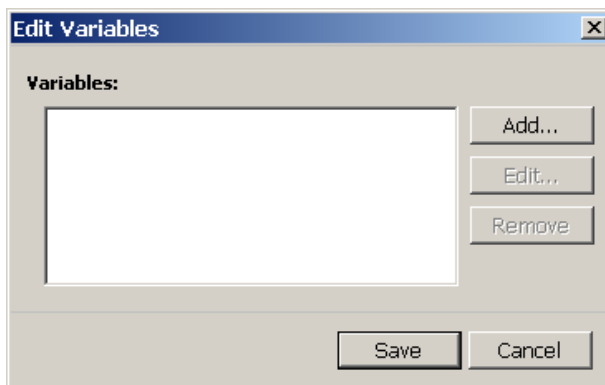
NOTE You can also create a script variable by clicking **Edit Variables** in the [Extract Text](#) and [Set Variable](#) commands.

To create a script variable:

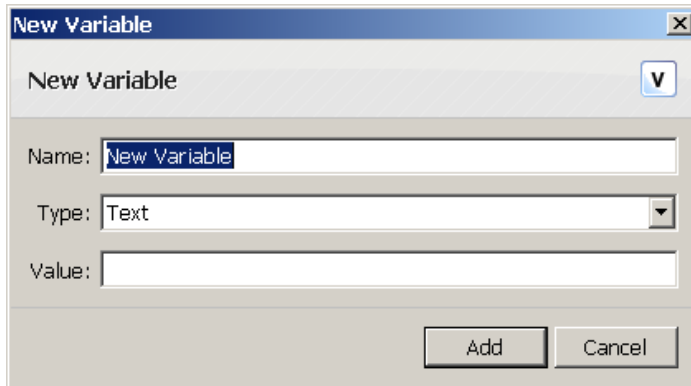
- 1 Check out your action implementation, test case, or test cycle.
- 2 Click **Advanced**, then **Edit Variables** above the script canvas.



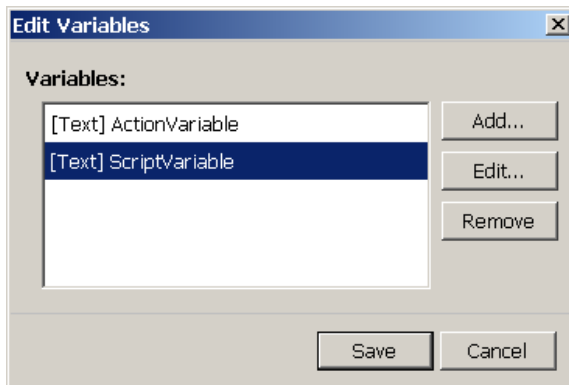
This brings up the Edit Variables dialog box.



- 3 Click **Add** (or **Remove** to remove a variable). This opens up the New Variable dialog box.



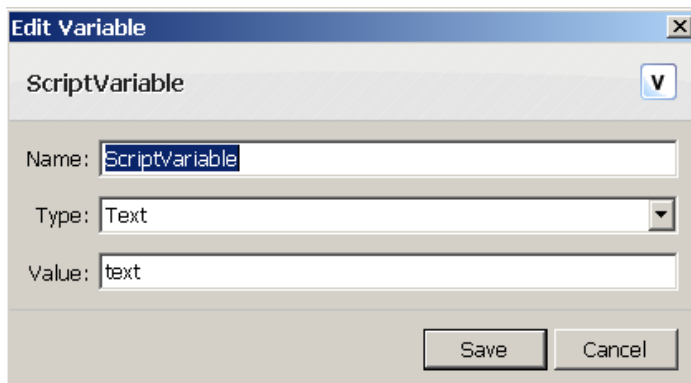
- 4 Enter a **Name** for the new variable.
- 5 Choose a variable **Type**. You can choose **Text**, **Number**, **Boolean**, or **DataSet**. (Script variables can be associated with a data set—see [Working with Data Sets](#).)
- 6 If you wish, enter a **Value** for the variable.
- 7 Click **Add**. The variable is now listed in the Edit Variables dialog box.



- 8 **Save** your changes.

To edit the default variable value:

- 1 Check out your action implementation or test case.
- 2 Click **Advanced > Edit Variables** above the script canvas. This brings up the Edit Variables dialog box.
- 3 Select a variable from the list and click **Edit**. This brings up the Edit Variable dialog box.



- 4 Change the **Value** and click **Save**. You can also change the variable **Name** and **Type**.
- 5 **Save** your changes in the Edit Variables dialog box.

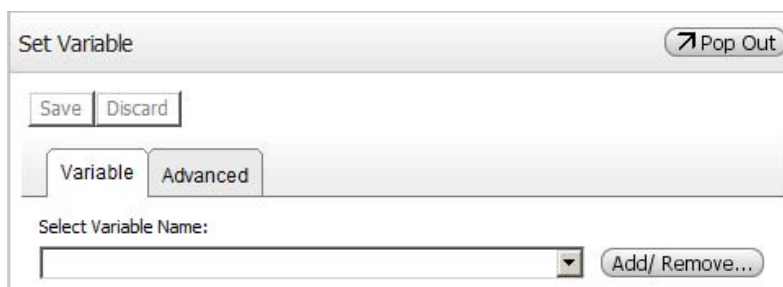
5.5.2.3 Setting Variable Values in the Set Variable Command

You can set different values for a variable at different points in your script by using the Set Variable command. For example, you can set a variable to different values depending on whether a reference point was found or not. You can then use the variable values as the basis for script branches. (Using variables in the [Branch](#) and [Loop](#) commands is discussed in detail in [Script Logic](#) below.)

NOTE You must [create a variable](#) first before you can set its value.

To set a variable value:

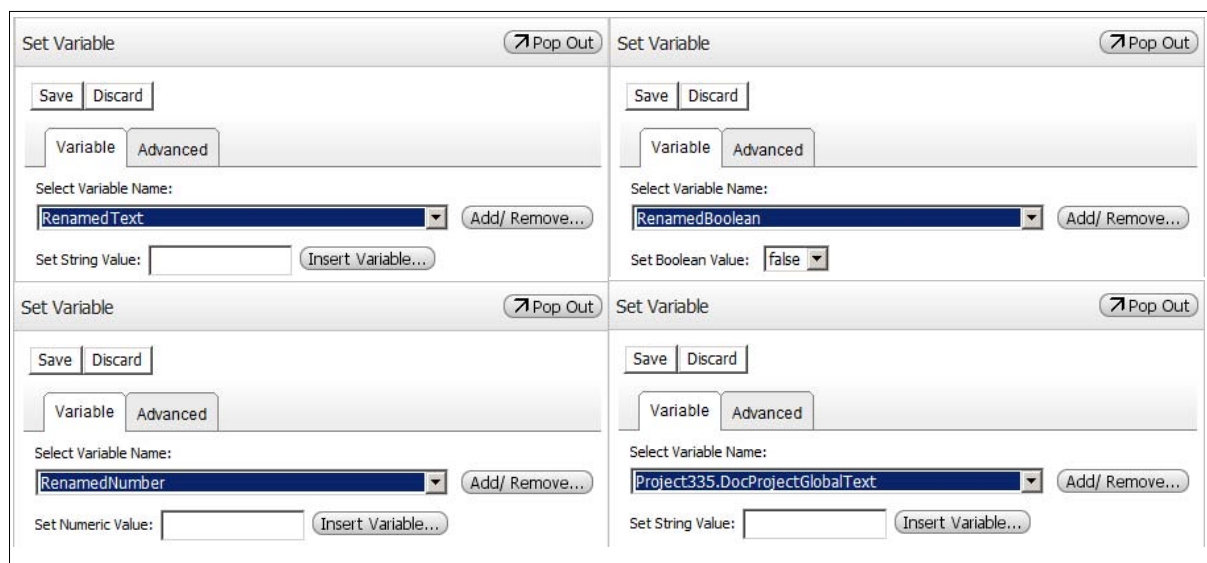
- 1 Drag the Set Variable  command onto your script canvas.



- 2 Choose the variable you wish to set a value for from the drop-down list. Project variables are prefixed by a project ID, e.g., Project88 . PhoneNumber.

This brings up a field for the variable value.

NOTE You cannot set a value in a data set using the Set Variable command.



NOTE You can [create and edit existing script variables](#)—click **Add/Remove**.

- 3 Enter a variable value. Depending on the variable type, you must enter a string, number, or choose a Boolean true or false value.

Instead of specifying a string or number, you can choose to pass the value stored in another parameter or variable:

- a Click **Insert Variable**. This brings up the Select Variable dialog box.
- b Choose a parameter, script variable, or global variable and click **Insert Variable**. The chosen parameter or variable (enclosed within percent signs) is displayed in place of the variable value.


- 4 **Save** your changes to the Set Variable command.

5.5.2.4 Calling Variables

Variables are used to [specify device input](#), as the basis for [script logic](#) or to define a string of [reference text](#). When you call a variable, your script will pick up the [default variable value](#) or the [value specified in the Set Variable command](#).

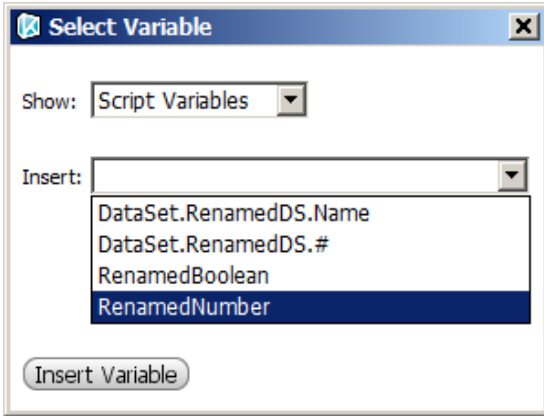
NOTES You must create a variable before you call it in a command.

If you have created a variable associated with a data set, you can only call it using the [Loop command](#)—see [Working with Data Sets](#).

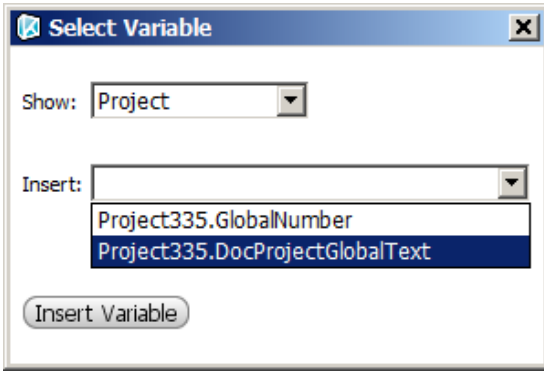
Several commands, (e.g., Send Keys, Execute Action, Navigate To, Extract Text) allow you to call a variable by clicking the **Insert Variable**  icon or button. Using Send Keys as an example, the procedure below describes how to call a variable from a command:

- 1 Drag the Send Keys command onto your script canvas and double-click to open it.
- 2 Click **Insert Variable** next to **Text to Send**. This brings up the Select Variable dialog box.

- 3 Select a variable:
 - To insert a script variable:
 - i Select **Script Variables** from the **Show** drop-down list.
 - ii Select a script variable from the **Insert** drop-down list. Do *not* select data set columns prefixed with DataSet. You can only use [data sets](#) with the Loop command.

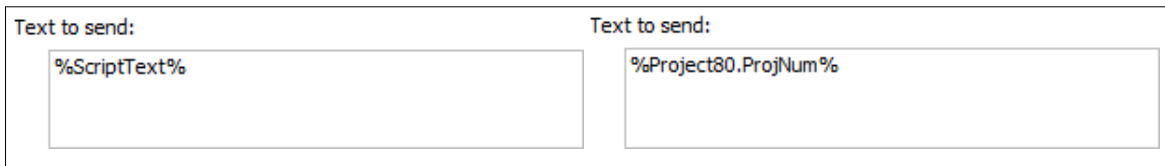


- To insert a global variable:
 - i Select **Project** from the **Show** drop-down list.
 - ii Choose the global variable from the **Insert** drop-down list. Do *not* select data set columns prefixed with DataSet. You can only use [data sets](#) with the Loop command.



4 Click **Insert Variable**.

This displays the selected variable (enclosed within percent signs) in **Text to Send**. Project variables are prefixed with the project ID, e.g., Project80.Variable.



NOTE Be sure that the value of the variable is appropriate for the field that text is being entered into. For example, if your variable passes a text string into a numeric field, your script will fail.

You can also type in the variable name in fields where it is appropriate to use a variable or parameter. Type variable names enclosed within percent (%) signs, e.g., %Website%. Be sure that the value of the variable is appropriate for the **Key Mode** of the device field that text is being entered into: An alphanumeric field supports both text and numeric parameter values; however, a numeric field supports only numeric parameter values (see [Specifying Device Input](#)).

CAUTION Typing in the name of a project variable is *not recommended* because of the special syntax required.


5.5.3 Extract Text Command

The Extract Text command uses character recognition technology to extract text from a device screen, which you can store in a [variable or parameter](#) for use elsewhere. This command can be used to store a value from one action in a global/project variable for use in another project action. For example, if an action consists of signing up to use a web site, you can use the Extract Text command to store the credentials in a global variable. You can then call the variable from another action that logs in to and performs a transaction on that site. Both actions must be part of the same run session.

As when defining text-based reference points, you must [transform text](#) to facilitate extraction.

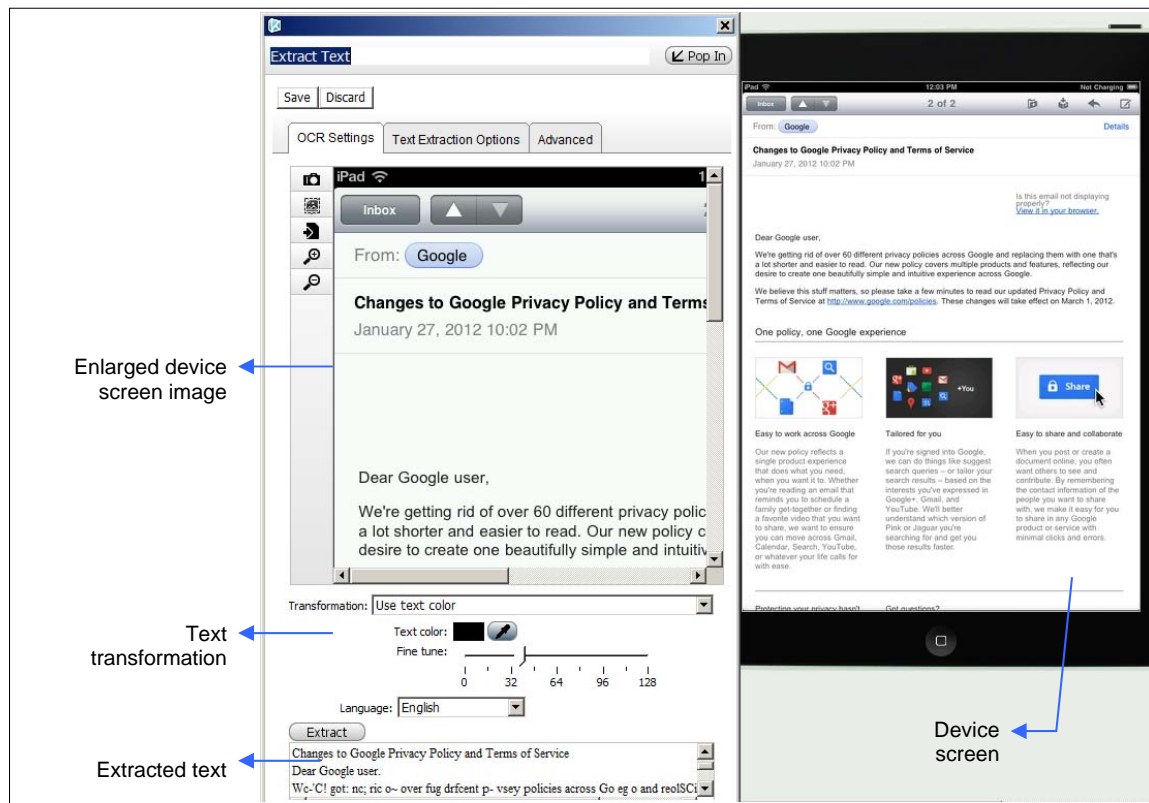
The Extract Text command offers great control in defining the text string to be stored.

- ◆ You can define the text string by specifying the terms that come before and/or after it.
- ◆ You can define the text string by the number of characters to extract after a given occurrence of a term.
- ◆ You can define the text string using a regular expression.
- ◆ You can also opt to pass all extracted text to a selected variable.

Figure 5-28 below shows the Extract Text command next to the device screen from which text is extracted. The default **OCR Settings** tab of the command allows you to view and adjust text extracted. If you navigate to a different device screen, click the camera icon  to update the command.

You can transform text by selecting text color or background, changing the screen image to black and white, or adjusting image contrast. All these operations are explained in detail in [Text Transformations](#).

Figure 5-28 Extract Text Command—OCR Settings



Next, in the **Text Extraction Options** tab (see Figure 5-29 below), click **Extract Text** and define a text string to be stored and the variable/parameter in which to store it.

To choose a variable/parameter to store the text string, click **Insert Variable** next to **Save processed text into variable**. In the Select Variable dialog box that appears, you can select a parameter, a script variable, or a global variable for storing your text string.

NOTES Be sure to select a variable or parameter of the appropriate type for the data you are extracting.

If you have created a variable that is associated with a data set, you can only call it using the [Loop command](#)—see [Working with Data Sets](#).

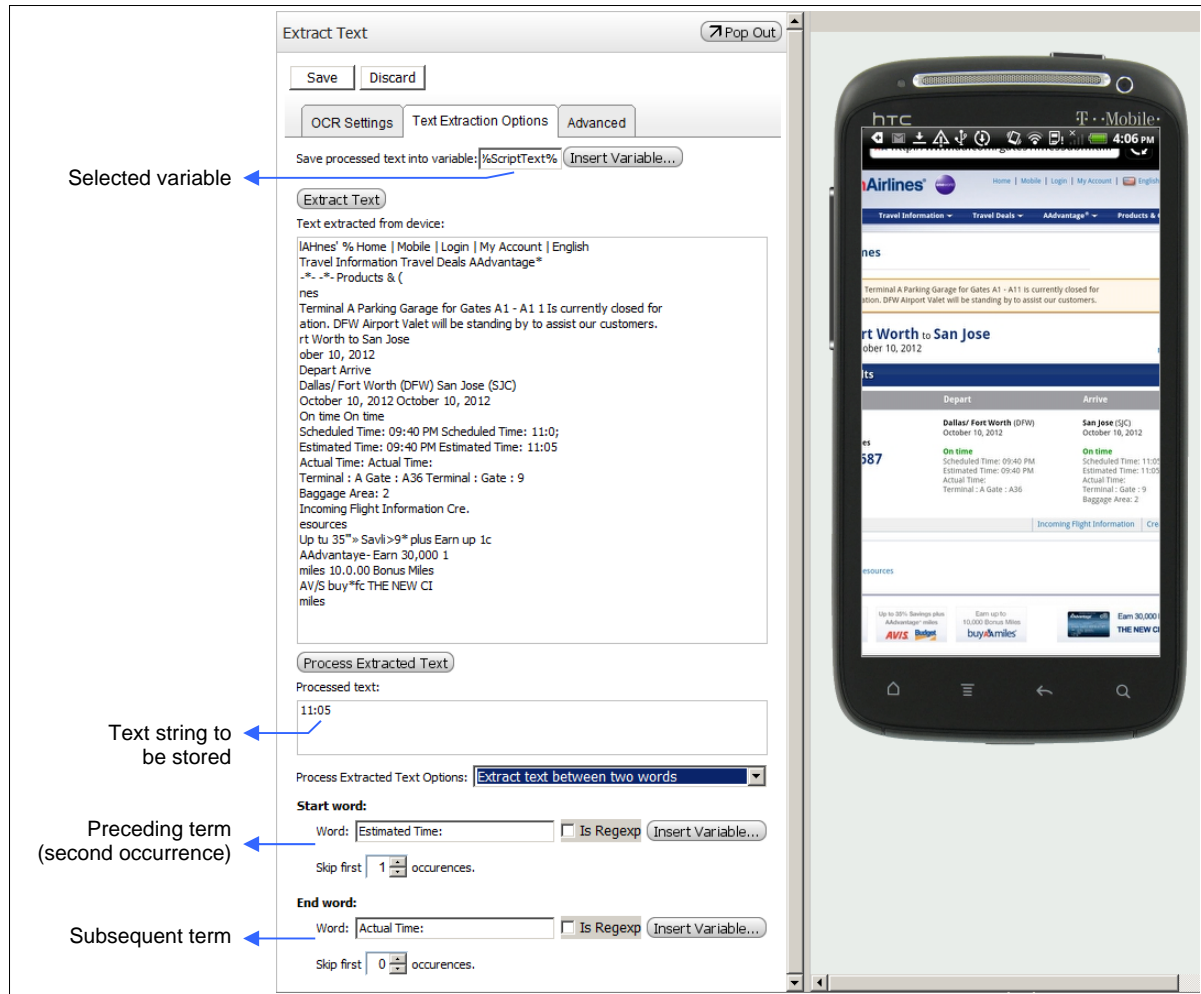
5.5.3.1 Extracting a Text String Between Two Words

You can define a text string to extract by specifying that it comes before a given occurrence of a term and/or after a given occurrence of a term. The term preceding and following your text string can also be contained in a parameter or variable. (In the image below, a text string is defined by specifying the terms that come before *and* after it.)

- 1 Select the **Extract text between two words** from the **Process Extracted Text Options** drop-down list.
- 2 If choosing a text string that comes after a word/phrase (**Start word**):
 - a Enter the **Word** that precedes the text string. Alternatively, you can define your text string as coming after the contents of a variable (click **Insert Variable** to choose the parameter/variable).
 - b If the preceding term (in this example, `Estimated Time :`) is repeated, you can opt to skip a given number of its occurrences from the top of the screen. Enter the number of occurrences to skip in the field provided.
 - c Optionally, you can use regular expressions (**Is Regex**) to define the preceding term, e.g., if you are not sure what exactly the term will be but know the character pattern it follows.
- 3 If choosing a term that comes after your text string (**End word**):
 - a Enter the **Word** that follows the text string. Alternatively, you can define your text string as coming before the contents of a variable (click **Insert Variable** to choose the parameter/variable).
 - b If the subsequent term (in this example, `Actual Time :`) is repeated, you can opt to skip a given number of its occurrences from the top of the screen or from the preceding term. Enter the number of occurrences to skip.
 - c Optionally, you can use regular expressions (**Is Regex**) to define the subsequent term, e.g., if you are not sure what exactly the term will be but know the character pattern it follows.
- 4 Click **Process Extracted Text** to see the string that will be passed to the variable.

In the image below, a text string is defined by specifying the terms that come before *and* after it.

Figure 5-29 Extract Text—Selecting Text Between Two Words



5.5.3.2 Extracting Characters After a Term

You can define a text string as a certain number of characters that come after a given occurrence of a term. This is useful, for example, in extracting a string that is composed of a fixed number of characters such as a phone number or flight time. The preceding term can also be contained in a parameter or variable.

- 1 Select the **Extract characters after a word** from the **Process Extracted Text Options** drop-down list.
- 2 Enter the number of characters to extract (**Include characters after following word**).
- 3 Enter the **Word** that precedes the text string. Alternatively, you can define your text string as coming after the contents of a variable (click **Insert Variable** to choose the parameter/variable).
 - a If the preceding term (in this example, `Estimated Time:`) is repeated, you can opt to skip a given number of its occurrences from the top of the screen. Enter the number of occurrences to skip in the field provided.
 - b Optionally, you can use regular expressions (**Is Regexp**) to define the preceding term, e.g., if you are not sure what exactly the term will be but know the character pattern it follows.
- 4 Click **Process Extracted Text** to see the string that will be passed to the variable.

In Figure 5-30 below, the text string, a flight time, is defined as 6 characters after the second occurrence of the term `Estimated Time:`. (One occurrence of `Estimated Time:` is skipped. The skipped occurrences are highlighted in red in the image below. The extracted text string is highlighted in green.

Figure 5-30 Extract Text—Selecting Characters After a Word

Extract Text Pop Out

Save Discard

OCR Settings Text Extraction Options Advanced

Save processed text into variable: %ScriptText% Insert Variable...

Extract Text

Text extracted from device:

```

|AHnes' % Home | Mobile | Login | My Account | English
Travel Information Travel Deals AAdvantage*
Terminal A Parking Garage for Gates A1 - A1 1 Is currently closed for
ation. DFW Airport Valet will be standing by to assist our customers.
rt Worth to San Jose
ober 10, 2012
Depart Arrive
Dallas/ Fort Worth (DFW) San Jose (SJC)
October 10, 2012 October 10, 2012
On time On time
Scheduled Time: 09:40 PM Scheduled Time: 11:0;
Estimated Time: 09:40 PM Estimated Time: 11:05
Actual Time: Actual Time:
Terminal : A Gate : A36 Terminal : Gate : 9
Baggage Area: 2

```

Process Extracted Text

Processed text:

11:05

Process Extracted Text Options: Extract characters after a word

Include 7 characters after following word:

Word: Estimated Time: Is Regexp Insert Variable...

Skip first 1 occurrences.

5.5.3.3 Other Options to Define Text String

You can use a regular expression to define the entire string of text to be extracted and stored in a variable. This is useful when you are not sure about the contents of the string but know the character pattern it follows, e.g., a 9-digit phone number, or an airline reservation number where the first three characters are letters. To use a regular expression, select **Use regular expression**. To pass all extracted text to the variable without additional processing, select **Leave text as is**.

5.5.4 Working with Data Sets

You can create script or global variables that refer to entire data sets instead of a single value. The data set can either be created/entered in DeviceAnywhere Studio or imported in CSV format from an external source.

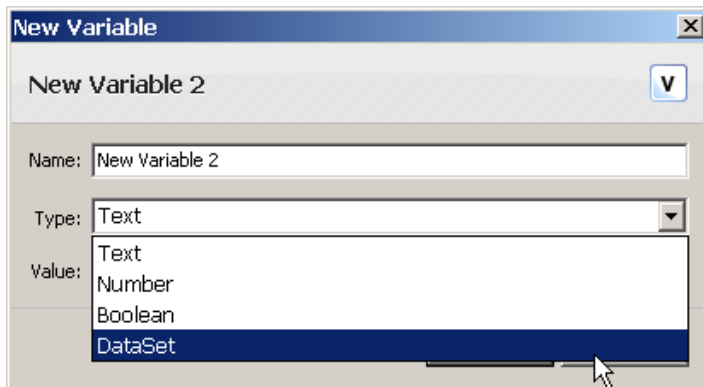
The Loop command can then be associated with the variable and used to loop iteratively through specified values from the data set. The Loop command allows you to define commands within the loop as well as commands when you exit the loop (see also [Script Logic](#)).

This section describes how to [create a variable associated with a data set](#) and then [use the Loop command with the data set](#).

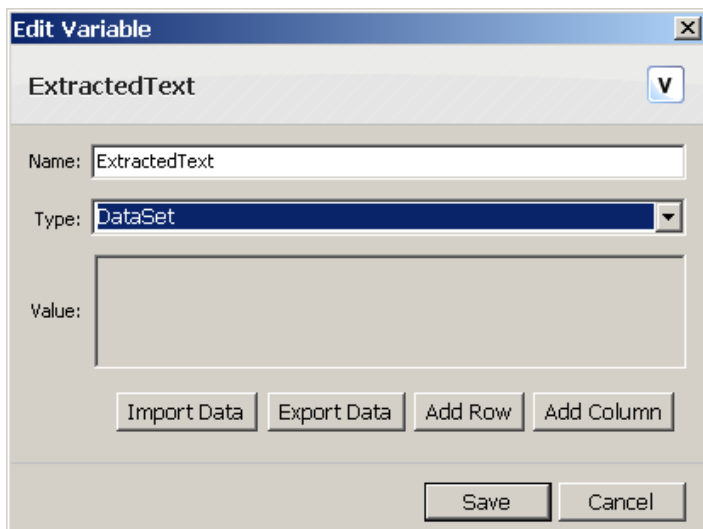
5.5.4.1 Creating a Data Set Variable

To create a [global variable](#) or [script variable](#) associated with a data set:

- 1 Select **DataSet** as the variable **Type**.

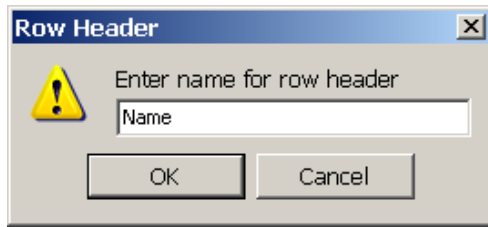


- 2 You are presented with the interface for entering/importing a data set.



- 3 Enter data:
 - To create a data set in DeviceAnywhere Studio:
 - i Click **Add Column** to create a data field.

- ii Enter the name of the field and click **OK**.



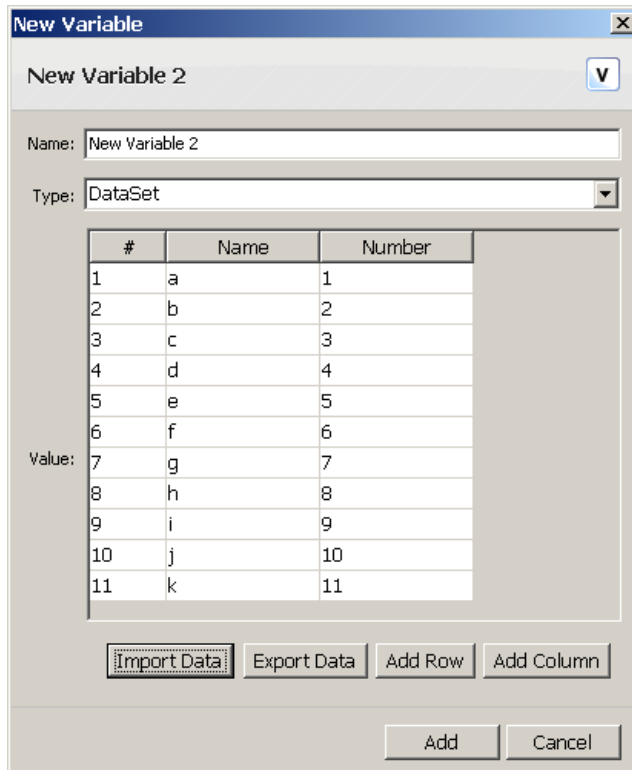
- iii Click **Add Row** to enter a record. A numbered row with placeholders for data is created. (To delete a record, right click a row and select **Delete Row**.)

#	Name	Number
1		

- iv Click in a cell and enter your data.

- To import data, click **Import Data** and select a CSV file from your file system.

Data values are displayed in the DeviceAnywhere Studio dialog box.




NOTE You can right-click a column and avail of controls to rename or delete it.

#	Number	Last Name	
1	23456	Nameabc	Rename Column
2	11234	Surname	Delete Column

You can also reorder your columns—click and hold a column head with your mouse and then drag it to the desired location.

- 4 Optionally, click **Export Data** to export your data set to a CSV file.
- 5 Click **Add** to finish adding the data set.
- 6 **Save** your changes to script variables, project properties, or test case properties.

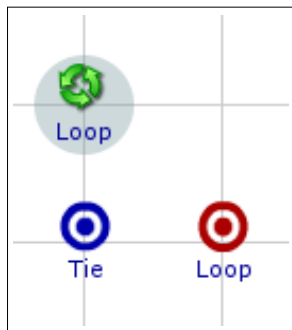
5.5.4.2 Using the Loop Command with a Data Set

The Loop command  allows a script to loop over a set of values stored earlier in a global variable or script (action or test case) variable.

NOTE Using Loop without calling a data set is discussed in [Script Logic](#).

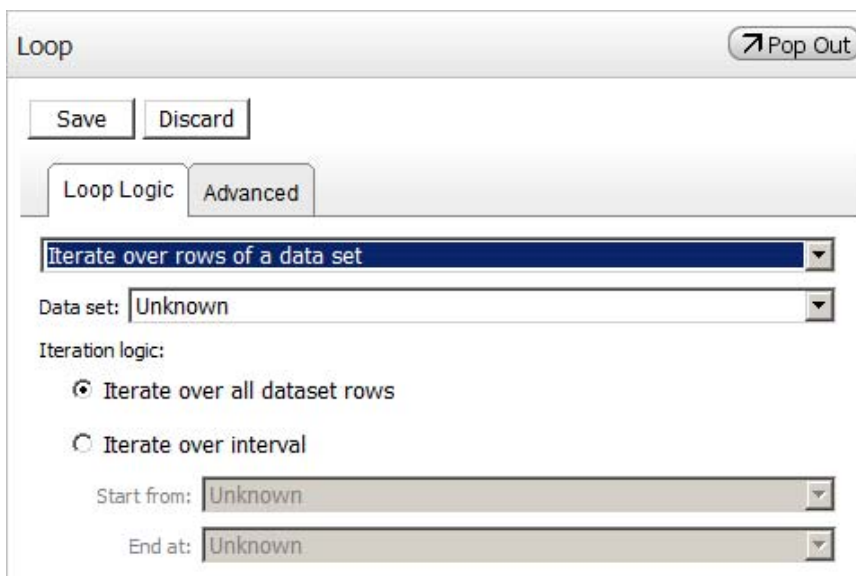
When you drag the Loop command onto the script canvas, it creates placeholders for the Loop and Tie branches. You must first select the variable (data set) you wish to use with the command. You can then drag commands to the Loop placeholder to define the command sequence that will loop iteratively through values in your data set. Use the Tie branch to define a command sequence after the script has exited the loop.

Figure 5-31 Loop Command Placeholders



To choose a variable in the Loop command:

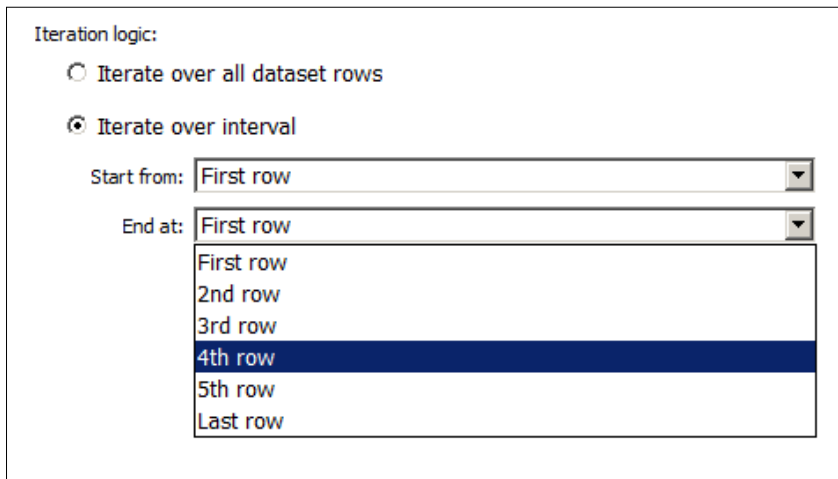
- 1 Drag the command to the script canvas.
- 2 In the **Loop Logic** tab, select **Iterate over rows of a data set**.



- 3 Select a **Data set** from the drop-down list. Project variables are prefixed with the project ID, e.g., Project80.ProjDataSet below.



- 4 Specify the data set records to loop over:
 - You can **Iterate over all data set rows**, or
 - You can loop through a custom range (**Iterate over interval**)—choose the **Start** and **End** record to loop through from the drop-down lists provided.



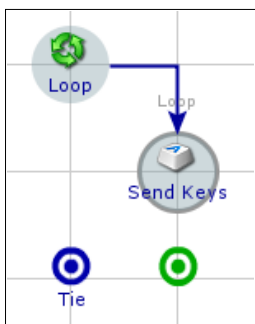
- 5 **Save** the command.

Next, you must drag commands to the Loop placeholder to define a command sequence within the loop. All commands in this sequence might not interact with data set values. However, for those that do, you must specify how they must use the fields in the data set.

For instance, you might want to update your contact list by iteratively looping through the values in your data set. Your command sequence could then consist of a Send Keys command to enter a name, a second Send Keys command to enter the phone number, and so on. At run time, the script loops through the commands to enter the selected fields and records.

To insert data set values in the Send Keys command (you should already have chosen the data set in Loop, and Send Keys must be part of the loop command sequence):

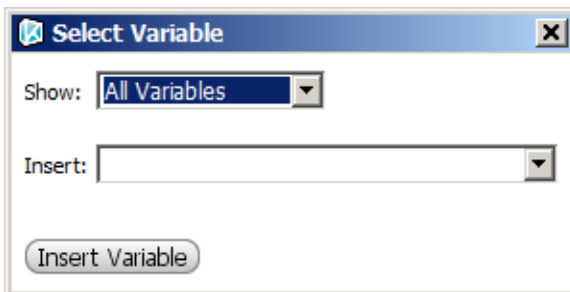
- 1 Drag Send Keys to the Loop placeholder and select to open command properties.



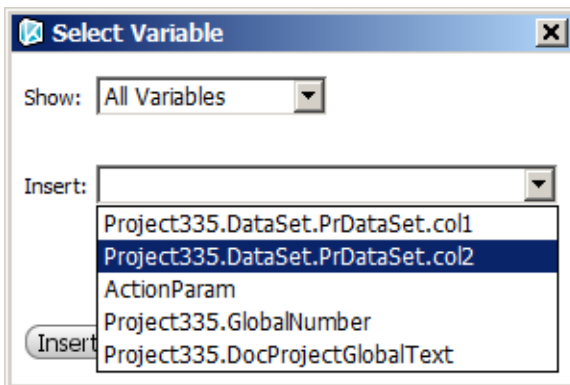
- Click **Insert Variable** next to **Text to Send** field to select the data set field you want the command to interact with.



- Ensure that **All Variables** is selected in the **Show** drop-down list so you can choose from script as well as project variables.

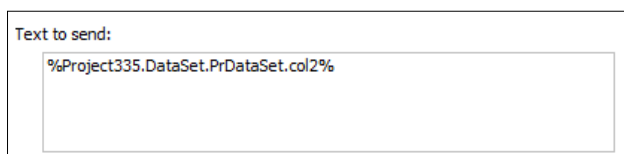


- Select a data set field/column from the **Insert** drop-down list. Data set fields are prefixed with DataSet and the variable name. If the variable is global, then the prefix also contains the project ID. For example, `DataSet.ScriptData.Color` pertains to a column called `Color` in a script variable called `ScriptData`. `Project335.DataSet.PrDataSet.col2` pertains to a column called `col2` in a global variable called `PrDataSet`.



CAUTION Be sure that the values in the chosen column are appropriate for the device field that data is being entered into. For example, if your variable passes a text string into a numeric field on your device, your script will fail.

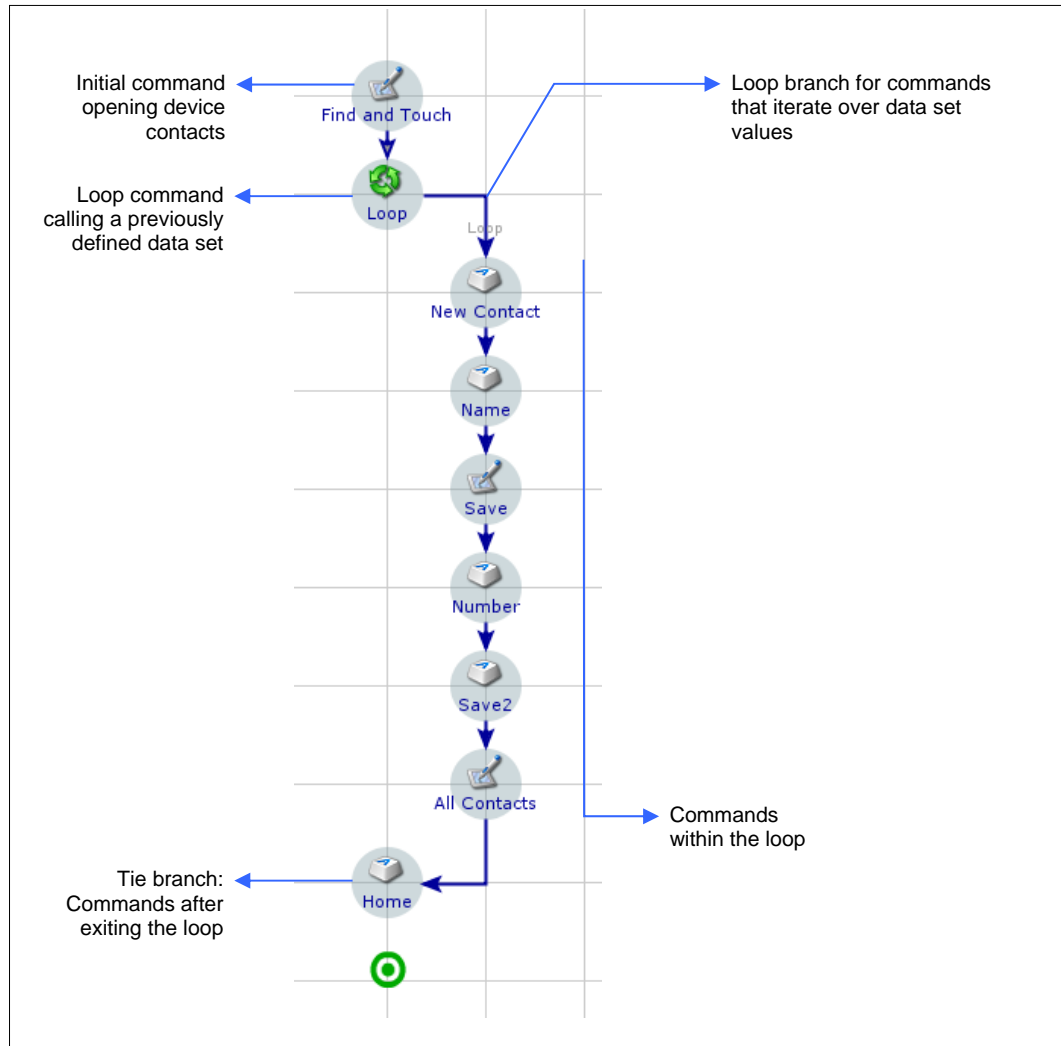
- Click **Insert Variable**. The data set column is displayed (enclosed within percent signs) in **Text to Send**.



3 Save your command.

When done populating your loop, Use the Tie branch to define a command sequence after the script has exited the loop.

Figure 5-32 Looping Over a Data Set in a Script



Refer to [Examples](#) for details about this script.

5.6 Error Definitions

You can create project-wide error definitions and then call them in various commands in your test scripts to generate error messaging when the script fails. The error management system consists of *error categories* containing *error types*.

You can create parent error categories as well as constituent error types in the [Error Types tab](#) of the [Project Properties dialog box](#). The errors you create are valid for the given project. Later when creating test scripts, you can select the error messaging triggered if your script fails at certain selected commands.

NOTE Mobile App Monitoring users cannot create new error categories; they may create [error types](#) in the DAP error category.

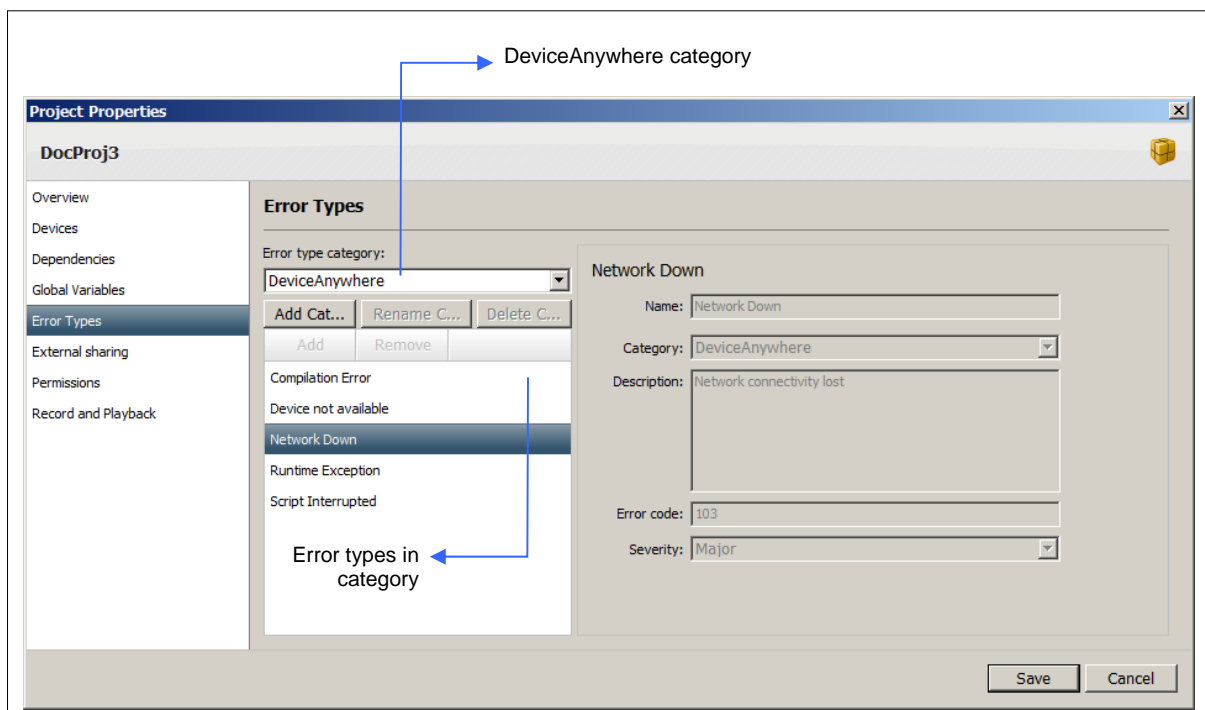
This section explains how to:

- ◆ [Create error categories.](#)
- ◆ [Create constituent error types.](#)
- ◆ [Implement error messaging in commands.](#)

5.6.1 Error Categories

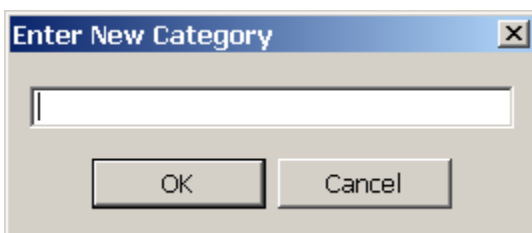
You can create your own custom error category in the **Error Types** tab of project properties.

- 1 Open the Project Properties dialog box (right-click your project in the project list > **Properties**).
- 2 Select the **Error Types** tab. When you first select the **Error Types** tab, it displays the list of error types in the default DeviceAnywhere category.

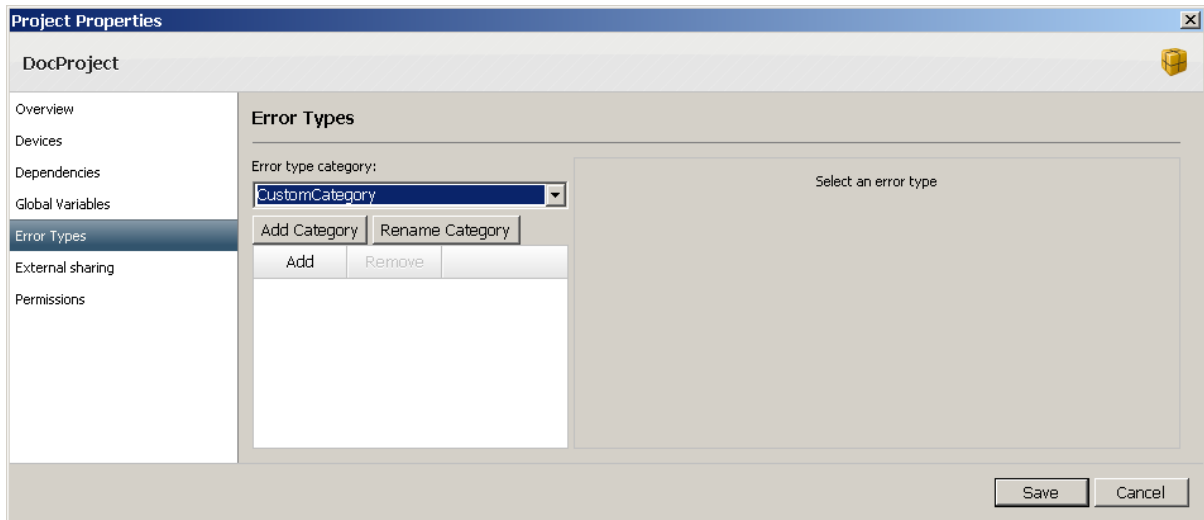


A DAE Automation or DAE Monitoring system comes with the default DeviceAnywhere error category. This category contains system-wide (i.e., available for all projects) error types. You cannot add or remove error types in this category or change their descriptions. This category is reserved for system errors and should not be used in test scripts.

- 3 Click **Add Category**.
- 4 Enter a name for the custom category in the box that appears and click **OK**.

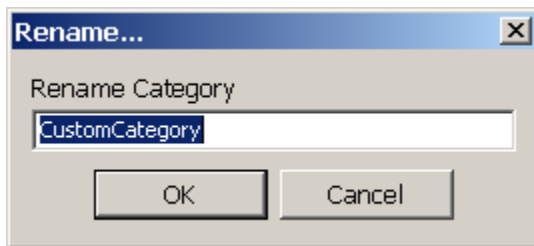


Your new category is listed in the **Error Types** tab of project properties.



- 5 **Save** the error category just created.
- 6 [Populate the error category with error types](#) (reopen project properties if you need to).

NOTES you can rename a custom error category by selecting it and clicking **Rename Category**. Edit the name in the dialog box that appears.



[The Fail command](#) provides a link to edit error categories in project properties.

5.6.2 Error Types

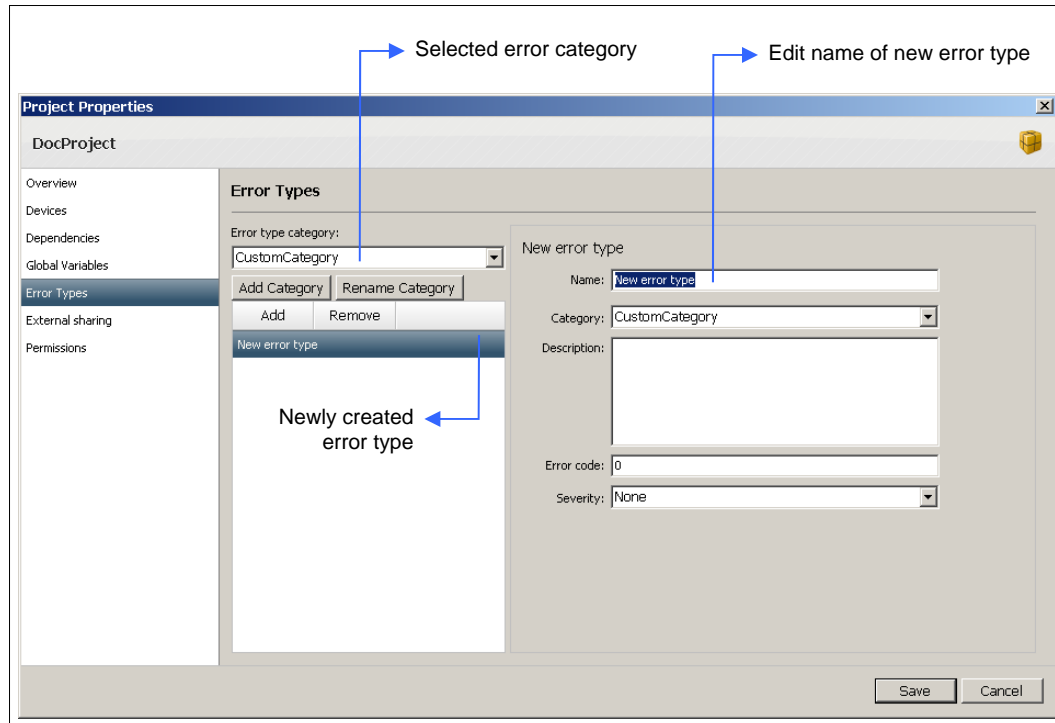
You can add, edit, and remove custom error types and default failure messages in the **Error Types** tab of project properties.

NOTE The **Timeout** tab in commands provides a link to edit error types in project properties.

To add an error type:

- 1 Select the **Error Types** tab in the [Project Properties](#) dialog box.
- 2 From the drop-down list, select the error category in which you would like to create an error type.
- 3 Click **Add**.

This creates a new error type in the selected error category.



- 4 Edit the default **Name** of the new error type.
- 5 Enter an error **Description**. You can add notes to this description when calling the error in commands.
- 6 If necessary, assign a numeric **Error code** to the error type.
- 7 Select an appropriate **Severity** level from the drop-down list, e.g., **Critical**, **Major**, **Minor**, **Normal**.
- 8 **Save** changes to project properties.

5.6.3 Implementing Error Messaging in Your Test Script

When you have defined error types (with descriptions, severity levels, and optionally, error codes), you can implement them by selecting the commands in which they are triggered in your test script. You can select the errors triggered and append their descriptions in the **Timeout** tab, which allows you to define script action in case of failure. This tab is available in the following commands:

- ◆ [Find and Touch](#)
- ◆ [Wait Event](#)
- ◆ [Web Element](#)
- ◆ [Web Wait](#)
- ◆ [Web Form](#)
- ◆ [Web Touch](#)

In addition, you can also select an error in the **Error** tab of the [Fail command](#).

The error messaging selected is displayed in script results in case of failure of these commands or when the Fail command is encountered in your script.

Using Wait Event as an example, the following procedure describes how to set up error messaging for script results:

- 1 Insert a command in your script.
- 2 Select the command's **Timeout** tab.

- 3 Enter a **Wait Time** (in seconds) within which the command must be completed, e.g., the time within which the command finds and matches [reference text](#) or a [reference image](#).
- 4 Select script action if the command times out – **Continue with script** to move on to the next command, or **Return failure immediately** to terminate the script with a “fail” result.


NOTE You can only select error messaging if you fail the script.

- 5 Select an error **Main Category** from the drop-down list. Leave the selection at **Select a category** if you do not want to trigger an error.
- 6 Select an **Error type**. The error type's predefined **Description**, **Severity**, and **Error Code** and are displayed.

NOTE You can choose an existing error type or select **Edit Error Types** to define a new error type in the [Error Types tab](#) of [project properties](#).

- 7 Optionally, Add **Notes** to the error description.

NOTE You can also add a **Comment** in the **Advanced** tab.

You can click the puzzle piece icon  to insert the contents of a variable, e.g., text extracted from the device. Choose your variable/parameter from the Select Variable dialog box that appears. The name of the variable/parameter appears enclosed within percent (%) signs in the field.

5.7 Web Elements in Commands







The new Web command category contains scripting commands that enable you to interact directly with elements in a web page, web application, or hybrid application. DeviceAnywhere Web commands support both native and other Web browsers. Special web utilities open or close the native browser from anywhere on the device (see [Requirements](#) below).

The advantages of working with Web commands and utilities are:

- ◆ The ability to create unpartitioned scripts that operate across supported device models and OS versions
- ◆ The ability to set up interactions with all elements on a Web page, even those not visible on screen
- ◆ The ability to perform form entry, searches, and other Web page interactions quickly, using a single command
- ◆ The ability to open and close browser sessions from anywhere on the device
- ◆ The ability to base verifications on Web elements, creating robust scripts that work across changes to the application UI (e.g., text size or color changes)

The table below describes the commands:

Table 5-3 Web Commands

Command	Description
 Web Element	Selects an element from a web page and performs an appropriate action. Supported actions include extracting a value to a variable and setting a value (e.g., in a text box).
 Web Touch	Finds and touches/clicks a web element, e.g., a button or hyperlink.
 Web Form	Selects/fills values in selected fields and submits a web form.
 Web Wait	Waits to find a web element, such as a form field, to come into focus or blur, or for an element to become visible or hidden based on previous actions.
 Browser Open	Opens the native device browser and navigates to a specified URL from anywhere on the device. Also includes options to first close any existing browser sessions and clear cache before opening a new browser instance.
 Close All Browser Sessions	Closes all open native browser sessions (with the option to clear browser cache) from anywhere on the device.

This section describes the [requirements for using Web commands](#), how to [search for an element](#) to interact with in a Web command, and how to [specify an action for an element](#). The simple operations to open and close browser sessions are discussed in the [Command Reference](#). The command reference also contains a field-by-field description of each Web command.

5.7.1 Requirements

Using Web commands requires the following:

- ◆ Dedicated customer environments require the DeviceAnywhere Central Server to communicate between DeviceAnywhere Studio and the device. For hosted environments, the address will be provided by your Keynote Solutions Consultant. The Central Server retrieves the web page DOM from the device and sends commands to operate on web elements.

NOTE The Access Server must be set up with the address of the Central Server—refer to the *DeviceAnywhere Private System Installation Guide* or contact your Solutions Consultant for details.

- ◆ Addition of a JavaScript tag to the test web page/application

Manually insert the following JavaScript tag just before the closing `</body>` tag of the Web page to be tested:

```
<script type='text/javascript'
src='http://webtest.deviceanywhere.com/da/js/client.js'></script>
```

In dedicated environments, replace `webtest.deviceanywhere.com` with the address of your own Central server, for example:

```
<script type='text/javascript'
src='http://yourCentralServer.yourDomain.com/da/js/client.js'></script>
```

◆ Browsers with support for the following HTML5 features:

- Cross document messaging (`window.postMessage`)
- Canvas tag (canvas element with 2D tag support)
- Cookies must be enabled.

Because browser features can vary on devices by different manufacturers, you can check compatibility by visiting `http://webtest.deviceanywhere.com/da/views/compat.html` from each device browser you wish to use. In dedicated environments, replace `webtest.deviceanywhere.com` in this URL with the address of your own Central Server, e.g., `http://yourCentralServer.yourDomain.com/da/views/compat.html`.

◆ Devices with supported browsers:

- iOS 4.x devices
- Android 2.3 and higher devices (Close All Browser Sessions requires 4.0.x or higher)
- BlackBerry 6/7 devices

◆ Requirements for using web utilities—Open Browser and Close All Browser Sessions:

- The DeviceAnywhere Agent must be installed on devices.


Minimum supported Agent versions are:


- Android: Agent version 1.11.159
- iOS: Agent version 3.6r4

Agents are not supported for web utilities on BlackBerry devices. Users must create visual (UI-based) scripts to open and close browsers on BlackBerry devices.

◆ Registration of device MCD with the Central Server at `http://webtest.deviceanywhere.com/da/views/mcd.html`

This is automatically accomplished when you navigate to this page using the Open Browser command. You must do this from each device browser you plan to use. In dedicated environments, replace `webtest.deviceanywhere.com` in this URL with the address of your own Central Server.

◆ Additionally, we recommend that you clear the browser cache before running scripts with Web commands. You can do this at the start of a script by selecting **Clear browser cache** in the Open Browser  command—this clears the cache before the browser navigates to the specified URL.

Similarly, you can also use the **Clear browser cache** option of the Close All Browser Sessions  command at the end of a script.

IMPORTANT The Open Browser and Close All Browser Sessions utilities cannot be used on BlackBerry devices—you must clear browser cache manually or by creating an automation script.

When you clear the browser cache on BlackBerry devices, you must also reset the device MCD with the Central Server (from each browser for which you have cleared the cache).

5.7.2 Searching for Elements

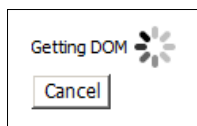
In order to interact with an element using a Web command, you must first select it from elements in the page being tested. You must select an appropriate element in the following Web commands:

- ◆ Web Element—you must select an element and then select the action you wish to perform on it.
- ◆ Web Touch—you must select an element to be touched. Only clickable elements are presented in selection filters.
- ◆ Web Form—you must select the form, i.e., the `<form>` element, you wish to work with (before selecting actions for form fields). Only forms are presented in selection filters.
- ◆ Web Wait—you must select the element you wish to use to verify a sequence of web page or web application interactions.

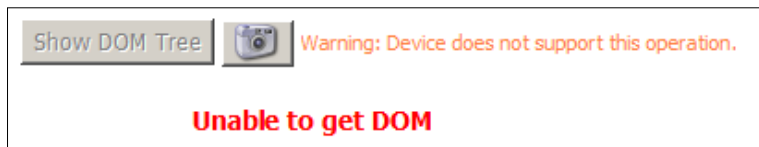
This section describes the filters available in Web commands to search for and select the element you wish to work with.

Before working with Web elements, you must:

- 1 Acquire your device and using a browser, navigate to the web page/application screen you wish to test. Web commands can only act on web elements.
- 2 Ensure that the device MCD is registered with the Central Server (see [Requirements](#) above).
- 3 Drag the command (e.g., Web Element) onto the script canvas and select to open up the command properties pane. A message indicates that the page markup is being parsed into a DOM. This enables the system to discover all the elements on the page.



If web testing is not supported on the device, you will see the following message:



NOTE If you click **Cancel** before the page is parsed, command properties are not displayed. You will

not be able to search for an element

A small rectangular box containing the text "Unable to get DOM" in bold, red font.

When the page is parsed, the message window closes and command properties are displayed.

The figure below shows the Web Element command when properties are first displayed after dragging it onto the script canvas. No element is selected for interaction. If the page's markup has not been parsed, you will not be able to see element properties.

Figure 5-33 Web Element Command Properties

5.7.2.1 Search Criteria and Values

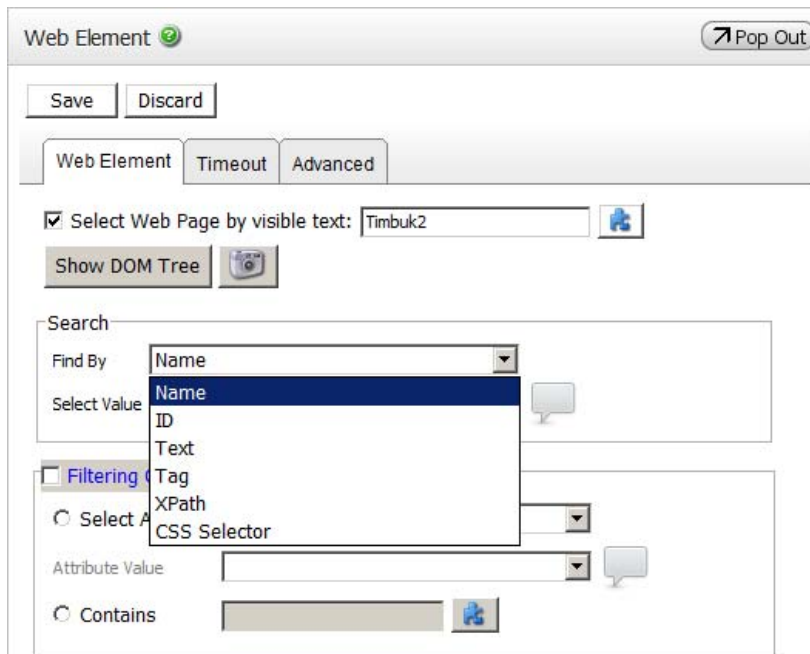
In order to locate an element, you must first let the parser know which of your open pages/sessions to parse:

- 1 Check **Select web page by visible text**.
- 2 Enter a unique text string to identify the page. Click the puzzle piece icon  pass in the value contained in a [variable or parameter](#).

If you have only one session open, you do not need to use this control.

Next, to locate an element, you must first specify a search criterion.

Figure 5-34 Selecting Search Criteria



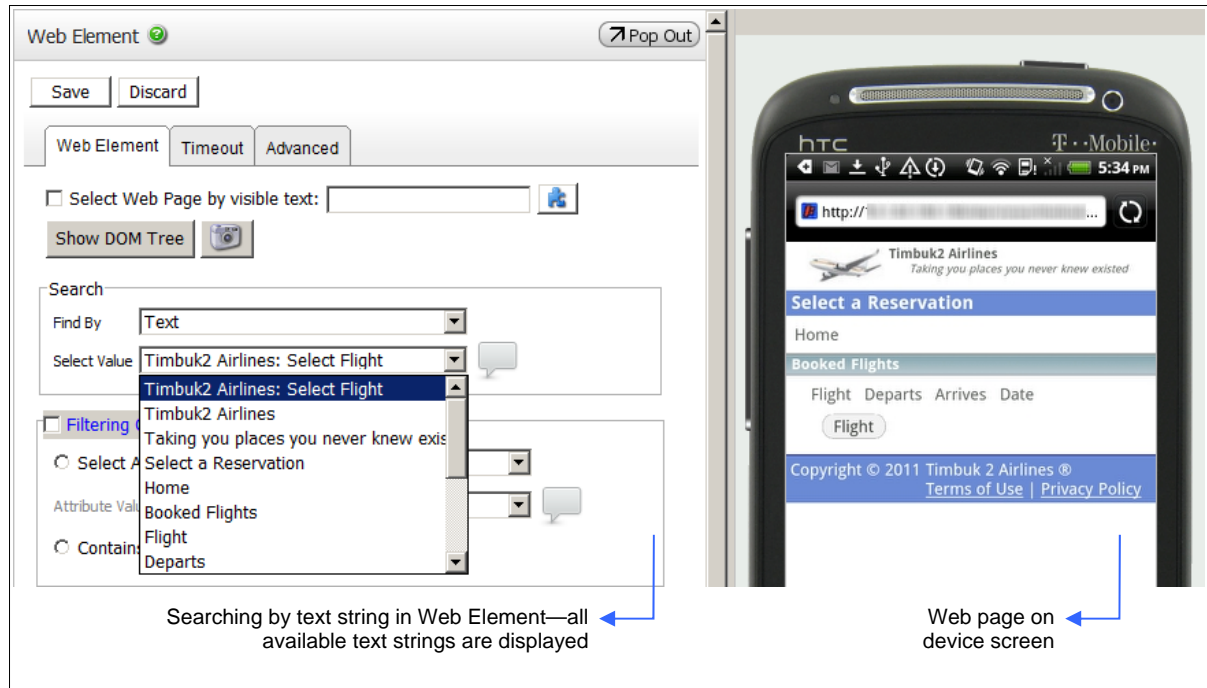
Search criteria available in the **Find By** drop-down list are:

- ◆ **Name**—the name attribute of an element
- ◆ **ID**—the id attribute of an element
- ◆ **Text**—the text between opening and closing tags of an element
- ◆ **Tag**—the tag applied to an element
- ◆ **XPath**—the path expression identifying the element in the DOM
- ◆ **CSS Selector**—the CSS Selector location of the element

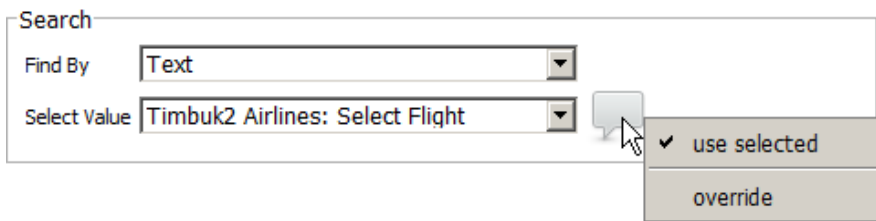
Next, select the criterion value to look for—choose an item from the **Select Value** drop-down list.


- ◆ If you choose to search by **Name**, **ID**, **Text**, or **Tag**, available values for the chosen criterion are listed. In some commands, the listed values are pre-filtered. For example:
 - If you choose to search by **Name** in the Web Element command, all values of the name attribute available in the page are listed.
 - If you choose to search by **Name** in the Web Form command, only values of the name attribute available for <form> elements are listed.
 - If you choose to search by **Name** in the Web Touch command, only values of the name attribute available for clickable elements are listed.

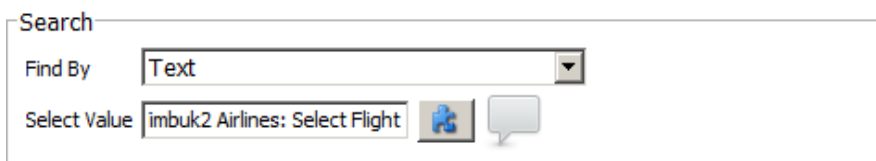
The image below shows all available values for a search by **Text** string in the Web Element command. The corresponding device screen is also shown.



If you are searching by **Name**, **ID**, **Text**, or **Tag**, you can use a value from the drop-down list provided or override it by clicking the icon to the right of the field. You would do this, for instance, to specify an element that is not currently displayed but appears on the page at run time.

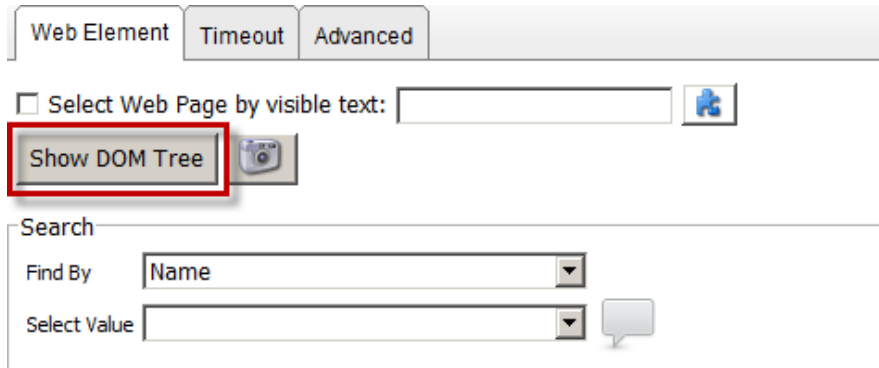


Click **override** and write in a value or click the puzzle piece icon  to pass in the value(s) contained in a variable containing a single value or an entire data set. For example, you could use a variable to loop through data set values to find elements.

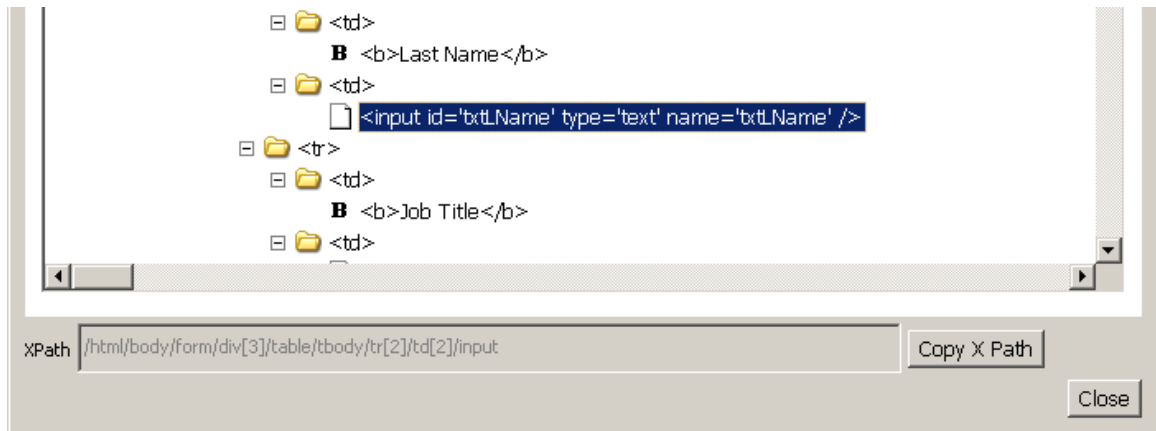


- ◆ If searching by **XPath**, you can copy the path expression from the DOM, available for viewing within the command:

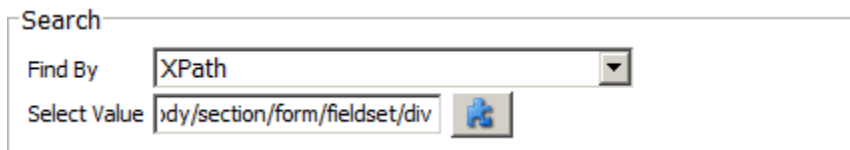
- a Click **Show DOM Tree** to view Web elements in a DOM (see [Search Aids—DOM Tree and Inspect Element](#) below for more information).




- b Select an element by selecting a node in the DOM. The corresponding path expression is displayed in the **XPath** field.



- c Click **Copy X Path**. The path expression is automatically copied to the **Select Value** field.



You can use regular expressions to allow variation in the path expression (e.g., login credentials). You can also pass in the value for a part of or the entire path expression from a variable—click the puzzle piece icon  next to **Select Value**. You could use a variable to loop through multiple path expressions contained in a data set.

- ◆ If searching by **CSS Selector**, you must type in the value in the field provided. You can also click the puzzle piece icon to pass in the value from a variable. Click **Evaluate** to view [search results](#). No results are displayed if the search string is incorrect.



NOTES If you cannot find the desired element, try other search criteria.

Name, **ID**, **Text**, and **Tag** are recommended over **XPath** or **CSS Selector** as search criteria. If an element's location in the markup is changed, its XPath and CSS selector expressions can change. Even if markup remains unaltered, XPath location can change from one device to another or when you change orientation on the same device.


As you specify search criterion and value, matching elements are listed in the **Results** pane. You can apply [additional search criteria](#) and filter results by index.

5.7.2.2 Additional Search Criteria and Results

If your initial [search criteria](#) produce multiple [results](#), you can apply additional search criteria by checking the **Filtering Options** box. You can filter by an attribute or by text between the opening and closing tags of elements in search results.

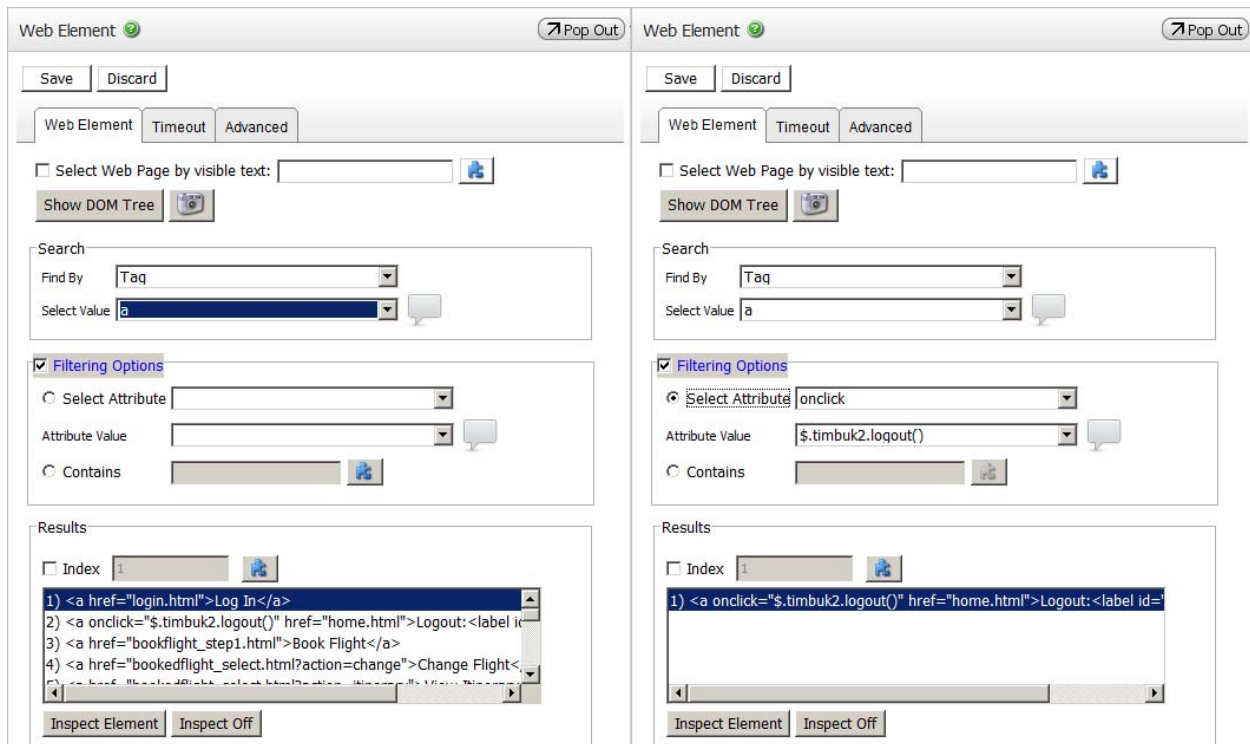
Figure 5-35 Additional Filtering Options

To filter by attribute:


- 1 Click the **Select Attribute** radio button.
- 2 Select an attribute from the drop-down list provided. If the resulting elements have no attributes, the list is empty.
- 3 Select an **Attribute Value** from the drop-down list provided. You can override a listed value by clicking the icon to the right of the field . You would do this, for instance, to specify an element that is not currently displayed but appears on the page at run time.

The image below shows initial search results before filtering by attribute and after additional filtering by attribute.

Figure 5-36 Search Results Before and After Additional Filtering by Attribute



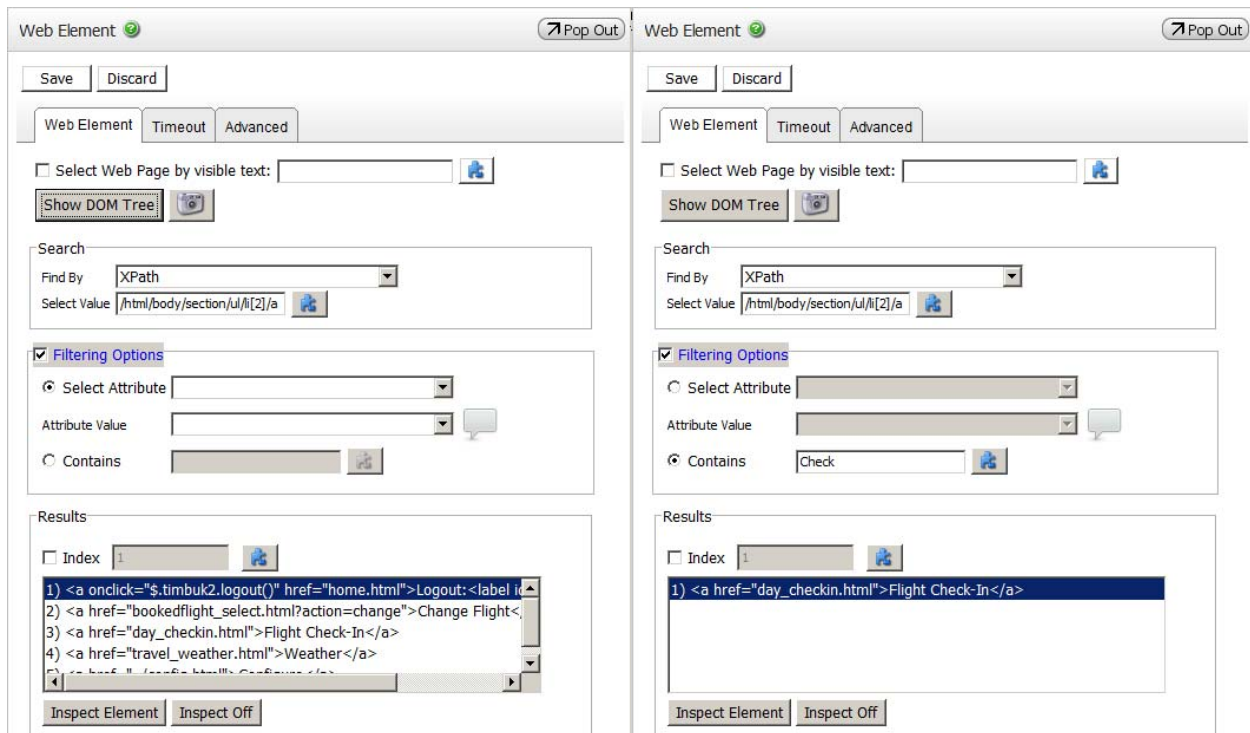
To filter by text string (this option is not available for `<form>` elements in the Web Form command):

- 1 Select the **Contains** radio button.
- 2 Enter a string of text—the string is case sensitive and must correspond to text between the opening and closing tags of elements in initial search results. You can also pass in the value(s) contained in a variable—click the puzzle piece icon . You could use a variable to loop through a data set to perform the same action for multiple elements.

The search results are dynamically filtered as you enter the text string.

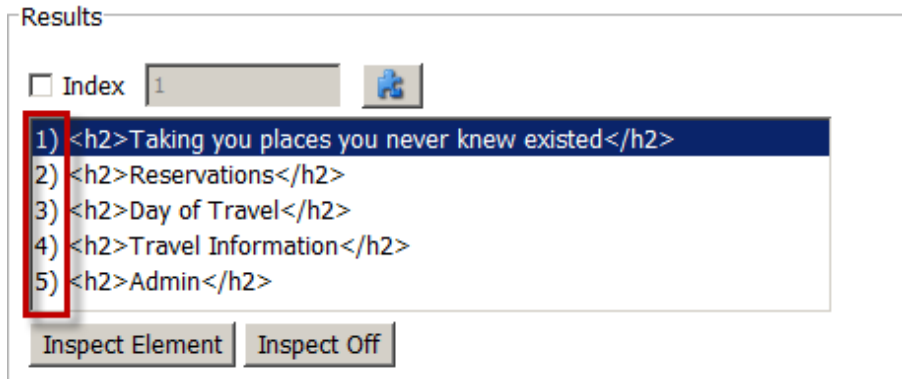
The image below shows initial search results before additional filtering and after additional filtering by element text.

Figure 5-37 Search Results Before and After Additional Filtering by Text



Results can be further filtered by index—the number indicating the order in which they appear. When you specify search criteria, all matching elements are displayed and numbered according to the order in which they appear in the DOM. Index numbers are highlighted in the image below.

Figure 5-38 Indexed Search Results



You can further filter elements by this assigned ordinal number.


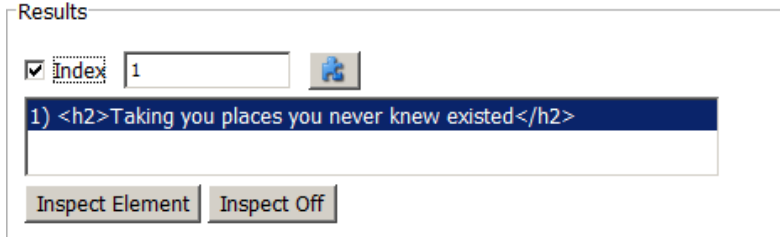
- 1 Check the **Index** box.
- 2 Enter the index number of the element you wish to work with. Click the puzzle piece icon  to pass in the index number(s) from a variable. You would do this, for instance, to loop through a data set to perform the same action for multiple elements. Alternatively, the element you wish to work with might change index number when you log in to the test page with different credentials.

Figure 5-39 Filtering Results by Index Number

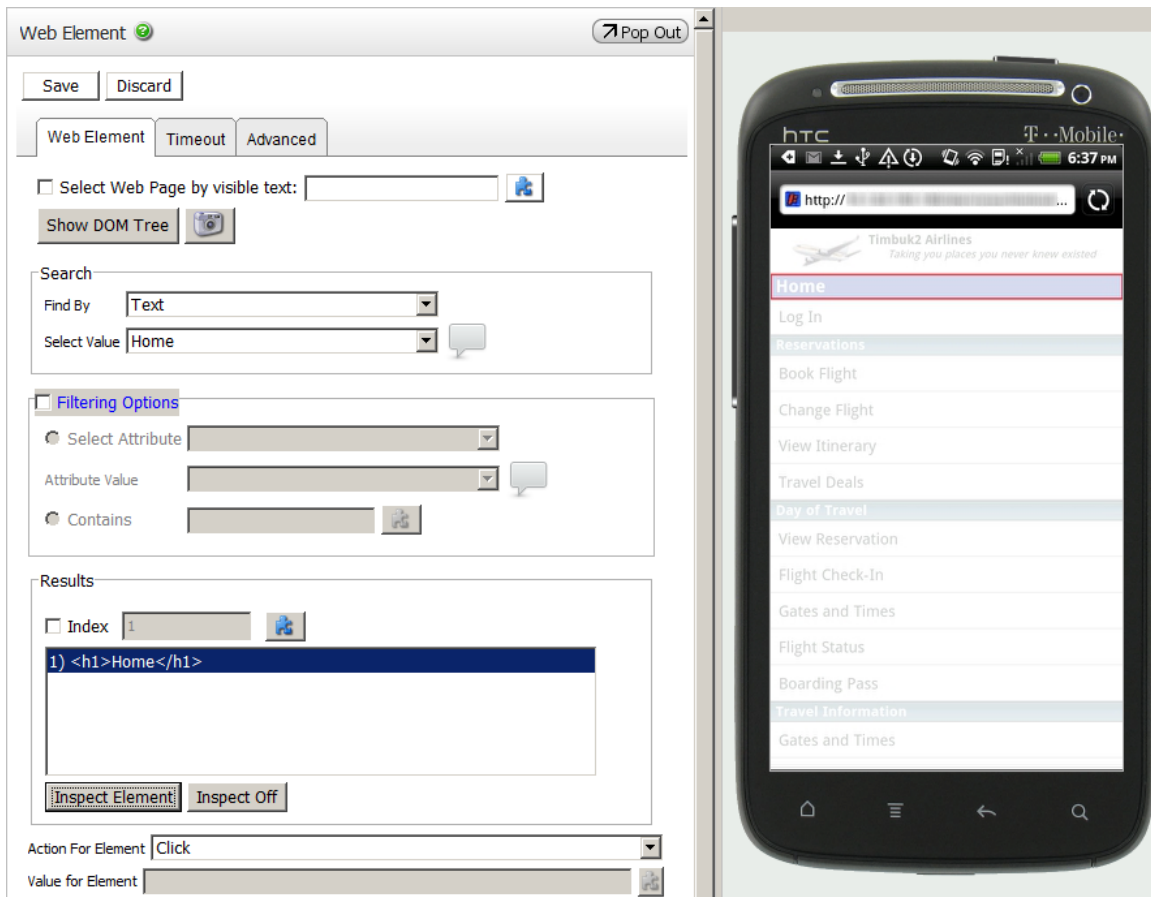


NOTE The index number is not an intrinsic characteristic of an element. The order of an element in search results can change depending on the search criteria you use or the credentials you use to access a web page. An element’s position in search results can also change from one device to another or when you change orientation on the same device. Source code can also be altered to change the relative position of an element in the markup.

5.7.2.3 Search Aids—Inspect Element and DOM Tree

You can turn on inspect mode from Web commands in order to help you select the correct element. To inspect an element:

- 1 Select an element in search results.
- 2 Click **Inspect Element**. The corresponding area of the device screen is highlighted in red. Be sure to turn inspect mode off (**Inspect Off**) when done.



At any point, you can view the structure of your test page in a DOM in order to help with element selection. If searching for an element by **XPath**, you can open the DOM viewer in order to select the correct path expression (see [Search Criteria and Values](#) above).


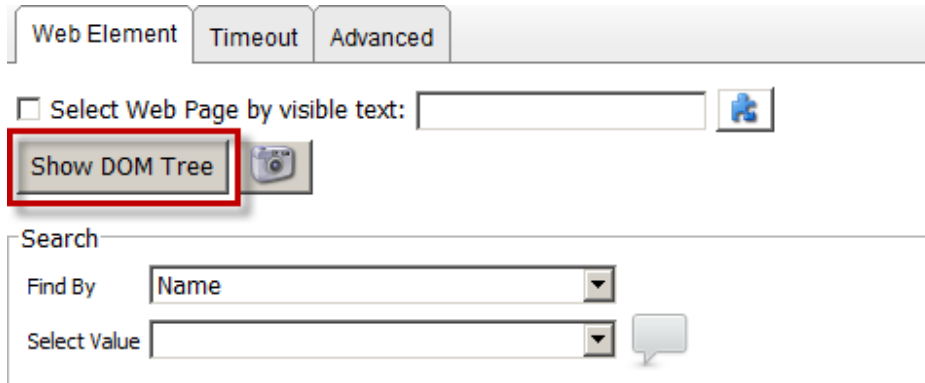
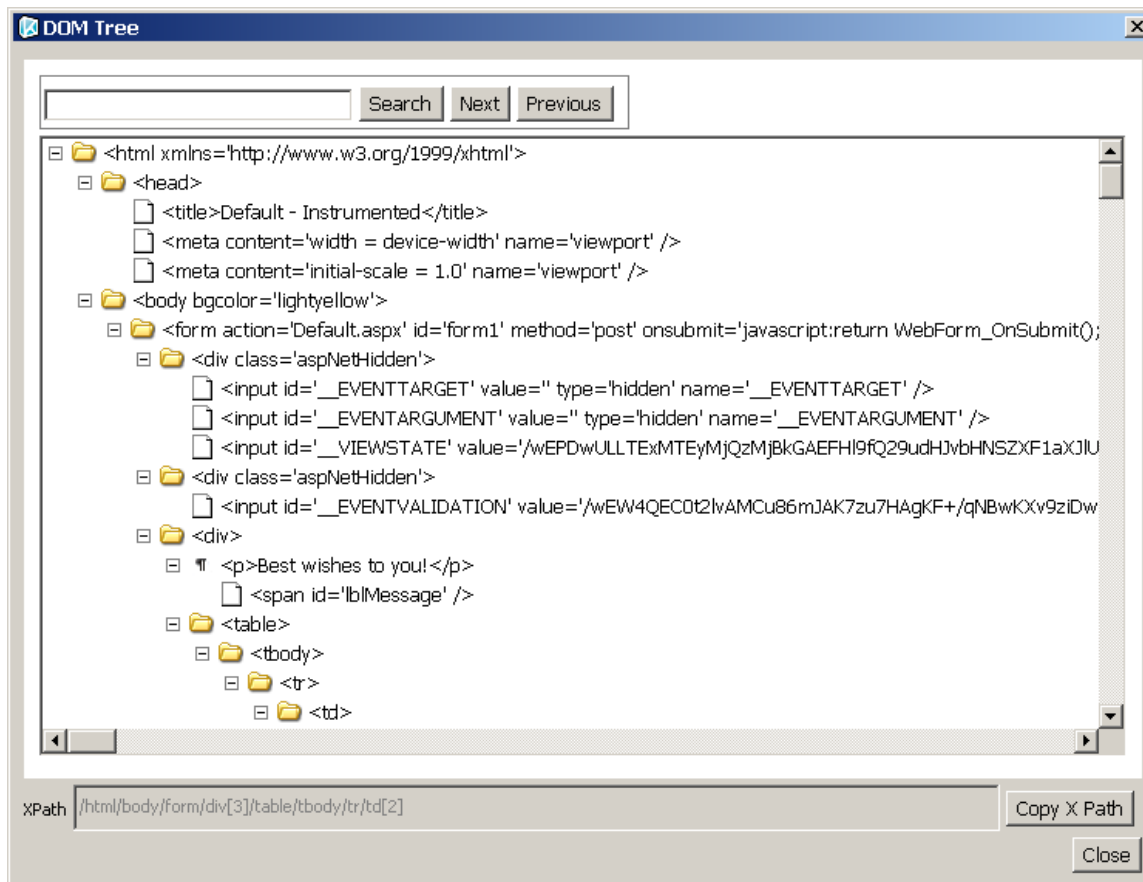
If you have navigated to a different test page on the device after opening the command, click the camera icon  to refresh the DOM *before* viewing it. Click **Show DOM Tree** from a Web command.

Figure 5-40 To View and Refresh the DOM



This opens up the DOM in a separate window (see image below).

Figure 5-41 DOM Viewer



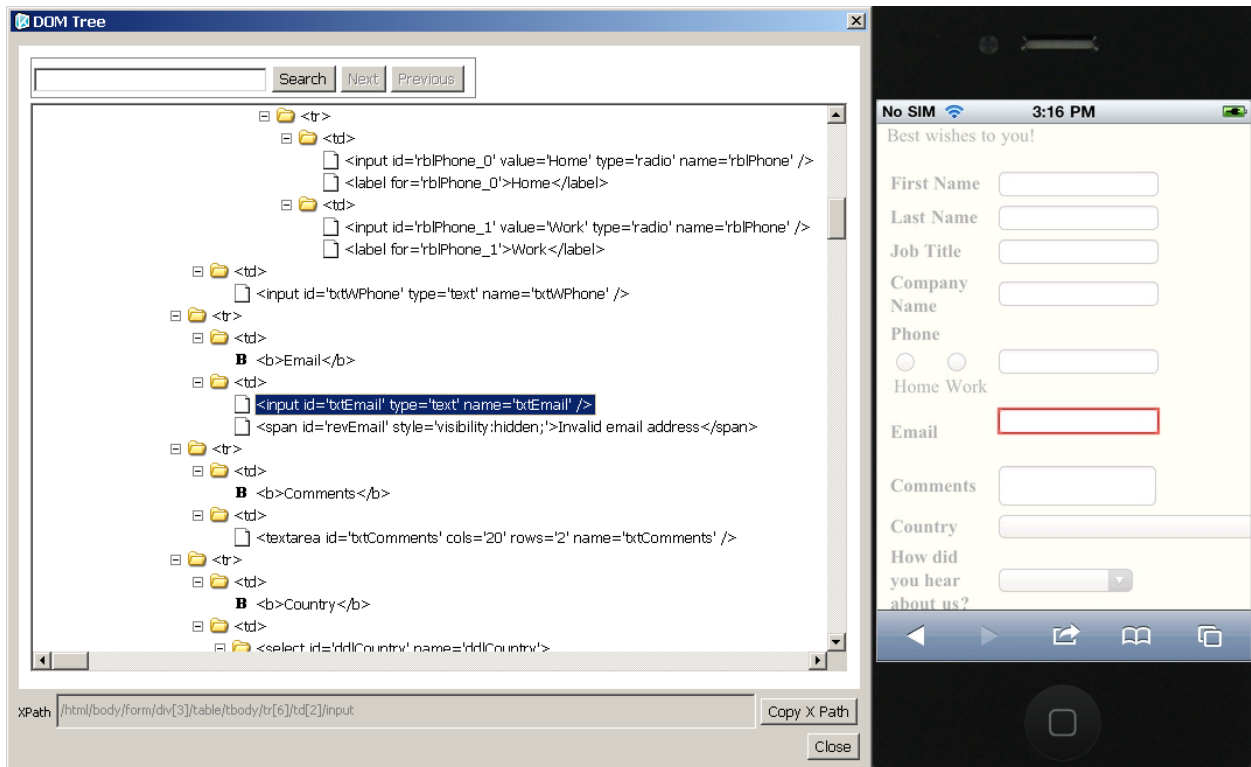
When you opt to view the DOM, inspect mode is automatically enabled on the device, and the entire page is highlighted in red.

Figure 5-42 Inspect Mode Enabled when Viewing the DOM



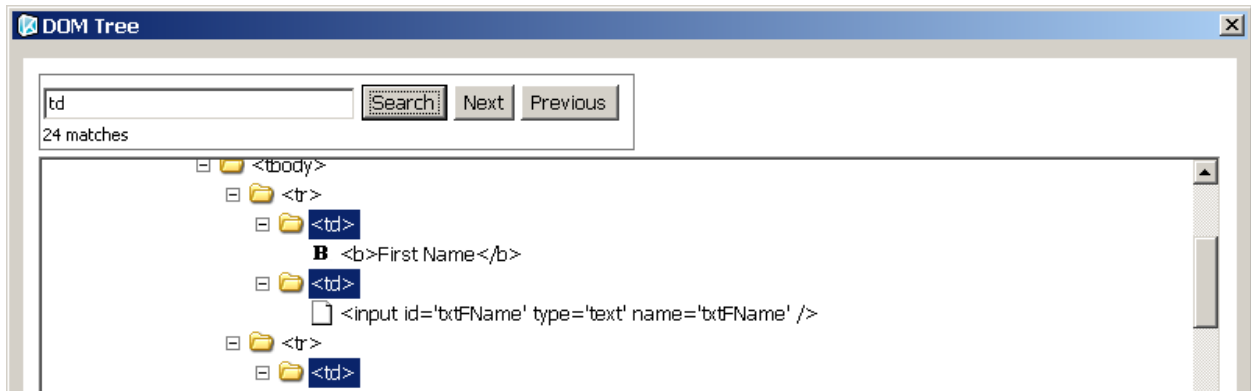
When you click on the device screen, the element is highlighted in red on the device screen, and the corresponding node is selected in the DOM. If you click an element on the device, it is highlighted in red and the corresponding node is selected in the DOM.

Figure 5-43 Using Inspect Mode in the DOM



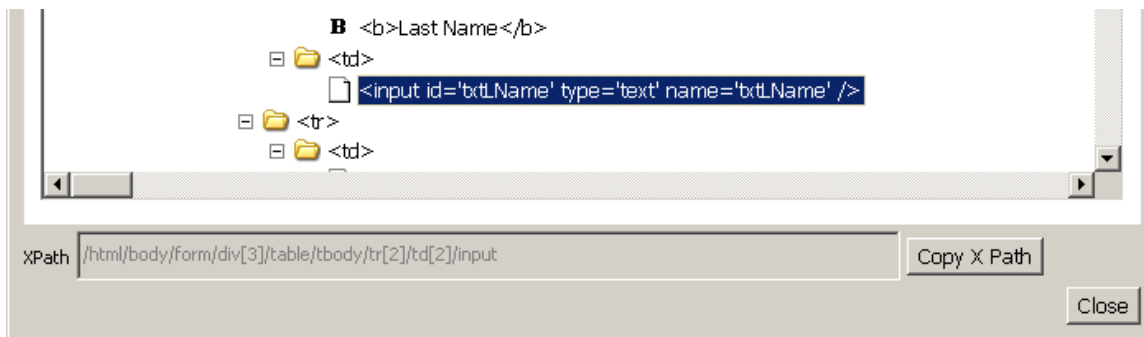
You can search for any string by typing it in and clicking **Search** in the DOM window. Results are highlighted. Use the **Next** and **Previous** buttons to scroll through search results.

Figure 5-44 DOM Tree Search



When you click on a node to select it, the corresponding **XPath** expression is displayed (see [Search Criteria and Values](#) for searching for an element by XPath).




Figure 5-45 Node Path Expression




5.7.3 Choosing an Action for a Selected Element

Once you have narrowed down the element you wish to work with, you must select it in the **Results** pane and select the action, i.e., the interaction that you wish to perform.

NOTE In many cases, the element you wish to work might only appear on the web page at run time and does not appear in search results when defining the command. However, you can still set up your command by choosing the correct search criteria and action to perform.

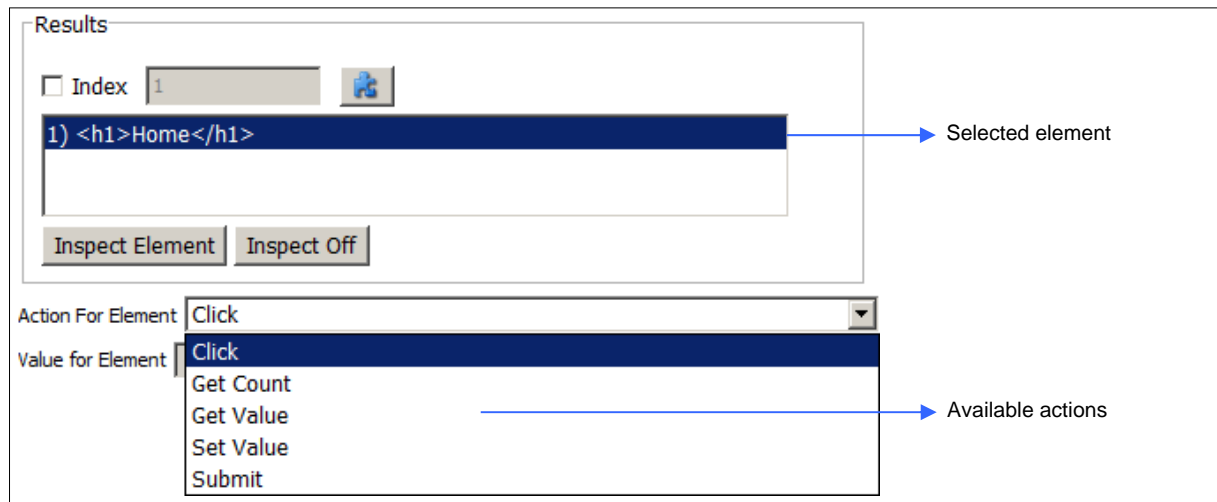
- ◆ You can choose one of several actions in the Web Element  command.
- ◆ In the Web Touch  command, the selected element can only be clicked—no other actions are available for selection.
- ◆ In the Web Form  command, the selected element is a form (<form> element) containing other elements. You can specify a different action for each form field (see [Working with Form Fields](#) below). Next, you can submit the form as a whole (see [Actions for Web Forms](#) below).


- ◆ In the Web Wait  command, the selected element can be used as a web-based [reference point](#) in several ways.

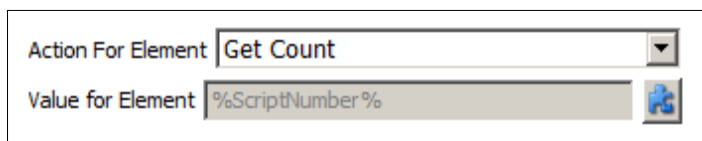
5.7.3.1 Actions in Web Element

To choose an interaction for an element in the Web Element command, select an option from the **Action For Element** drop-down list:

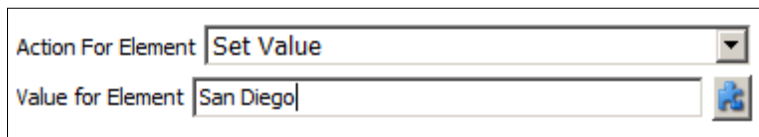
Figure 5-46 Choosing an Action for an Element in Web Element



- ◆ **Click**—clicks the selected element (or element that matches search criteria used) at run time.
- ◆ **Get Count**—counts the number of elements on the page that match your search criteria. This value can be stored in a variable—click the puzzle piece icon  to select a variable. The variable name is displayed in **Value for Element**.



- ◆ **Get Value**—extracts the text between opening and closing tags or the value entered into a field. This value can be stored in a variable. This value can be stored in a variable—click the puzzle piece icon to select a variable. The variable name is displayed in **Value for Element**.
- ◆ **Set Value**—enters a specified element value, e.g., in a field. Specify the element value in **Value for Element**. You can also click the puzzle piece icon to pass in the value from a variable.

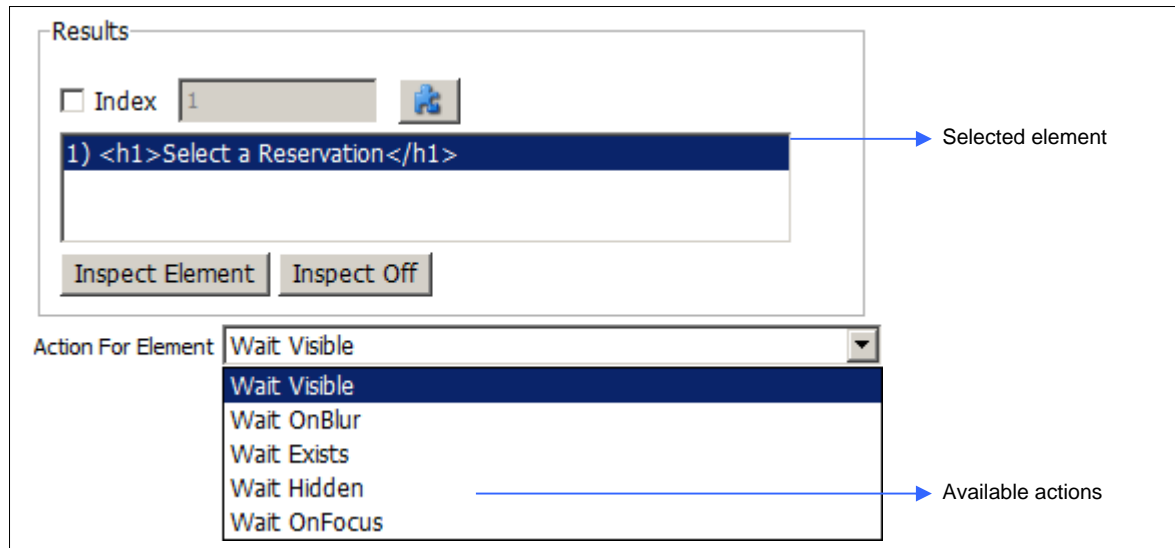


- ◆ **Submit**—clicks a button of type submit, e.g., at the bottom of a form.

5.7.3.2 Actions in Web Wait

The Web Wait command allows you to choose the following wait actions for an element:

Figure 5-47 Choosing a Wait Action for an Element in Web Wait



- ◆ **Wait Visible**—checks that an element is displayed on the page (even if you have to scroll the device screen to view it).
- ◆ **Wait OnBlur**—checks that an element loses focus, e.g., after clicking away from a form field.
- ◆ **Wait Exists**—checks to find an element on the page.
- ◆ **Wait Hidden**—ensures that an element is not displayed on the page (even the portion not visible on the device screen).
- ◆ **Wait OnFocus**—checks that an element comes into focus, e.g., after clicking in a form field.

5.7.3.3 Working with Form Fields

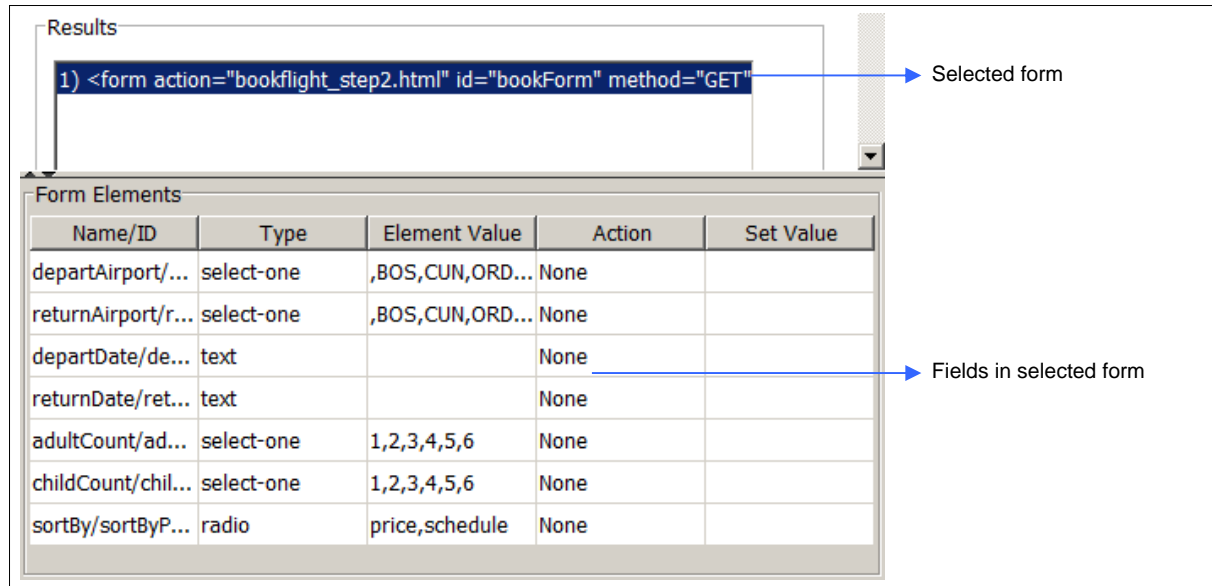
You can interact differently with each field in a form using the Web Form command. The command allows you specify a value or action for each field and thus, fill out an entire form. You can even interact with fields that are not visible, e.g., that you have to scroll the page to see.

When using the Web Form command:

- 1 You must first use available filters to select the form you wish to work with (see [Searching for Elements](#) above).
- 2 Next, you must specify an action for individual fields, which this section describes.
- 3 When you have specified form values, choose an appropriate action (e.g., submission) for the form as a whole—see [Actions for Web Forms](#) below.

When you have selected a form to work with from the **Results** section of the command, all **Form Elements** are listed. Click the bar at the top edge of the pane to resize it.

Figure 5-48 Form Fields in Web Form

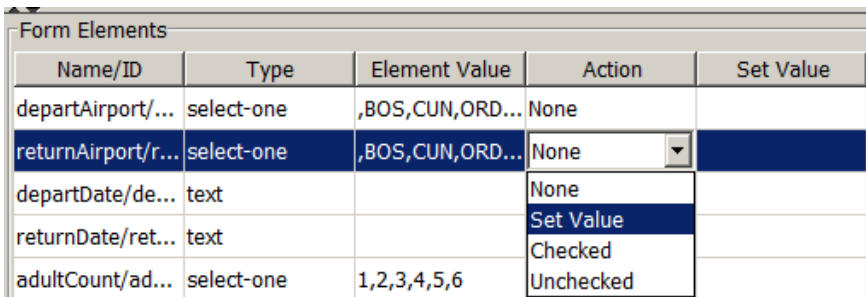


The following information is displayed for each form field:

- ◆ **Name/ID**— values of the corresponding element’s name and id attributes
- ◆ **Type** of field:
 - **text**—text line
 - **radio**—radio button
 - **textarea**—text box
 - **select-one**—drop-down list
 - **checkbox**—check box
- ◆ **Element Value**—the default value or selectable values of the field
- ◆ **Action**—the interaction to perform with the field—set by default to **None**.
- ◆ **Set Value**—the value specified for the field—blank by default.

To work with a field:

- 1 Click to select the field in the **Form Elements** list.
- 2 Click the **Action** column and choose an appropriate action for the field from the drop-down list:



- **None**—no interaction
- **Set Value**—choose to specify a value by selecting a radio button option, a drop-down list option, or by entering a value.
- **Checked**—choose to check a check box.
- **Unchecked**—choose to uncheck a check box.

NOTES Submit buttons are *not* listed with other form elements. To click this button after filling out form fields, choose the **Submit** action for the form as a whole (see [Actions for Web Forms](#) below).

Any other buttons at the bottom of your form are listed with form elements, e.g., the “Next” button at the bottom of the first page of a three-page form. Do not choose any action for this button. Instead, use a Web Touch command after the Web Form command to click the button. This ensures that the button is clicked after all form fields are filled out.

- 3 Enter or select a value in the **Set Value** column. This is only activated when you choose the **Set Value** action for the field.

If the field is a text box or text area, click in the **Set Value** column and enter a value. You can also pass in the value from a variable—select **Use Variable** from the drop-down list.

Enter a value in the **Set Value** column. ←

Form Elements				
Name/ID	Type	Element Value	Action	Set Value
returnDate/ret...	text		Set Value	11/12/12
adultCount/ad...	select-one	1,2,3,4,5,6	Set Value	Use Variable
childCount/chil...	select-one	1,2,3,4,5,6	None	

Alternatively, select **Use Variable** to pass in the value from a variable. ←

If the field is a radio button or drop-down list, click the **Set Value** drop-down list to select a value. You can also enter your own value or select **Use Variable** to pass in the value contained in a variable.

Form Elements				
Name/ID	Type	Element Value	Action	Set Value
rbPhone/rblPhon...	radio	Home,Work	Set Value	
txtWPhone/txtWP...	text		None	Home
txtEmail/txtEmail	text		None	Work
				Use Variable

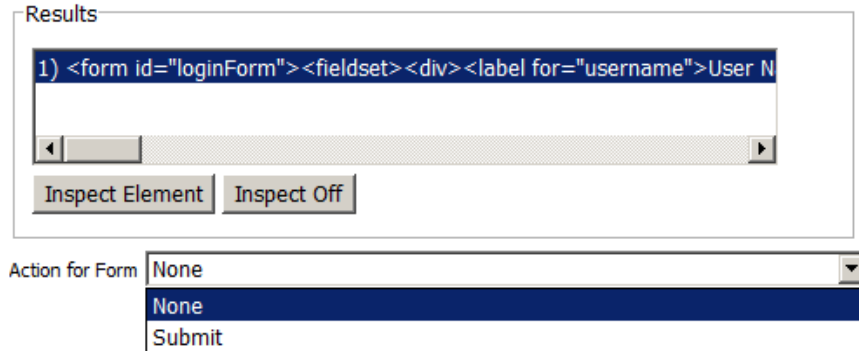
Select a value from available choices, enter your own value, or click **Use Variable** to pass in the value from a variable. ←

Repeat these steps for each field you wish to interact with. If appropriate, specify an action for the form as a whole in the **Action for Form** field—see [Actions for Web Forms](#).

5.7.3.4 Actions for Web Forms

After [working with form fields](#) and choosing interactions for individual elements in a form, you may select the following actions for the form as a whole:

Figure 5-49 Actions for a Form in Web Form



- ◆ **None**—performs no additional action for the form as a whole (see [Working with Form Fields](#) to specify actions for individual form fields).
- ◆ **Submit**—submits the form as a whole, by clicking a button of type submit after individual form fields have been filled out. Select this action only if there is a button of type submit on the test page. You can select this action for the form even if the page has no visible submit button.

NOTES If your form is followed by any other type of button than submit, do not use the **Submit** action for your form. For example, the button at the bottom of the first page of a three-page form does not submit the form; it moves to the next form page. In such a case, use a Web Touch command after Web Form to click the button. This ensures that the button is clicked after all form fields are filled out.

Buttons of type submit are *not* listed with other form elements when you select a form in the Web Form command. To click this button after filling out form fields, choose the **Submit** action for the form as a whole.

5.8 Native Objects in Commands






The new Object command category contains scripting commands that enable you to interact directly with the building blocks of native applications (or native objects in hybrid applications). Native object utilities also open or close native applications from anywhere on the device (see [Requirements](#) below).

The advantages of working with Object commands are:

- ◆ The ability to create unpartitioned scripts that operate across supported device models and OS versions
- ◆ The ability to open and close applications from anywhere on the device
- ◆ The ability to base verifications on native objects, creating robust scripts that work across changes to the application UI (e.g., text size or color changes)

The table below describes the commands:

Table 5-4 Native Object Commands

Command	Description
 Launch App	Opens a native application.
 Close App	Closes a native application.
 Object Touch	Finds an object by its text and touches it.
 Object Edit	Enters text in a native object after finding it and optionally, setting focus in it.
 Object Extract Text	Extracts text from a selected object and stores it in a variable for use elsewhere.

This section describes the [requirements for using Web commands](#), how to [search for an element](#) to interact with in a Web command, and how to [specify an action for an element](#). The simple operations to open and close browser sessions are discussed in the [Command Reference](#). The command reference also contains a field-by-field description of each Web command.

5.8.1 Requirements

Using Object commands requires the following:

- ◆ Devices must be configured for this feature to work—please contact your Keynote Solutions Consultant.
- ◆ Supported devices:
 - Android 4.0.4 and higher devices
- ◆ The DeviceAnywhere Agent must be installed on devices. Minimum supported Agent versions are:
 - Android: Agent version 1.11.188
- ◆ Accessibility API installer, `KDAService.apk`, provided by Keynote

After installation on your device, navigate to **Settings > Accessibility > KDAService** and enable the service.

5.8.2 Selecting a Native Object

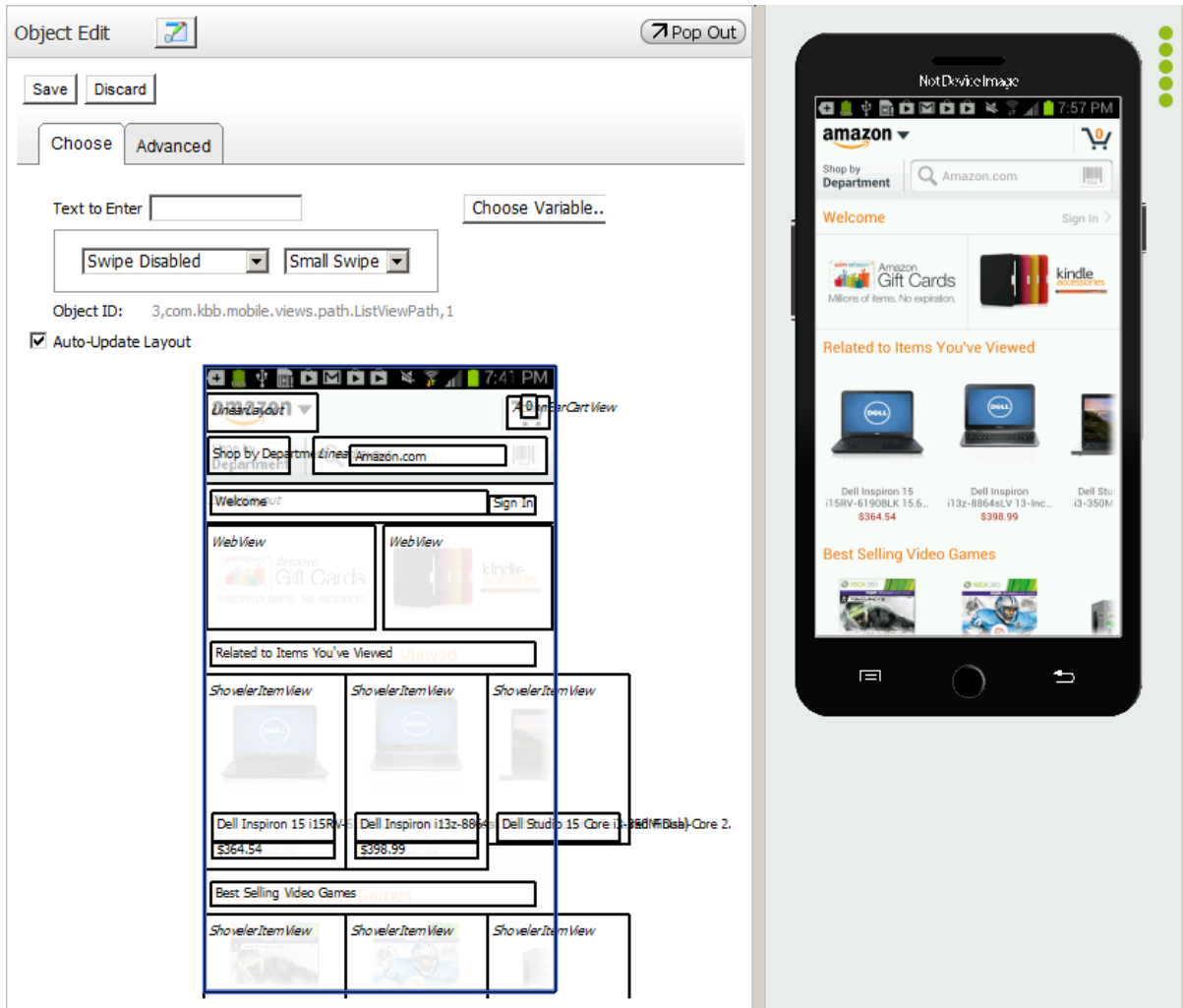
In order to interact with an object in a native application using an Object command, you must first select it from the object layout of the page being tested. You must select an appropriate element in the following Web commands:

- ◆ Object Touch—you must select an object to be touched.
- ◆ Object Edit—you must select a field object and specify text to be entered in it.
- ◆ Object Extract Text—you must select an object from which the displayed text is extracted and stored in a variable.
- ◆ In Wait Event and in states, you must select the object you wish to use as a [reference point](#) to verify a sequence of native application interactions.

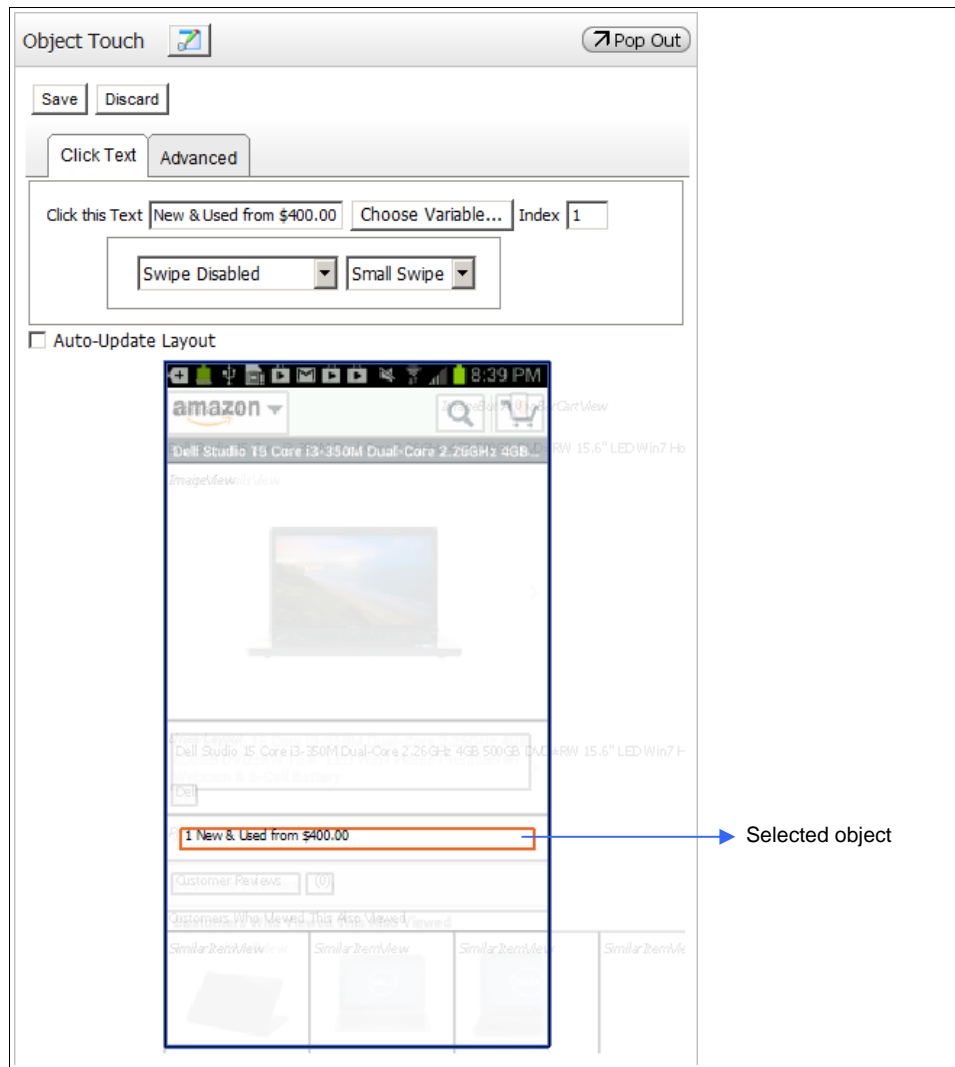
This section describes how to work with the object layout presented in Object commands when you navigate to a page in a native or hybrid application.

To work with native objects, you must:

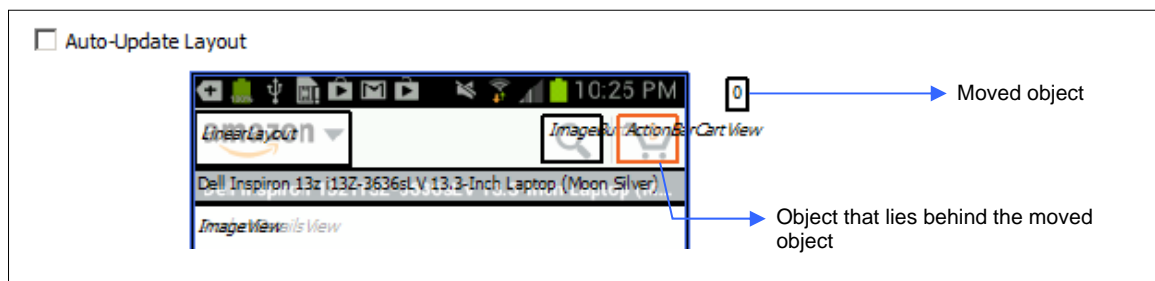
- 1 Acquire your device.
- 2 Use the Launch App command or directly launch the application you wish to work with on the device.
- 3 Drag a command (e.g., Object Touch) onto the script canvas. You will see a grid representing the object layout in the command.



- 4 Click to select the object you wish to work with. The object is outlined in orange.



- When you select an object, the center of the selected region is touched at run time for any action to be performed. The object layout of a page sometimes has many layers of objects. You can even move an object out of the way to clearly see another that lies behind it.



NOTE When selecting an object, ensure that no other object lies at its center or the wrong object might be selected at run time.

- In some application pages, objects in the layout appear outside the edges of the device screen. The object you select must at least be partially contained within the edges of the device screen or it will not be correctly selected during run time.

Object Touch Pop Out

Save Discard

Click Text Advanced

Click this Text Choose Variable... Index 1

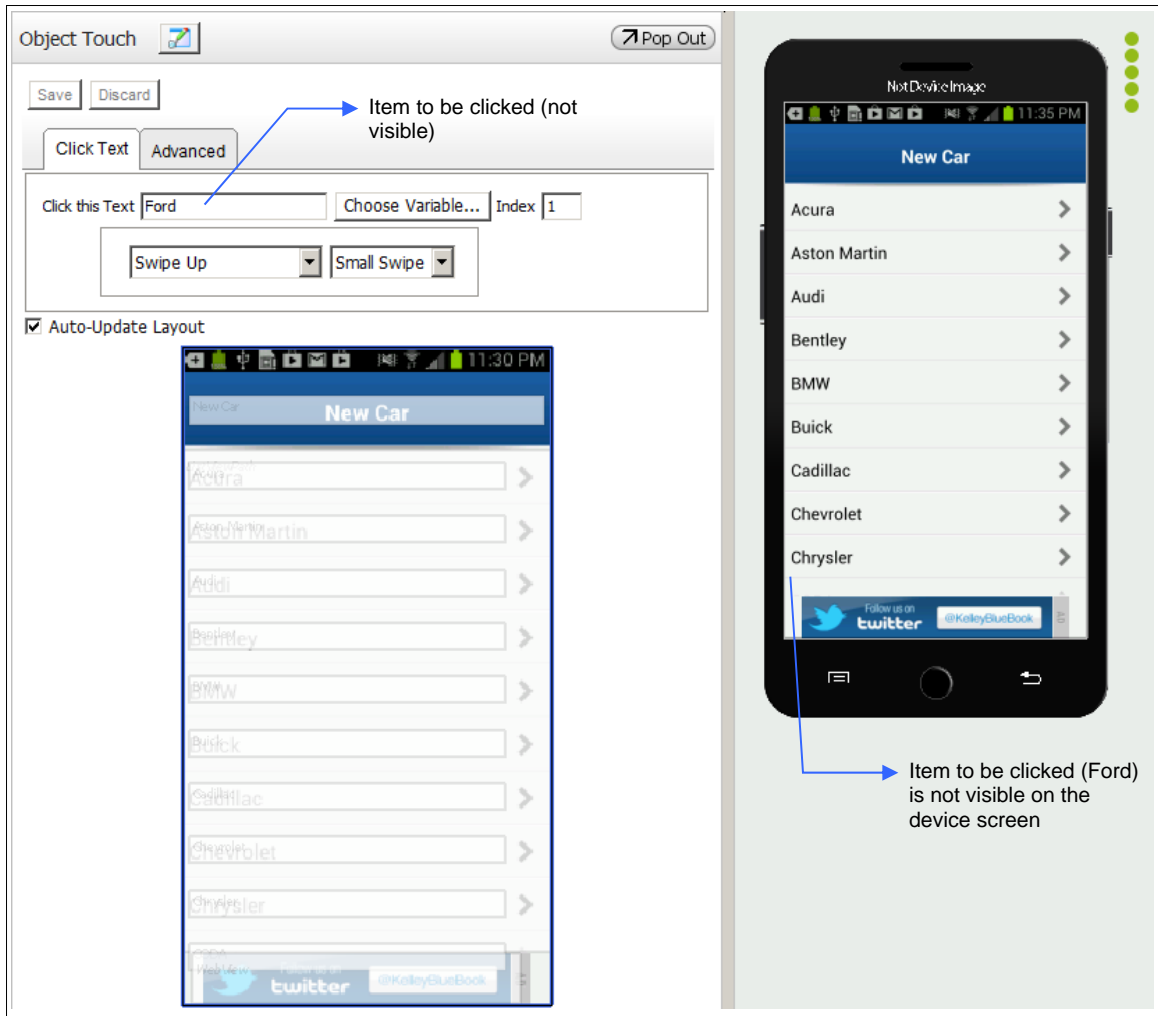
Swipe Disabled Small Swipe

Auto-Update Layout

Do not select objects that lie off screen

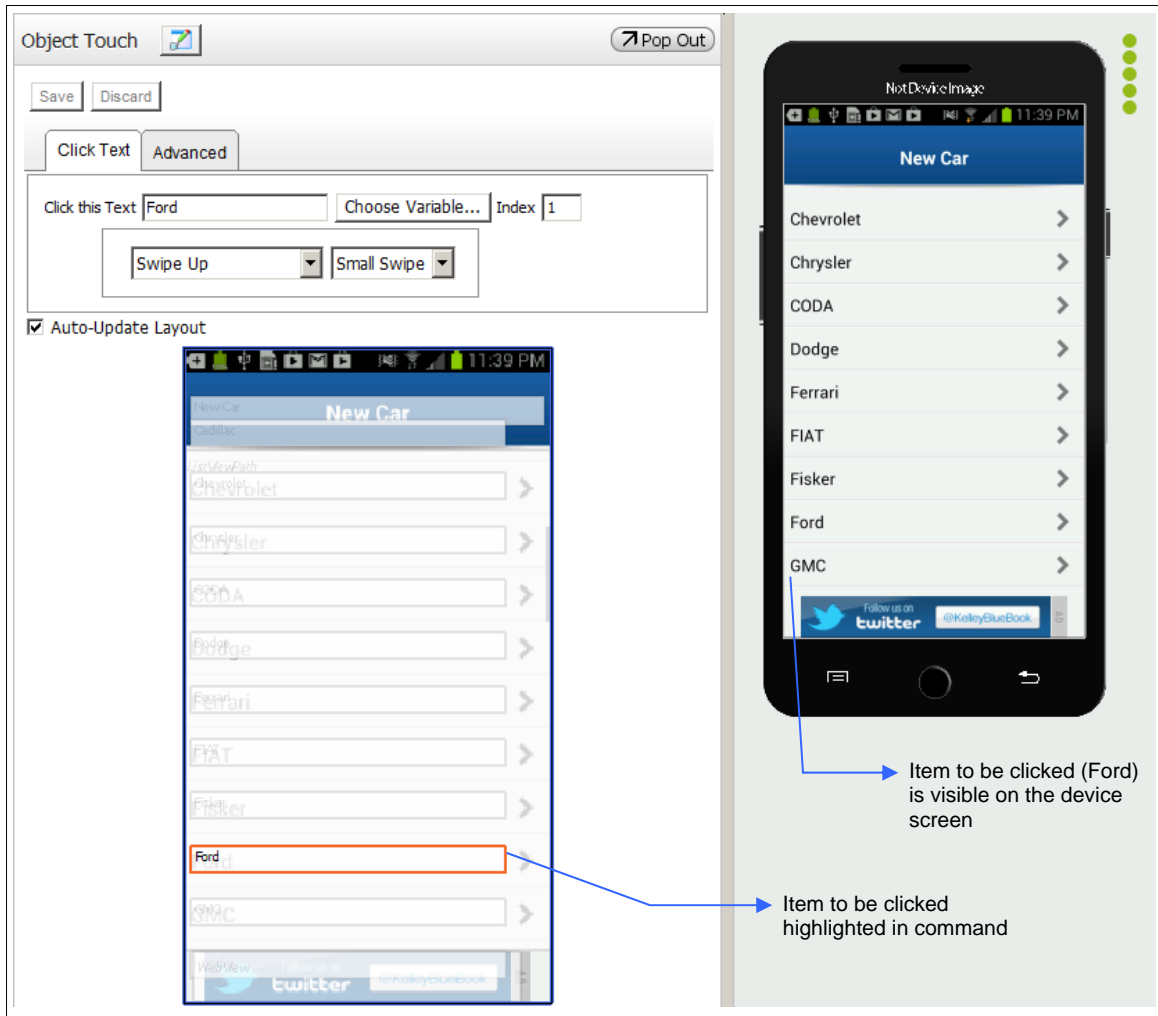
Object that lies partially on screen—test to ensure it is selected correctly during execution

- As you navigate from screen to screen on the device, the object layout in the command is updated if **Auto-Update Layout** is checked.
- When you are on an application page with a long list of selectable items, the object you want might only be visible after scrolling up/down or left/right. For example, on the page shown below, you would need to swipe the screen up before you can see the automobile make “Ford.” You can opt to implement small or large vertical or horizontal swipes so that the object you want is visible on the device screen. Choose the same swipe direction as your finger would move if you were physically controlling the device.

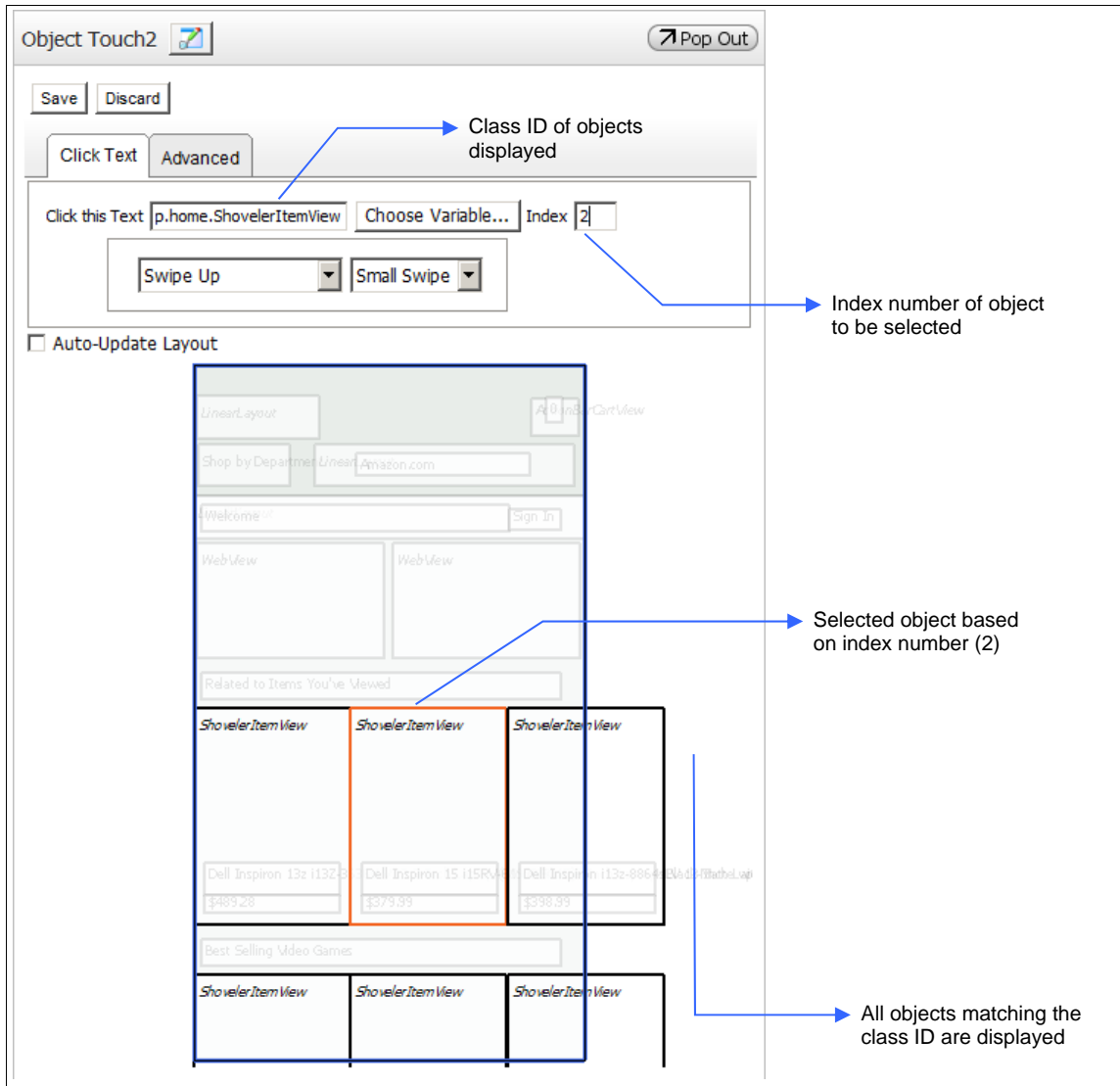


At run time, the screen is scrolled in the direction chosen until the selected text is found or until the command times out. You can increase the **Timeout** value in the **Advanced** tab when implementing a scroll to allow for adequate time to find the selected object.

If you scroll the device screen to view the item you want (and **Auto-Update Layout** is checked in the command), you will see the object highlighted and selected correctly.







- In the Object Touch command, you can select an object by entering object text or by passing in the value contained in a variable. You might want to do this to loop through data set values to select objects. Click **Choose Variable** to pass in the value(s) contained in a variable.
- In the Object Touch command, when you select an object with no visible text, its class ID is displayed instead. If there are several objects that match the class ID, they are displayed. You can then use the **Index** number to indicate which of the displayed objects you want selected.



5.8.3 Acting on a Selected Object

Once you have chosen down the object you wish to work with, you can perform one of the actions available in Object commands or in Wait Event or a state:

- ◆ In the Object Touch  command, the selected object is clicked.
- ◆ In the Object Edit  command, you can [enter a value in the selected field object](#)—ensure that you have selected an object that can accept the entered value.
- ◆ In the Object Extract Text  command, you can [select a variable to store extracted text](#) from the selected object—ensure that you have selected an object with a text value.
- ◆ In the Wait Event  command or a state, the selected object can be used as an [object-based reference point](#).

5.8.3.1 Entering Text in an Object

You might want to enter text in a form field or a search field in a native application—use the Object Edit command. Be sure to correctly select the field object in your command. For example, in the image below, the search field is selected, not the placeholder text in front of it.

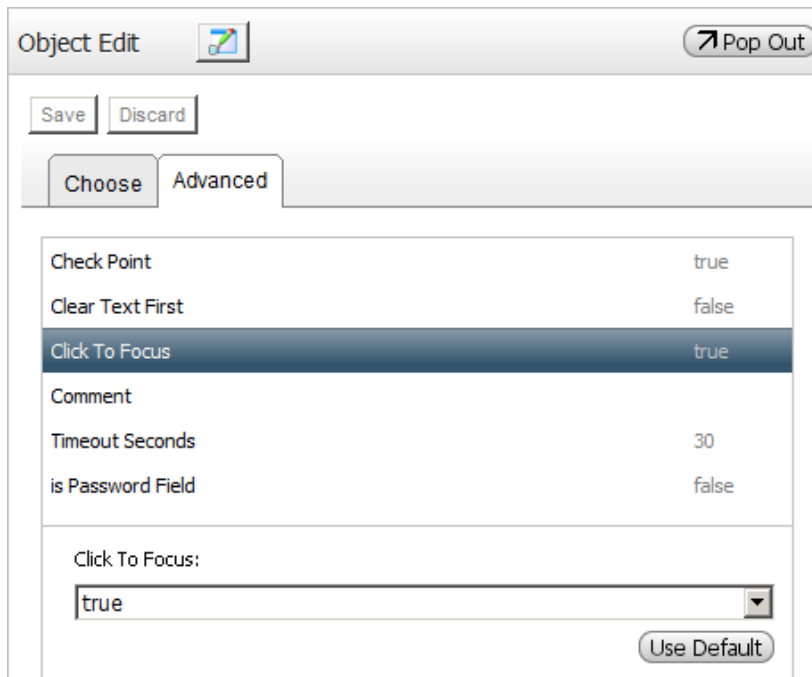
Specify **Text to Enter**; you can also select **Choose Variable** to pass in the text contained in a variable.

Figure 5-50 Entering Text in a Selected Object



In the **Advanced** tab, an option to set focus in the selected field (**Click to Focus**) is set by default to **true**. You can also specify if the field is for password entry (**is Password Field**) so that text can be encrypted.

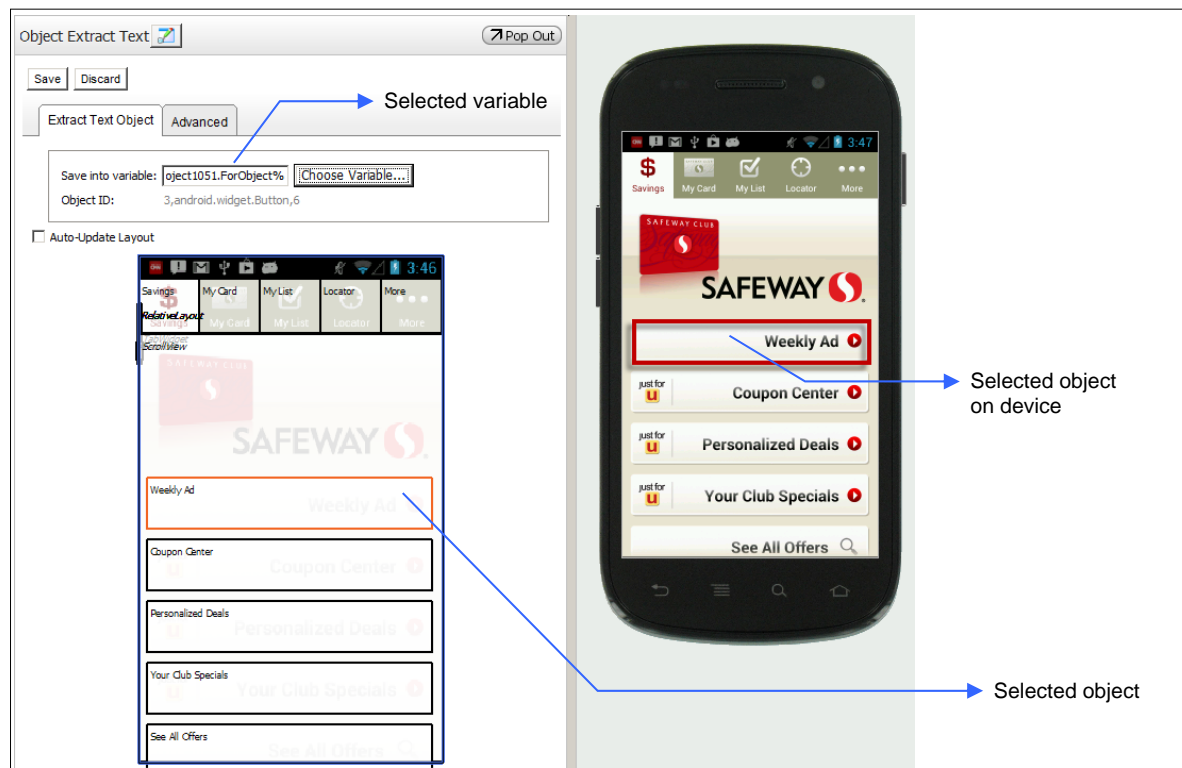
Figure 5-51 Advanced Options for Entering Object Text



5.8.3.2 Extracting Object Text

In the Object Extract Text command, select an object and then click **Choose Variable** to select a variable in which to store the text value.

Figure 5-52 Storing Object Text in a Variable

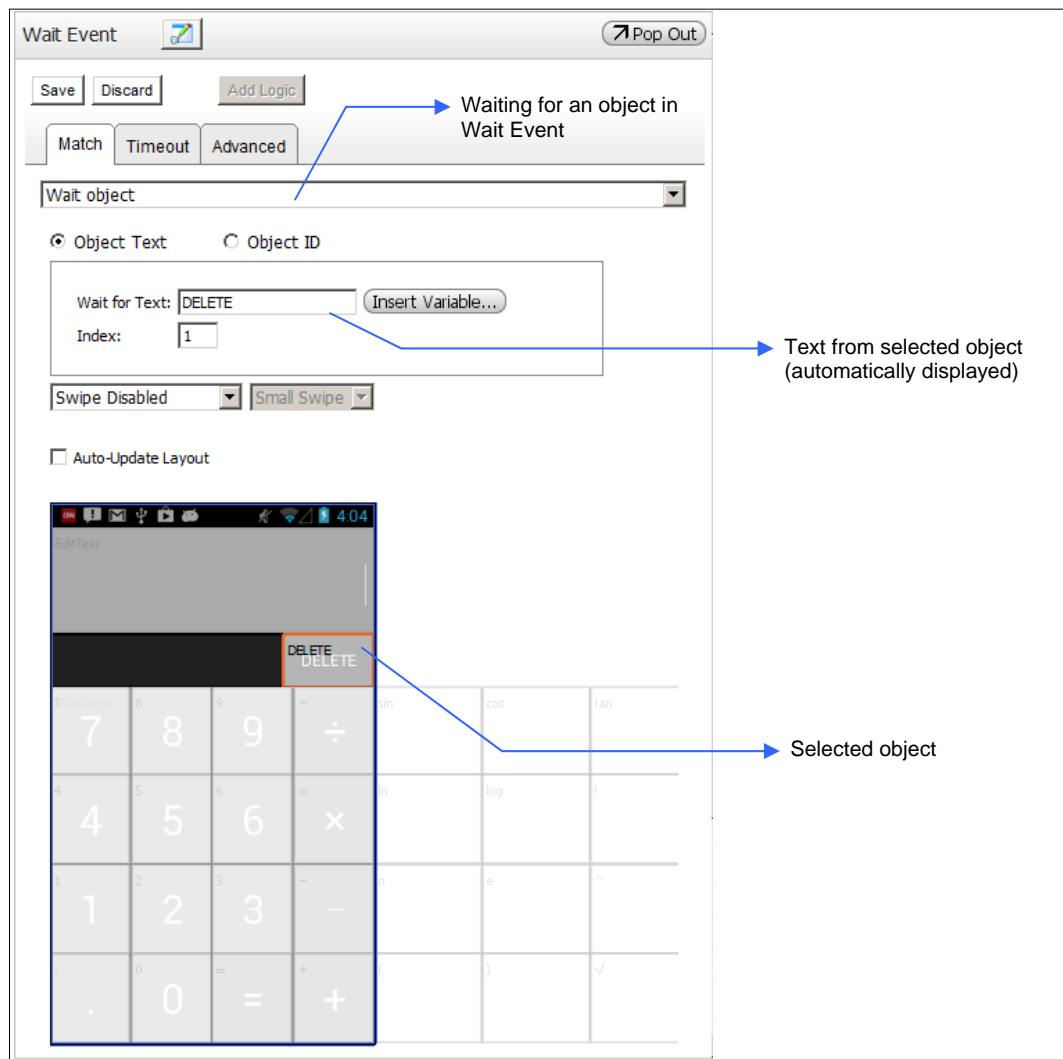


5.8.3.3 Waiting for an Object

To use an object as a reference point, use the Wait Event command and select **Wait object** from the drop-down list. Select an object.

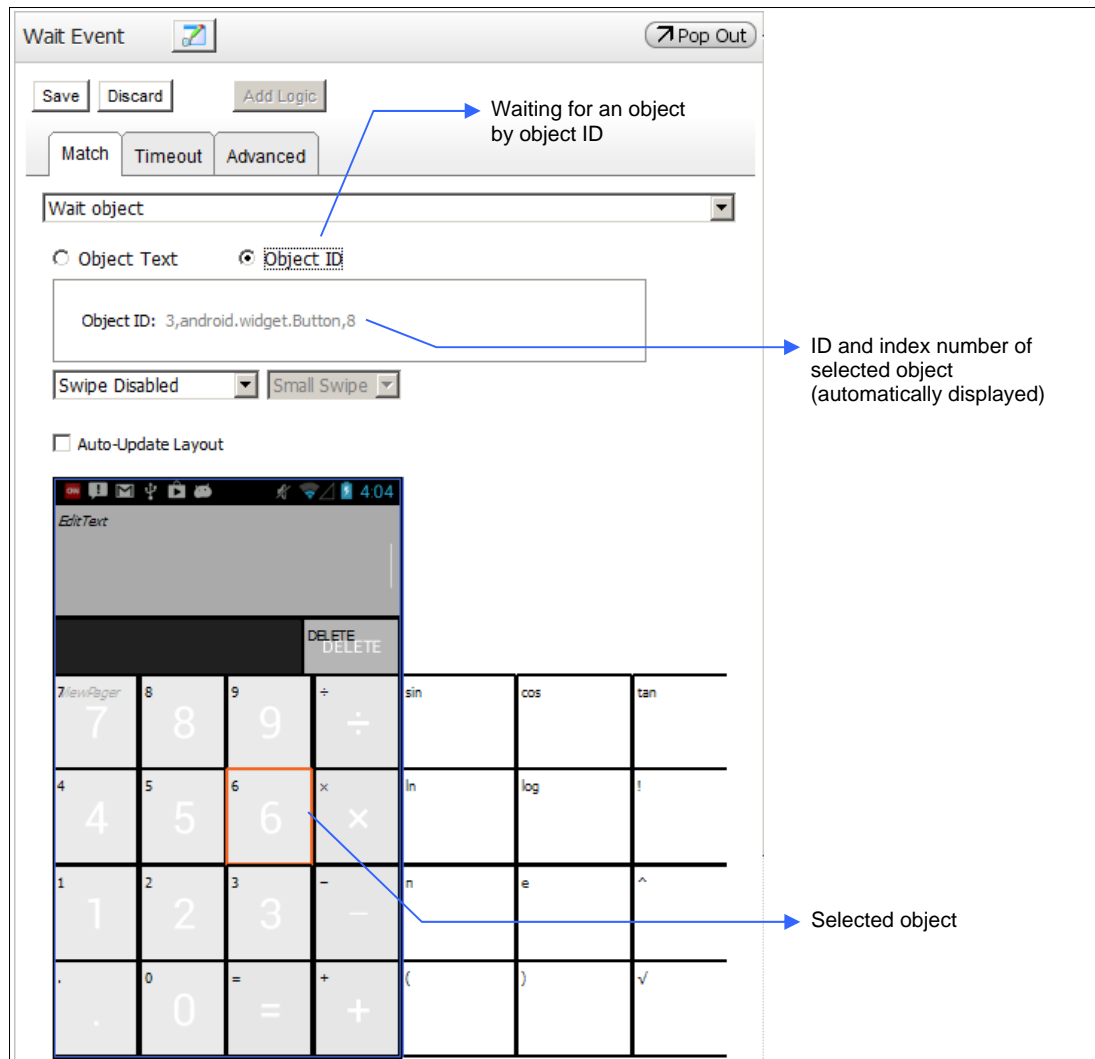
- ◆ If you opt to wait for **Object Text**, the text is automatically displayed. The system waits to find this text for script verification at run time. You can even pass in the text to wait for from a variable—click **Insert Variable**. You would do this, for instance, if the object you want to wait for does not appear on the screen during command definition. If multiple objects have the same text, choose the **Index** number of the specific object you want to wait for.

Figure 5-53 Wait Event—Object Text



- ◆ You can select an object to wait for by its **Object ID**. When you select an object from the layout, its ID, followed by the index number, is automatically displayed.

Figure 5-54 Wait Event—Object ID



5.9 Script Logic


DeviceAnywhere Studio enables you to construct scripts with varying levels of complexity, ranging from a simple linear progression of commands to multi-path scripts involving [branches](#) and [loops](#) based on various conditions. You can also explicitly [terminate a script](#) in success or failure and define customized termination messages for display in test results.

5.9.1 Branches

DeviceAnywhere Studio allows you to set the if-then conditions for script branches based on:

- ◆ The outcome of a command (as in [Navigate To](#))
- ◆ The reference point found (as in [Wait Event](#))
- ◆ The value of a parameter or variable (see the [Branch](#) command in the [Command Reference](#)).

5.9.1.1 Navigate To Command

The Navigate To  command presses a key (or key sequence) and then waits for a reference point. It repeats this procedure for a fixed number of times or until the reference point is found, whichever comes first. The Navigate To command creates Success and Failure branches based on its outcome, i.e., on whether the reference point has been found. Refer to [Examples](#) for a script that implements Navigate To.

When you drag the Navigate To command onto your script canvas, it creates placeholders for three branches: Success, Failure, and Tie.

Figure 5-55 Default Branches in Navigate To



You can drop commands in the Success branch to define action if the reference point is found. Likewise, you can drop commands in the Failure branch to define action if the reference point is not found. When done defining the Success and Failure branches, you can use the Tie branch to tie branches together and continue with the script. For example, you might want to set a variable to one value in the Success branch and to another value in the Failure branch, after which you want both branches to follow the same command sequence—use the Tie branch to define this command sequence and continue with the script.

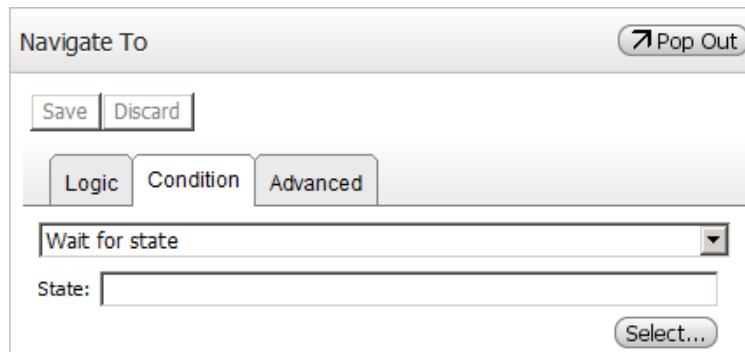
Select the command to view command properties (see Figure 5-56 below) and fill out the fields as follows:

- 1 Enter a **Key Sequence**. The command will press this key sequence and wait for a reference point up to the maximum number of times specified. As in *Send Keys*, you can **Record** device interaction to save a key/touchscreen sequence. The captured interaction is displayed in the **Key Sequence** field. Click **Insert Variable** to insert the key sequence contained in a variable or parameter.
- 2 In the **Repeat no more than** field, select the number of times Navigate To must press the key sequence and then wait for a reference point.
- 3 Specify an appropriate **Input mode** depending on the type of field text is being entered into on the device—see [Specifying Device Input](#) for more on key modes.

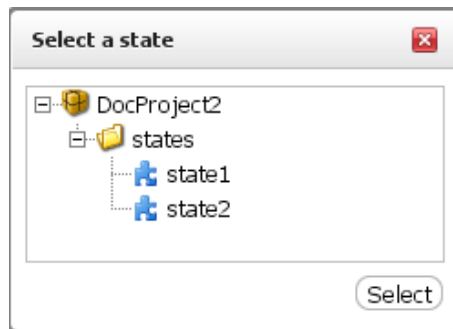
NOTE Use the **Alpha** key mode when entering *navigation keys* in the **Key Sequence** field.

- 4 In the **Advanced** tab, select the length of time (in milliseconds) each key is pressed for in **Hold Time** and the time between key presses in **Delay Time**.
- 5 Select a reference point to wait for in the **Condition** tab.
 - Select **Wait for image** from the drop-down list provided to define a device-specific, image-based reference point. The current device screen is automatically captured in the command. You can then select a screen region as a reference image.
 - Likewise, selecting **Wait for text** enables you to define a device-specific, text-based reference point to wait for. Refer to [Image-Based Reference Points](#) and [Text-Based Reference Points](#) for instructions.

- You can also choose to use a reference point already defined in a state. The correct implementation of the state is automatically loaded and used at runtime.
 - i Select **Wait for state**.



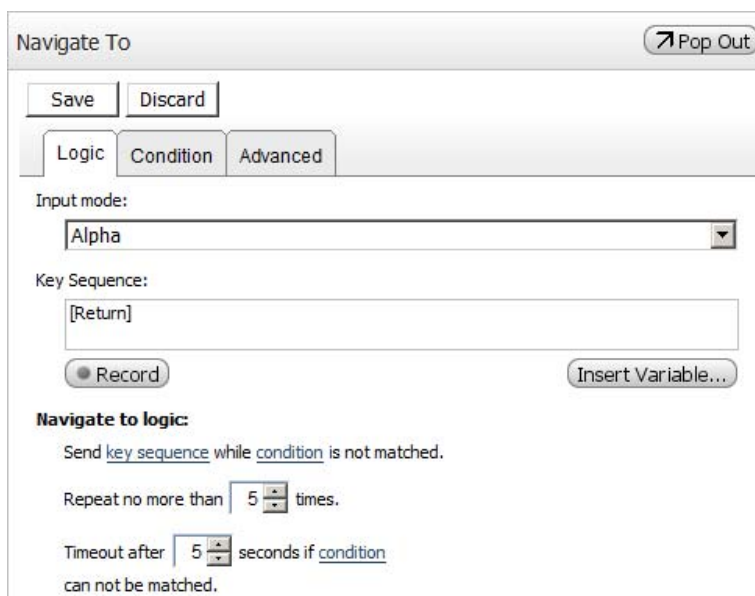
- ii Click **Select** to choose a **State**.
- iii Choose a state from the list provided and click **Select**.




The state is now listed in the **State** field.

- 6 In the **Logic** tab, specify the **Timeout** (in seconds) within which the reference point must be found.

Figure 5-56 Navigate To Command Properties

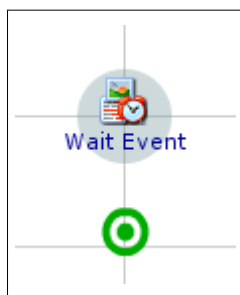


5.9.1.2 Wait Event Command

The Wait Event  command searches for and uses a combination of [reference points](#) as the basis for creating script branches. The reference points serve as “if/then” conditions for passing script control to a branch, e.g., if a reference image is found, the script takes the associated branch. You can create script branches based on both device-specific reference points as well as device-independent states. This command is useful when there are multiple possible outcomes for a command sequence, and a different course must be followed depending on the outcome.

When you first drag the Wait Event command onto your script canvas, the default type of reference point is image based. The current device screen is automatically captured in the command. No additional branches/placeholders are created.

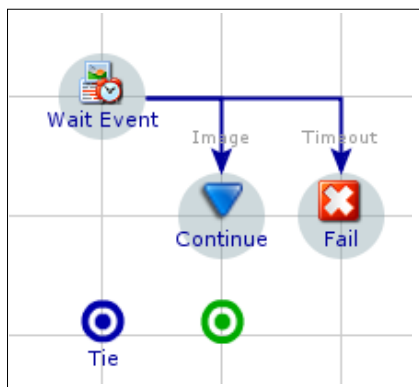
Figure 5-57 Wait Event in Script Canvas



You can change the default reference point from image based to some other type.

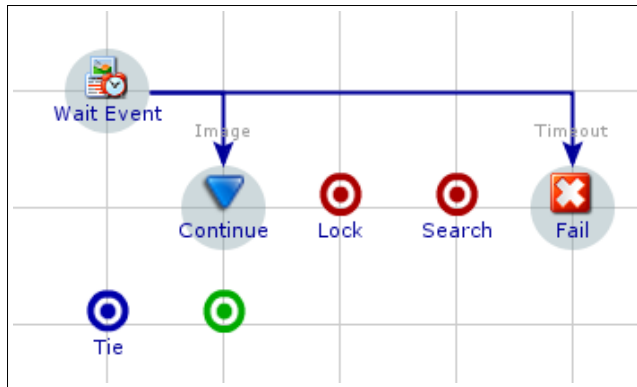
You can also choose to add logic, creating additional branches for script control. Use the Timeout branch can be used to define a command sequence if Wait Event times out, i.e., if no reference point is successfully matched. This branch has a Fail command in it by default. There is also a branch for defining a command sequence if the default, image-based reference point is met. This branch has a Continue command by default. You can use the Tie placeholder to tie these branches together and continue with the script. In other words, after your script branches have run their course, use the Tie branch to define a command sequence for all branches to follow and continue with the script.

Figure 5-58 Wait Event with Additional Logic



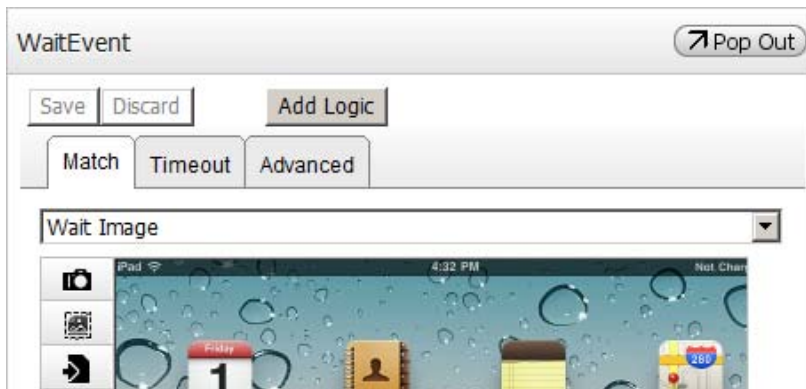
An additional placeholder is created for each reference point you add to Wait Event. You can then drop commands in the placeholders to define script action based on the reference points. (The actual Wait Event reference point found and the branch taken during a script run will depend on the outcome of preceding commands.)

Figure 5-59 Wait Event with Logic and Added Reference Points

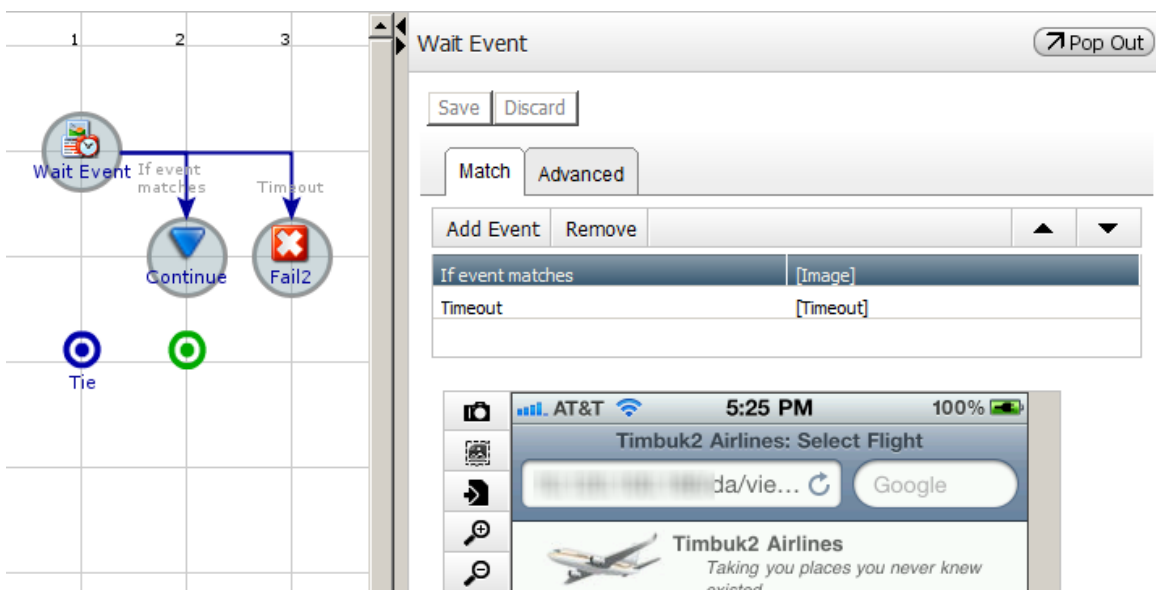


To add logic and define a Wait Event composed of multiple reference points:

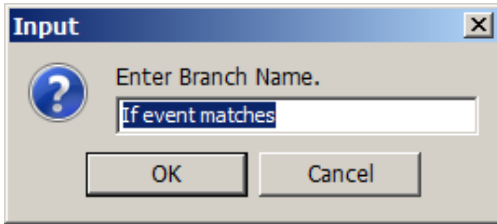
- 1 Select the command to open up command properties. The command displays controls for an image-based reference point. The current device screen is automatically captured in the command.



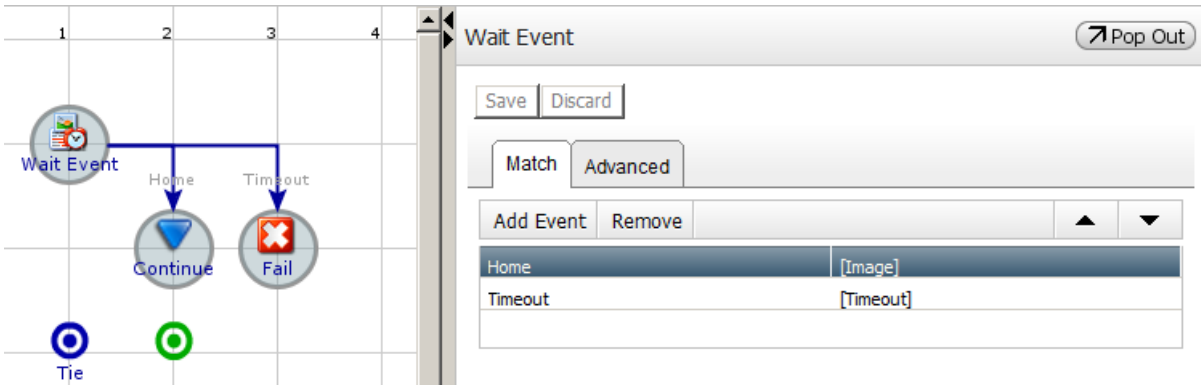
- 2 Click **Add Logic** to display the default reference point in a branch of its own along with the Tie and Timeout branches. The command displays an event list that includes the default reference point and the Timeout branch. The script canvas shows additional placeholders.



- 3 You can select the default event and double-click to change the name of the branch. Enter a name in the dialog box that appears and click **OK**.

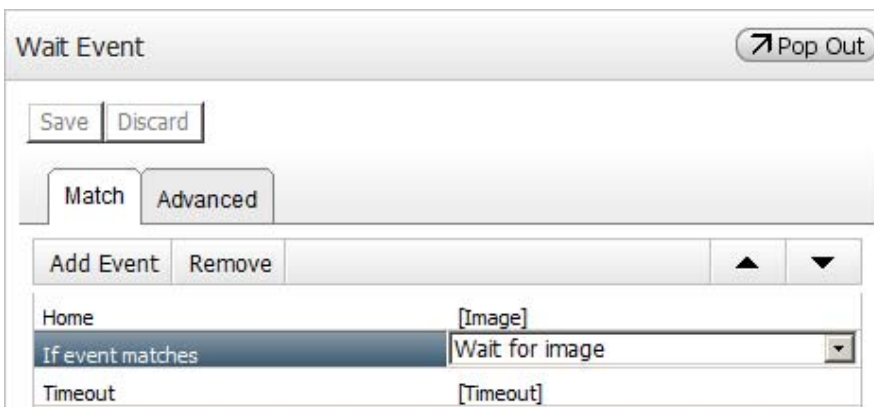




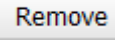
The new name is reflected in the script canvas and in the command event list.



- 4 You can also choose/change the type of reference point from the drop-down list:
 - For an image-based reference point, leave selection as **Wait for image**.
 - For a text-based reference point, select **Wait for text**.
 - To call a reference point stored in a state, select **Wait for state**.
 - For an audio-based reference point, select **Wait for audio**.
 - For a web element-based reference point, select **Wait for web element**.
- 5 Click **Add Event** to add a new branch. By default, a branch based on an image-based reference point is created. You can change the type of reference point and add as many events as required.

The added reference point is now listed in the **Events** list (see also Figure 5-60).

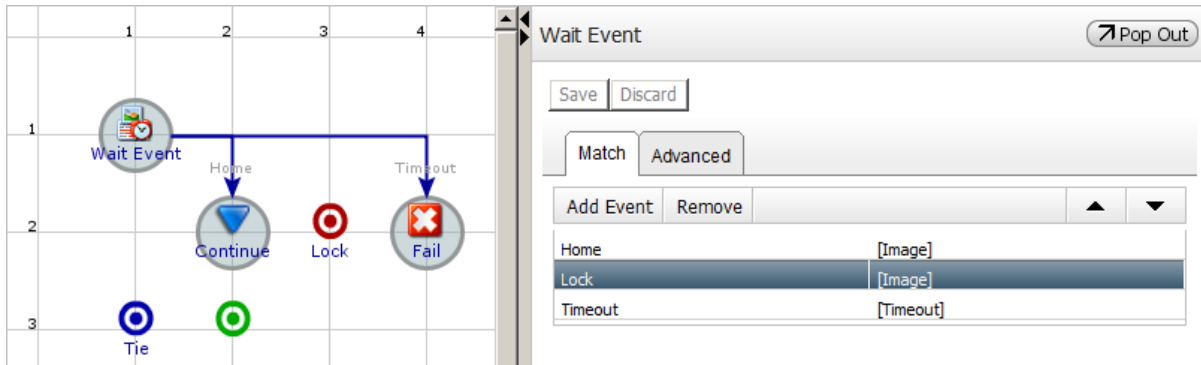


NOTES You can rearrange your event list: Select an event and move it up in the list , move it down in the list , or delete it . You cannot delete or move the Timeout event.

The system checks for reference points starting from the top of the event list—arrange your events in the order that you want them checked. The first reference point found determines the branch taken.

- 6 Click an event in the event list to rename it in the dialog box that appears.

The script canvas and event list in the command reflect the name changes.

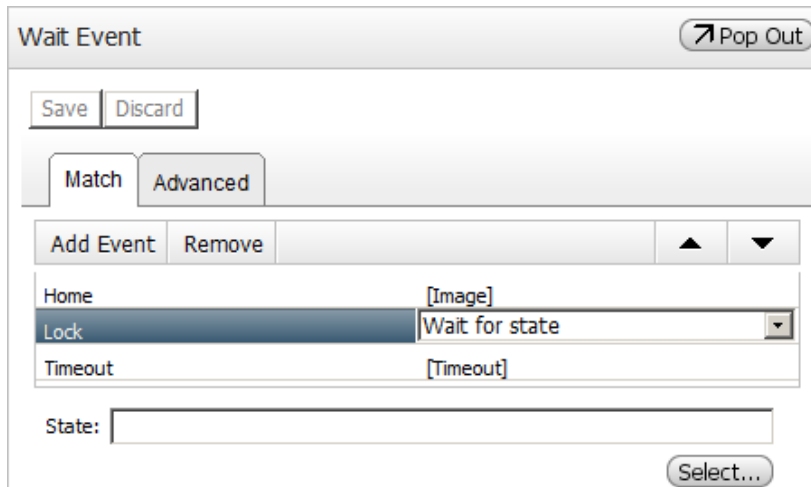


You cannot rename the Timeout branch.

- 7 Select an event in the event list and define your text-, image-, web- or audio-based reference point. See [Image-Based Reference Points](#), [Text-Based Reference Points](#), [Audio Reference Points](#), and [Web Elements](#) for detailed instructions.

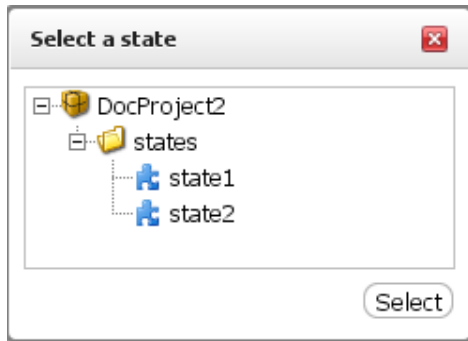
If adding a reference point already defined in a state, select the state.

- a Select that state-based reference point from the event list.



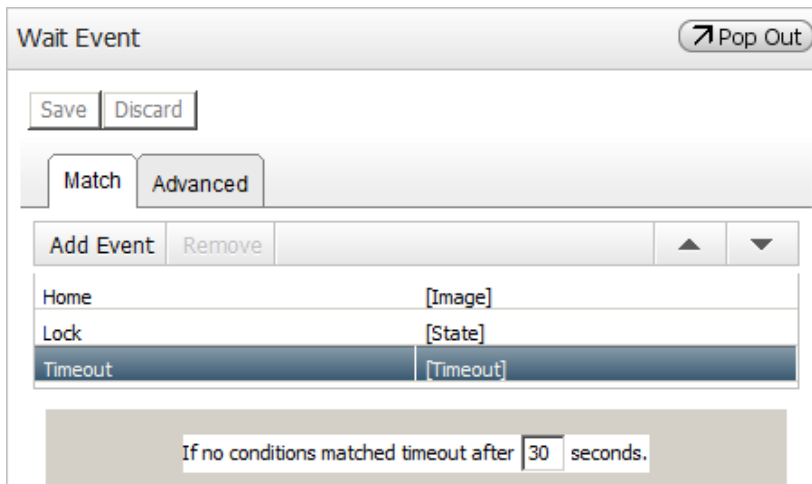
- b Click **Select** to choose a **State**.

- c Choose a state from the list provided and click **Select**.



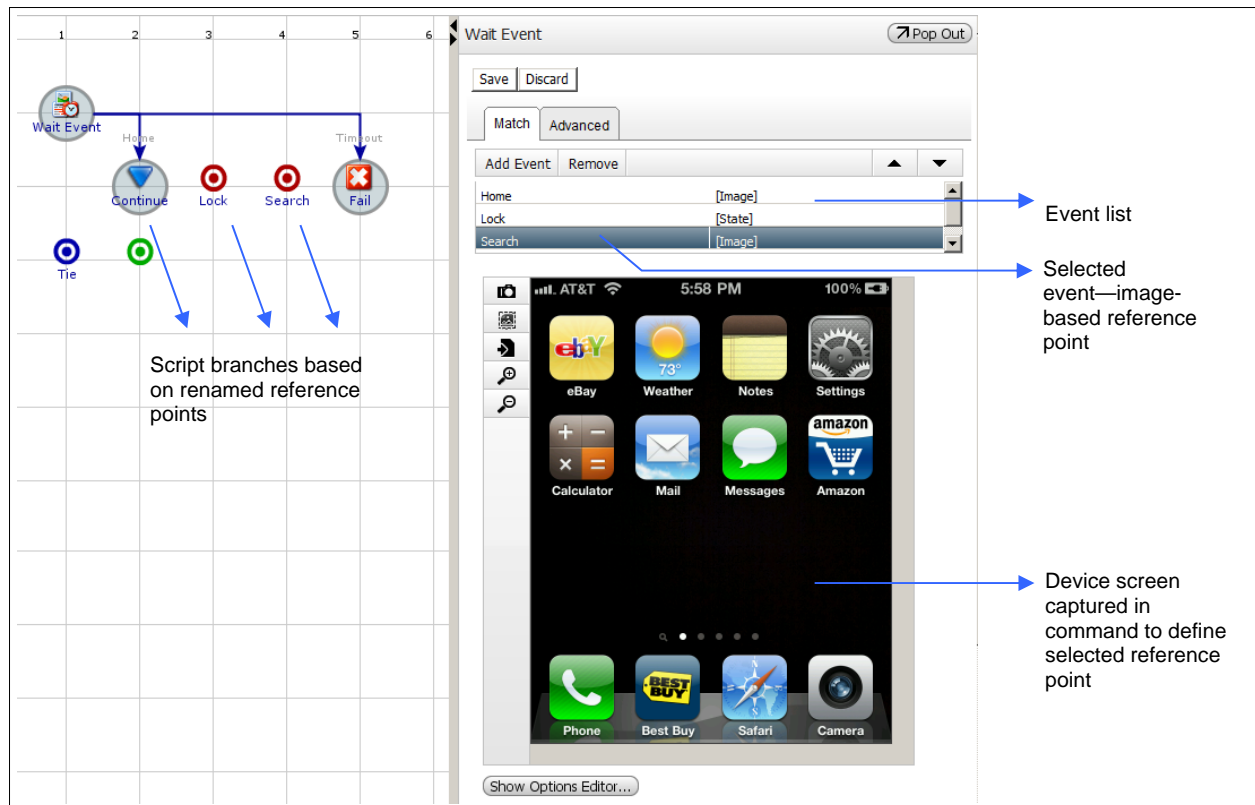
The state is now listed in the **State** field.

- 8 Select **Timeout** from the event list and specify the time (in seconds) within which any reference point must be matched.



- 9 **Save** your settings. The script canvas displays a branch for each reference point.

Figure 5-60 Wait Event Command Properties and Script Canvases



Refer to [Examples](#) for a script that implements the Wait Event command.

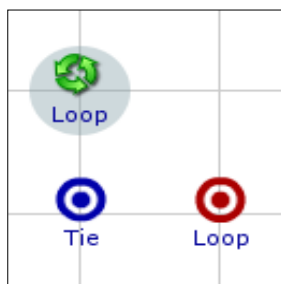
5.9.2 Loops

The Loop command allows a script to loop over a set of commands for a fixed number of times or until certain conditions are met. A loop condition is set by the value of one or many parameters/variables. The variable or parameter value(s) must be set earlier in the script and then reset within the loop.

NOTE See [Working with Data Sets](#) for a description of using Loop with data set records.

When you drag the Loop command onto the script canvas, it creates placeholders for Loop and Tie branches.

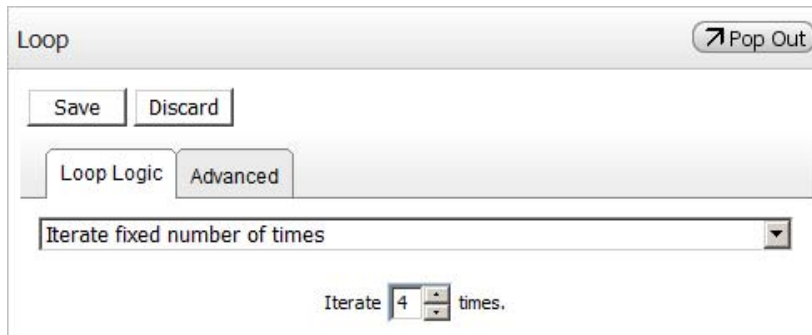
Figure 5-61 Loop Command Script Canvases



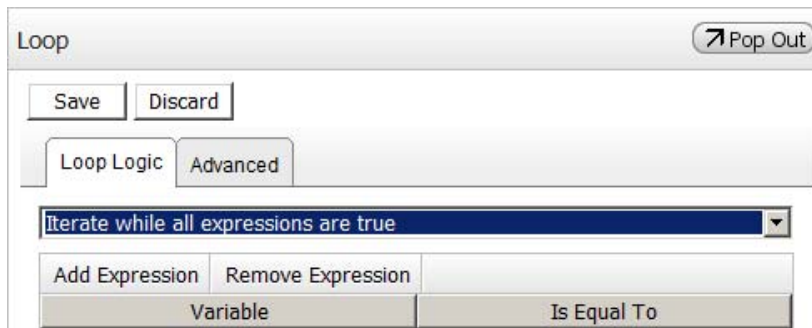
You can drag commands to the Loop placeholder to define a command sequence within the loop. Use the Tie branch to define a command sequence after the script has exited the loop.

You must set a loop condition in the Loop command:

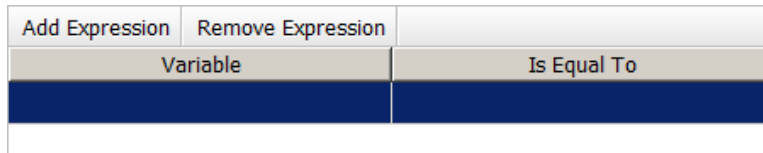
- 1 Select the command to view command properties.
- 2 Set a loop condition.
 - To have your loop iterate for a fixed number of times:
 - i Select **Iterate fixed number of times**.
 - ii Enter a loop count in the field next to it.



- To set a loop condition based on the value of parameters or variables (all expressions must no longer be true for the script to exit the loop at runtime):
 - i Select **Iterate while all expressions are true**. This enables you to set the value of one or more variables/parameters as the loop condition.



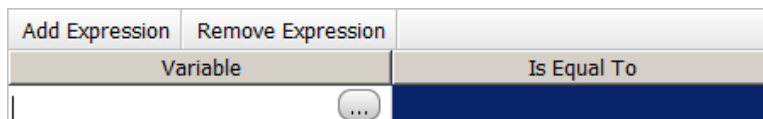
- ii Click **Add Expression** to select a variable/parameter as a loop condition. This creates a blank row in the expression list.



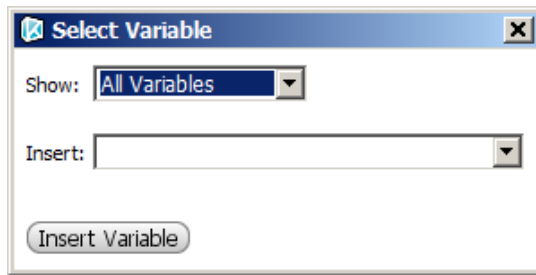
NOTES You can select an expression in the list and click **Remove Expression** to delete it.

You can add several expressions as the loop condition.

- iii Double-click in the **Variable** column and click the  button to select a variable.



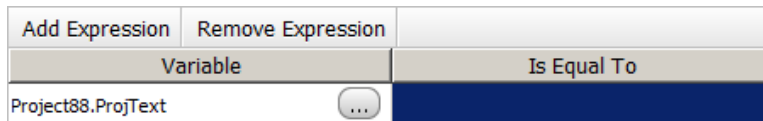
This brings up the Select Variable dialog box.



- iv Choose a variable/parameter and click **Insert Variable**.

NOTE Do *not* choose a data set column for use as a loop condition. Data set values cannot be reset using the Set Variable command.

The selected **Variable** name is displayed. Global variables are prefixed with the project ID, e.g., `Project88.ProjText`.



- v Double-click in the **Is Equal To** column and enter a value for the variable/parameter as a loop condition. You should have set the variable to this value earlier in your script, e.g., using the Set Variable command. You must also reset the value within the loop so that the loop can be exited.

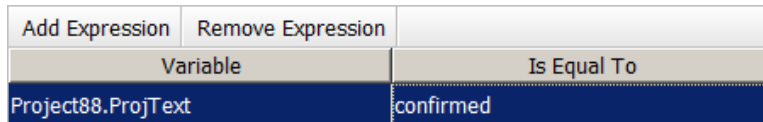
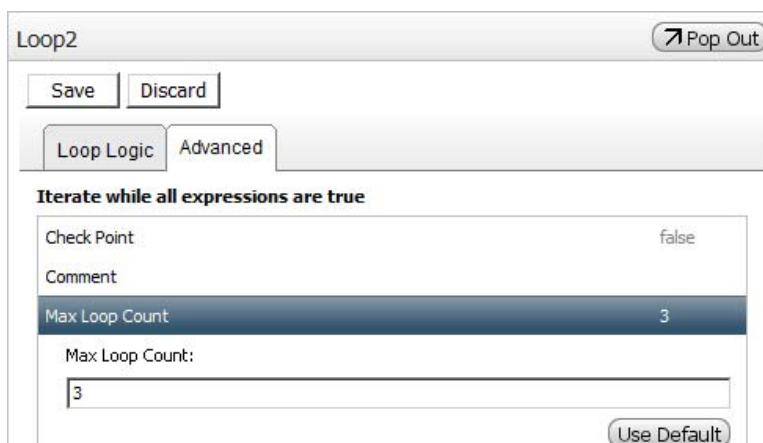


Figure 5-63 below shows a sample script that sets variable values, creates a loop, and resets the value within the loop.

- vi Optionally, to have the loop iterate for a fixed number of times *after* loop conditions have been met, check **Max Loop Count** in the **Advanced** tab. Enter a count.



- 3 **Save** your settings.

Figure 5-62 Loop Condition Using Multiple Expressions

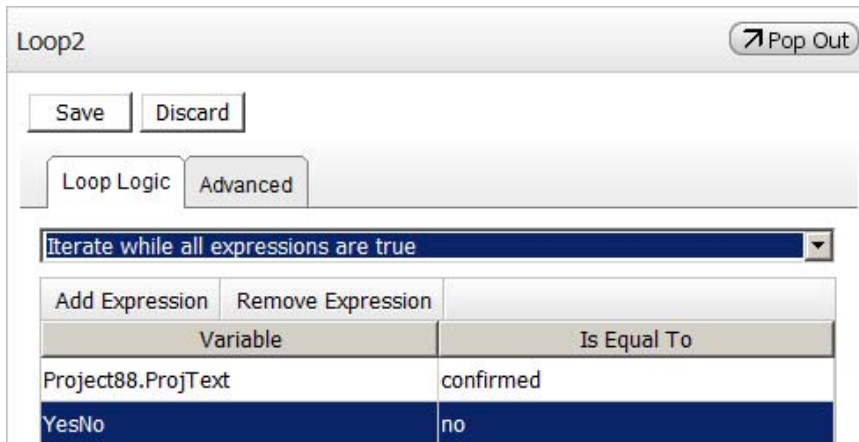
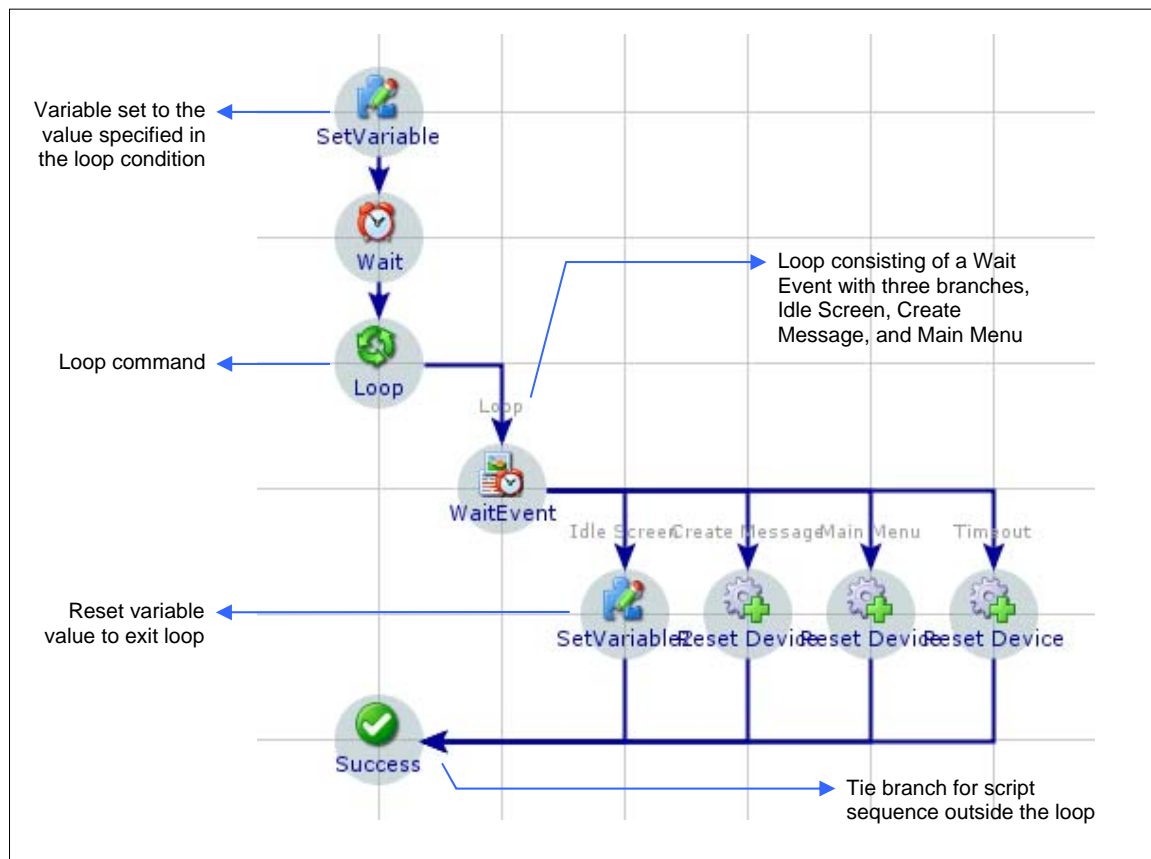


Figure 5-63 Using the Loop Command in a Script



Refer to [Examples](#) for a script that implements the Loop command.

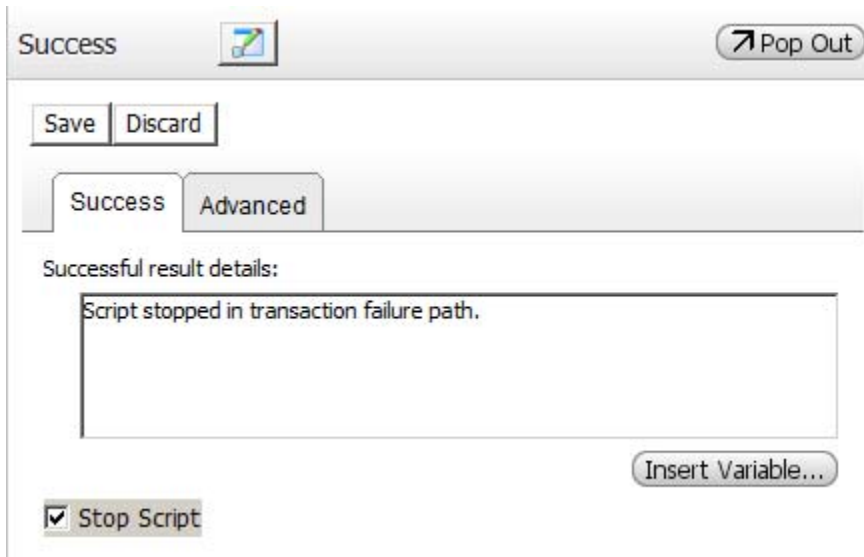
5.9.3 Script Termination

While a script automatically terminates at the end of your command sequence, you can explicitly terminate your script at specific points using the Success or Fail commands. You can use these commands if your script has several branches that you want to define outcomes for. You can specify a customized termination message in these commands for display in test results.

5.9.3.1 Success

To use the Success command:


- 1 Drag the Success  command onto your script canvas:

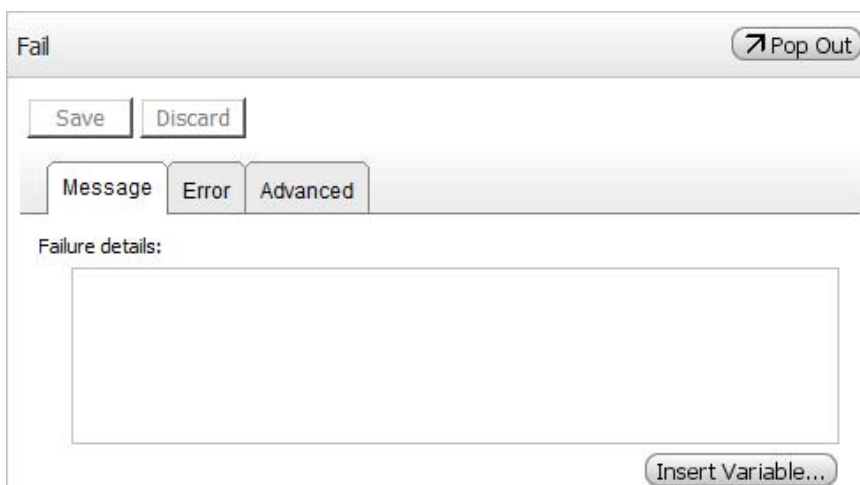


- 2 If desired, overwrite the default termination message for display in test results. Optionally, click **Insert Variable** to include the contents of a parameter or variable in the termination message.
- 3 Use the command in a branch that the script fails and check **Stop Script** to stop the script without triggering an error.

5.9.3.2 Fail

You can insert the Fail command to explicitly terminate the script with failure. You can choose to display error messaging.

- 1 Drag the Fail  command onto your script canvas.



Enter a termination message for display in test results. You can click **Insert Variable** to insert the contents of a variable/parameter in the failure **Message**. The name of the variable/parameter appears enclosed within percent (%) signs in the field.

- To display error messaging in script results, select the **Error** tab.

The screenshot shows the 'Fail' dialog box with the 'Error' tab selected. The 'Category' dropdown menu is set to 'Do not add error code' and the 'Error type' dropdown menu is set to 'Unknown'. The main content area displays 'No error code selected'. At the top, there are 'Save' and 'Discard' buttons, and a 'Pop Out' button with an arrow icon. At the bottom right, there is a link that says 'Edit Error Types...'. The 'Message' and 'Advanced' tabs are also visible but not selected.

- Choose an error **Category** and select an **Error type** from the list displayed. The error type's predefined description, severity, and error code are displayed.

The screenshot shows the 'Fail' dialog box with the 'Error' tab selected. The 'Category' dropdown menu is set to 'CustomCategory' and the 'Error type' dropdown menu is set to 'Type 3'. The main content area displays 'Type 3' with a description, 'Severity: Notice', and 'Code: 0'. At the top, there are 'Save' and 'Discard' buttons, and a 'Pop Out' button with an arrow icon. At the bottom right, there is a link that says 'Edit Error Types...'. The 'Message' and 'Advanced' tabs are also visible but not selected.

NOTE You can choose an existing error type or select **Edit Error Types** to define a new error type in the [Error Types tab](#) of [project properties](#).

- If desired, add a **Comment** in the **Advanced** tab.
- Refer to [Error Definitions](#) for details on how to implement error definitions.

6 States

States define known device conditions (based on text, images, objects, or audio from the device) that can be referenced to verify the result of a sequence of device interactions. [Reference points](#) serve as expected results against which the outcome of scripts can be verified. For instance, if your test script navigates to the device idle screen, you can use the idle screen to define an image-based expected result.

States are device-independent, that is, they are defined for all project devices. They can be unpartitioned, as when creating an object-based state, or consist of device-specific [implementations](#) to account for differences in interfaces. If an expected result changes over time, a state can be updated in one place, without having to update every script containing a reference to it. Once you create a state, you can use the [Wait Event](#), [Find and Touch](#), or [Navigate To](#) command to call the appropriate state implementation in an action script.

An image-based reference point specifies an area of a device screen to be used for script verification. A text-based reference point specifies a text string from a device screen for script verification. An audio reference point waits to hear any device audio (or specified DTMF sequence) for script verification. An object-based state selects a native object found on an application page for script verification.

NOTES Web element-based reference points can only be defined in the Wait Event and Web Wait commands.

Device- or script-specific reference points can be defined in commands such as Wait Event and Navigate To. These reference points differ from states in that they cannot be reused across scripts.

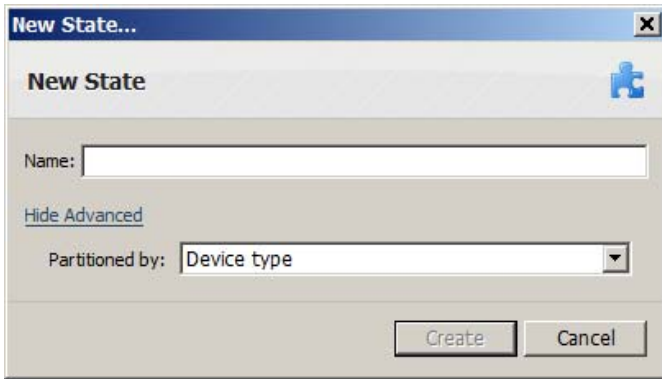
This chapter contains step-by-step instructions for [creating states](#) and [implementing them](#). (The mechanism for defining reference images, text, or audio in device-specific commands or device-independent states is the same, and is explained in the section [Reference Points](#).) This chapter also describes the [state properties dialog box](#).

6.1 Creating a State

To create an automated state:

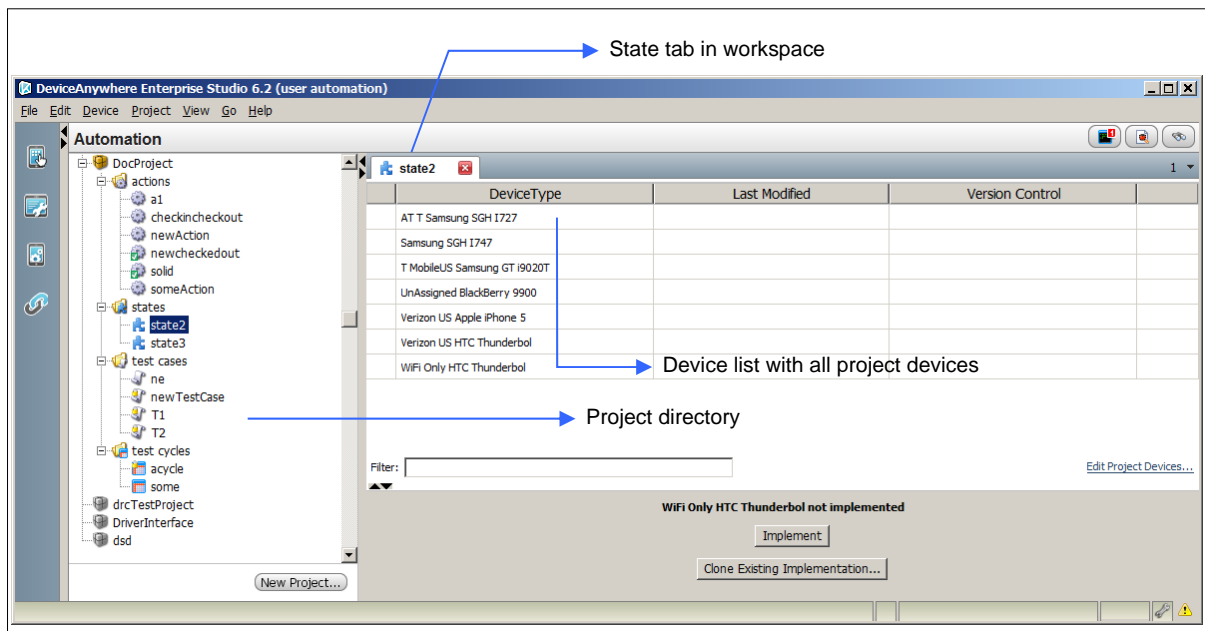
- 1 Make one of the following selections in the Automation/Monitoring view:
 - Right-click your project `states` folder (or sub-folder) and select **New State**.
 - While in your project, select **File > New > State**.
- 2 Enter a name for your state in the New State dialog box.
- 3 Optional: Click **Show Advanced** to display additional controls.


Choose to create a state that is **Partitioned by** device type or that is unpartitioned (as when creating an object-based state). The default is a partitioned state with implementations for each device make/model. (If you have two instances of the same device in your project, they share one single implementation.)



NOTE You would create an unpartitioned state if your project consisted of like devices that did not require separate implementations.

- 4 Click **Create**. This opens up a tab for it in the workspace, where you can view the list of project devices.



You can **Filter** the device list. To resize the pane, hover over the bottom edge of the device list and drag the sizing handle that appears. You can also expand or collapse the device list using the buttons provided .

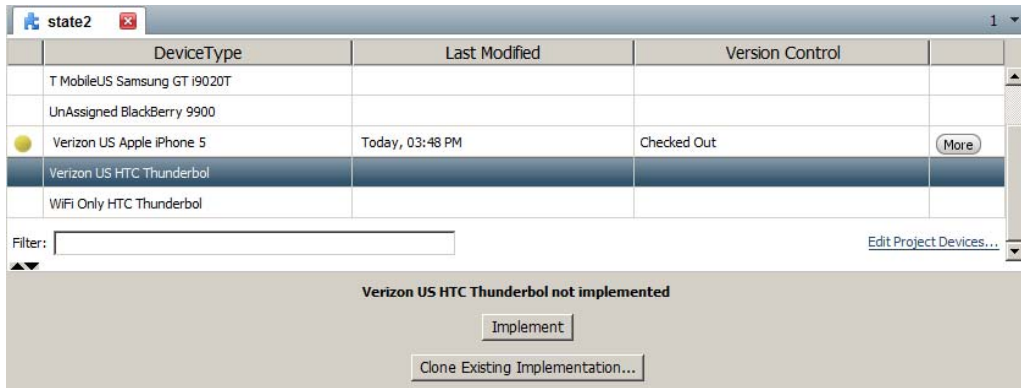
Click **Edit Project Devices** at the bottom of the device list to bring up the [Devices tab](#) of the [Project Properties](#) dialog box.

- 5 Create state [implementations](#) for project devices.

6.1.1 Implementing a State


An implementation is the image region, text string, or audio setting that defines a state on a particular device or device type. When you create a state, you must also implement it in order to specify a reference point on each device. To implement a state:

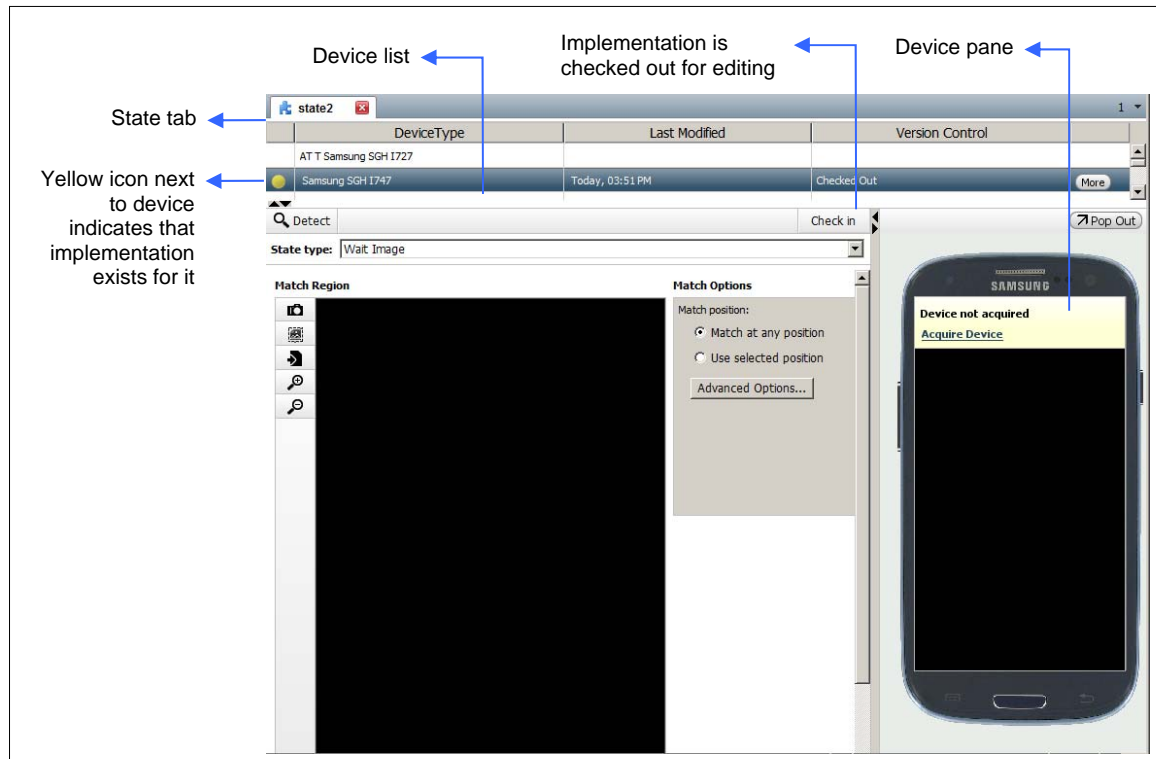
- 1 [Open the state](#) if it isn't already. Each device is listed with its name (carrier, manufacturer, model) in the **Device Type** column, date the implementation was **Last Modified**, and **Version Control** status of the script (checked in or checked out).
- 2 Select a device from the device list and click **Implement**. You can also click **Clone Existing Implementation** to copy and modify the implementation of a like device in the same state.



In addition to the device list, the tab now displays a device pane on the right and controls to define an image-based reference point. (Change the **State type** to select a text or audio reference point if required.)

NOTE If you do not see the device, hover over the right edge of the script canvas and drag the sizing handle that appears.

The device list is updated to display a yellow icon  next to the device, indicating that an implementation exists for it. The implementation is also listed as **Checked out** for editing.



- 3 If your project contains two instances of the selected device, you must select one from the device pane and click **Save Selected**.
- 4 Right-click the device in the device pane and select **Acquire Device** to interact with it live.
- 5 Select a **State type**—you can choose an image, text, or audio reference point.
- 6 Define your reference point—see [Image-Based Reference Points](#), [Text-Based Reference Points](#), [Audio Reference Points](#), and [Waiting for an Object](#) for details.
- 7 [Check in](#) your implementation and release the device when done editing.

See [Managing Scripts](#) for information on copying, checking in, and deleting states.

Check [Examples](#) for sample state implementations.

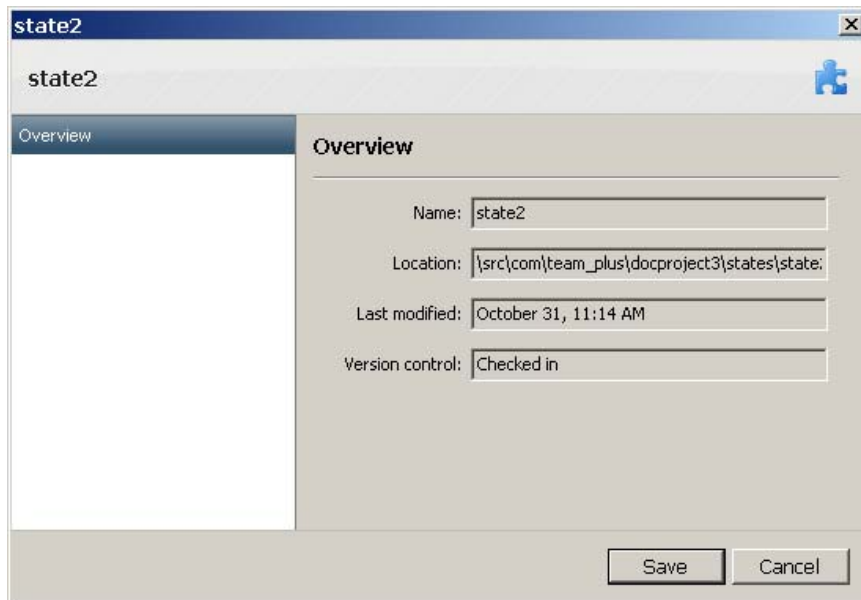
6.2 State Properties Dialog Box

The state properties dialog box displays state metadata. To open state properties, right-click the state in the project directory and click **Properties**.

The state properties dialog box has only one tab—**Overview**. This tab contains non-editable fields for state **Name** and **Location** within the project directory in the local file system, the date **Last modified**, and whether checked in or out of **Version control**.

The **Last modified** field is updated when an implementation is added/deleted. Click **Save** or **Cancel** to close the dialog box.

Figure 6-1 State Properties – Overview



7 Automated Test Cases

A test case is an end-to-end process that accomplishes a specific goal in order to test an application or device. For example, a test case might consist of viewing and deleting call records from a device or performing a banking transaction.

Test cases are modular and generally consist of calls to [actions](#) and [states](#), with controls for [script logic](#). The larger test case goal can be broken down into discrete procedures, represented by actions. In DAE Monitoring and Mobile App Monitoring, test cases are deployed as monitor scripts. In DAE Automation, test cases are the building blocks of [test cycles](#).

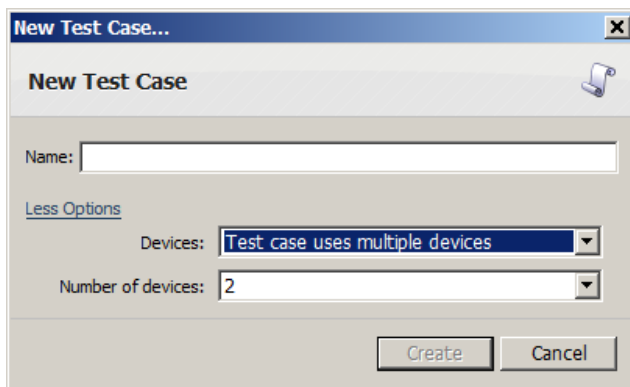
Test cases are device-independent. Several commands for device-specific interactions available in the action toolbar are not available in the test case toolbar (see [Working with Commands](#)). A test case script is valid for all project devices—test cases do not have device-specific implementations. However, you can create test case scripts for multi-device scenarios, e.g., to send and receive a text message from one device to another. DeviceAnywhere Studio supports multi-device test cases for up to ten devices.

This chapter contains step-by-step instructions for [creating test cases](#) and the [test case properties dialog box](#). (After creating a test case, you must add commands to it—refer to [Working with Commands](#).)

7.1 Creating a Test Case

To create an automated test case:

- 1 Make one of the following selections:
 - Right-click your project test cases folder (or sub-folder) and select **New Test Case**.
 - While in your project, select **File > New > Test Case**.
- 2 Enter a name for your test case in the New Test Case dialog box.
- 3 Optional: Click **More Options** to display additional controls.
 - a From the **Devices** drop-down list, choose **Test case uses one device** to create a single-device test case. Choose **Test case uses multiple devices** to set up a multi-device test case (e.g., to test sending and receiving a message from one device to another).
 - b In a multi-device test case, choose the **Number of Devices** (up to 10) from the drop-down list.

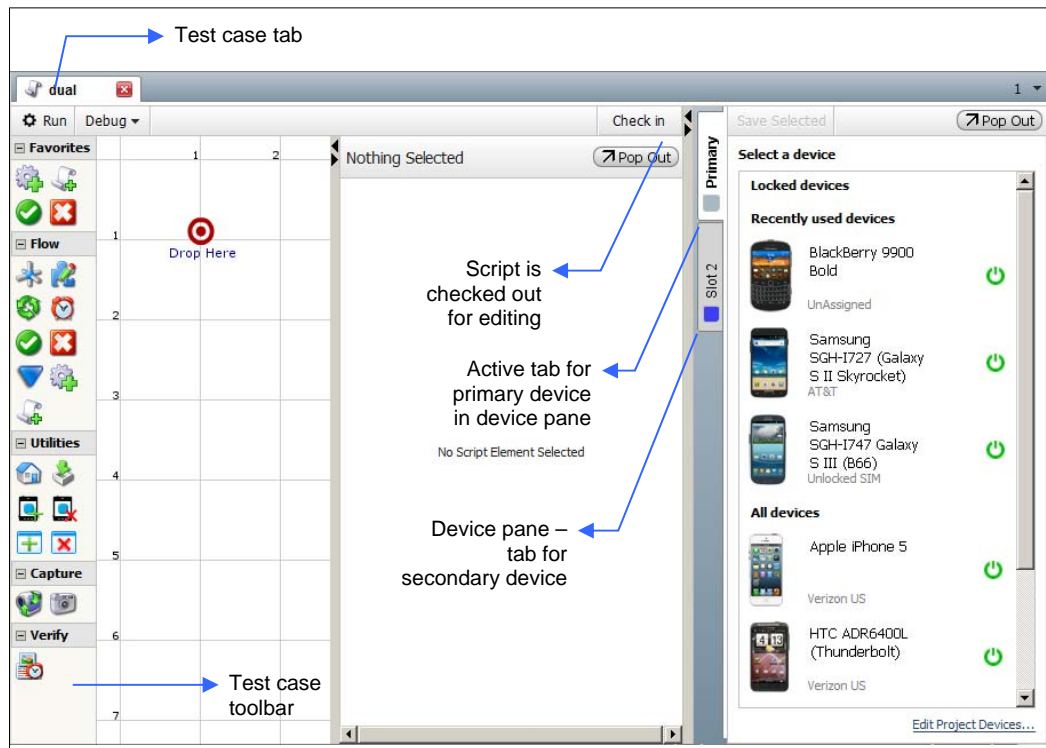


NOTE If you do not click **More Options**, the default is a single-device test case.

- Click **Create**. This [checks out the test case](#) and opens it up in the workspace. The device pane displays all project devices. In a multi-device scenario, the device pane displays a tab for each slot. For example, in a test case with a two-device scenario, the device pane contains a tab each for the primary and secondary device. Each tab lists all project devices.


To resize panes, hover over the edge and drag the sizing handle that appears.

Click **Edit Project Devices** below the device list to edit [project devices](#) in project properties.



- Select the device you wish to interact with while defining the test case and click **Save Selected**. (Repeat this step for each device slot.) You can change your selection at any time by clicking **Change Device** above the device pane.
- Right-click the device in the device pane and select **Acquire Device** to interact with it live.
- Drag commands from the toolbar onto the script canvas to create your script. Arrows connect the commands, indicating the order in which they will be performed.

To call previously defined actions:

- Drag the Execute Action command  onto the script canvas and use it to select an action.
- Drag and drop the action from the project directory onto the script canvas. This automatically inserts the action into the script in an Execute Action command.

NOTE To call a state, you can use the Wait Event command or you can drag and drop the state onto the script canvas.

See the chapter on [Working with Commands](#) for detailed information on using commands.

- [Check in your test case](#) and release the device when done editing.

Check [Examples](#) for sample test case scripts.

See [Managing Scripts](#) for information on copying, checking in, and deleting test cases.

7.2 Test Case Properties Dialog Box

The test case properties dialog box displays test case metadata and allows you to specify script parameters and device slot names.

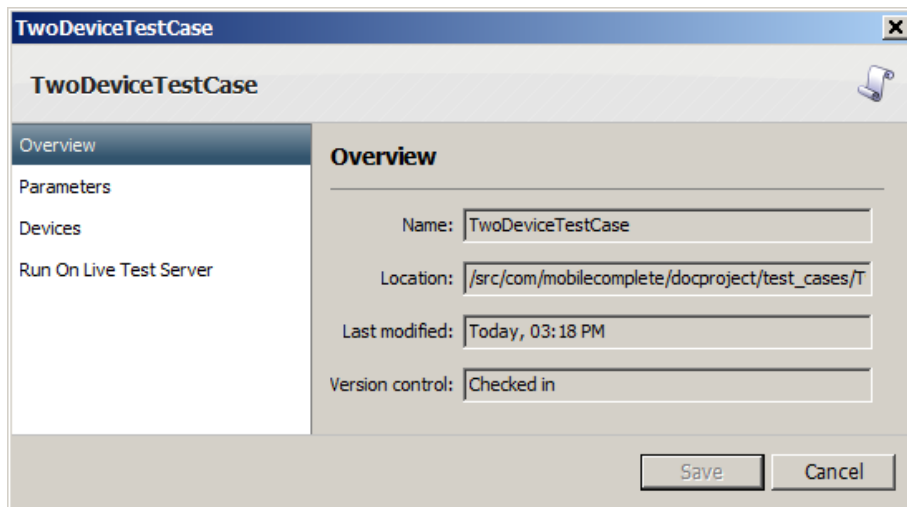
To open test case properties, right-click the test case in the project directory and click **Properties**. The test case must be checked out to edit properties. Be sure to **Save** any changes.

7.2.1 Overview Tab

The **Overview** tab contains non-editable fields for test case **Name** and **Location** within the project directory in the local file system, the date **Last modified**, and whether checked in/out of **Version control**.

The **Last modified** field is updated when a script is checked in or out or when properties are saved. The **Version control** field is updated when you check in/out the test case script. Click **Save** or **Cancel** to close the dialog box.

Figure 7-1 Test Case Properties – Overview



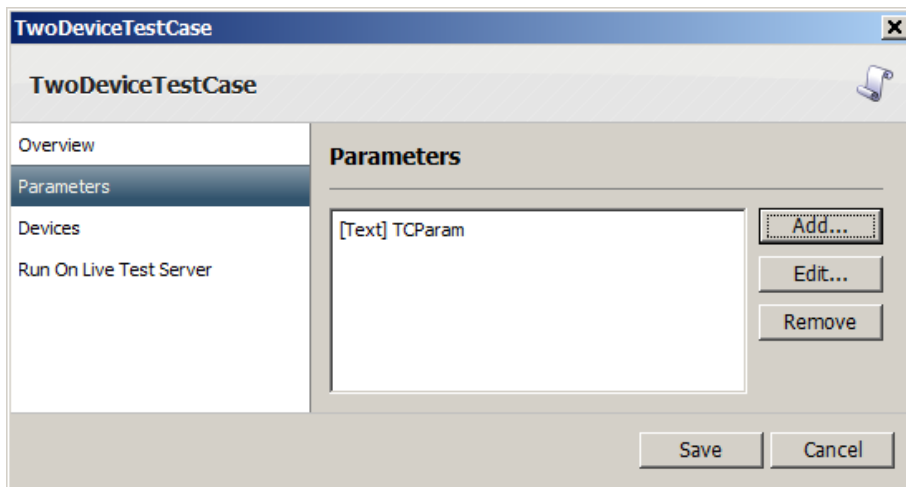
7.2.2 Parameters Tab

Defining script [parameters](#) allows you to dynamically enter data into test case devices at the start of a script run, i.e., specify a different data value for each script run. You can also create loops and branches based on the value of a parameter.

NOTE Test case parameters are not available for use in constituent actions.

The **Parameters** tab of the test case properties dialog box displays a list of existing parameters. You can **Add** or **Remove** script parameters and **Edit** parameter values.

Figure 7-2 Test Case Properties – Parameters

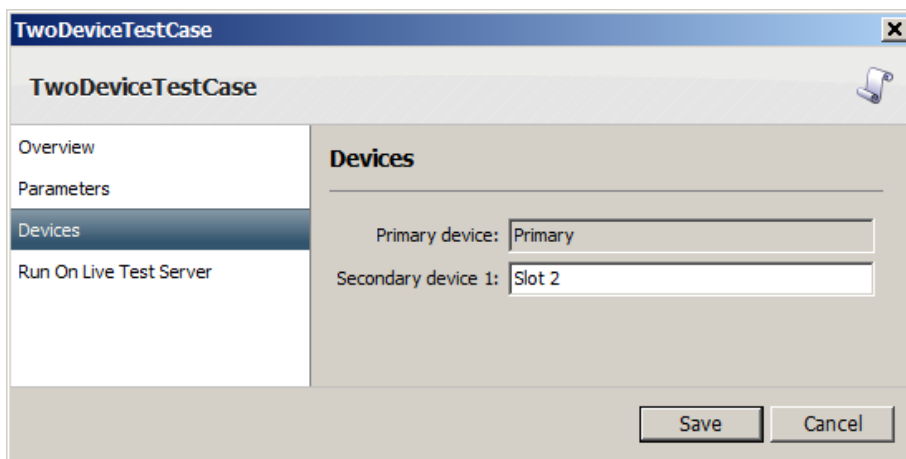


Refer to the chapter on [Working with Commands](#) for detailed information on using parameters—defining them, calling them from commands, and specifying values.

7.2.3 Devices Tab

If your test case contains multiple slots (e.g., for device-to-device interaction), the **Devices** tab is displayed in test case properties. You can customize the name of additional slots in this tab.

NOTE Be sure to use the same slot name when using the DeviceAnywhere utility to schedule the test case from the command line.



7.2.4 Run On Server Tab

Settings in the **Run On Server** tab enable the DeviceAnywhere command-line utility or any other third-party scheduler to run the test case on a Live Test server. You can specify the Live Test server, device(s) to run the test case on, and addresses to send [execution summaries](#) to in this tab.

NOTES Contact your system administrator to set up a LiveTest server for scheduled runs.

Ensure that you make default device selections for each test case slot in test case properties. You can overwrite device selections when running the command-line scheduling utility.

Refer to [Test Cases](#) in [Scheduling Script Runs](#) for a step-by-step description of how to enable a test case to be scheduled.

Figure 7-3 Test Case Properties—Run On Server—Single-Device Test Case

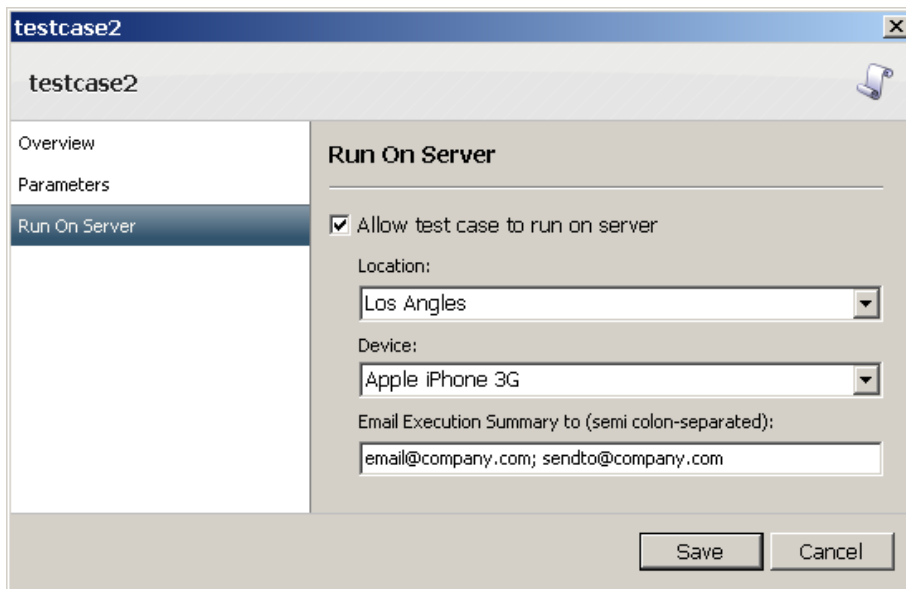
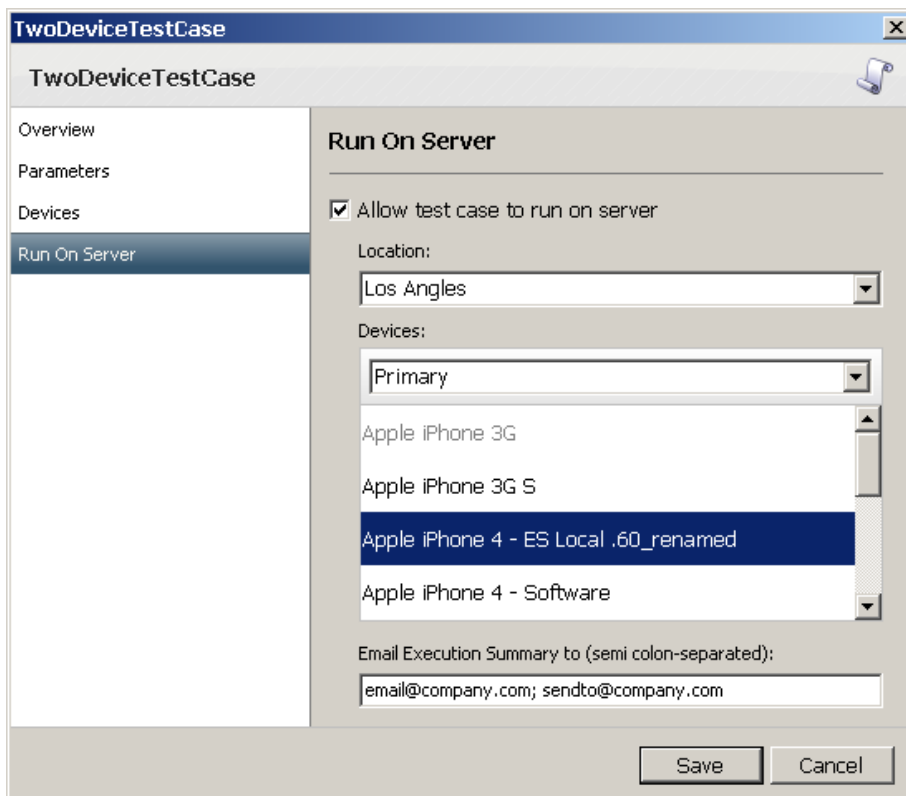


Figure 7-4 Test Case Properties—Run On Server—Multi-Device Test Case



8 Automated Test Cycles

In DAE Automation, a test cycle defines a test plan in terms of set of test cases to be executed on selected project devices. Test cycles consist of a group of test cases assigned to run on all or a subset of project devices so that users can focus testing on particular functional areas or test targets.

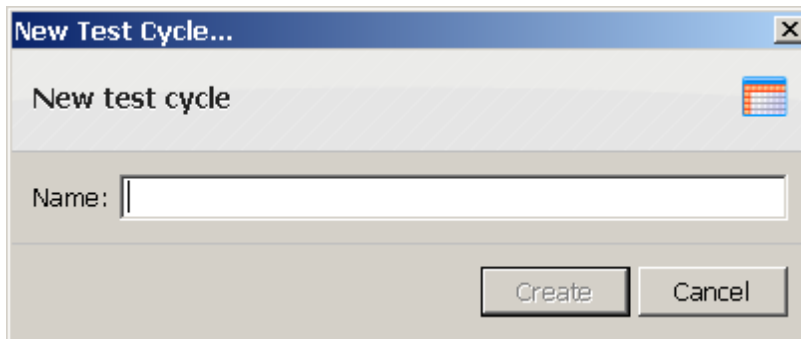
You can create and run automated test cycles consisting of automated test cases from the [Automation](#) view. This chapter contains step-by-step instructions for [creating automated test cycles](#) and [managing them](#). (Executing automated test cycles is covered in [Executing Scripts and Viewing Results](#).)

NOTE You can create test cycles consisting of *both* manual and automated test cases in the [Test Case Editor](#) interface of [Test Case Manager](#). As you execute these test cycles in the [Test Case Runtime](#) interface of Test Case Manager, you can track progress across devices and resume testing from where you last left off. See [Manual and Partially Automated Test Cases and Test Cycles](#) for information on creating and running test cycles in the Test Case Manager view.

8.1 Creating a Test Cycle

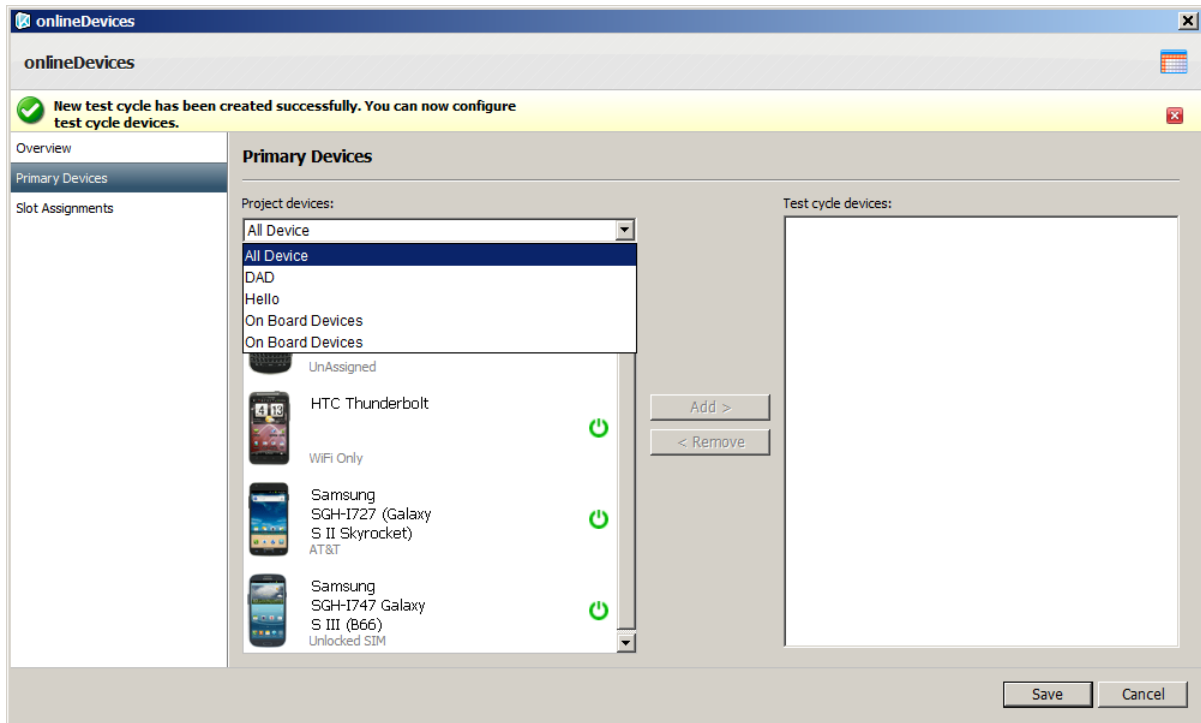
To create an automated test case:

- 1 Make one of the following selections in the [Automation](#) view:
 - Right-click your project test cycles folder (or sub-folder) and select **New Test Cycle**.
 - While in your project, select **File > New > Test Cycle**.
- 2 Enter a name for your test cycle in the New test cycle dialog box.

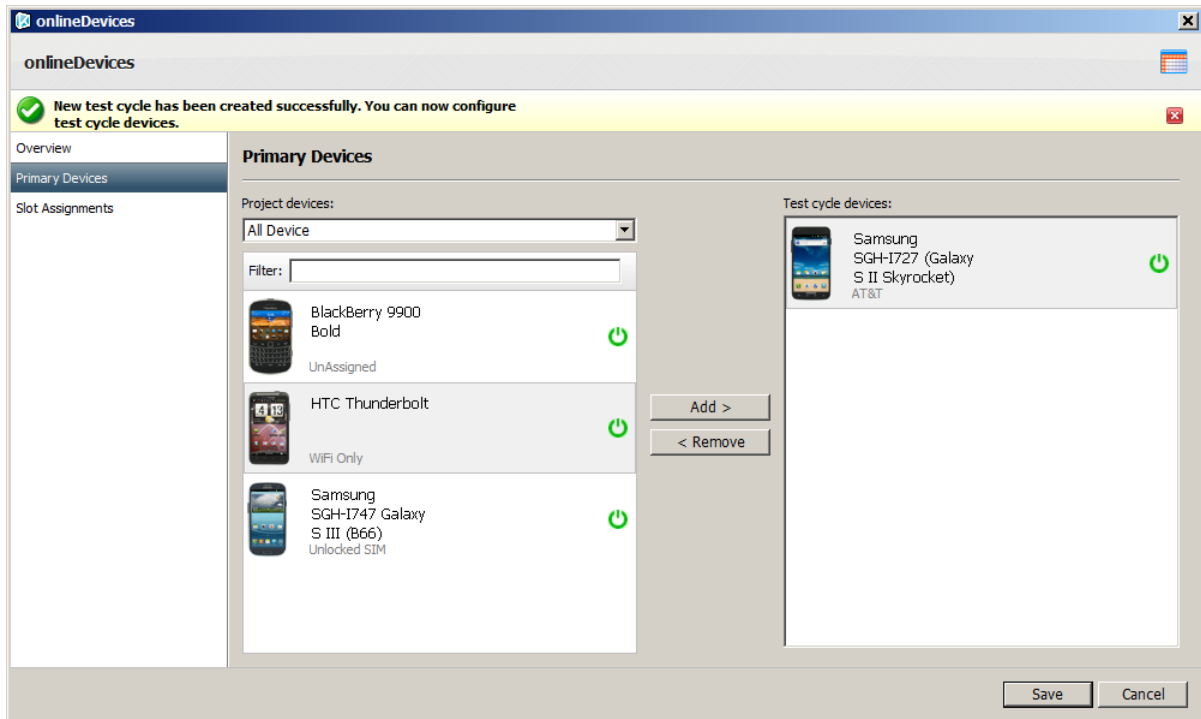


- 3 Click **Create**. This checks out the test cycle and opens up a tab for it in the Automation workspace. This also opens up the **Devices** tab of the test cycle properties dialog box where you must choose from among project devices.

- Select a package from the list of device packages in your test environment. When you choose a package, its available devices are listed in the left pane.



- Choose the device you wish to interact with in the left pane and click **Add**. You will only be able to add project devices. This displays the device in the **Test cycle devices** pane on the right.

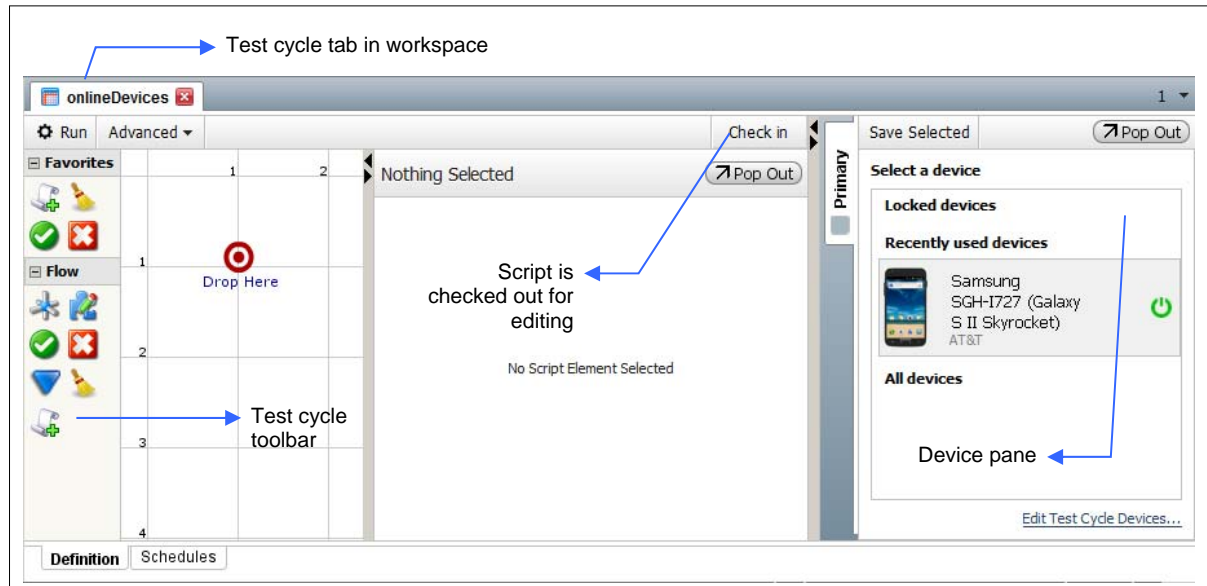


Repeat this step for all devices you wish to add to the test cycle. You can add or change test cycle devices at any time in the [Primary Devices tab of test cycle properties](#).


6 Save your changes.

The tab for the new test cycle displays the blank script canvas, the test cycle toolbar on the left, and the device pane on the right for test cycle devices.

You can resize a pane by hovering over the edge and dragging the sizing handle that appears. Click **Edit Project Devices** below the device list to edit project devices in the [Project Properties](#) dialog box.



- 7 Select a test cycle device from the right pane and click **Save Selected**.
- 8 Right-click the device in the device pane and select **Acquire Device** to interact with it live.
- 9 Drag commands from the toolbar onto the script canvas to create your script. Arrows connect the commands, indicating the order in which they will be performed.

To call a previously defined test case, drag the Execute Test Case command  onto the script canvas and use it to select a test case.

See the chapter on [Working with Commands](#) for detailed information on using commands.

NOTE The test cycle toolbar contains only a few commands for script logic, script termination, and for executing test cases and cleanup scripts.

- 10 As you add commands to the script, right-click and select **Run from here**. This allows you to verify recently added commands and moves the device into the correct state for defining the next command.
- 11 [Check in your test cycle](#) and release the device when done editing.

See [Examples](#) for a sample test cycle.

See [Managing Scripts](#) for information on copying, checking in, and deleting test cycles.

8.2 Test Cycle Properties Dialog Box

The test cycle properties dialog box displays test cycle metadata and allows you to choose devices.

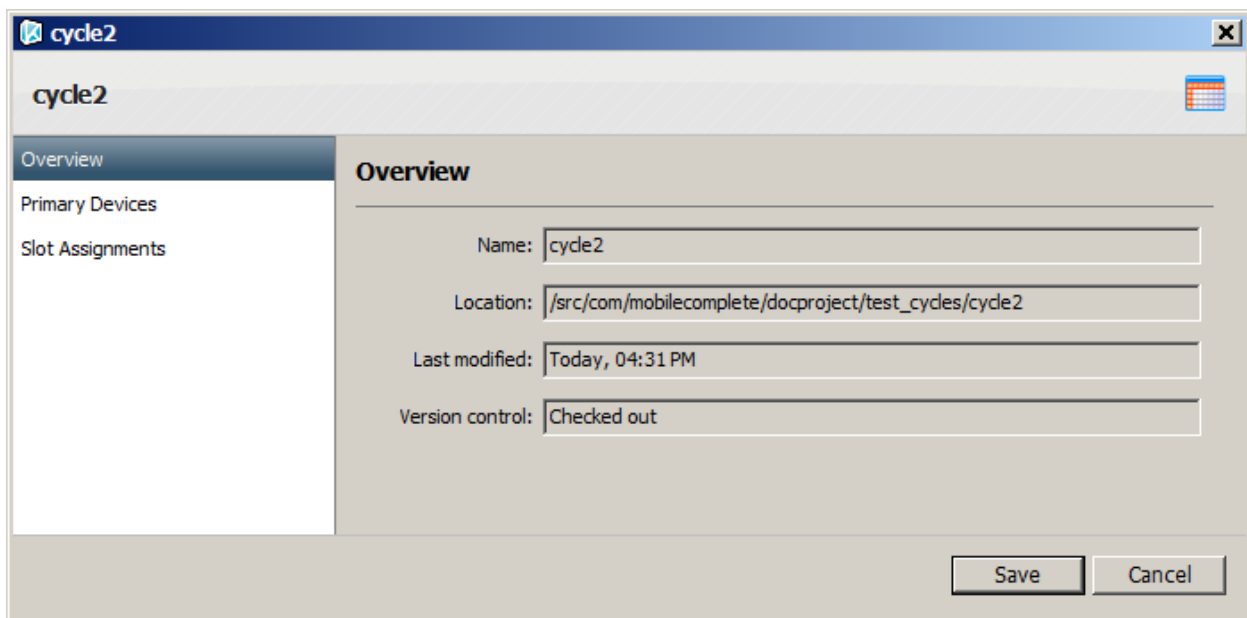
To open the test cycle properties dialog box, right-click the test cycle in the project directory and click **Properties**. The test cycle must be checked out in order to edit properties. Be sure to **Save** any changes.

8.2.1 Overview Tab

The **Overview** tab contains non-editable fields for test cycle **Name** and **Location** within the project directory in the local file system, the date **Last modified**, and whether checked in/out of **Version control**.

The **Last modified** field is updated when the script is checked in or out. The **Version control** field is updated when you check in/out the test cycle script. Click **Save** or **Cancel** to close the dialog box.

Figure 8-1 Test Cycle Properties – Overview

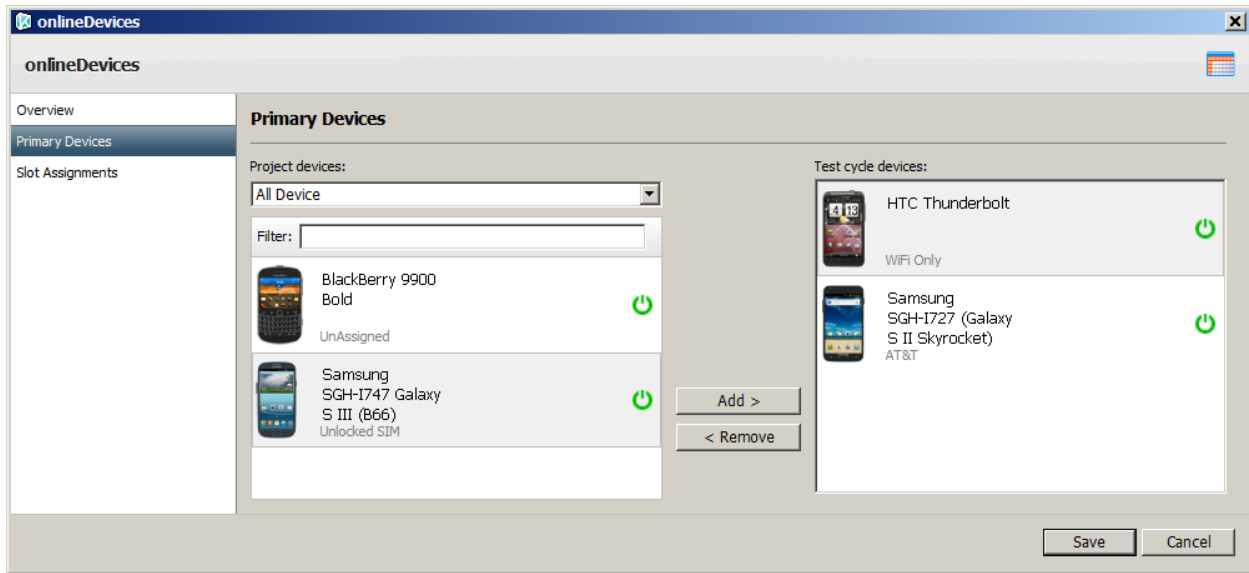


8.2.2 Primary Devices Tab

The **Primary Devices** tab of the test cycle properties dialog box displays the list of test cycle devices. You can **Add** or **Remove** devices from here.

If [scheduling the test cycle to run unattended](#) (from the DeviceAnywhere Studio UI or using the DeviceAnywhere command-line utility), it is executed consecutively on each of these devices. If your test cycle contains a multi-device test case, you must choose devices for each additional slot in the [Slot Assignments Tab](#) of test cycle properties.

Figure 8-2 Test Cycle Properties – Devices

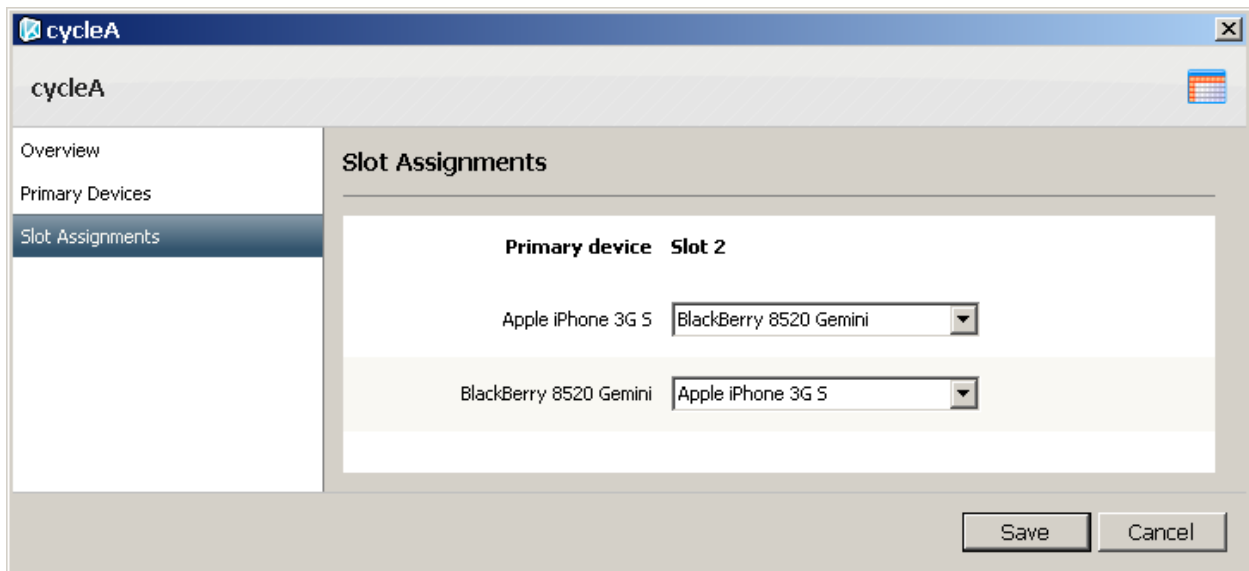


Refer to [Creating a Test Cycle](#) for instructions on adding devices while creating a test cycle.

8.2.3 Slot Assignments Tab

If your test cycle contains a multi-device test case, and you wish to schedule the test cycle to run unattended, you must assign devices for additional slots in this tab. For each additional slot, choose a device from the drop-down list provided next to each primary device.

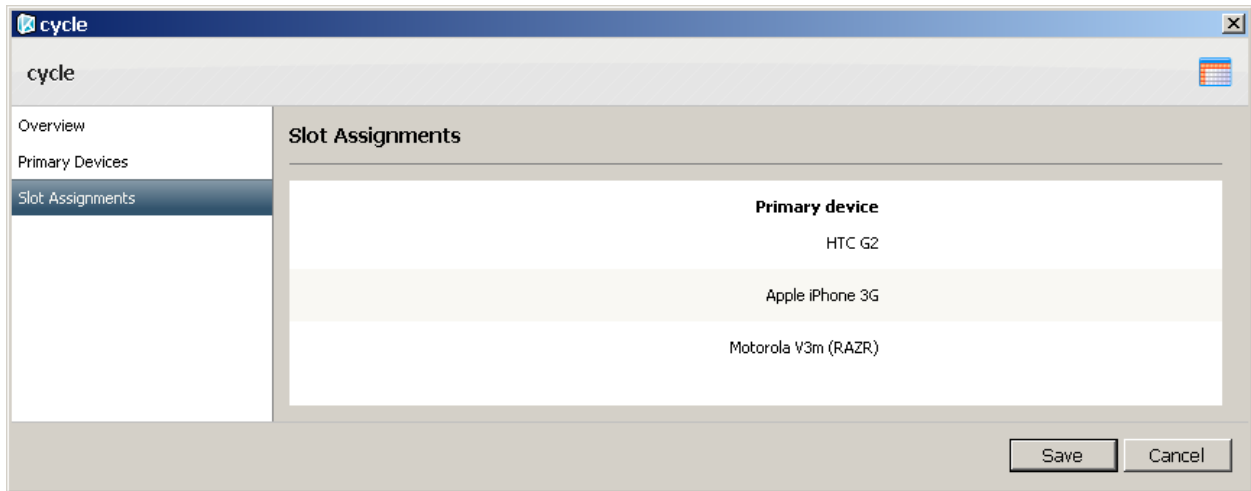
Figure 8-3 Test Cycle Properties — Slot Assignments



Refer to [Test Cycles](#) in [Scheduling Script Runs](#) for a step-by-step description of how to schedule a test cycle.

If your test cycle consists only of single-device test cases, this slot simply displays primary test cycle devices.

Figure 8-4 Test Cycle Properties—Slot Assignments (No Additional Slots)



9 Managing Scripts and Schedules

This chapter contains information on how to manage your script and scheduling assets, including [opening and checking out a script for editing](#), [checking in and closing scripts](#), [merging and splitting implementations](#) (actions and states only), [rolling back scripts](#) to earlier versions, [deleting actions and implementations](#), [searching for references](#), and [other functions](#).

9.1 Opening and Checking Out an Asset

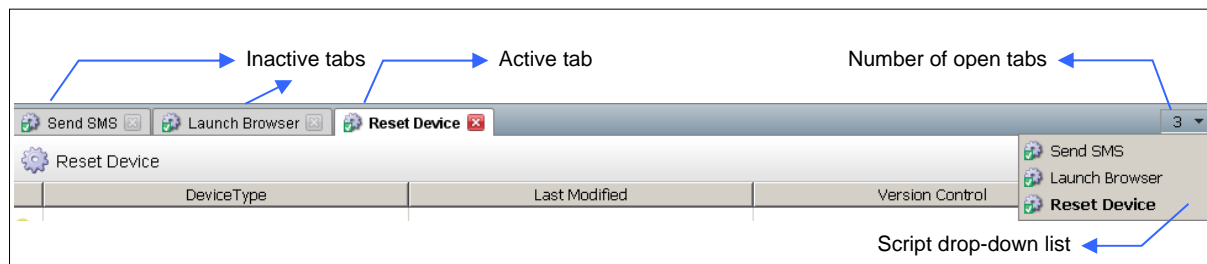
This section describes all available methods for opening and checking out a script, implementation, or DAE Automation schedule for editing.

1 Open a script:

- Select and double-click the script in the project directory.
- Select the script in the project directory, right-click, and select **Open**.

This will open a tab in the workspace. Script tabs display the device list. (See also [Opening a Project](#).)

You may open multiple test assets. The workspace displays tabs for each open asset and lists the number of open tabs. Use the drop-down list next to this number to tab through your assets.



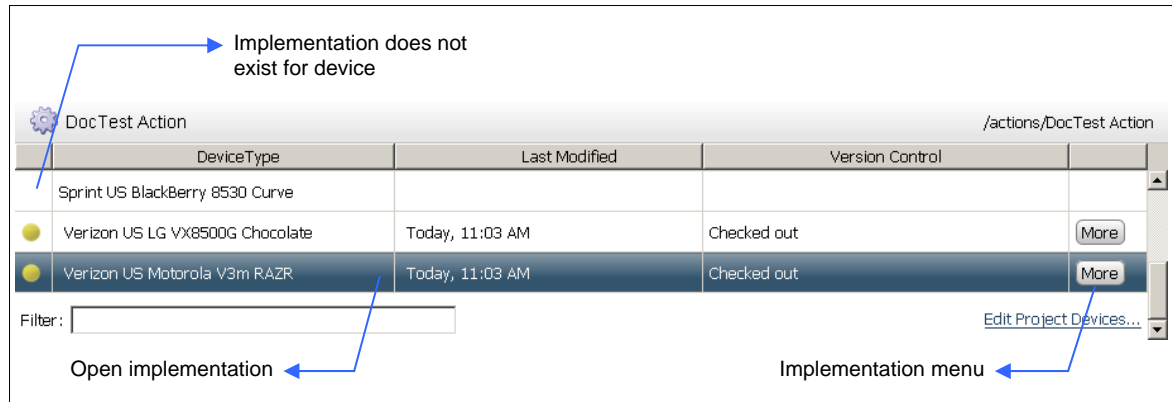
- #### 2 In actions and states, you must also open an implementation. To open an existing implementation, select a device in the device list. Devices with implementations are listed with a yellow icon.

This opens the implementation below the device list.

NOTE If you do not see the device, hover over the right edge of the window and drag the sizing handle that appears.

If you select a device without an implementation, you are given the option to create one.

You can only have one implementation of an action or state open at any time. To switch between implementations, select the appropriate device from the device list. The figure below shows devices with and without implementations in an action device list.



- Check out the script or schedule from the version control system for editing:
 - Right-click the asset in the project directory and click **Check Out**. For actions and states, this checks out all available implementations.
 - Click **Check Out** above the canvas to check out an open asset.
 - For action and state implementations, you can also right-click the device in the device list or click **More**. Then **Check Out** the open implementation.

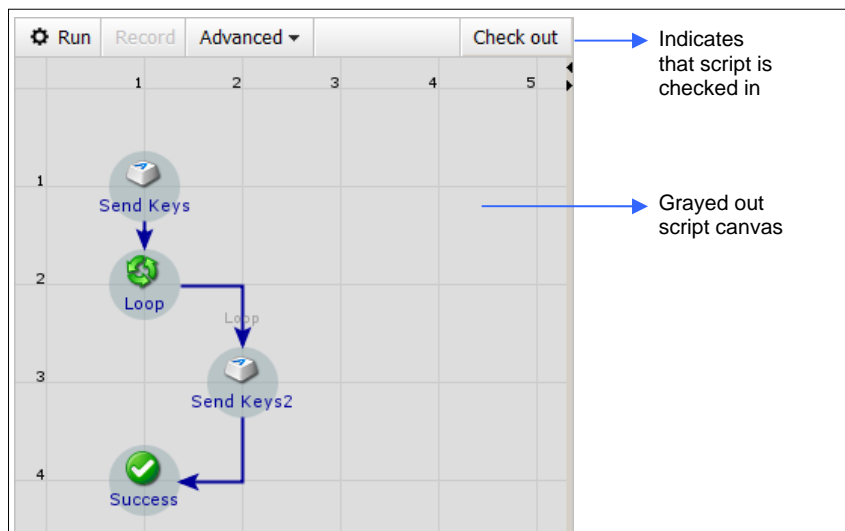
This displays the command toolbar, enabling you to edit the script.


NOTE Ensure that you acquire the device so you can interact with it while scripting.

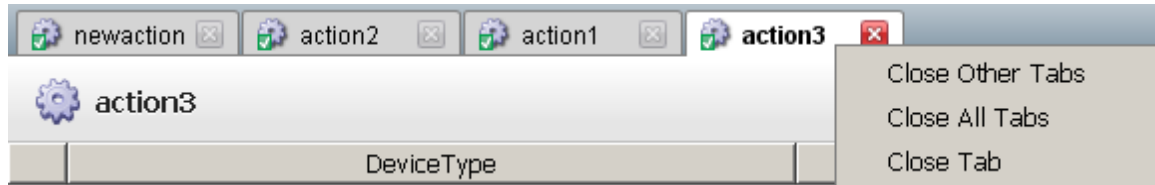
9.2 Checking In and Closing an Asset

This section describes all available methods for closing and checking in a script or schedule.

- Check the asset in:
 - Click **Check In** above the script canvas to check in an open schedule, script, or implementation.
 - For action and state implementations, you can also right-click the device in the device list or click **More**. Then, **Check In** the implementation. This grays out the script canvas and hides the command toolbar.



- Right-click the asset in the project directory and click **Check In**. For actions and states, this checks in all available implementations.
- 2 Close the asset:
 - Click the close button  in the script tab.
 - Right-click the tab, and from the drop-down list that appears, select **Close Tab** to close the tab, **Close All Tabs** to close all open tabs, or **Close Other Tabs** to close other open tabs.



NOTE Be sure to release any acquired devices if you are no longer using them.

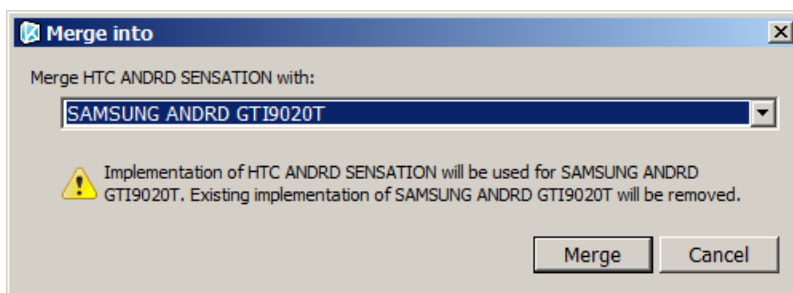
9.3 Merging and Splitting Implementations

In DeviceAnywhere Studio, you can merge action and state implementations when the implementation is constant over multiple devices. For instance you would merge implementations when navigation on devices is identical and any [reference points](#) being used are audio based. You can also split a previously merged implementation when device interfaces start to differ.

To merge implementations:

- 1 From the device list of an open action or state, select the device whose implementation you would like to merge with others. This implementation will override the implementation of any merge target(s) you select.

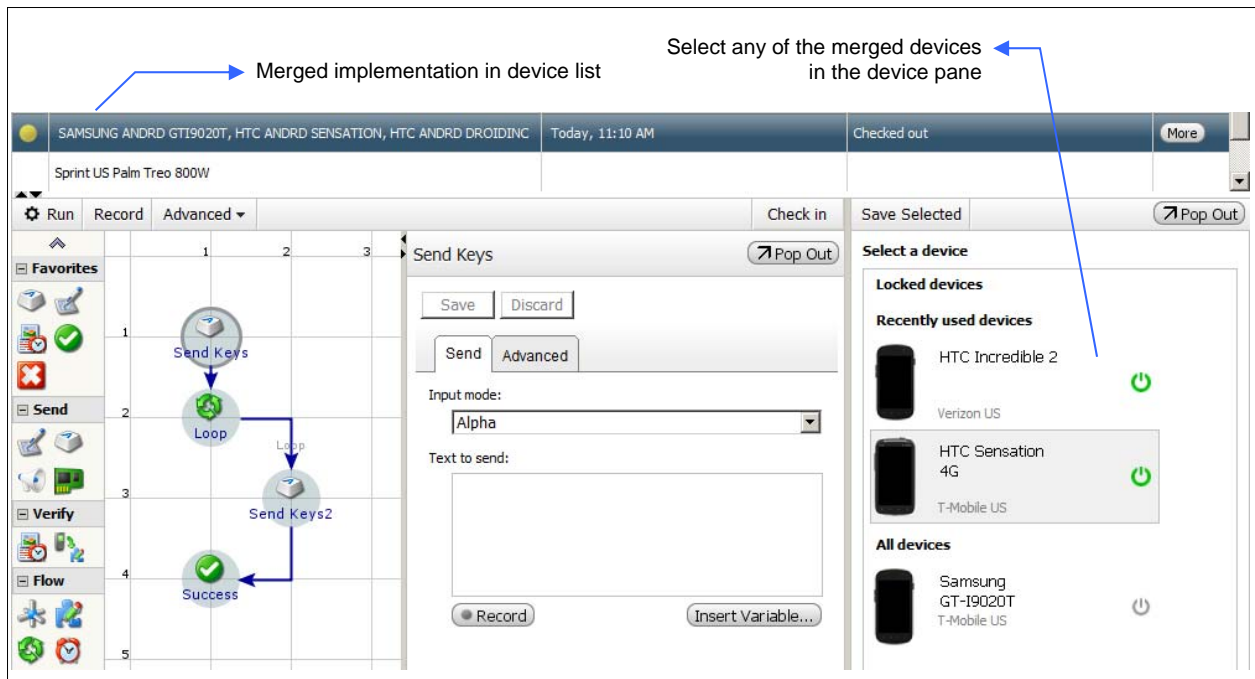
You can merge any number of implementations but you can only select one merge target at a time.
- 2 Right-click the device in the device list or click **More**. Next, select **Merge with**.
- 3 Select a target device (merge target) from the drop-down list provided and click **Merge**.



The implementation you selected in step 1 above is now the effective implementation for this device.

The merged implementation appears in the device list with the names of merged devices. The device pane now allows you to select and interact with any of the merged devices—to check that your implementation works for each of them.

Figure 9-1 Merged Action Implementations

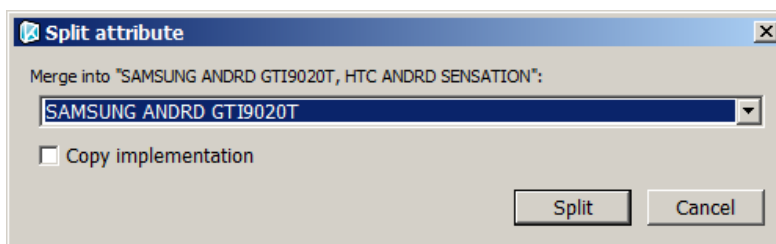


At times, you might need to split a merged implementation (as the one created above) because device interfaces have diverged. To split a merged implementation:

- 1 Select the merged implementation from the device list of an open action or state.
- 2 Right-click the implementation or click **More**. Next, select **Split away**.
- 3 From the drop-down list provided, select the device for which you would like to separate out an implementation and click **Split**.

NOTE The drop-down list contains the list of merge targets, i.e., devices merged into the original implementation.

Optionally, check **Copy implementation** to copy and modify the merged implementation for the device.



The device for which you separated out an implementation now appears in the device list.

- 4 You can restore the most recent implementation prior to being merged—select the device from the device list and click **Implement**. Then select **More > Show History** to select a script version to [roll back](#) to.

9.4 Rolling Back to Previous Versions

DeviceAnywhere version control tracks and saves incremental changes to schedules, scripts, and implementations when they are checked in, enabling you to roll back an asset to an earlier version:

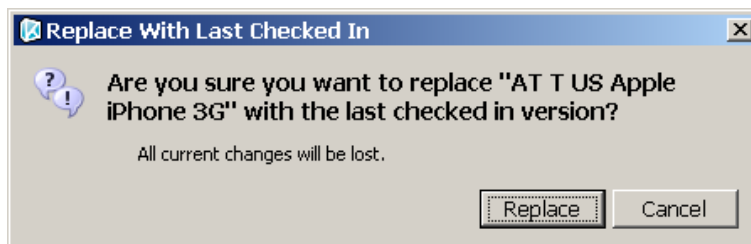
- 1 Open the script you wish to roll back to a previous version.
- 2 If working with an action or state, select the implementation you wish to roll back.
- 3 Check the asset out.
- 4 Roll back your asset:

To roll back to the most recently checked in version:

- a For schedules, test cases, and test cycles, right-click the script in the project directory and select **Replace With Last Checked In**. For action and state implementations, click **More** next to the implementation in the device list and select **Replace With Last Checked In**.

CAUTION Rolling back cannot be undone. Any changes not checked in are replaced by contents of the earlier version.

- b Click **Replace** in the dialog box that appears.

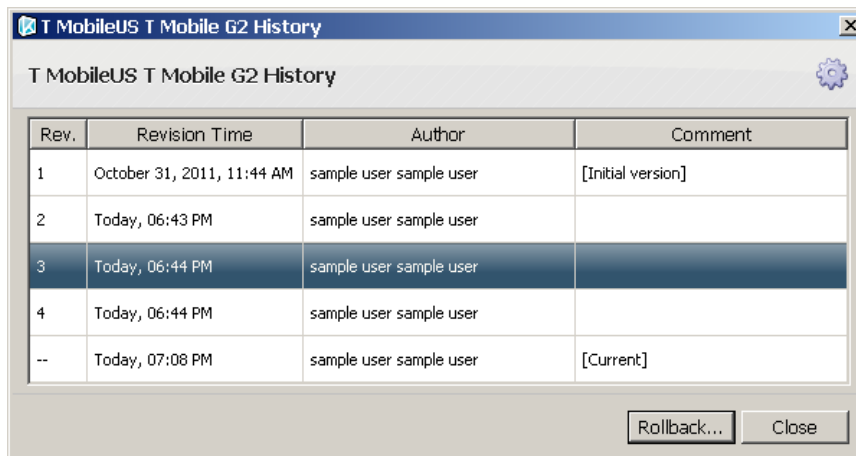


This displays the command sequence or settings from the earlier version on the canvas.

To select the version you wish to roll back to:

- a For schedules, test cases, and test cycles, right-click the asset in the project directory and select **Show History**. For action and state implementations, click **More** next to the implementation in the device list and select **Show History**.

This opens up a revision history dialog box where asset versions are listed by revision number, with date checked in, author, and comments.

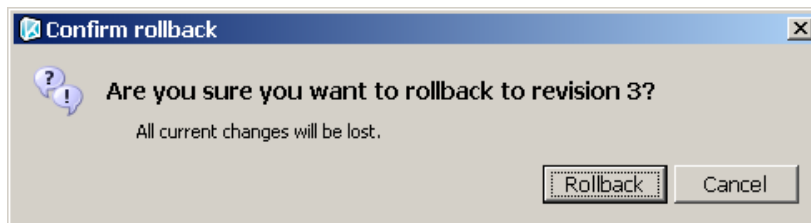


NOTE The most recent checked out version is listed as **Current** with the time checked out—it has no revision number. The revision number and time are updated when an asset is checked back in.

- b Select the version you would like to roll back to and click **Roll back**.

CAUTION Rolling back cannot be undone. Any changes not checked in are replaced by contents of the earlier version.

- c Click **Roll back** to confirm your selection.



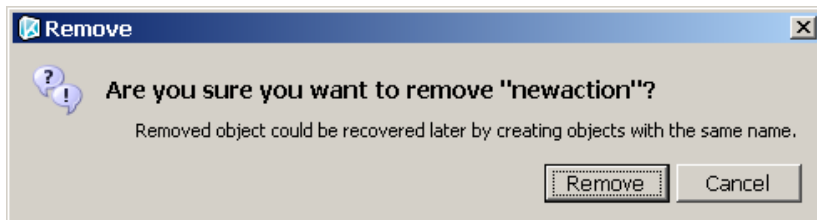
This displays the command sequence or settings from the earlier version on the canvas and updates the asset version number.

- 5 Be sure to check the asset back in to save changes.

9.5 Removing Assets

To remove a script:

- 1 Check out the asset. In the case of actions or states, check out any implementation.
- 2 Right-click the asset in the project directory and select **Remove**.
- 3 Click **Remove** in the dialog box that appears to confirm your selection.

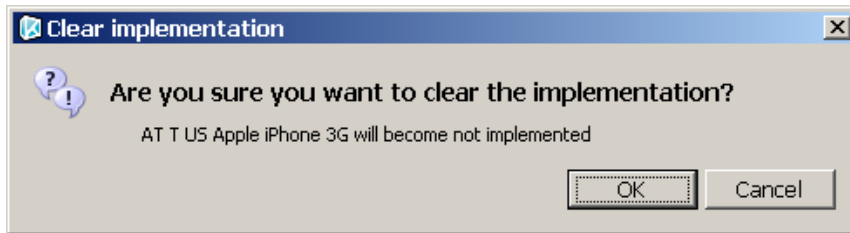


NOTE Removing an asset clears it from the project directory. However, you can recover it by creating a new asset with the same name. You can then restore the asset to older versions in the **Show History** dialog box.

9.6 Removing Implementations

To remove an implementation:

- 1 [Open the action or state](#) and check out the implementation.
- 2 Click **More** next to the implementation in the device list and select **Clear implementation**.
- 3 Click **OK** in the dialog box that appears to confirm your selection.



The device appears in the device list without a yellow icon, indicating that it does not have an implementation.

NOTE You can restore a cleared implementation by selecting the device from the action device list and clicking **Implement**. You can then access and restore the script to older versions in the **Show History** dialog box.

9.7 Searching for References

You can find all other scripts or schedules that make a reference to (call) a selected script.

- 1 Right-click your script in the project directory and select **Find References**.

A status bar above the project list displays the outcome of the search and the number of references found, if any.



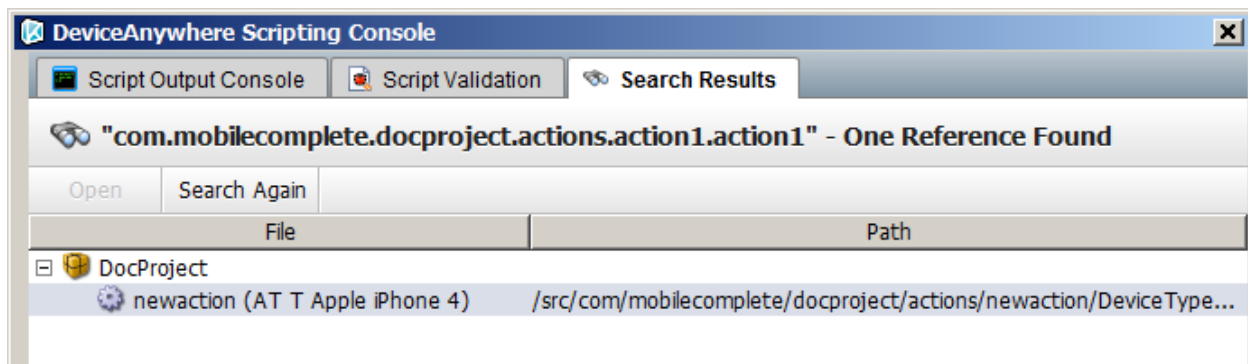
In the top-right corner of Studio, the search results icon displays the number of references found.



- 2 To open up search results in the DeviceAnywhere Scripting Console Window:
 - Click **View Search Results** in the status bar or
 - Click the search results icon at the top-left corner.

The **Search Results** tab of the DeviceAnywhere Scripting Console window displays the results of your search. The results of the most recent search persist until the next search.

Figure 9-2 Results of a Search for References




Double-click a search result to open up the script or schedule containing it (the asset must not be checked out by another user). You can also select the reference and click **Open**.

9.8 Other Functions

The table below describes script and schedule management commands available by right-clicking a script in the project directory:

Table 9-1 Script Management Commands

Command	Description
Add Comment	<p>Allows you to add notes for the script or schedule version (must be checked out). For actions and states, this command is accessible from an implementation More menu.</p>  <p>Notes are displayed in the Comments column of the revision history dialog box. Click Edit Comment to change a note before checking the script or implementation back in.</p>
Cut "<script>"	Cuts a script for pasting at another location, e.g., in the corresponding folder of another project. Not available for schedules.
Copy "<script>"	Copies a script for pasting at another location. Not available for schedules.
Paste	Pastes a copied script at the selected location. Not available for schedules.
Paste with Dependencies	Pastes a copied script containing calls to other scripts from dependent projects at the selected location. Not available for schedules.
Rename	<p>Allows you to specify a new name for the script. Not available for schedules.</p> <p>NOTE Version history is reset if you rename a script.</p>
Edit Variables	<p>Accessible from the Advanced or Debug option above an action implementation, test case, or test cycle script canvas. Also accessible from an action implementation More menu.</p> <p>Allows you to create and manage script variables. This is discussed in Variables in Working with Commands.</p>
Show History	Displays revision history from where you can choose a script/schedule version to roll back to.
Replace With Last Checked In	Replaces currently checked out script/schedule version with the most recently checked in version.
Properties	Opens action, state, test case, test cycle, or schedule properties.

10 Executing Scripts and Viewing Results

This chapter covers how to [validate your scripts before execution](#), [schedule script execution](#) (DAE Automation and DAE Monitoring only) [execute tests ad hoc](#), send [execution summaries](#), and [view results](#).

NOTE Provisioning a Mobile App Monitoring measurement takes place in MyKeynote.

10.1 Validation

Before executing scripts, you can run project- or script-level validation that flags errors such as:

- ◆ Circular references, e.g., two projects that depend on each other
 - ◆ Project build errors
 - ◆ References to entities that do not exist, e.g., calling an action that has been deleted
 - ◆ References to scripts that are not implemented, e.g., calling an action that is not implemented for the chosen device
 - ◆ Invalid reference images, e.g., using a reference image from one device for another device with a different screen size
 - ◆ Undefined reference images, e.g., an empty Wait Event command
 - ◆ Keys not mapped for a given key mode, e.g., sending a text string in Send Keys in the Numeric key mode
-

NOTE Validation is supported on single-device Java test scripts. Device-specific validations such as key mapping are not supported on Java test scripts.

Validation flags issues as:

- ◆ Errors—will cause a script to fail if not fixed.
- ◆ Warnings—might cause a script to fail and should also be fixed.

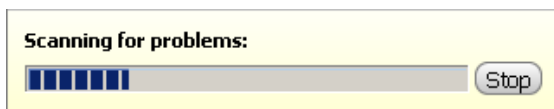
10.1.1 Running Validation

You can run a validation at the project- or script-level.

To run a *project-level validation*:

- 1 Load your project.
- 2 Right-click your project in the project list and choose **Validate All Files**. This checks all project files for errors. Select this option before building or publishing a project.

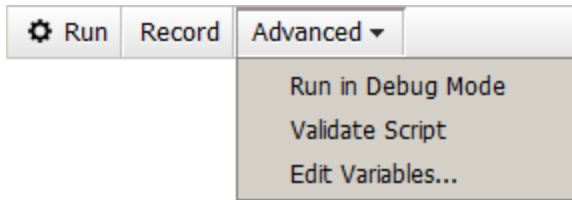
A status bar above the project list indicates the progress of validation. (You can **Stop** validation.)



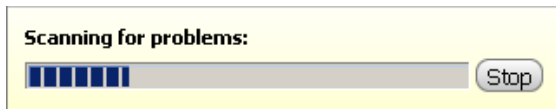
- 3 [View validation results](#).

To run a *script-level validation*:

- 1 Open your script.
- 2 From the **Advanced** drop-down list above the script canvas, select **Validate Script**.



A status bar displays the progress of script validation above the script canvas.



- 3 [View validation results.](#)

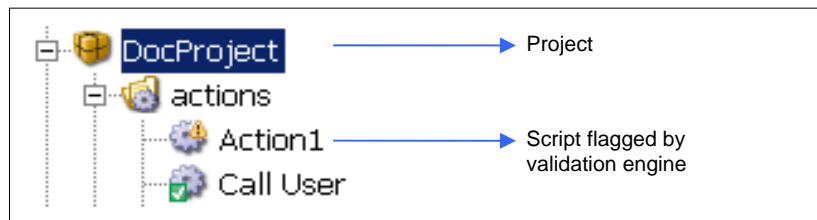
10.1.2 Viewing Validation Results

This section describes the different areas of the Studio interface that display [notifications after a project- or script-level validation](#). It also describes viewing validation results in the [DeviceAnywhere Scripting Console window](#).

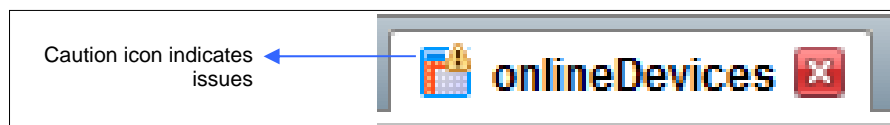
10.1.2.1 Validation Notifications

Running the validation engine places notifications in several areas of the Automation interface.

- ◆ *Project list:* Any script with an issue is flagged with a caution icon in the project directory.

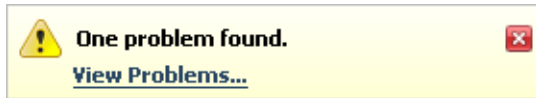


- ◆ *Script tab:* If a script is flagged by the validation engine, the open tab displays a caution icon.



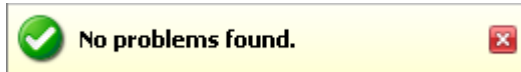
- ◆ *Status bar:* A status bar above the project list or script canvas (depending on the type of validation) displays the outcome of validation.

If the validation engine flags any issues, the status bar lists the number of issues and displays a link to view them.



Click **View Problems** to open a detailed list of issues in the [DeviceAnywhere Scripting Console window](#).

If there are no validation flags, the status bar displays a green check mark.



- ◆ *Output console icons:* The output console icons are located at the top-right corner of the program and enable you to access the [DeviceAnywhere Scripting Console window](#) to view results of a validation. The icons also display notifications each time you [run a script](#), [detect a state](#), [build a project](#), or [perform a search](#).



Hover over each icon to see a tooltip describing the output console you can see by clicking the icon.



Whenever you run validation, the output console and validation icons indicate the number of notifications to view.



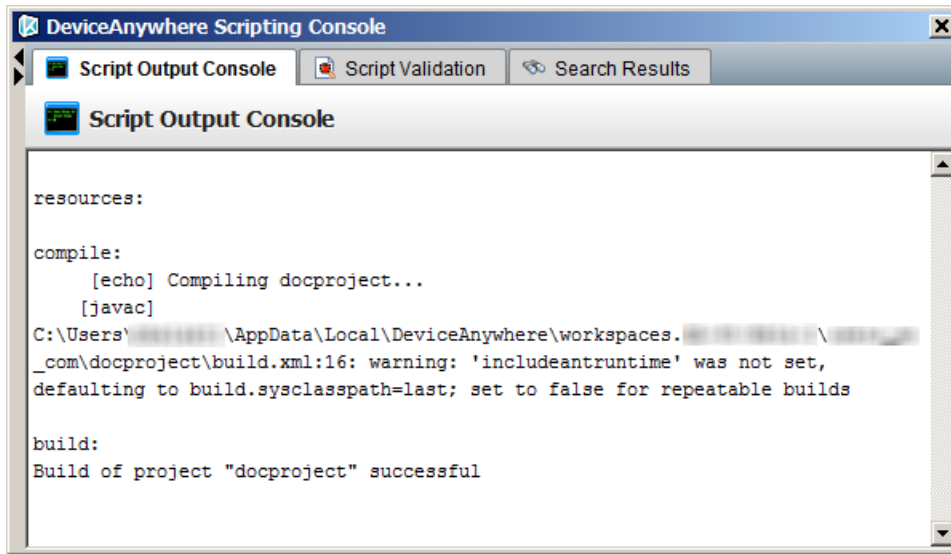
Click any of the icons to open the [DeviceAnywhere Scripting Console window](#) and view validation results.

10.1.2.2 DeviceAnywhere Scripting Console Window

The Scripting Console window contains three tabs to display the output of project builds, script validation, and searches. You can access this window by clicking in the [status bar](#) displayed after performing [validations](#) or searches. You can also open this window by clicking any of the [output console icons](#) at the top-right corner of the program.

The first tab displays raw output in a console after you build a project or run validation.

Figure 10-1 Script Output Console



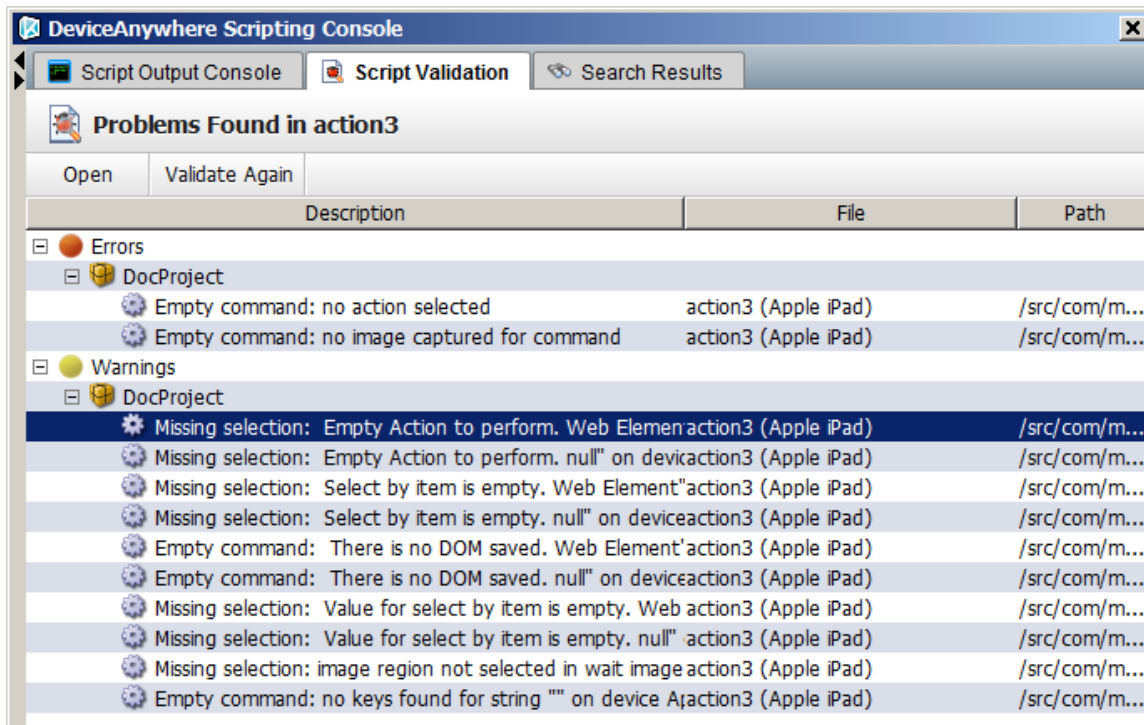
This information is useful for debugging scripts and persists until the next project build or validation.

The **Script Validation** tab displays the results of project- or script-level validation. Results from the most recent validation persist until the next run.

Issues are listed either as **Errors** or **Warnings** (see Figure 10-2 below). Errors will cause script failures if not fixed. Warnings might cause failures and should also be fixed.

Each issue is listed with a description, script/file, and script location within the local project directory.

Figure 10-2 Validation Results



Double-click an issue to open up the script containing it (the script must not be checked out by another user).

10.2 Scheduling Script Runs

Scheduling DAE Automation runs from DeviceAnywhere Studio can be implemented as follows:

NOTE Deploying DAE Monitoring test cases as monitors is discussed in the [DAE Monitoring Best Practice Workflow](#). Provisioning MAM measurements is discussed in [MyKeynote online help](#).

- ◆ [Test cycles](#)
 - Click the **Schedules** tab at the bottom of a test cycle script canvas.
 - Click the **Scheduling tab** of the Automation view to create test cycle schedules in a project.
- ◆ [Test cases](#)
 - Select the **Run On Server** tab of test case properties to enable a third-party scheduler to run the test case on a LiveTest server.

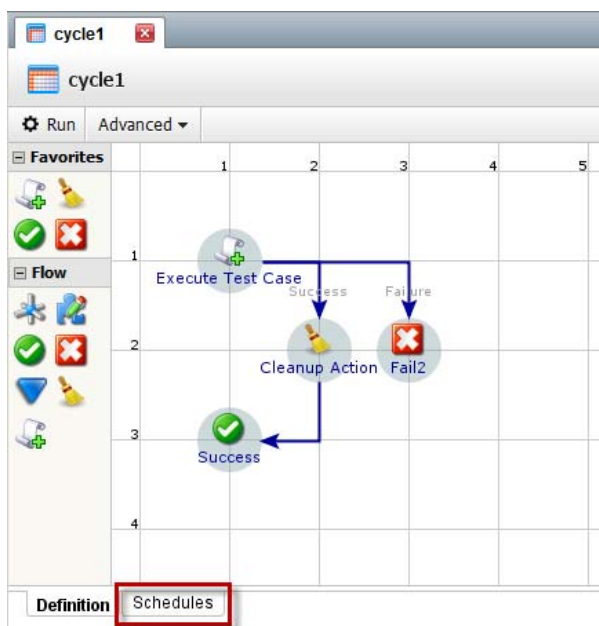
You can also use the DeviceAnywhere command-line utility to schedule test cycles and test cases.

10.2.1 Test Cycles

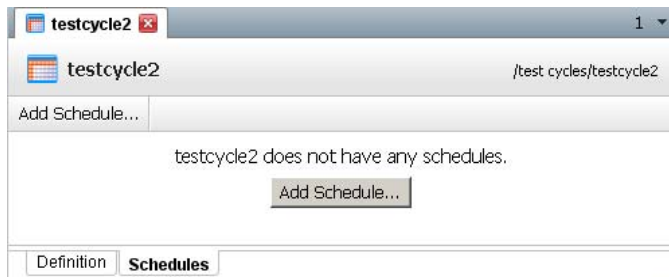
Test cycles can be scheduled in both the [Scripting](#) and the [Scheduling](#) tabs of the Automation view. The [scheduling form](#) for creating a schedule is the same in both tabs.

The **Scripting** tab displays all existing schedules for a selected test cycle; the **Scheduling** tab displays all existing schedules for all project test cycles, including ones created in the **Scripting** tab—you can [manage your schedules](#) in the **Scheduling** tab.

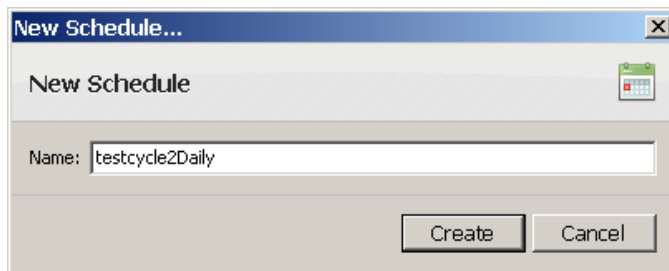
- ◆ To schedule a test cycle in the **Scripting** tab:
 - a Open and check out your test cycle.
 - b Select the **Schedules** tab at the bottom of the script canvas.



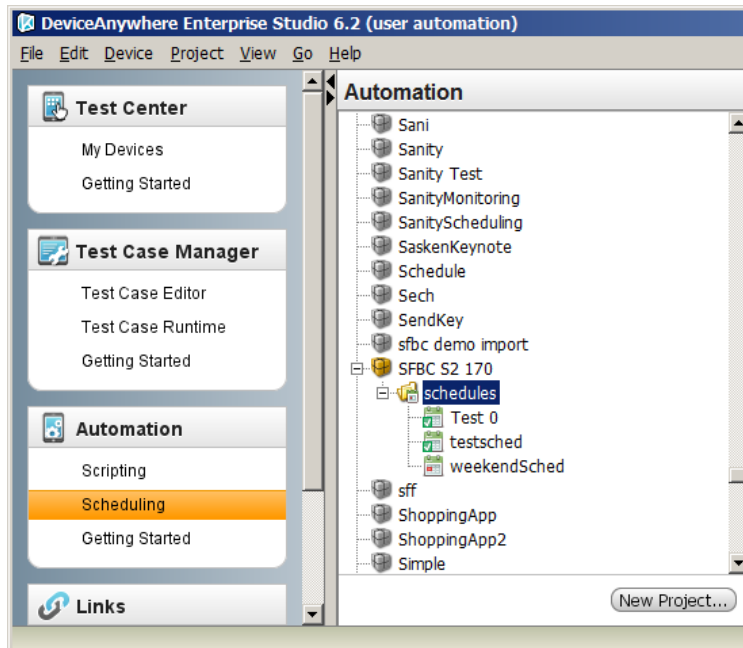
- c Click **Add Schedule**.



- d Enter a name for the schedule and click **Create**.



- e Fill out the [scheduling form](#) as described below.
- f Optionally, validate your schedule and check it in.
- g If your test cycle contains a multi-device test case, assign a device for each additional slot in the [Slot Assignments tab](#) of [test cycle properties](#).
- h Publish your project for your schedule to become effective (right-click the project in the project directory > **Publish Project**).
- ◆ To schedule a test cycle in the **Scheduling** tab:
- a Open your project. The schedules folder displays all existing schedules for project test cycles.



- b Right-click the schedules folder and select **New Schedule**.
- c Enter a name for the schedule and click **Create**.
- d Fill out the [scheduling form](#) as described below
- e Optionally, validate your schedule and check it in.
- f If your test cycle contains a multi-device test case, assign a device for each additional slot in the [Slot Assignments tab](#) of [test cycle properties](#).
- g Publish your project for your schedule to become effective (right-click the project in the project directory > **Publish Project**).

10.2.1.1 Scheduling Form

In either the **Scripting** and **Scheduling** tabs of the Automation view, you must fill out the scheduling form shown in the image below to set up a test cycle schedule.


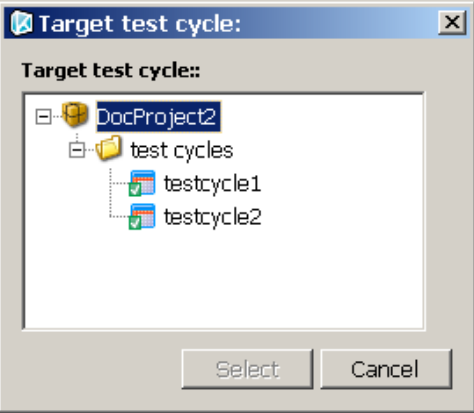


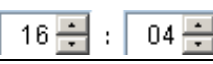


Figure 10-3 Scheduling Form

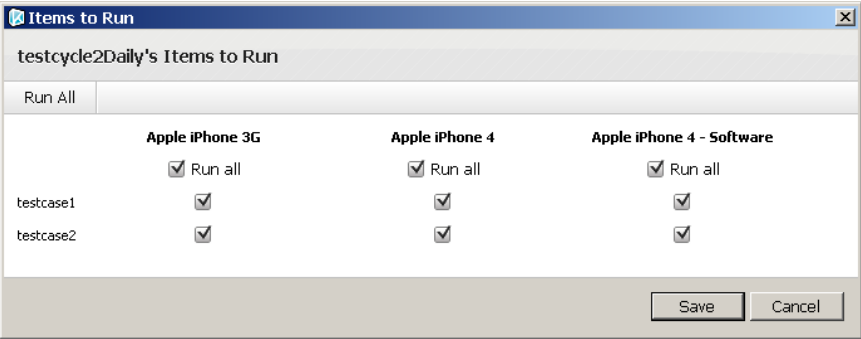
The screenshot shows the 'testcycle2Daily' scheduling form. At the top, there are buttons for 'Validate' and 'Check in'. The main section is titled 'testcycle2Daily's status is unknown'. It contains the following fields and options:

- Target test cycle:** A text box containing 'testcycle2' with a search icon and a close icon.
- Run schedule in:** A dropdown menu showing 'San Francisco'.
- Execution recurrence:**
 - Run once: Includes a date field 'Wed 11/02/2011' and a time field '10 : 49'.
 - Run recurrently: Includes a date field 'Wed 11/02/2011' and a time field '22 : 00'. Below this is a 'Frequency' section with checkboxes for Mon, Tue, Wed, Thu, Fri, Sat, and Sun, all of which are checked.
 - Run is triggered externally
- Description:** A text area containing the text: 'Schedule for com.team.plus.docproject2.test cycles.testcycle2. Runs daily at 10:00 p.m.'
- Advanced execution options:**
 - Run all test cases on all devices. Below this is a link 'Edit Items to Run...'
 - Maximum schedule execution time: A text box with '4' followed by 'hours'.
 - Try for up to: A text box with '0' followed by 'hours', another text box with '0' followed by 'minutes to acquire devices for a test cycle run.'
 - Email Execution Summary to (semi colon-separated): An empty text box.

Fill out form fields as described in the table below—you can opt to schedule your test cycle to **Run Once**, **Run Recurrently**, or trigger the run externally (using a third-party scheduler).

Table 10-1 Scheduling a Test Cycle

Control/Field	Sub-Field	Description
Target test cycle		<p>Ellipsis button  — Click to select a test cycle to schedule.</p>  <p>Click the delete button  to remove selected test cycle.</p>
Run once radio button		Select for a one-time run schedule.
	Run time	<p>Choose the run date from the drop-down calendar.</p>  <p>Choose a run time from the hour and minute selectors (24-hour clock).</p> 
Run recurrently radio button		Select to specify a repeating run schedule.
	Start time	<p>Choose the start date from the drop-down calendar.</p>  <p>Choose a start time from the hour and minute selectors (24-hour clock).</p> 

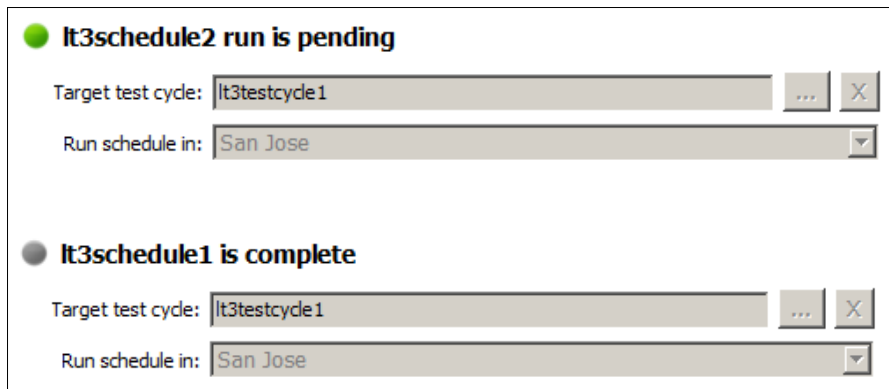
Control/Field	Sub-Field	Description
	Frequency check boxes	Select the days of the week on which to execute the test cycle—the script will be started at the same time each day. Frequency: <input type="checkbox"/> Mon <input type="checkbox"/> Tue <input type="checkbox"/> Wed <input type="checkbox"/> Thu <input type="checkbox"/> Fri <input checked="" type="checkbox"/> Sat <input checked="" type="checkbox"/> Sun
Run is triggered externally radio button		Select to specify that the script will be executed by the DeviceAnywhere command-line utility or any other third-party scheduler.
Description		Enter a description for the test cycle schedule, e.g., Runs on Saturday and Sunday at 9:00 a.m.
Run Schedule In		Select the location of a LiveTest server from the drop-down box provided. <input type="text" value="San Francisco"/> NOTE Contact your system administrator or Keynote Solutions Consultant to set up a LiveTest server for scheduled runs.
Edit Items to Run		Check the boxes for the specific test cases and test cycle devices you wish to execute as part of the schedule. 
Terminate test cycle execution after		Enter a test cycle execution time in hours, e.g., 3 . This is the maximum allowable time for each test cycle run, excluding the time taken to acquire a device. NOTE The execution time clock begins ticking as soon as devices for the run become available. The Live Test Server tries for up to two hours to acquire all devices required for a scheduled run.
Email Execution Summary		Enter addresses (semi-colon separated) to send execution summaries to.
Validate		Click to check your schedule for errors.
Check in/out		Check your schedule in to version control when you are done editing it, or check it out for editing.

See [Other Functions](#) in [Managing Scripts and Schedules](#) for a description of commands available by right-clicking a schedule in the project directory (**Scheduling** tab of the Automation view).

10.2.1.2 Schedule Status

After you have created a test cycle schedule, the status on the scheduling form is “pending” before the scheduled run. After the run is complete, the schedule status is updated to “complete.” In the case of a recurring schedule, the status reads as “running” during a scheduled run and reverts to “pending” before the next scheduled run.

Figure 10-4 Schedule Status



10.2.2 Test Cases

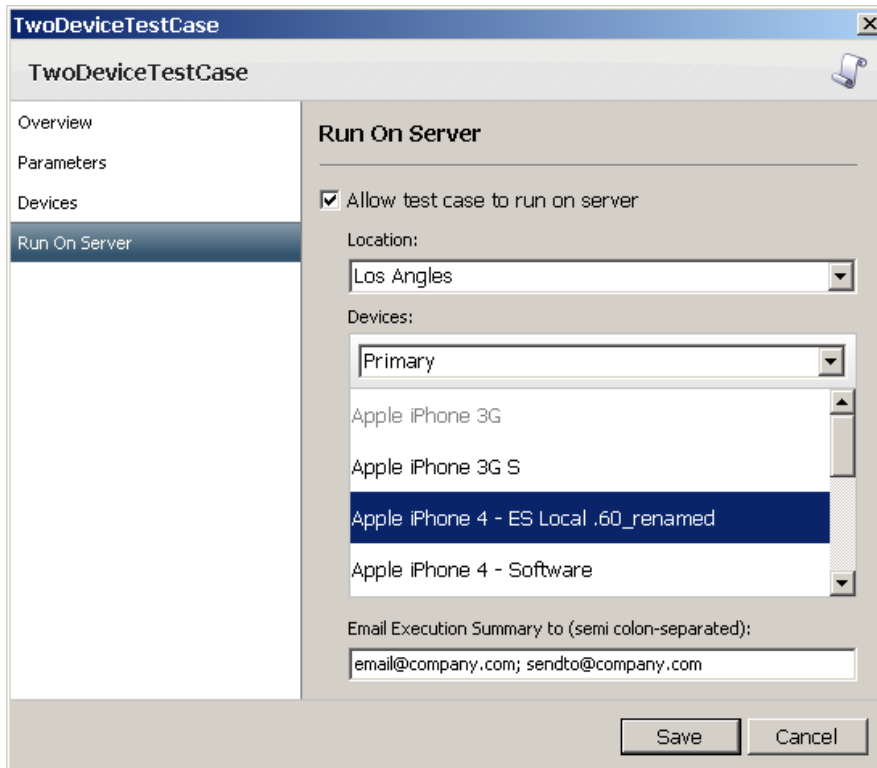
While test cases cannot be explicitly scheduled in the DeviceAnywhere Studio interface, you can enable a test case to be executed on a LiveTest server by DeviceAnywhere's command-line utility or any other third-party scheduler.

NOTE Contact your system administrator to set up a LiveTest server for scheduled runs.

- 1 Navigate to the **Scripting** tab of the Automation view.
- 2 Select and check out a test case.
- 3 Right-click the test case and select **Properties**.
- 4 Select the **Run On Server** tab of [test case properties](#).
- 5 Make the following selections:
 - a Check **Allow test case to run on server**.
 - b Select a LiveTest server **Location** from the drop-down list provided.
 - c Select **Devices** for each test case slot. Select a slot name from the drop-down list and click to select a device. You can override default device selection(s) when using the command-line scheduler.

NOTE Names for additional slots appear as defined in the [Devices tab](#) of test case properties. Be sure to use the same slot name when using the DeviceAnywhere utility to schedule the test case from the command line.

- 6 Enter email addresses (semi-colon separated) to send [execution summaries](#) to.



- 7 **Save** changes to test case properties.
- 8 Publish your project (right-click project in project list > **Publish Project**).

10.3 Executing Scripts Ad Hoc

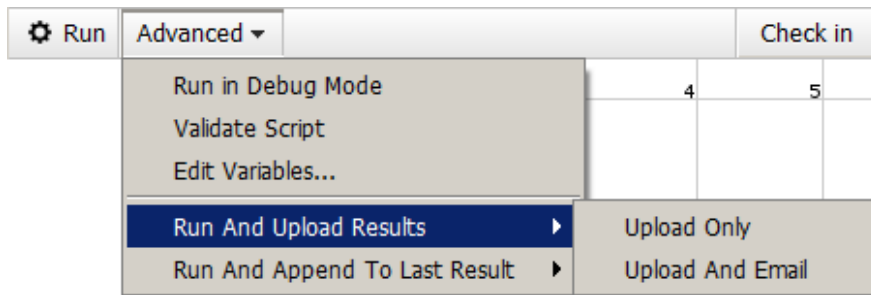
There are several options for ad hoc script runs:

- ◆ Click **Run** above the script canvas to run your script ([action](#), [test case](#), [test cycle](#)).
- ◆ Click **Advanced** or **Debug** and then **Run in Debug Mode** above the script canvas to run your script (action, test case, test cycle) in debug mode. This opens any embedded scripts (called using the Execute Action or Execute Test Case commands) in separate tabs as they are being executed.

Test cycles in DAE Automation have additional ad hoc run options:

- ◆ Click **Advanced** > **Run And Upload Results** to run a test cycle and upload results automatically to the web portal as each test case is completed. (When using this option, the [Script Result window](#) displays a **View** button instead of an **Upload** button as results have already been uploaded.)
 - **Upload Only** sends results to the web portal.
 - **Upload and Email** sends results to the portal and sends out an [execution summary](#).
- ◆ Click **Advanced** > **Run And Append To Last Result** if you have already run a test cycle on one device and want the results of a subsequent run on another device appended to previous results.
 - **Append Only** appends results in the portal to the previous run of the test cycle.
 - **Append And Email** appends results to those of the previous test cycle run and sends out an [execution summary](#).

Figure 10-5 DAE Automation Test Cycle Run Options



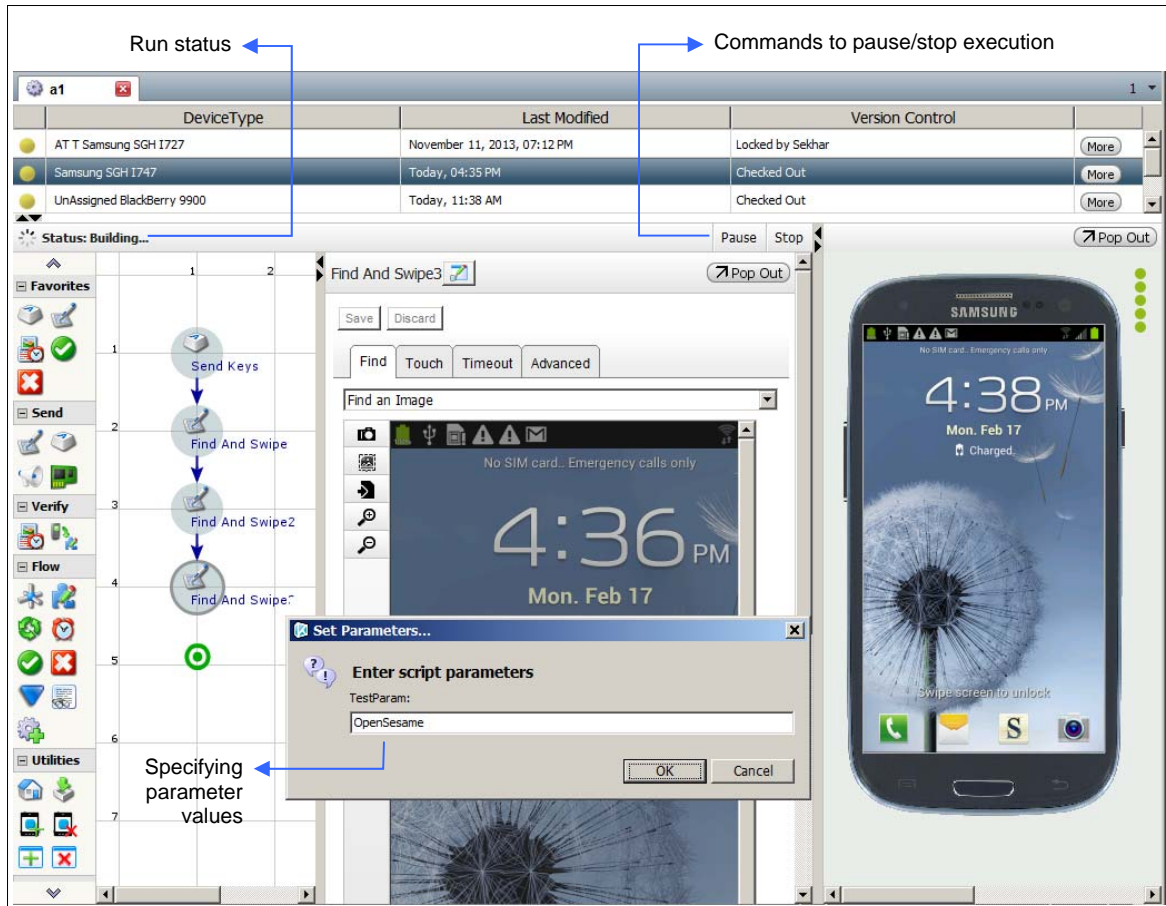
NOTE You can perform a partial script run by right-clicking a command in the script canvas and choosing [Run from here](#). This runs the script from the selected command and is useful when writing a test script to run through recently added commands.

To run or debug a script (action, test case, test cycle):

- 1 Double-click your script in the project directory to open it.
- 2 Choose an implementation/device:
 - For actions, select and open the implementation you wish to run/debug.
 - For test cases, if you haven't already, choose a project device to run the script on from the device pane. Click **Save Selected** above the device pane.
 - For test cycles, if you haven't already, choose a device to run the test cycle on from the device pane on the right. Click **Save Selected** above the device pane.
- 3 Acquire the device (right-click the device > **Acquire Device**). Your script run will fail if you do not acquire the device.
- 4 Optionally, check the script out. You need not check out a script in order to execute it. That way, testers without edit permissions can execute scripts. However, you might want to have the script checked out for editing if you are running it in debug mode.
- 5 Choose the **Run** or **Run in Debug Mode** option above the script canvas. (For DAE Automation test cycles, you can also choose **Run And Upload Results** or **Run And Append To Last Result**.)



- 6 Specify runtime values for any [script parameters](#) in the Set Parameters dialog box and click **OK**. You can change these values for each script run. Click **Cancel** to cancel your script run.



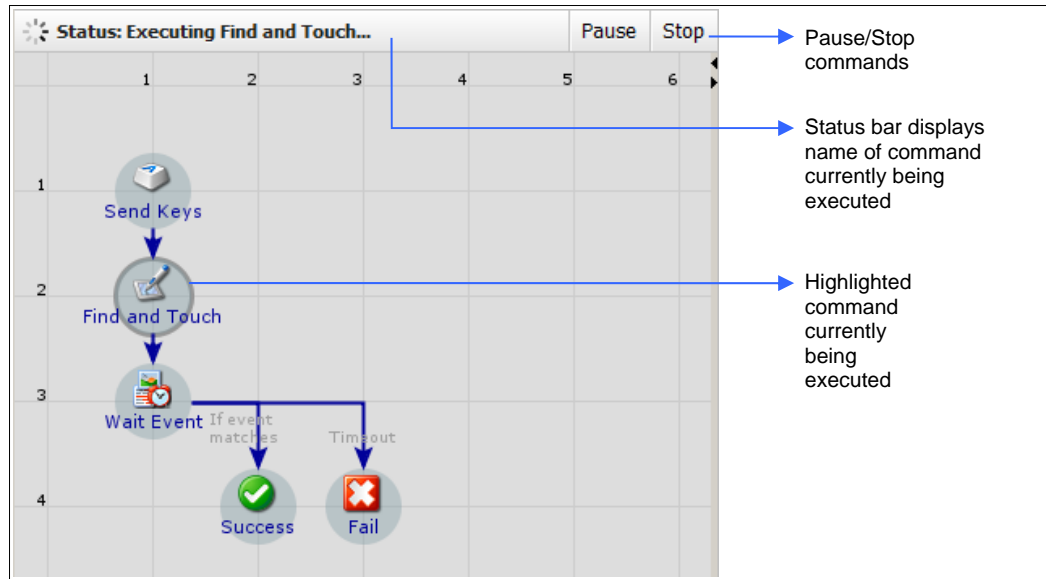
The status bar now indicates that the script is running.



If your script contains embedded scripts (called using the Execute Action or Execute Test Case commands), the correct implementation for the device you have acquired is automatically loaded.

If running a script in *debug mode*, any embedded scripts are automatically opened in separate tabs and run one by one so you can view script execution at a granular level.

As the run progresses, the script canvas highlights the command currently being executed and the status bar displays its name (in both the script being executed and any embedded scripts that have been opened in debug mode).



- 7 Optionally, click **Pause** or **Stop** above the script canvas to halt your script run. (Stopping your script causes it to fail.)

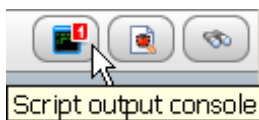
The status bar indicates when a script has been paused. Click **Resume** to continue.



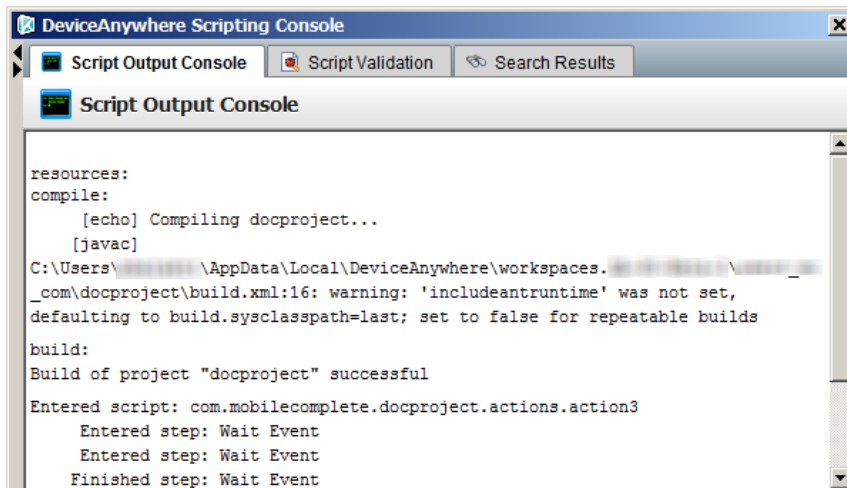
The status bar indicates if a script has been stopped.



- 8 When a project is built, including when a script is executed, the script output console icon in the top-right corner of the Automation interface displays a notification.



Click the icon to view console output in the DeviceAnywhere Scripting Console window. The contents persist until the next build.



9 [View test results.](#)

10.4 Execution Summaries

Email execution summaries with screenshots of the last proof image collected from the device can be sent out for test cases and test cycles scheduled to run either from DeviceAnywhere Studio or using the command line scheduling utility. Email summaries are also available for ad hoc test cycle runs (**Run And Upload Results > Upload And Email** and **Run And Append To Last Result > Append And Email**).

Besides proof screenshots, email summaries contain information on script name, device, and execution time. A single email is sent out for scheduled test cases. For scheduled test cycles, an overall summary and a completion email for each primary test cycle device are sent out. For test cycles executed using the **Run And Upload Results** or **Run And Append To Last Result** options, a completion email is sent out.

Recipients of execution summaries are specified in the [Run On Server tab](#) of [test case properties](#) and in the [scheduling form](#) for [test cycle schedules](#).

Figure 10-6 Test Cycle Overall Execution Summary

keynote DeviceAnywhere™ Any Device. Any Network. From Anywhere.
Mobile App Lifecycle Management™

Summary info

Test Cycle: ThridTestcycle
Time started: Tue Jan 03 10:56:55 PST 2012
Time completed: Tue Jan 03 11:01:42 PST 2012
Total time: 00:04:47
Live Test server location: Los Angeles

Details

Device	Pass	Fail
Samsung A900M	0	1
Apple iPhone 4	0	1

[Results Page](#)

Failed test cases

Test Case	Device	Time
CallingSecondAction	Samsung A900M	00:03:55
CallingSecondAction	Apple iPhone 4	00:04:43

Figure 10-7 Execution Summary for a Test Cycle Device (with Device Screen Proof)

keynote DeviceAnywhere™ Any Device. Any Network. From Anywhere.
Mobile App Lifecycle Management™

Summary info

Test Cycle: ThridTestcycle
 Device: Apple iPhone 4
 Secondary: N/A
 Time started: Tue Jan 03 10:56:57 PST 2012
 Time completed: Tue Jan 03 11:01:40 PST 2012
 Total time: 00:04:43
 Live Test server location: Los Angles

Details

Test Case	Status	Last Command	Last Image Proof
CallingSecondAction	Fail	WaitState - Failed	

The image below is a sample of the execution summary mailed out when a test cycle is executed from DeviceAnywhere Studio (Run And Upload Results > Upload And Email or Run And Append To Last Result > Append And Email options).

Figure 10-8 Execution Summary for Test Cycle Executed from DeviceAnywhere Studio

keynote DeviceAnywhere™ Any Device. Any Network. From Anywhere.
Mobile App Lifecycle Management™

Summary info

Test Cycle: NewTestcycle
 Device: HTC EVO Shift 4G
 Secondary: N/A
 Time started: Wed Jan 18 16:03:41 PST 2012
 Time completed: Wed Jan 18 16:03:43 PST 2012
 Total time: 00:00:02

Details

Test Case	Status	Last Command	Last Image Proof
CallinAction03DBProject	Fail	ExecuteAction - Failed	N/A

Keynote DeviceAnywhere Test Center
 777 Mariners Island Blvd., San Mateo, CA94404 | 650-403-2400
 © 2012 Keynote DeviceAnywhere. All rights reserved.

Figure 10-9 Execution Summary (with Device Screen Proof) for Test Case Scheduled from Command Line

10.5 Viewing Results

A script run is deemed successful when all commands have been completed, or when an explicit Success command is encountered. A script run is deemed unsuccessful when all commands have not been completed, an explicit Fail command is encountered, a device is not acquired for the script run, or execution is stopped.

After a script run has been completed, a [result bar](#) above the script canvas displays execution status. You can then view the [Script Result window](#) and [upload results and view them in the web portal](#). At any point, you can [log in to the web portal and view previously uploaded test results](#).

10.5.1 Result Bar

After a script (action implementation, test case, test cycle) has been executed (or stopped), a result bar above the script canvas displays whether the run has succeeded, failed, or errored out. A new result bar is displayed for each run. Result bars for consecutive runs persist until you dismiss them (see Figure 10-10).


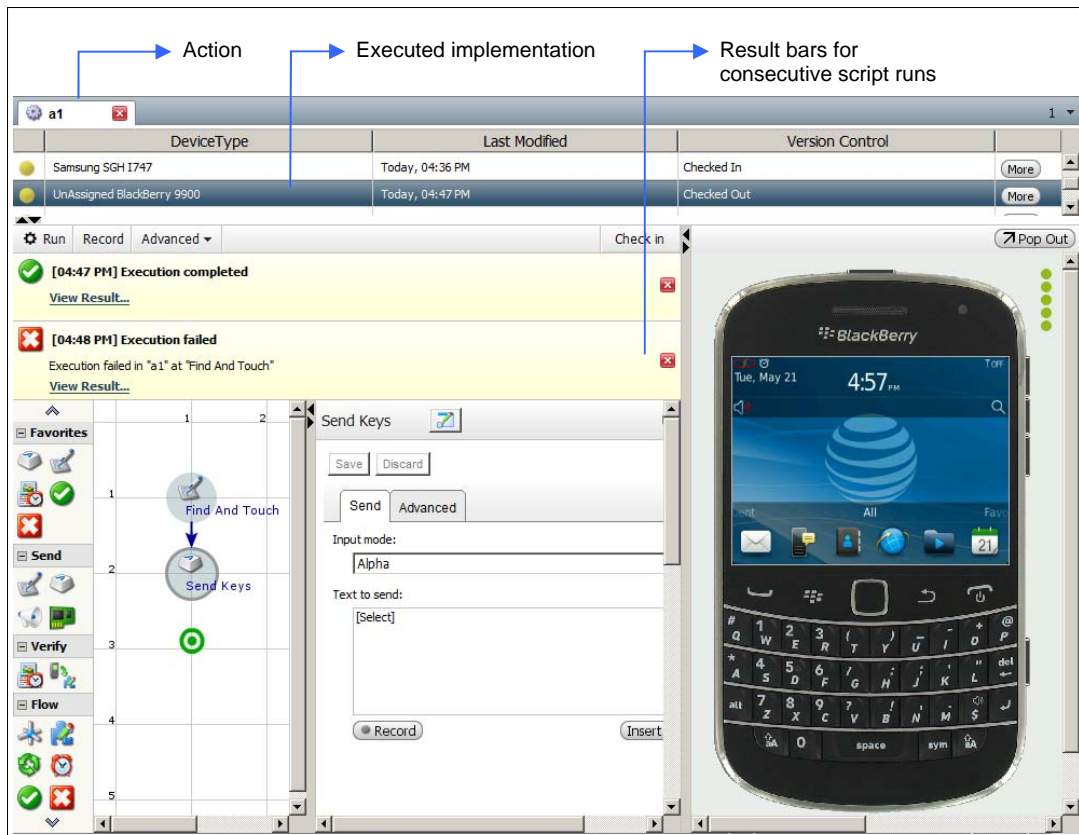
- ◆ Click the close button  to close the result bar.
- ◆ Click **View Result** to open the [Script Result window](#).

Figure 10-10 Result Bars for Consecutive Script Runs



The figures below show the result bars for successful, unsuccessful, and stopped script runs, respectively.

Figure 10-11 Result Bar – Success

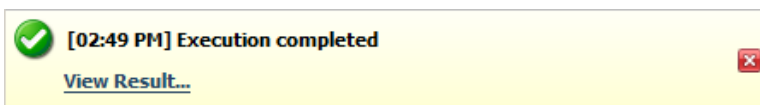


Figure 10-12 Result Bar – Fail

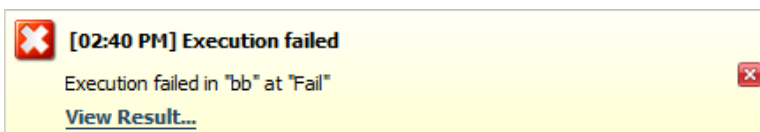
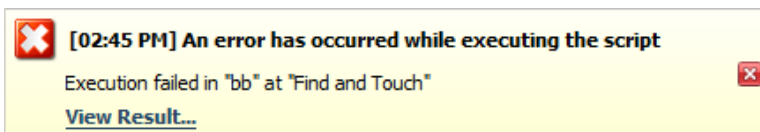


Figure 10-13 Result Bar – Stopped Execution



Script runs can also fail because the device is not acquired (see [Executing Scripts](#) above) or, in the case of test cases and test cycles, not selected.

Figure 10-14 Result Bar—Device Not Acquired

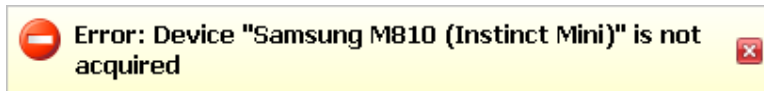
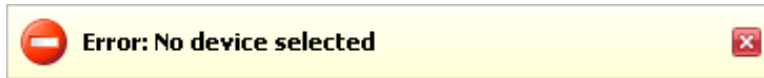
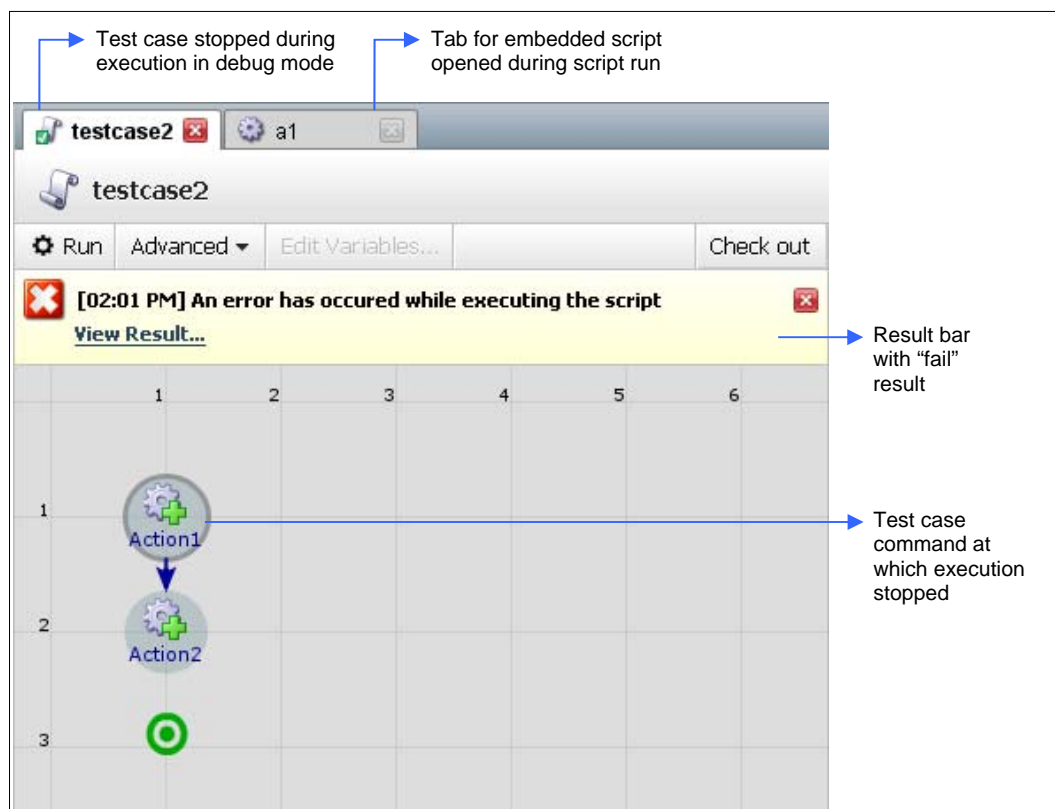


Figure 10-15 Result Bar—No Device Selected



The figure below shows a test case script that has been stopped during execution in debug mode. The script canvas highlights the command at which execution was stopped. There is also an open tab for an embedded action. The result bar indicates that the script run has been stopped and has therefore failed.

Figure 10-16 Stopped Execution in Debug Mode: Test Case with Embedded Actions





10.5.2 Script Result Window

The Script Result window opens up when you click **View Result** in the [result bar](#) of a [script run](#) from DeviceAnywhere Studio (see Figure 10-17 below).

The Script Result window lists details of the script executed (script name, location in the local project directory, device) and the overall result. You can customize the overall result displayed by:

- ◆ Implementing a custom message in the [Success](#) command, or
- ◆ Choosing the [error type](#) triggered in the [Fail](#) command and the [Timeout tab](#) of other commands.

The left pane of the Script Result window displays the script executed, any embedded scripts, and the commands and their branches within them in a tree-like structure. Icons indicate if a node in the tree structure is successful  or unsuccessful . From the tree, click the script name to view top-level results, or click a specific command to view granular, command-level results in the right pane. **Upload Result** uploads the run results to the web portal.

Notes If the particular result has already been uploaded, the button changes to **View Result**.

The **Upload Result** option is not available for Mobile App Monitoring script runs in Studio.

Export Result enables you to save the results as an XML file to your file system.

Figure 10-17 below shows the Script Result window for an [action](#), “bb.” The left pane shows the action and its constituent commands in a tree structure with the action name selected. The right pane shows top-level results for the run including a custom script termination message (from the [Success command](#)), the script duration, and the device.

Figure 10-17 Script Result Window—Successful Action

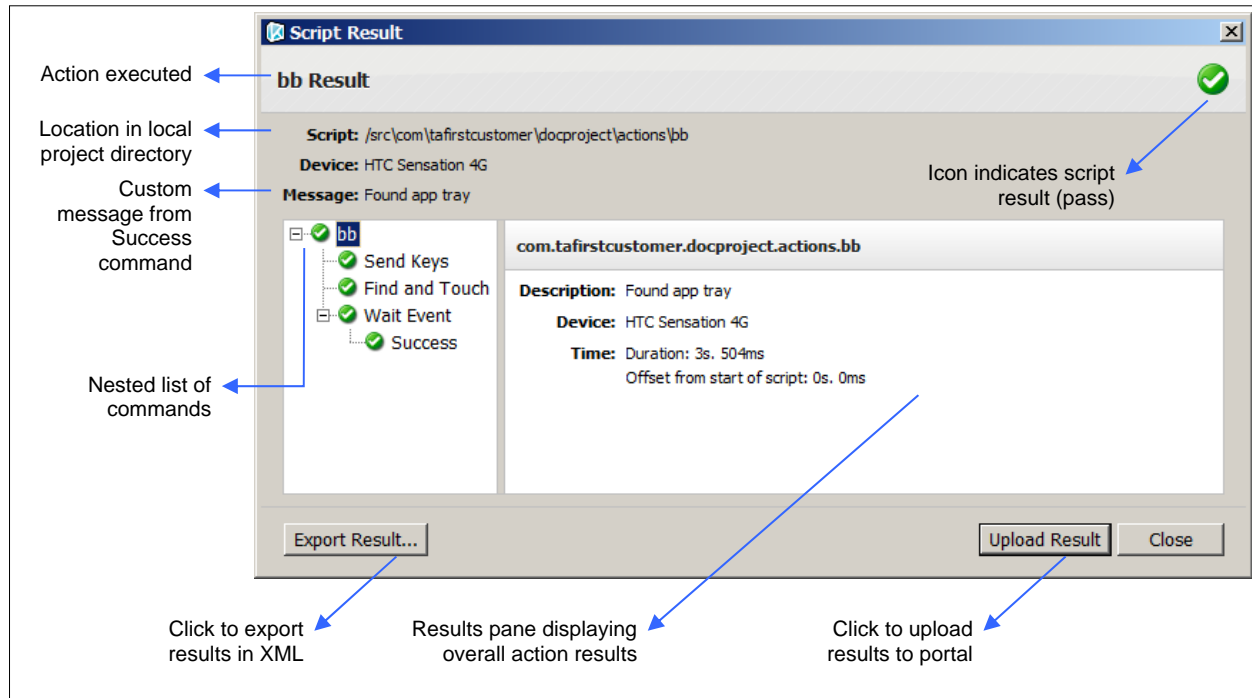
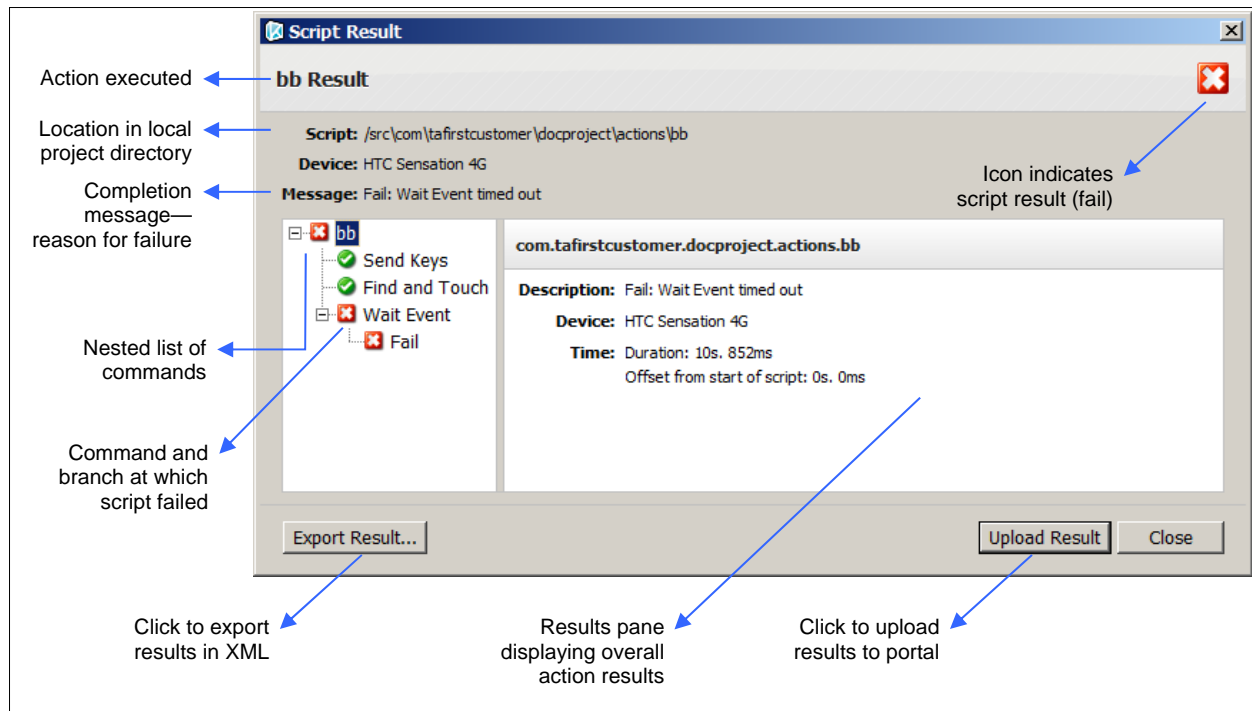


Figure 10-18 Script Result Window—Failed Action



When you select an individual command in the left pane, any [proofs](#) captured for the command are displayed in the right pane of the Script Result window (see Figure 10-19 below). A drop-down list enables you to select images for expected results, actual results, and a comparison of the two with differences highlighted in pink. Use the second drop-down list provided to select and view proofs. By default, proofs consist of the first and last screen of the device screen during command execution.

Figure 10-19 Viewing Proofs Captured

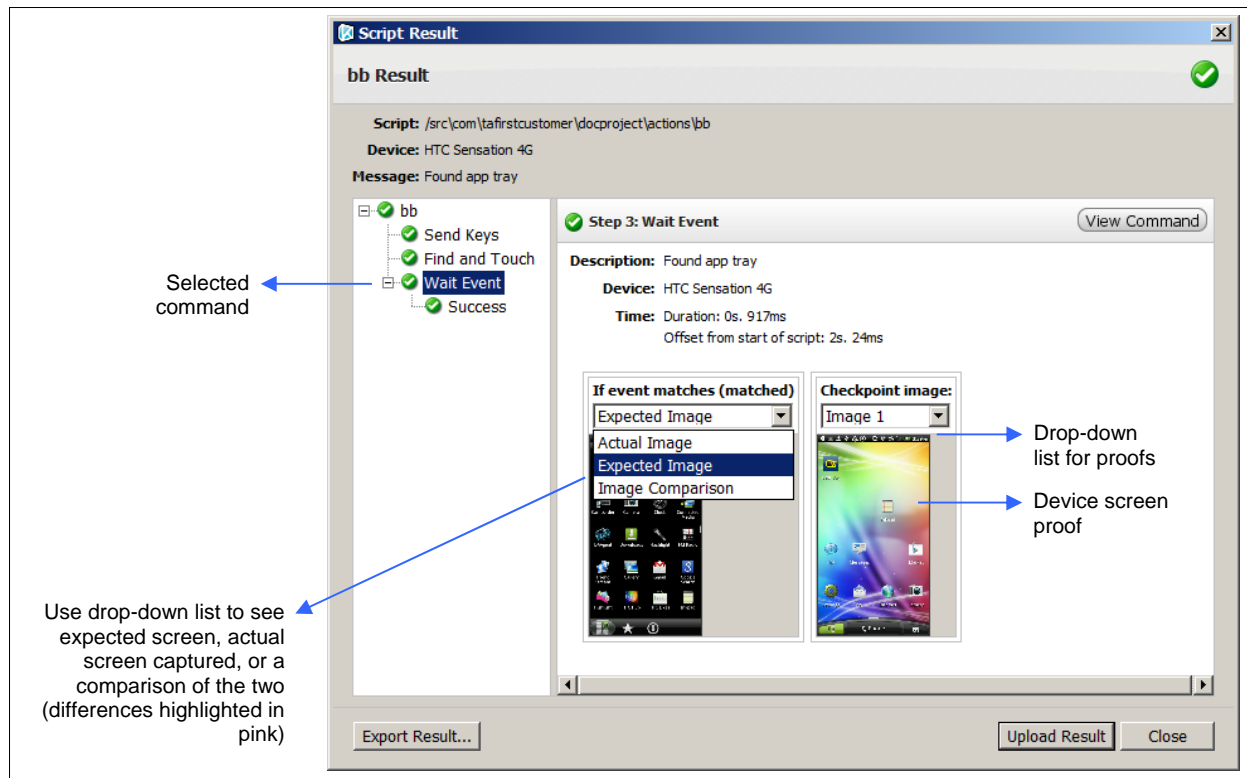
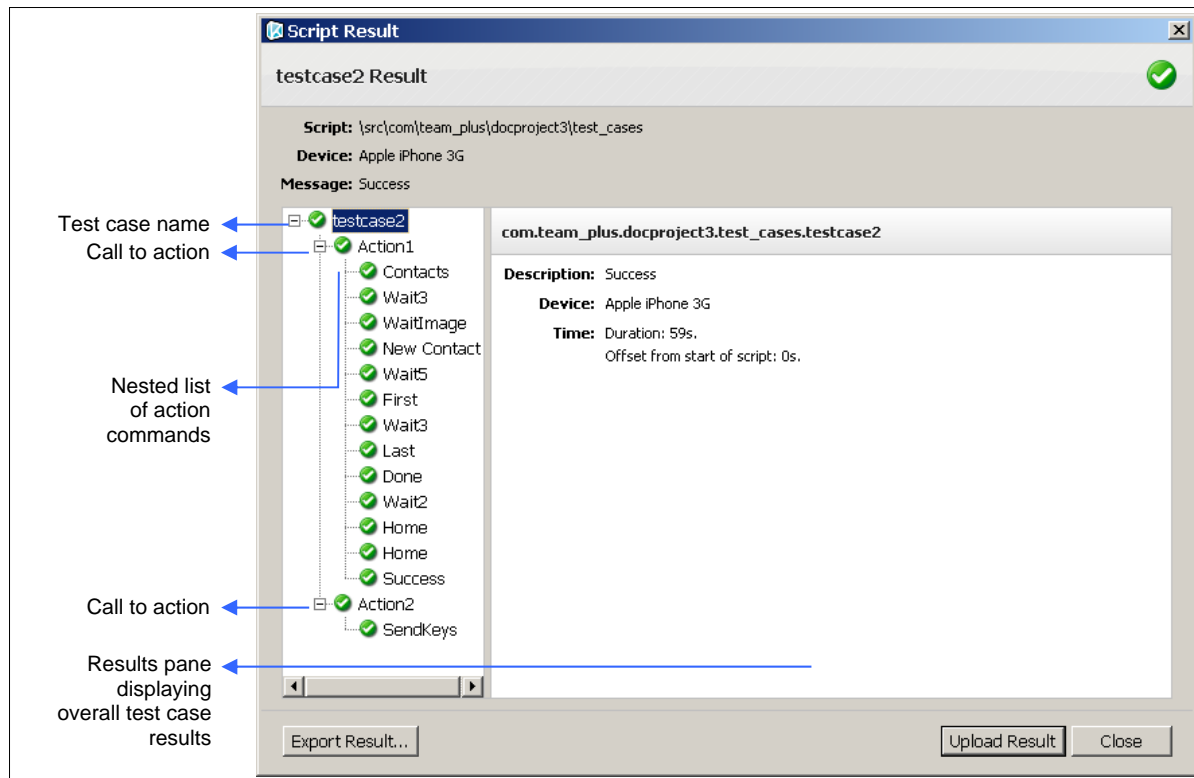


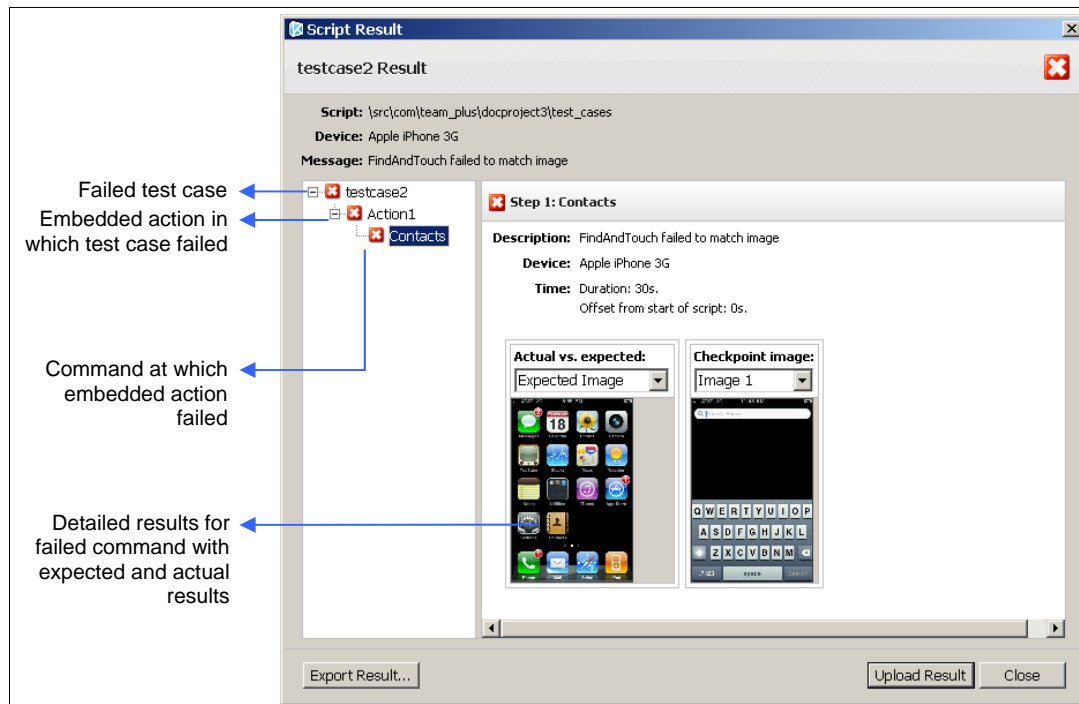
Figure 10-20 shows the Script Result window for a successful [test case](#). The test case contains calls to two actions. The test case name, the action calls, and commands in the actions are displayed successively nested in the left pane with icons indicating success. The right pane below displays the overall test case results. As in the results window for actions, you can select any script/command in the left pane to view its detailed proofs in the right pane.

Figure 10-20 Script Result Window—Successful Test Case



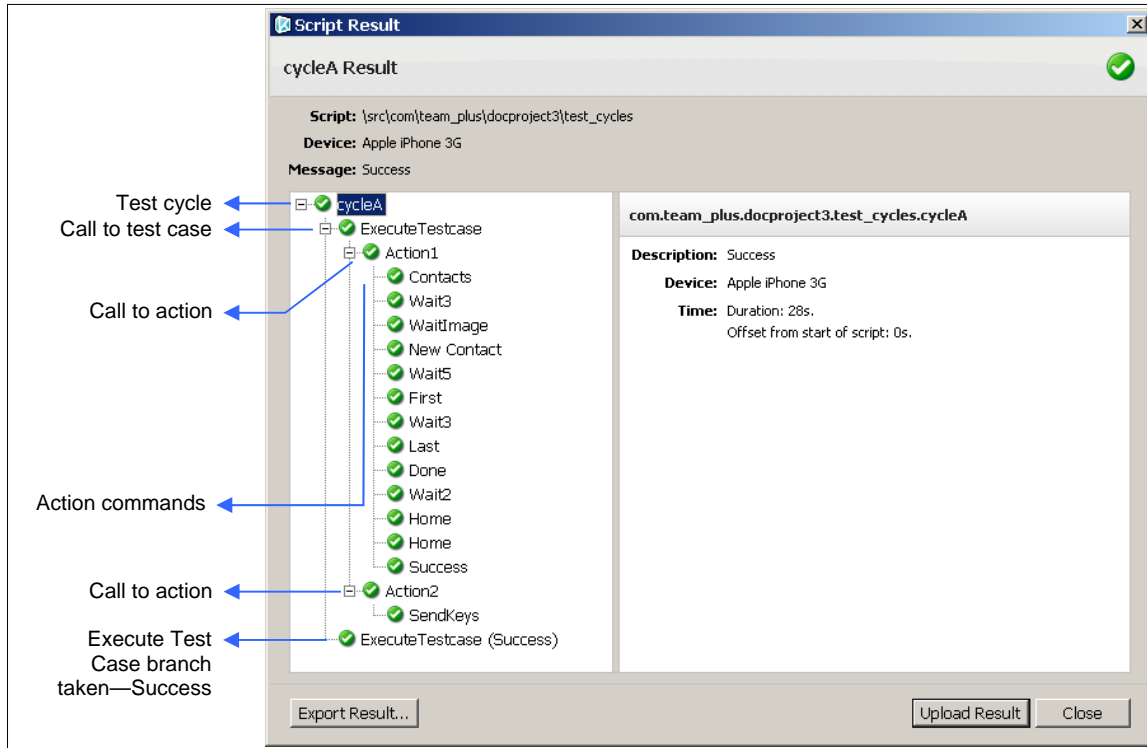
In the case of failed test cases, red icons indicate the embedded action and the specific command at which the run failed.

Figure 10-21 Script Result Window—Failed Test Case



In the Script Result window for [test cycle](#) runs (DAE Automation only), the left pane displays the names of the test cycle, test case(s), action(s), and commands executed. The left pane also displays the branch taken (Success or Failure) in an [Execute Test Case command](#). Figure 10-22 shows the results window for a test cycle consisting of a single test case, which, in turn, calls two actions.

Figure 10-22 Script Result Window for a Test Cycle

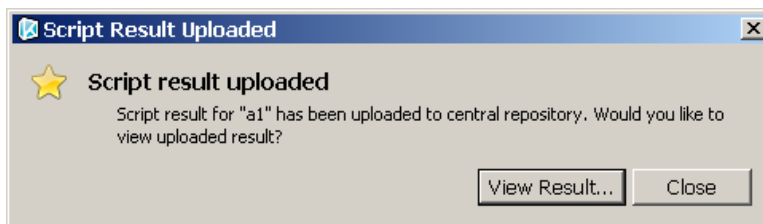


10.5.3 Uploading Results to the Web Portal

This section describes uploading DAE Automation script runs to the DeviceAnywhere results and administration web portal. (Results of script runs from the [Test Case Runtime](#) interface of Test Case Manager are automatically uploaded to the web portal.)

To upload test results:

- 1 After [running your script](#), click **Upload Result** in the [Script Result window](#). This stores test results in your customer account on the web portal.
- 2 Click **View Result** in the dialog box that appears to view the uploaded result in the portal.



This automatically logs you in to the web portal and displays the uploaded script result.

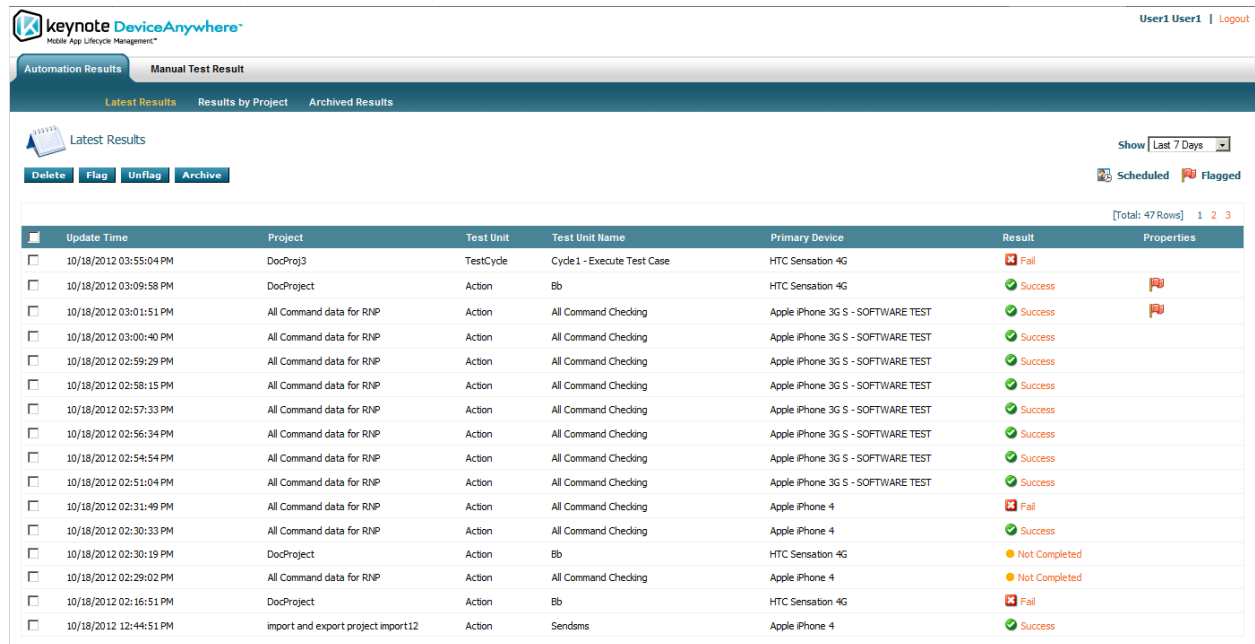
Portal results pages for actions, test cases, and test cycles are described next in [Viewing Results in the DeviceAnywhere Web Portal](#).

10.5.4 Viewing Results in the Web Portal

You can log in to the DeviceAnywhere results and administration web portal at any time to view previously [uploaded results](#) for ad hoc or scheduled DAE Automation runs:

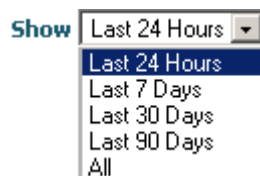
- 1 Click **Result Portal** from the DeviceAnywhere Studio sidebar.



You are directed to the results page for the most recent script runs (**Latest Results**) from the Automation view and scheduled runs. You can also view [results sorted by project](#).



Update Time	Project	Test Unit	Test Unit Name	Primary Device	Result	Properties
10/18/2012 03:55:04 PM	DocProj3	TestCycle	Cycle1 - Execute Test Case	HTC Sensation 4G	Fail	
10/18/2012 03:09:58 PM	DocProject	Action	Bb	HTC Sensation 4G	Success	Flagged
10/18/2012 03:01:51 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	Flagged
10/18/2012 03:00:40 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:59:29 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:58:15 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:57:33 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:56:34 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:54:54 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:51:04 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 3G S - SOFTWARE TEST	Success	
10/18/2012 02:31:49 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 4	Fail	
10/18/2012 02:30:33 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 4	Success	
10/18/2012 02:30:19 PM	DocProject	Action	Bb	HTC Sensation 4G	Not Completed	
10/18/2012 02:29:02 PM	All Command data for RNP	Action	All Command Checking	Apple iPhone 4	Not Completed	
10/18/2012 02:16:51 PM	DocProject	Action	Bb	HTC Sensation 4G	Fail	
10/18/2012 12:44:51 PM	import and export project import12	Action	Sendsms	Apple iPhone 4	Success	

You can select the period for which you want to view results from the **Show** drop-down list.



Icons in the **Properties** column indicate whether a result is for a scheduled run  or has been flagged  for some reason.

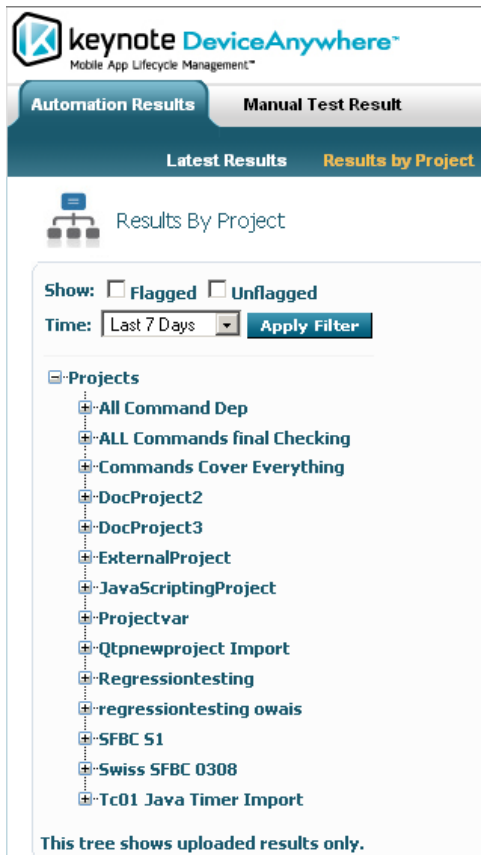
From this view, you can check the box next to a run to **Delete**, **Flag**, **Unflag**, or **Archive** it. Archived results are displayed in the **Archived Results** tab from where they can be deleted or removed from the archive.

- 2 Click the **Success** or **Fail** link next to a run to [view detailed results](#).

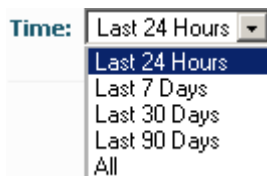
10.5.4.1 Viewing Results by Project

To view results sorted by project and then by script type:

- 1 Click **Result Portal** in the DeviceAnywhere Studio sidebar to access the portal results page.
- 2 Click the **Results by Project** tab. The tab displays projects in your environment for which results have been uploaded.

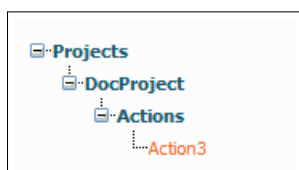


- 3 Select the period for which you want to view results from the **Time** drop-down list.



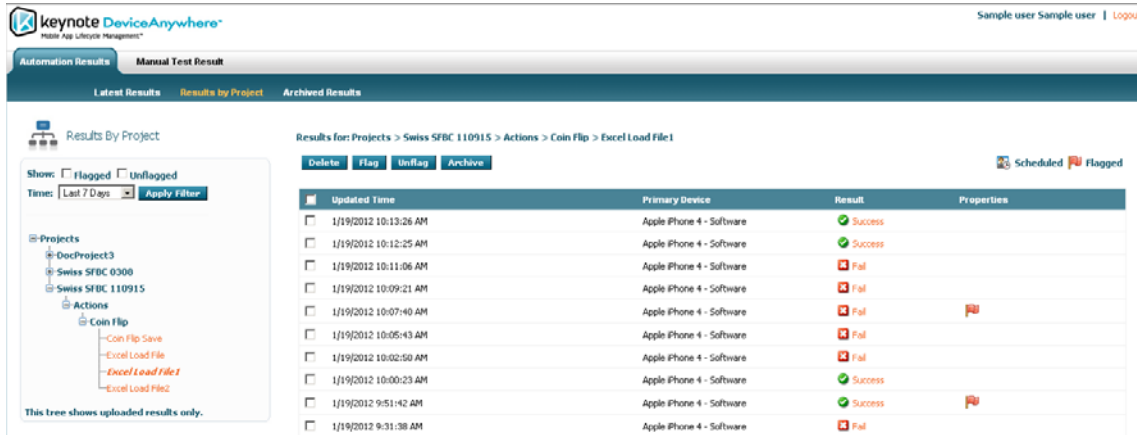
You can also opt to view **Flagged** or **Unflagged** run results.



- 4 Click **Apply Filter**.
- 5 Select and expand a project to view its directory structure and a list of any scripts for which results have been uploaded.



- 6 Select the script for which you want to view results.

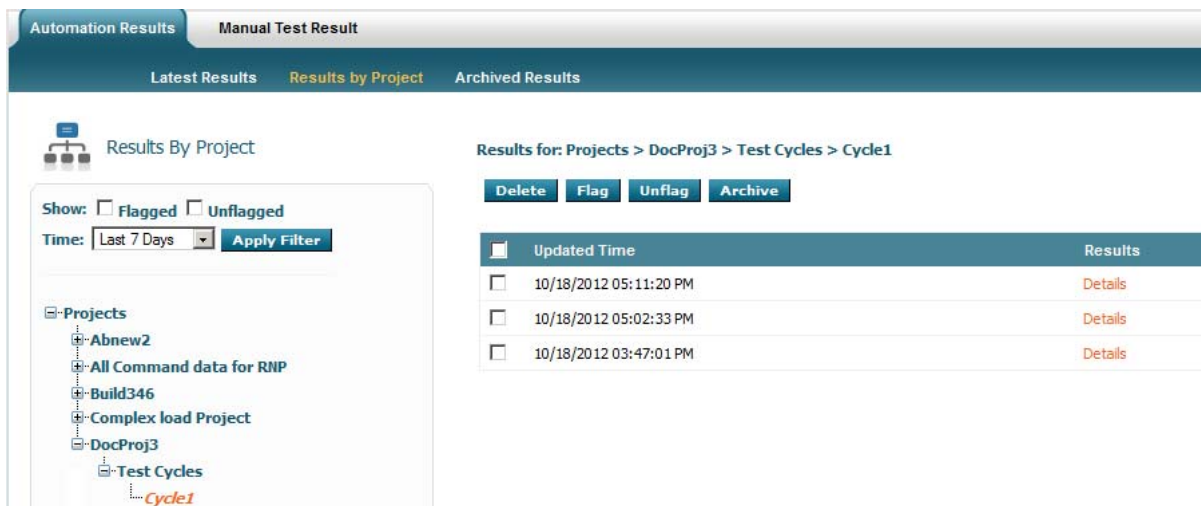
A list of runs for which results have been uploaded is displayed in the right pane.



Icons in the **Properties** column indicate whether a result is for a scheduled run  or has been flagged  for some reason.

From this view, you can check the box next to a run to **Delete**, **Flag**, **Unflag**, or **Archive** it. Archived results are displayed in the **Archived Results** tab.

- 7 If you have selected a test cycle with multiple runs, click **Details** next to a run to view a list of test cases covered in the run.



A test case-device matrix is displayed. Multiple devices are displayed if you have scheduled the test cycle to run on multiple primary devices or have used the **Run And Append To Last Result** execution option.

Results for: Projects > DocProj3 > Test Cycles > Cycle1 > Testcase/ Device Summary





Testcycle Run Summary


User : user1 user1

Scheduled Run : No

Properties : N/A

Results

Test Case	HTC Sensation 4G	Apple iPhone 4 - Software
Execute Test Case2	 Success	 No Result
Execute Test Case	 Success	 Fail

No Result  indicates that a test case was not executed on a device, e.g., because the previous test case in the test cycle failed.

- Click the **Success** or **Fail** link next to a run to [view detailed results](#).

10.5.4.2 Detailed Results

When you click the **Success** or **Fail** link next to a script run, you can view detailed, command-by-command results, including device screen proofs and comparisons between expected and actual results.

Script commands, including commands in embedded scripts are displayed on the left in a tree-like structure. If you run a test case containing calls to actions and upload the results to the web portal, you can expand the test case results displayed to view step-by-step results for embedded actions as well.

Select a command on the left to view results for it in the right pane. Proofs are displayed with actual and expected results and a comparison of the two—areas of mismatch are highlighted in pink.

Figure 10-23 below shows the detailed results for a test case that calls two actions. The tree structure on the left displays commands within the embedded actions. For the Wait Event command selected on the left, the right pane displays proofs as well as actual vs. expected results.

Figure 10-23 Detailed Results – Test Case

keynote DeviceAnywhere
Mobile App Lifecycle Management™

Proofs for Test Case: Tc1
Primary Device: HTC Sensation 4G

Script name and device script was executed on

Links to archive, email, export, and flag results

Project Name : DocProj3
List of Secondary devices : N/A
Run Start time : 10/18/2012 05:47:00.000 PM

Executed By : user1 user1
Portal user ID
Properties : N/A
Run Completion time : 10/18/2012 05:47:07.297 PM

Action: Wait Event -- Success

Description	Command Type
Script completed successfully.	AdvancedWaitEvent
Active Device: HTC Sensation 4G	Slot: N/A
Comments: N/A	Time: Duration: 825 ms Offset from start of script: 1 secs 982 ms
Start Time: 10/18/2012 05:47:00.000 PM	End Time: 10/18/2012 05:47:00.825 PM

Image Proof(s)

Checkpoint Image 1

Checkpoint Image 2

Profs—images of device screen

Script commands in a tree-like structure with selected command highlighted

Matched Event: If event matches (Matched)

Actual Image
Branch: If Event Matches (Matched)

Expected Image
Branch: If Event Matches (Matched)

Image Comparison
Branch: If Event Matches (Matched)

Expected results

Copyright © 2011 Keynote DeviceAnywhere™. All rights reserved.

If you opt to collect proofs using the [Checkpoint control](#) of a command, the first and last screen of the device during command execution are displayed in test results.

Figure 10-24 Proofs Elected Using the Checkpoint Control of Send Keys

keynote DeviceAnywhere™
Mobile App Lifecycle Management™

Proofs for Action: Act1
Primary Device: HTC Sensation 4G

Project Name: DocProj3
List of Secondary devices: N/A
Run Start time: 10/18/2012 06:31:00.000 PM

Executed By: user1 user1
Properties: N/A
Run Completion time: 10/18/2012 06:31:04.007 PM

Act1
Send Keys
Find and Touch
Wait Event

Action: Send Keys -- Success

Description	Send Keys - [Home]	Command Type	SendKeys
Active Device	HTC Sensation 4G	Slot	N/A
Comments	N/A	Time	Duration: 891 ms Offset from start of script: 0 ms
Start Time	10/18/2012 06:31:00.000 PM	End Time	10/18/2012 06:31:00.891 PM

Image Proof(s)

Checkpoint Image 1 Checkpoint Image 2

Download original Download original

Copyright © 2011 Keynote DeviceAnywhere™. All rights reserved.

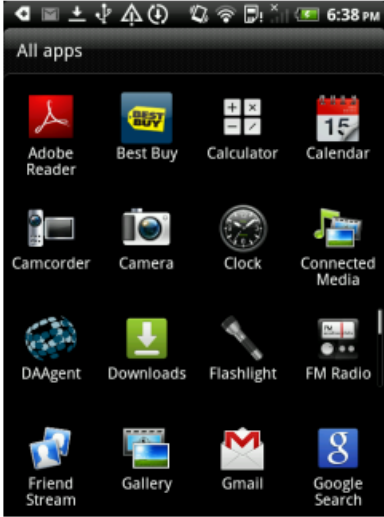
You can store proof images to disk (**Download Original**), e.g., if you want to import it into the [reference image wizard](#). Click an image to view a larger version of it.

Figure 10-25 Enlarged Proof Image



If your command uses a text-based reference point for verification, results for both extracted and expected text are displayed. You can download expected or extracted text as a text file.

Figure 10-26 Text Matching Results in the Portal

Matched Event: If event matches (Matched)		
<p>Actual Image</p> <p>Branch: If Event Matches (Matched)</p>  <p>Download original</p>	<p>Expected Text</p> <p>Branch: If Event Matches (Matched)</p> <p>Camera</p> <p>Download original</p>	<p>Extracted Text</p> <p>Branch: If Event Matches (Matched)</p> <p>Camcorder Camera Clock Connected Media i DAAgent Downloads Flashlight FM Radio Friend Gallery Gmail Google Stream Search</p> <p>Download original</p>

Click **Email Results** at the top left of the window to email a link to the current results page. You can **Select Recipients** from users in your account or enter a **Custom Email** address as well as mark a copy to yourself. Enter a **Custom Message** in the field provided and click **Send**.

Figure 10-27 Emailing Test Results from the Portal

Email
✕

Project: DocProj3

Action: Act1 **Result:** Success

Device: HTC Sensation 4G

Select Recipients

Ensemble Server (Ensemble)
 Script Runner (ScriptRunner)

Custom Email (if any)

(e.g. name1@xyz.com, name2@xyz.com, ...)

Send me a copy

Custom Message

Send
Cancel

Click **Archive** to archive test results. **Archived Results** are displayed in a separate tab. You can view detailed results by clicking on the link next to a run. You can also delete or remove results from the archive from here.

Figure 10-28 Archived Results

	Update Time	Project	Test Unit	Test Unit Name	Primary Device	Result	Properties
<input type="checkbox"/>	01/19/2012 2:07 PM	DocProject3	Action	A1	Apple iPhone 3G	Success	
<input type="checkbox"/>	01/19/2012 12:28 PM	DocProject3	TestCase	Testcase2	Apple iPhone 3G	Success	
<input type="checkbox"/>	01/19/2012 10:36 AM	Swiss SFBC 110915	Action	Excel Load File2	Apple iPhone 4 - Software	Fail	
<input type="checkbox"/>	12/30/2011 11:07 AM	SFBC S1	TestCase	HTML DataSet	Apple iPhone 4 - Software	Success	

Click **Export** to save results as a PDF file. You can opt to save the currently displayed page (**This page only**) or results for all commands in the script (**All pages**).

Figure 10-29 Exporting Test Results from the Portal

Export to PDF

File Name:

Export Range

This page only

All pages (May take few minutes to generate PDF)

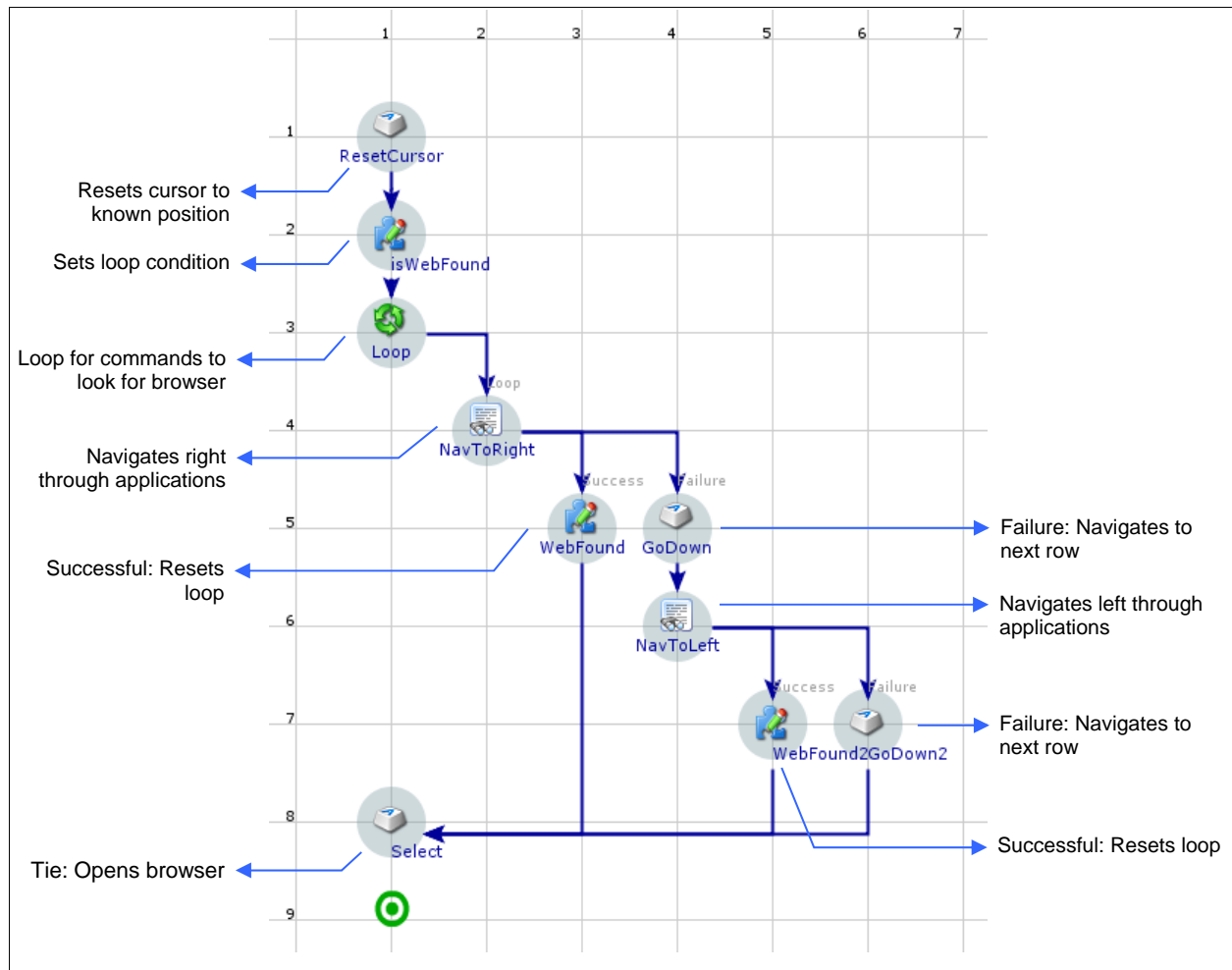
11 Examples

This chapter presents sample action and state implementations, and test case and test cycle scripts that exemplify the use of scripting commands.

11.1 Action: Navigating to the Web

In this sample [action](#) implementation, the script resets a BlackBerry device and then navigates to and opens the web browser. This script exemplifies the use of [Set Variable](#), [Loop](#), and [Navigate To](#).

Figure 11-1 *Navigate To, Set Variable, and Loop Example*



The table below explains each command in detail:

Table 11-1 Navigating to the Browser on BlackBerry—Commands

Name	Command	Description
ResetCursor	Send Keys	Repeatedly presses Power key to wake the device. Presses Menu key to access application menu. Repeatedly presses Left key, then Up key to return cursor to the top left of the application menu.
isWebFound	Set Variable	Sets a Boolean variable, isWebFound, to “false,” as the objective of finding the browser has not yet been met.
Loop	Loop	Uses isWebFound=false as a loop condition. Maximum number of loop iterations is set to 4: The commands within the loop will iterate 4 times or until isWebFound=true, whichever comes first. <i>See Figure 11-2 below.</i>
NavToRight	Navigate To	Within the loop, presses the Right key (for a maximum of 5 times) and checks to find the browser. <i>See Figure 11-4 below.</i>
WebFound	Set Variable	In the Success branch of NavToRight, resets isWebFound to true, thereby exiting the loop.
GoDown	Send Keys	In the Failure branch of NavToRight, navigates to the next row of application icons.
NavToLeft	Navigate To	Still in the Failure branch of NavToRight, presses the Left key (for a maximum of 5 times) and waits to find the browser.
WebFound2	Set Variable	In the Success branch of NavToLeft, resets isWebFound to true, thereby exiting the loop.
GoDown2	Send Keys	In the Failure branch of NavToLeft, navigates to the next row of application icons. The script then returns to the top of the loop (NavToRight).
Select	Send Keys	Continuing with the script outside the loop, presses Select to open the browser.

The figures below show the Loop and NavToRight command properties.

Figure 11-2 Loop Command: Loop Condition

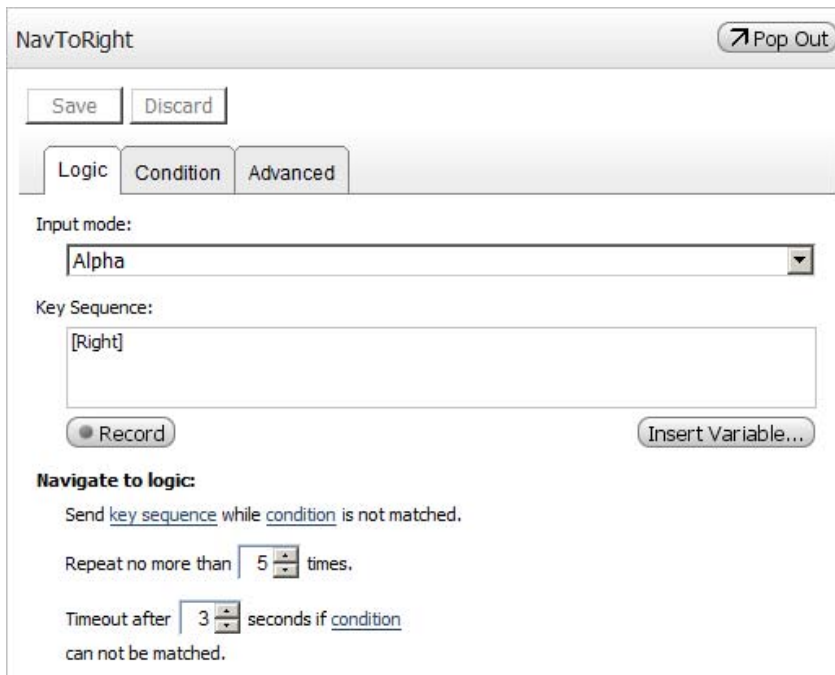
The screenshot shows the configuration for a Loop command. The 'Loop Logic' tab is active, displaying a dropdown menu set to 'Iterate while all expressions are true'. Below this, there are 'Add Expression' and 'Remove Expression' buttons. A table lists the current loop condition:

Variable	Is Equal To
isWebFound	False

Figure 11-3 Loop Command: Max. Iterations



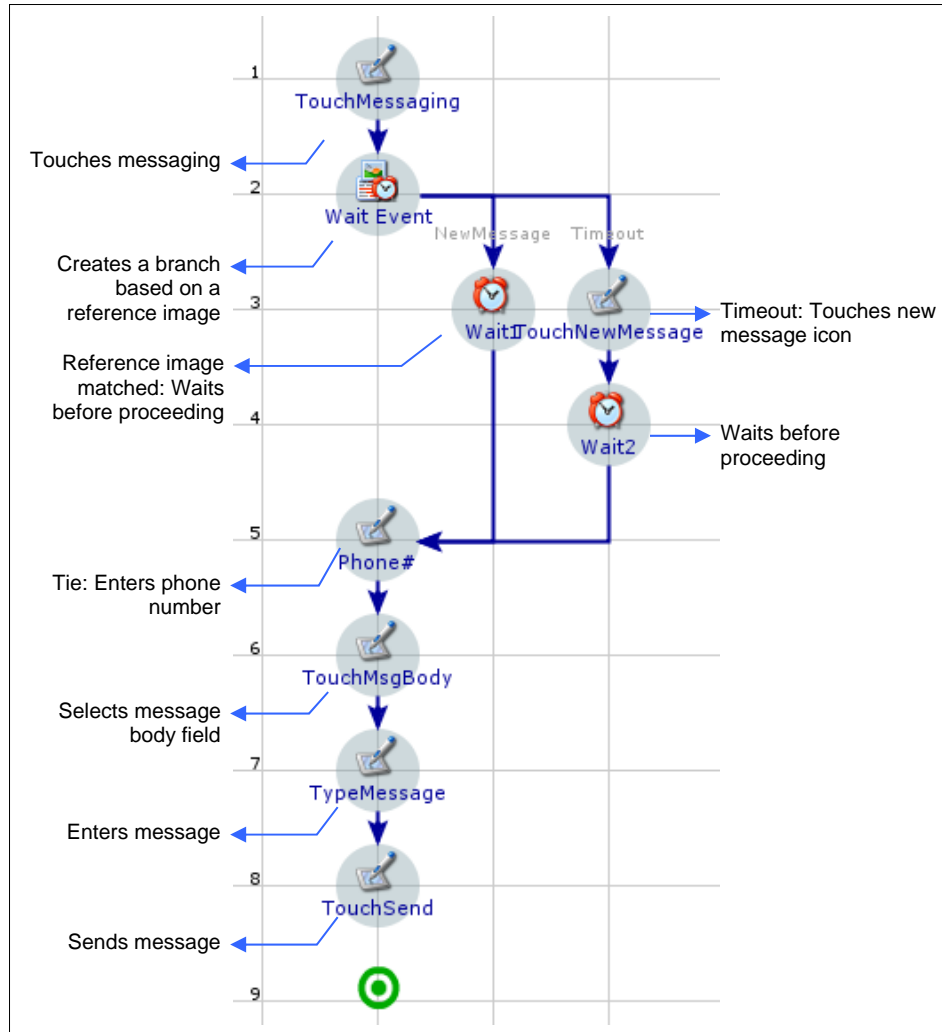
Figure 11-4 NavigateToRight Command—Key Sequence and Iterations



11.2 Action: Sending an SMS Message

In this sample [action](#) implementation, the script sends a message on an iPhone device. This script exemplifies the use of [Find and Touch](#) and [Wait Event](#) (with additional logic for the Tie and Timeout branches).

Figure 11-5 Wait Event, Find and Touch Example



The table below explains each command in detail:

Table 11-2 Sending an SMS Message on Android – Commands

Name	Command	Description
TouchMessaging	Find and Touch	Presses the messaging icon on the touchscreen. <i>See Figure 11-6 below.</i>
WaitEvent	Wait Event	Creates a branch based on a reference image to verify that a new message has been created. If the image is found, script control passes to the branch, NewMessage. <i>See Figure 11-7 below.</i>
Wait1	Wait	In the NewMessage branch, waits for a second after the new message has been created.

Name	Command	Description
TouchNewMsg	Find and Touch	In the Timeout branch (if a new message is not created), touches the new message icon.
Wait2	Wait	In the Timeout branch, waits for a second after a new message has been created.
Phone#	Find and Touch	In the Tie branch of Wait Event, enters a phone number.
TouchMsgBody	Find and Touch	Selects the field for the message body.
TypeMessage	Find and Touch	Enters the message text.
TouchSend	Find and Touch	Selects Send to route the message.

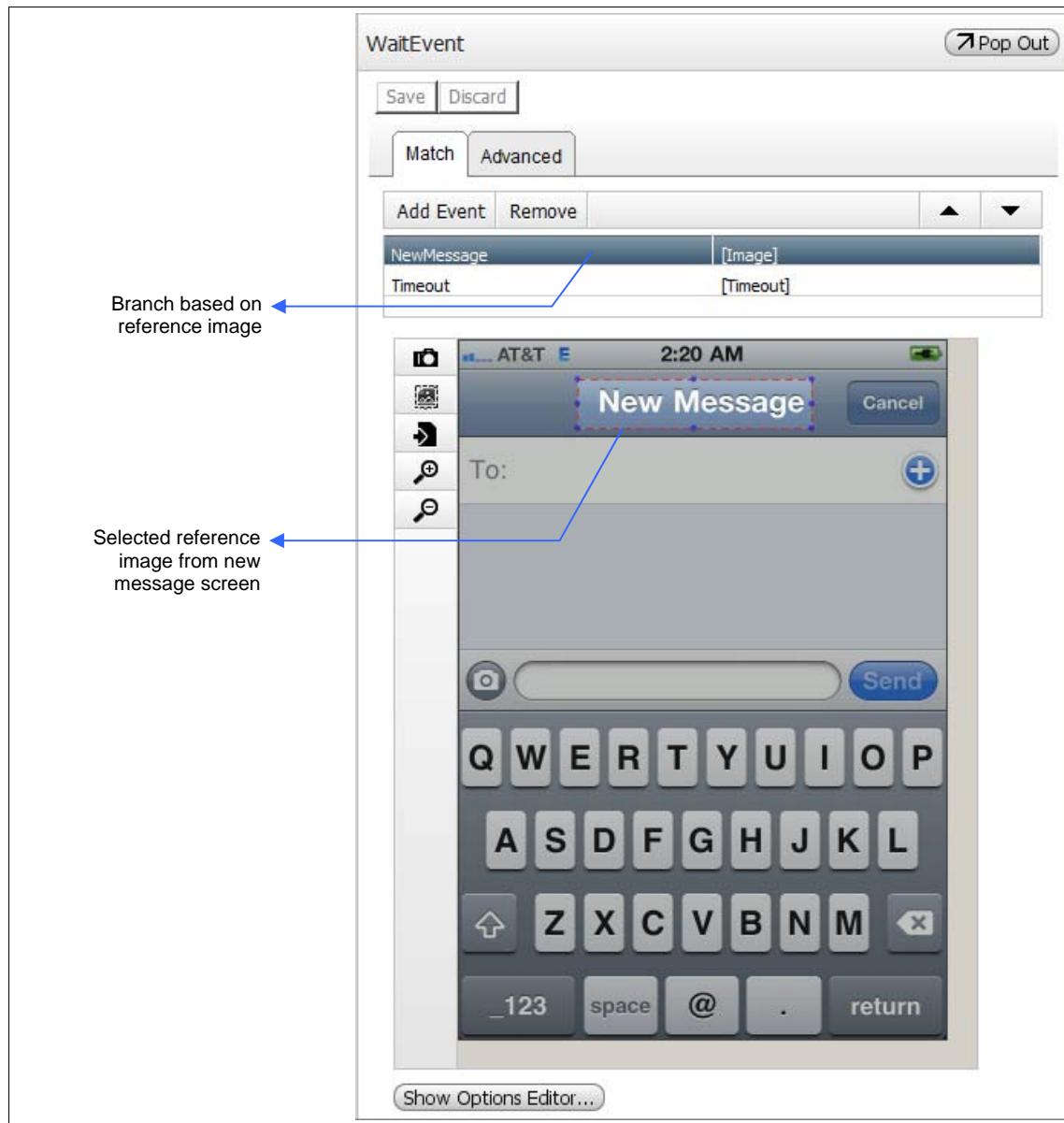
The figure below shows the Find and Touch command for selecting the messaging application.

Figure 11-6 Find and Touch Messaging Application



The figure below shows the Wait Event command with the branch NewMessage, based on a reference image. The reference image is a small area of the new message screen.

Figure 11-7 Wait Event Command Properties



11.3 Action: Adding a Contact

This [action](#) implementation adds several contacts on an iPhone 3GS device and exemplifies the use of the [Loop](#) command with a data set. The data contains one field for the contact name and another for a phone number. Within the loop, Send Key and Find and Touch commands enter the contact name and then the contact number.

NOTE As a best practice, separate out the commands for entering data set values from commands for other key presses.

The commands in the loop run iteratively until all records in the data set have been into the contacts list. Outside the loop, a command returns the device to the home screen.

For more information on using Loop with a data set, see [Working with Data Sets](#) in the section [Parameters and Variables](#) and [Loop](#) in the [Command Reference](#).

Figure 11-8 Loop Data Set Example

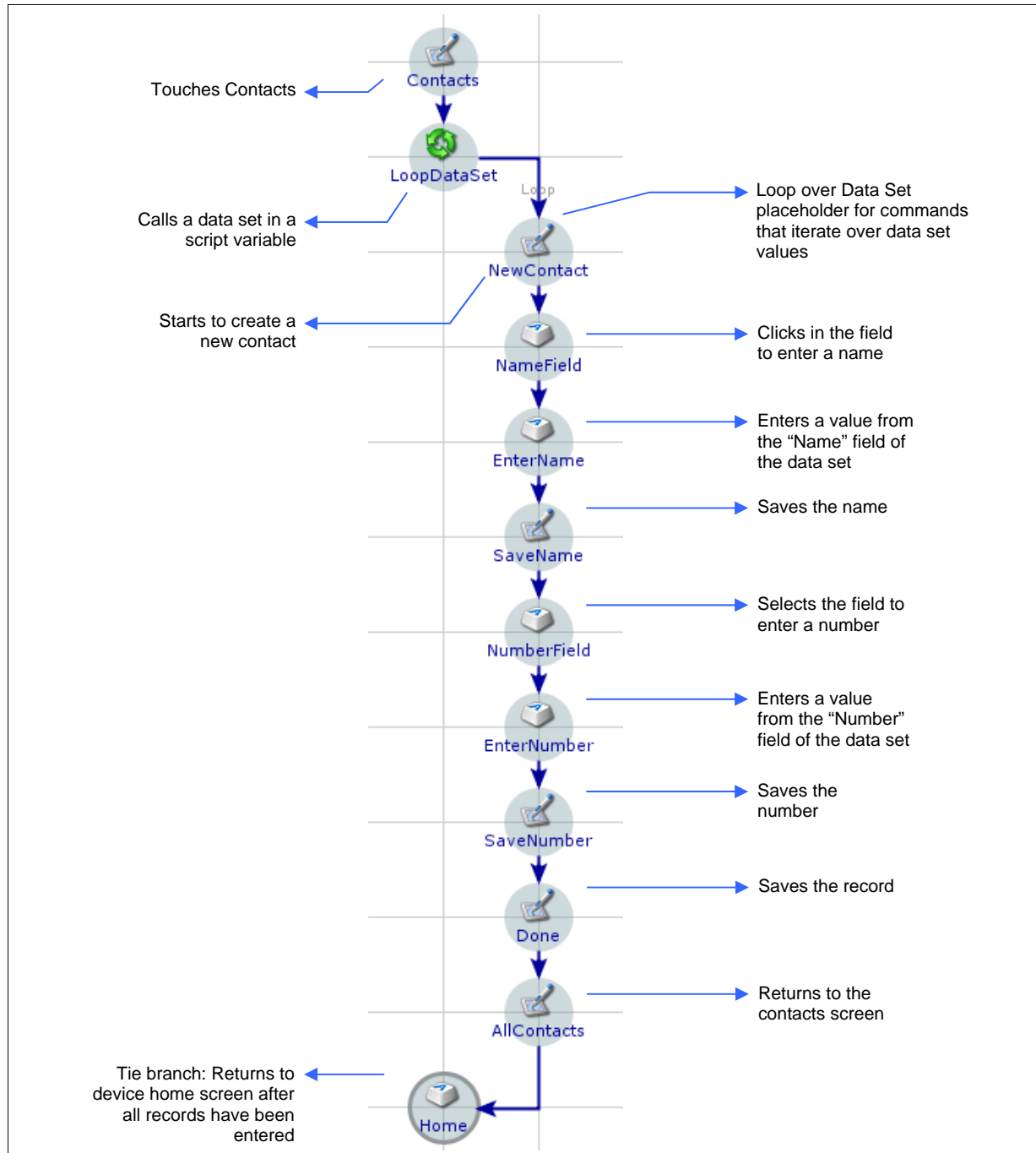


Table 11-3 Adding a Contact on an iPhone—Commands

Name	Command	Description
Contacts	Find and Touch	Opens the Contacts application.
LoopDataSet	Loop	Calls an action variable associated to a data set with two fields and three records. Opts to loop over all records (see Figure 11-9).
NewContact	Find and Touch	Touches button to create a new contact.
NameField	Send Keys	Clicks in the field to enter contact name.
EnterName	Send Keys	Inserts a value from the “Name” field/column of the data set specified in the Loop command (see Figure 11-10).
SaveName	Find and Touch	Saves the name just entered.
NumberField	Send Keys	Clicks in the field to enter a contact number.
EnterNumber	Send Keys	Inserts a value from the “Number” field/column of the data set specified in the Loop command.
SaveNumber	Find and Touch	Saves the number just entered.
Done	Find and Touch	Saves the contact.
AllContacts	Find and Touch	Returns to the list of all contacts.
Home	Send Keys	Returns the device to the home screen.

Figure 11-9 Loop Command Calling All Records in a Data Set

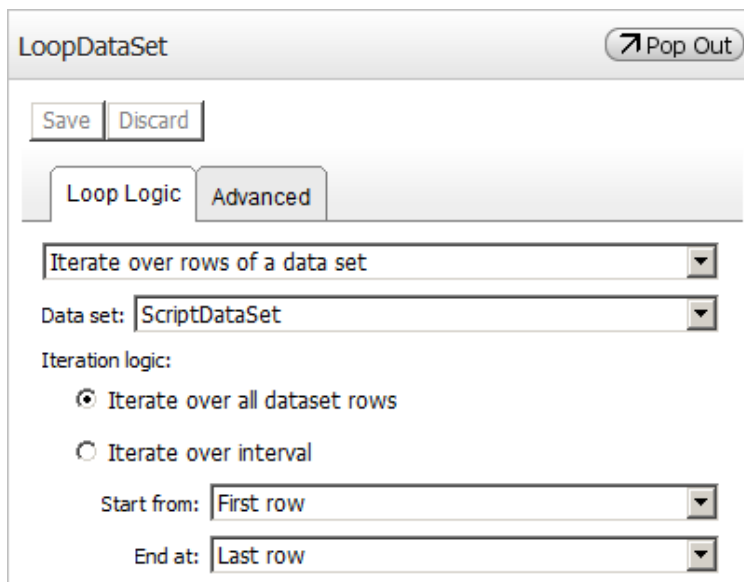
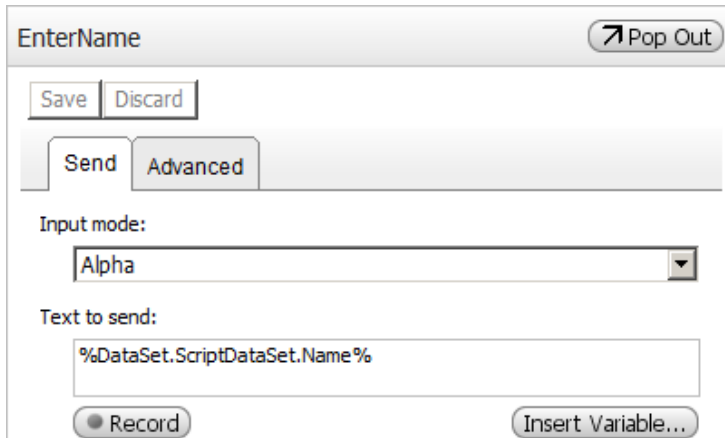


Figure 11-10 Send Keys Inserts Values from the "Name" Field of a Data Set



11.4 Action: Unpartitioned Web Action

This unpartitioned action implementation uses Web commands to navigate to an airline site, enter search criteria, and verify results. The action is unpartitioned as it does not contain any device-specific commands. As a best practice, isolate [Web commands](#) in a separate unpartitioned action so that you do not have to create an implementation for each device in your project.

Figure 11-11 Web Commands Example

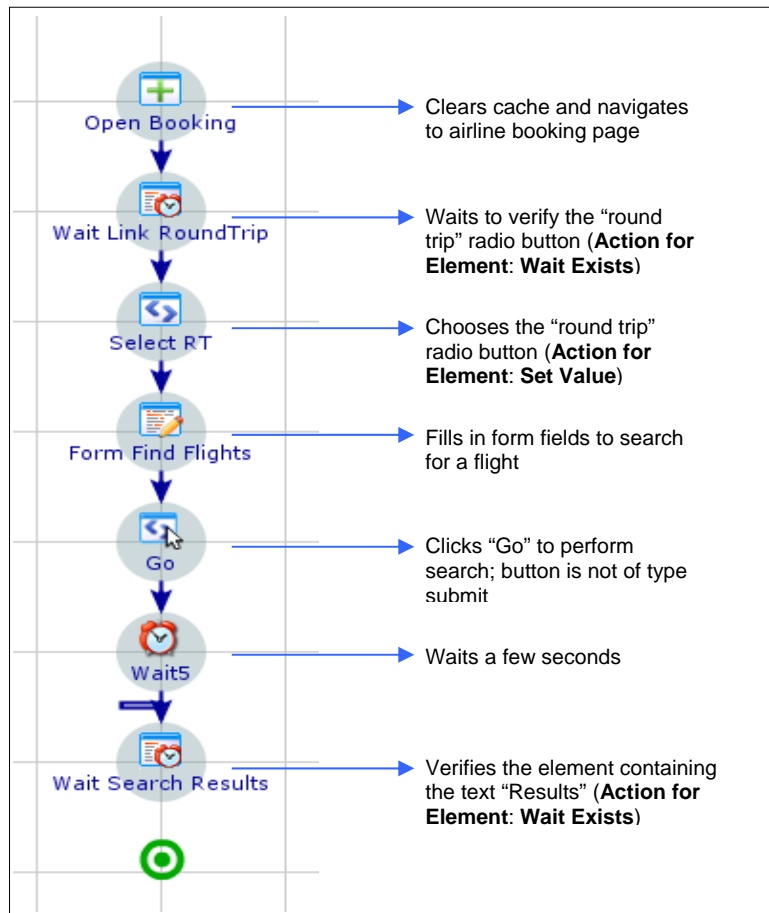


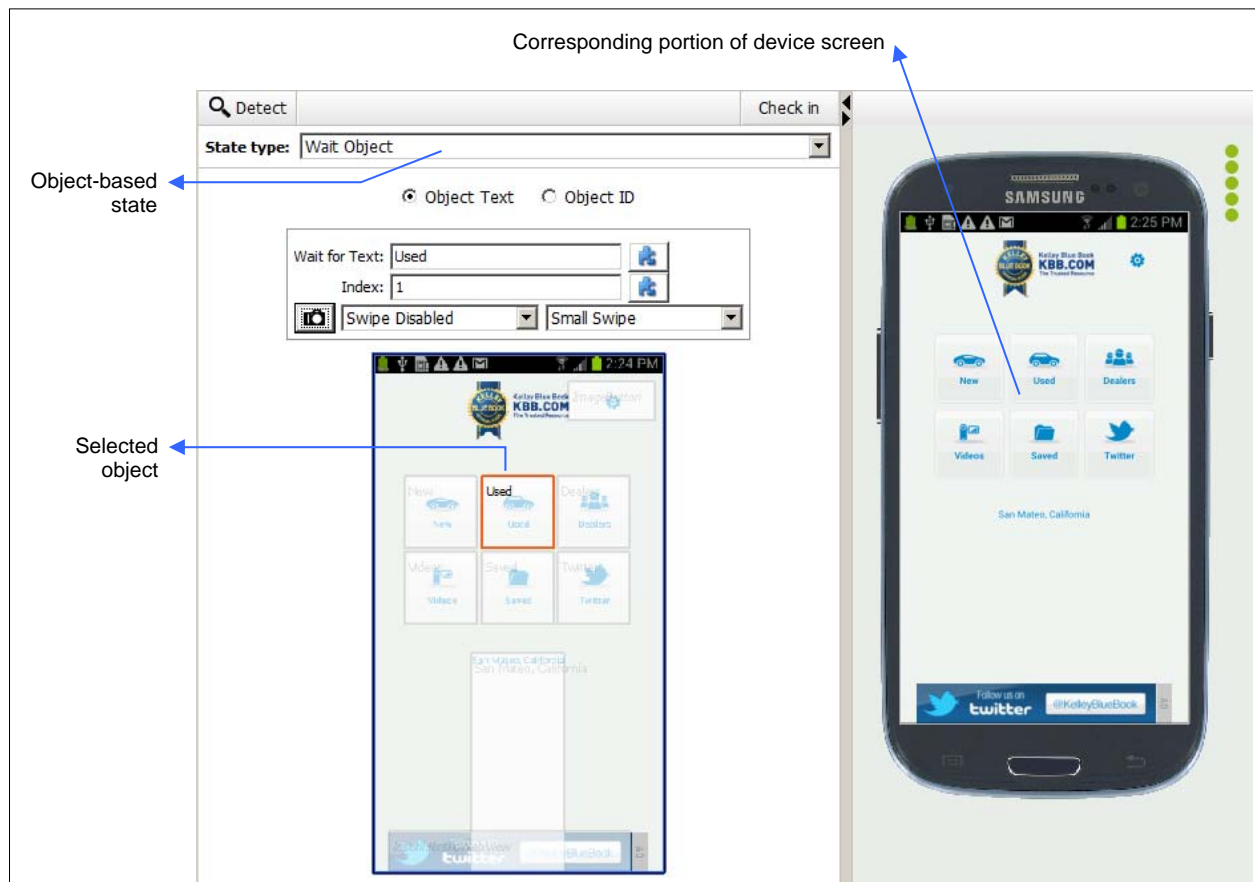
Table 11-4 Searching for Flights—Commands

Name	Command	Description
Open Booking	Open Browser	Clears native browser cache and navigates to airline booking page.
Wait Link Round Trip	Web Wait	Waits to verify the “round trip” radio button. The corresponding element is chosen in the command and the action Wait Exists is specified.
Select RT	Web Element	Chooses the element for fare type (round-trip or one-way) and clicks the “round-trip” radio button. The action for the element is Set Value .
Form Find Flights	Web Form	Fills in form fields with drop-down lists to specify travel cities, dates, and number of passengers.
Go	Web Touch	Clicks a button to perform a flight search after all form fields have been filled out; the button is not of type submit.
Wait5	Wait	Waits for a few seconds while the search is performed.
Wait Search Results	Web Wait	Verifies the element containing the text “Results” by choosing the action Wait Exists .

11.5 State: Application Home Screen

The figure below shows an object-based [state](#) implementation to verify the home screen of a native application. The selected object is highlighted in command properties. See [Waiting for an Object](#) for detailed instructions on defining object-based reference points.

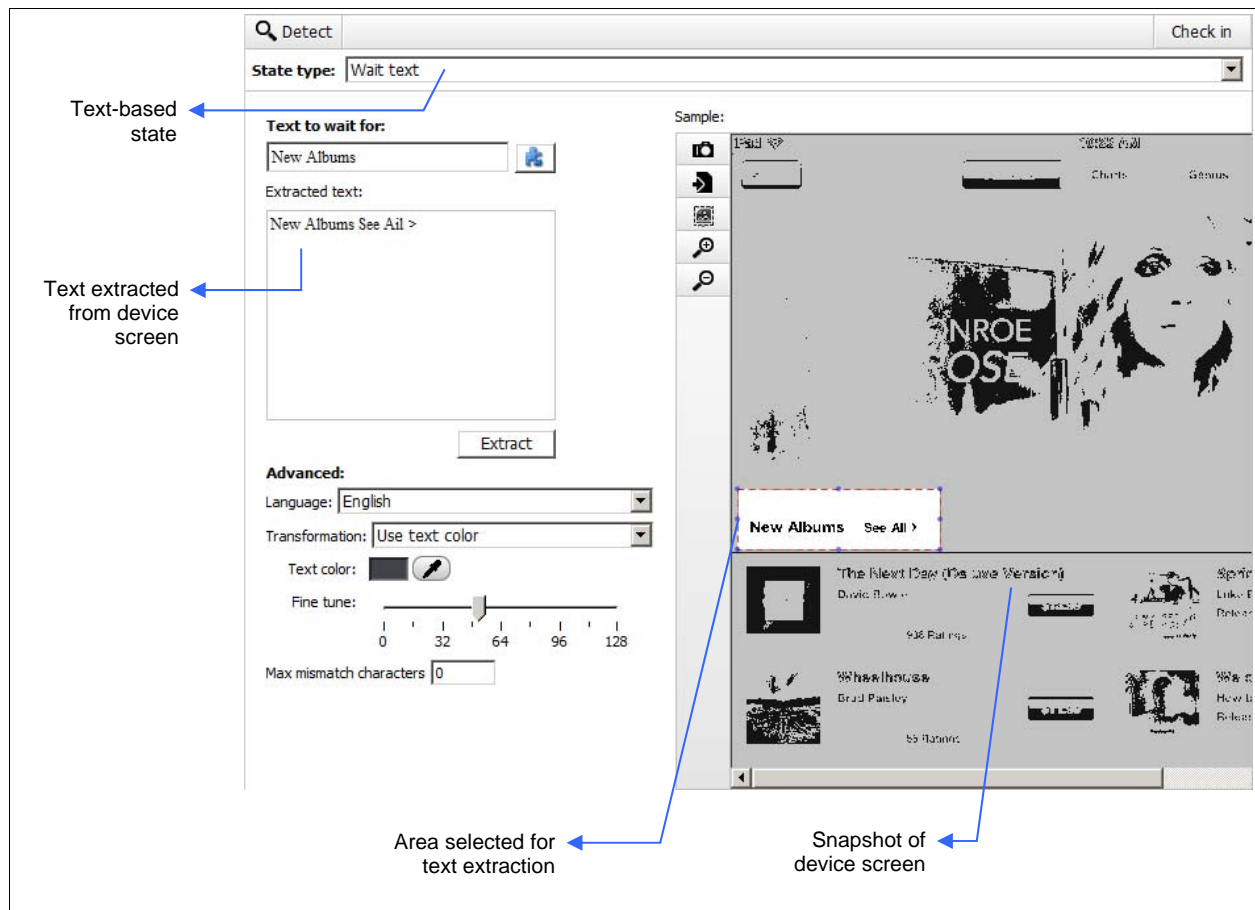
Figure 11-12 Object-Based State



11.6 State: Create Email Message Screen

The figure below shows a text-based [state](#) implementation to verify the message creation screen on an iPhone 4 device. A subset of text extracted from the device screen is used as a reference string. TCE Automation uses optical character recognition (OCR) technology to match the reference string to the actual device screen. See [Text-Based Reference Points](#) for detailed instructions on defining reference strings.

Figure 11-13 Text-Based State



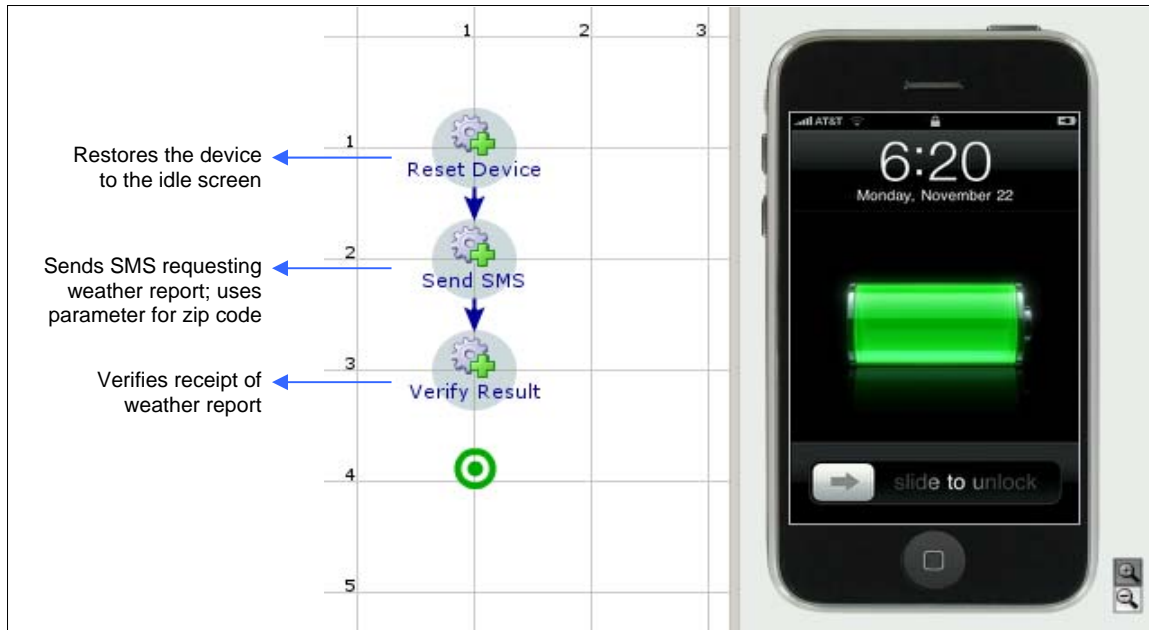
11.7 Test Case: Requesting and Receiving the Weather Forecast

The [test case](#) script below requests the weather report for a zip code from a short code service. It then receives the weather report. This script exemplifies the use of [parameters](#) and the [Execute Action](#) command. The test case calls three actions, the first to reset the device to the idle screen, the second to send an SMS requesting the weather report, and the third to verify receipt of the weather report.

This test case does not contain any device-specific commands. At runtime, the correct implementation of each action for the chosen device (in this case, an iPhone 3G) is loaded.

NOTE If an implementation for the chosen device does not exist, the test case script will fail.

Figure 11-14 Test Case Example

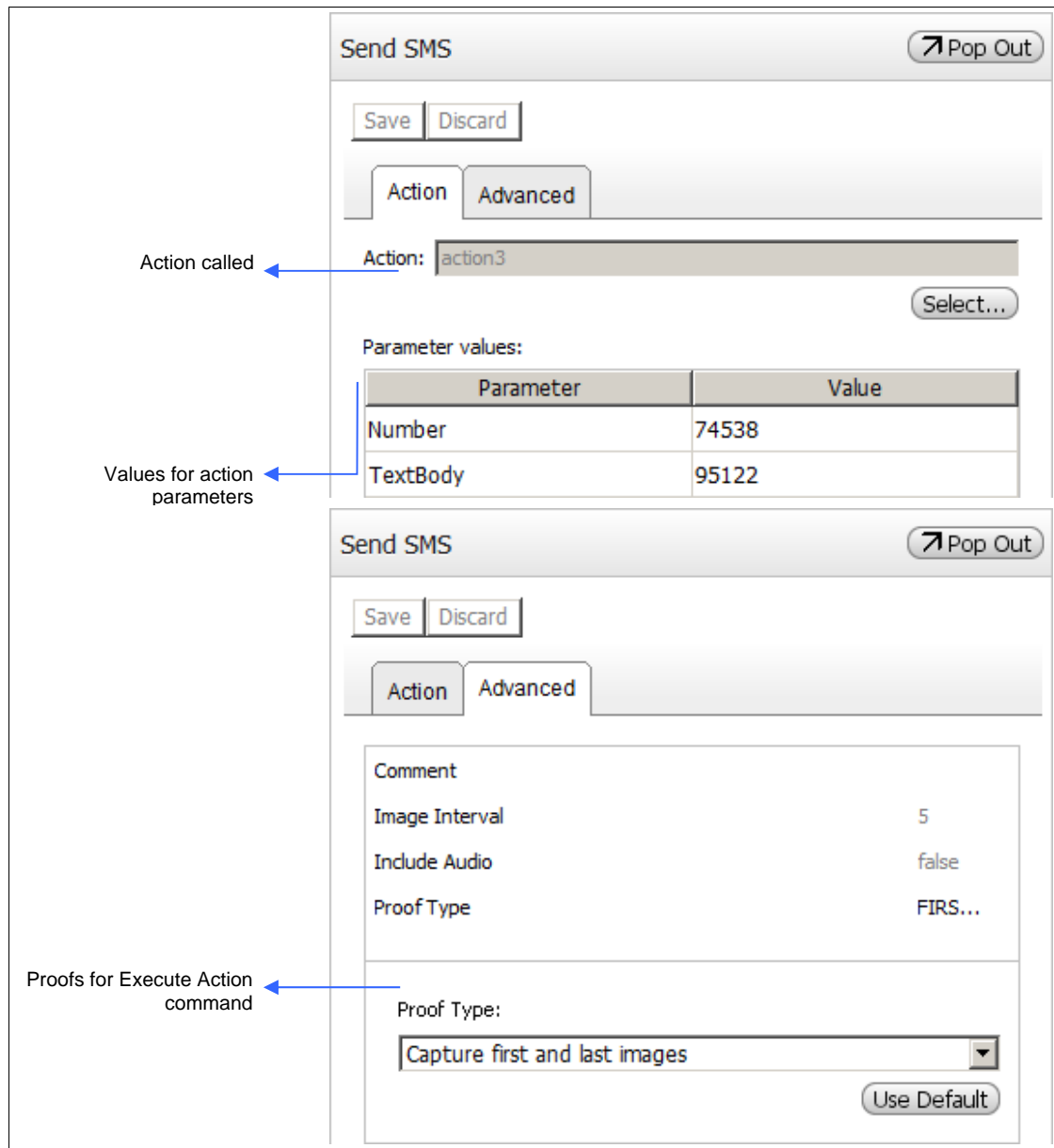


The table below explains each command in detail:

Table 11-5 Requesting the Weather Report Example—Commands

Name	Command	Description
Reset Device	Execute Action	Calls an action to reset the device to the idle screen and specifies the last image as proof.
Send SMS	Execute Action	Calls an action to send an SMS to a short code weather service, specifying a zip code in the message body. <i>See Figure 11-15 below.</i> This command also specifies values for action parameters: <ol style="list-style-type: none"> 1 The phone number to send the message to, i.e., the short code service 2 The body of the message, i.e., the zip code
Verify Result	Execute Action	Calls an action to verify that a new message with the weather report for the specified zip code has been received. Specifies a value for an action parameter for the name of the corresponding city.

Figure 11-15 Execute Action Command Properties

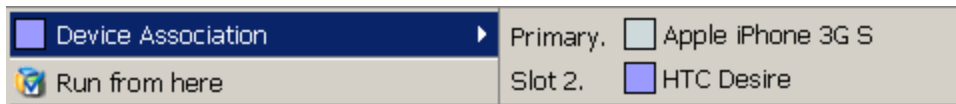


11.8 Multi-Device Test Case: Audio Validation

The script below is a two-device [test case](#) for audio validation. While the calling device plays a DTMF sequence, the receiving device listens for the tones. This script exemplifies the concurrent use of the [Play Audio](#) and [Wait Event \(audio\)](#) commands on two different devices. These commands

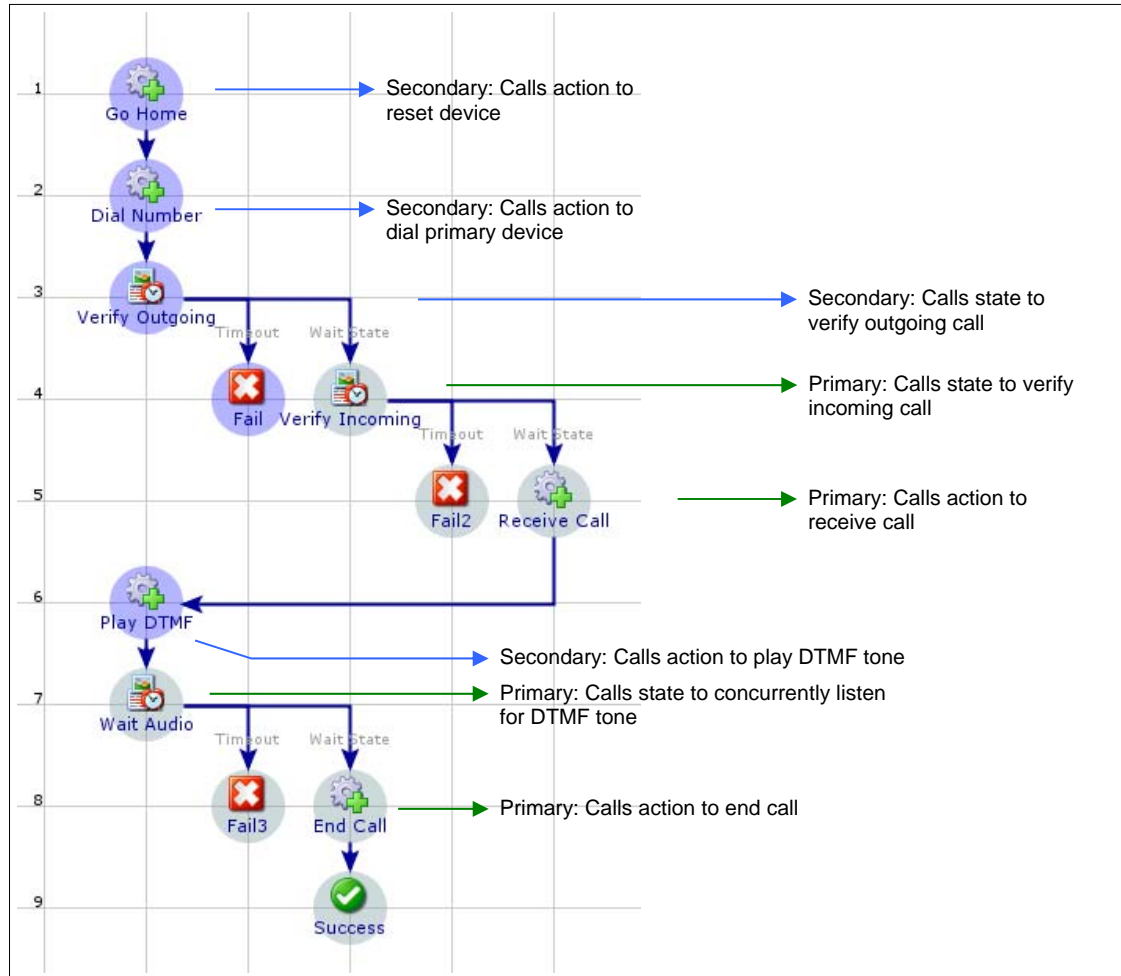
The commands for each device are color coded. Right-clicking any command in the script and selecting **Device Association** displays the primary and secondary devices selected to run the test case on.

Figure 11-16 Device Association in Two-Device Test Case



NOTE To use a reference point in a test case, you can call an action containing a reference point. You can also drag in a state from the project directory, which automatically inserts a Wait Event command in the test case script (shown in Figure 11-17 below).

Figure 11-17 Play and Wait for Audio Example



The table below explains each command in detail:

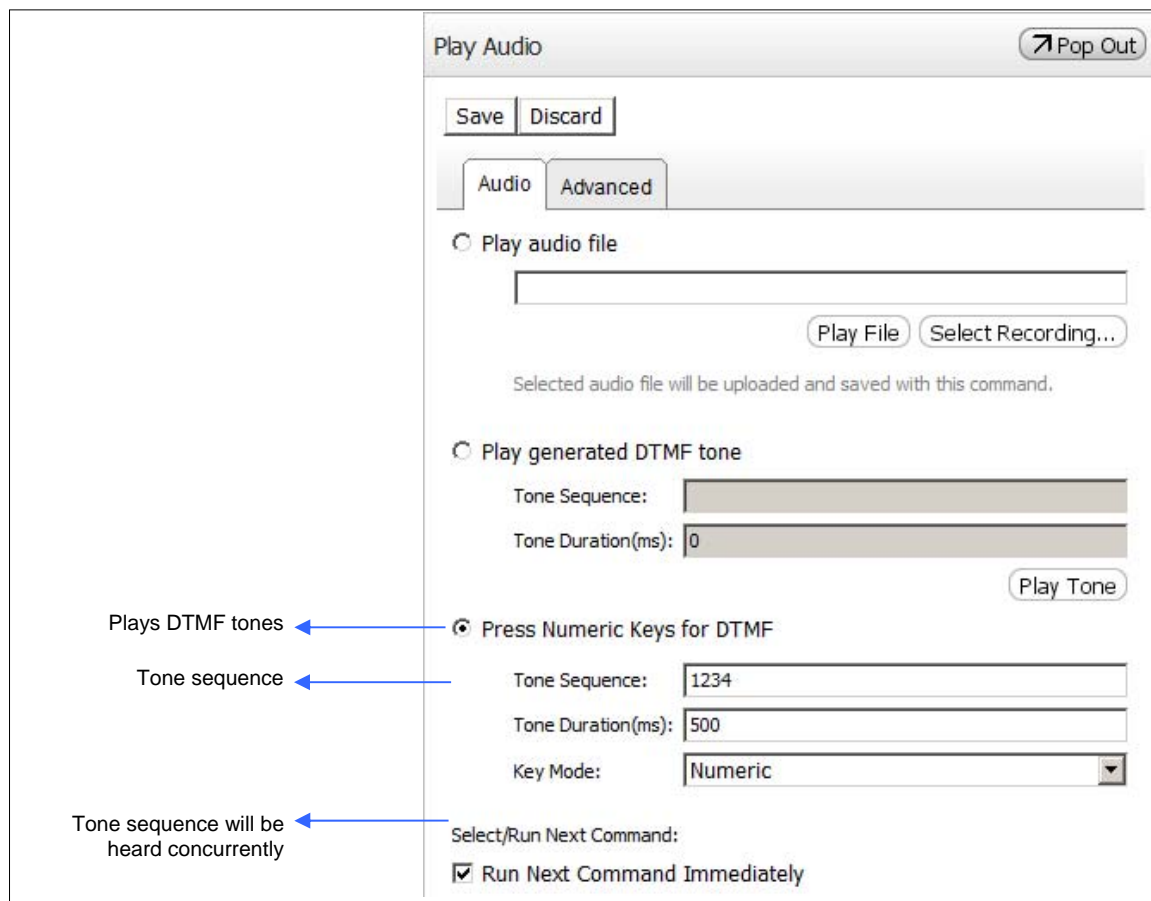
Table 11-6 Audio Validation Test Case for Two Devices—Commands

Name	Command	Device	Description
Go Home	Execute Action	Secondary	Calls an action to reset the device to the home screen.
Dial Number	Execute Action	Secondary	Calls an action to launch the key pad, type in the number of the primary device, and place the call.
Verify Outgoing	Wait Event (state)	Secondary	State to verify outgoing call dragged into script
Verify Incoming	Wait Event (state)	Primary	State to verify incoming call dragged into script

Name	Command	Device	Description
Receive Call	Execute Action	Primary	Accepts incoming call and verifies active call.
Play DTMF	Execute Action	Secondary	The action contains Play Audio to play a DTMF sequence to the listening device. This command runs concurrently with the audio verification in the following command. <i>See Figure 11-18 below.</i>
Wait Audio	Execute Action	Primary	The action contains Wait Event, which listens for the DTMF sequence played by the secondary device.
End Call	Execute Action	Primary	Ends the call.

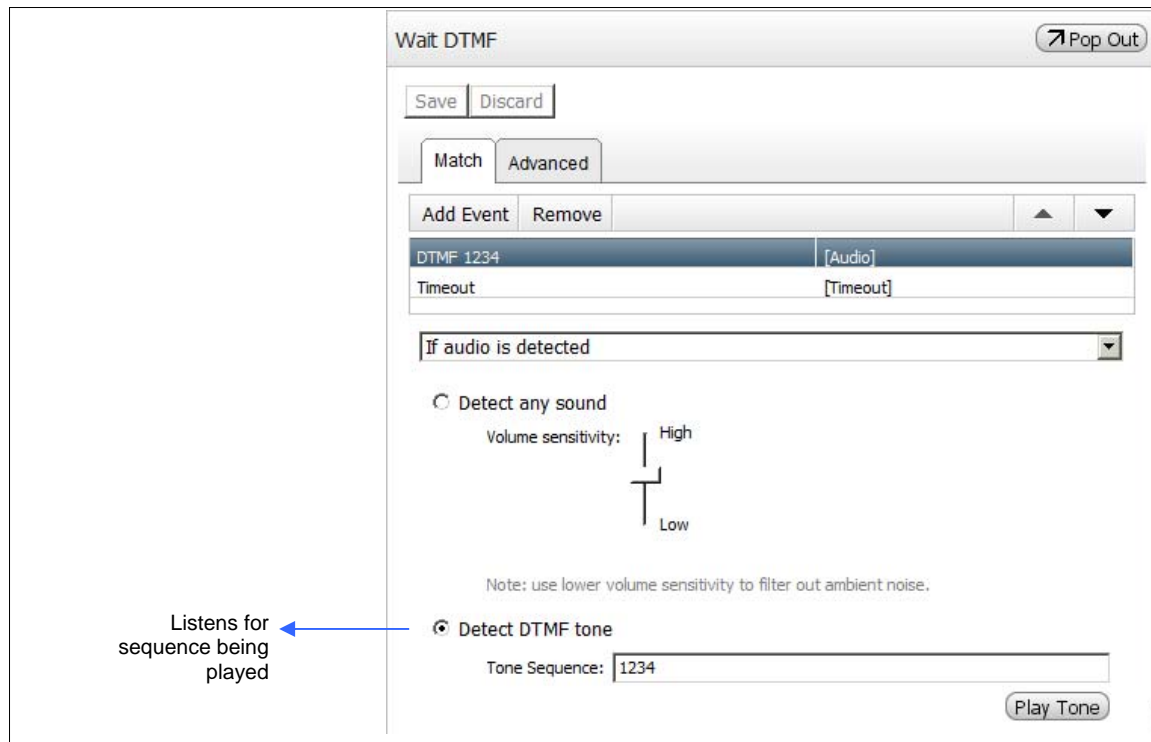
The figure below shows the Play Audio command on the secondary device for playing a DTMF sequence to the primary device. The **Run Next Command Immediately** option is checked to ensure that the primary device can concurrently listen for the DTMF tones.

Figure 11-18 Play Audio Properties



The figure below shows the Wait Event command that enables the primary device to listen for DTMF tones as they are being played.

Figure 11-19 Wait Audio Properties



11.9 Test Cycle: Saving and Using Extracted Text

The [test cycle](#) script below opens an iPhone weather application to view an updated weather report, then extracts the report timestamp to a variable and uses it in a note. This test cycle exemplifies the use of the [Execute Test Case](#) and [Execute Cleanup Action](#) commands. The test cycle calls three scripts:

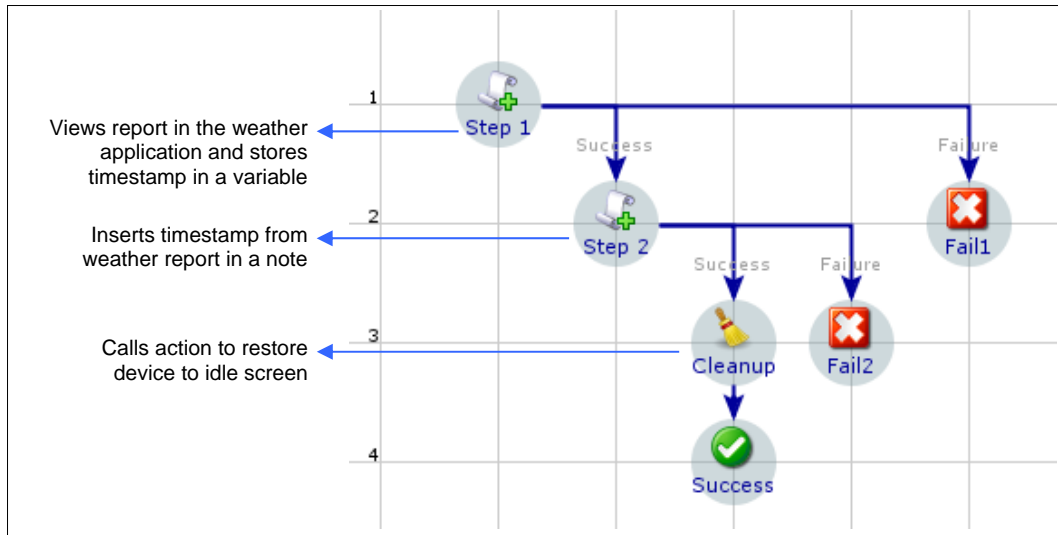
- ◆ A test case that views the updated weather report and extracts the timestamp to a variable (Step 1)
- ◆ A test case that inserts the extracted timestamp in a new note (Step 2)
- ◆ An cleanup action that restores the device to the idle screen (Cleanup)

This test cycle does not contain any device-specific commands. At runtime, the correct implementation of each script for the chosen device (in this case, an iPhone device) is automatically loaded.

NOTE If an implementation for the chosen device does not exist, the test cycle script will fail.

The test cycle script also contains explicit success and failure points.

Figure 11-20 Test Cycle Example



The table below explains each command in detail:

Table 11-7 Test Cycle Example—Commands

Name	Command	Description
Step 1	Execute Test Case	Calls a test case to open and verify the weather application, view and verify an updated weather report, and extract the timestamp to a variable.
Step 2	Execute Test Case	Executed if Step 1 succeeds – calls a test case to open the note application and create a new note containing the extracted timestamp from the weather report.
Fail 1	Fail	Fails the test cycle if Step 1 is not successful.
Cleanup	Execute Cleanup Action	Executed if Step 2 succeeds – Calls an action to restore the device to the idle screen. Proofs from this command and the action called are <i>not</i> inserted into test results.
Fail 2	Fail	Fails the test cycle if Step 2 is not successful.
Success	Success	Terminates the test cycle successfully after Step 1, Step 2, and Cleanup have been executed.

12 Command Reference

This reference describes visual scripting commands. For each command, images and tables describing controls on each tab are displayed. (The controls common to all commands are described in [Command Properties Pane](#).)

12.1 Find and Touch

On touchscreen devices, this command searches for and touches the center of a specified screen region, a text string, or either of these captured in a state. You can also opt to touch a point at an offset from the image or text selected. You would do this for instance, when you want to click in a specific data entry field on a page where all the fields look alike. You then define the field as an x and/or y offset from a unique screen region next to the field.

The screen region or text to touch is defined in much the same way as image- or text-based [reference points](#).

12.1.1 Touching an Image

To use the command to touch an image:

- 1 Acquire the device and navigate to the appropriate screen.
- 2 Drag the command onto the script canvas.
- 3 Select **Find an Image**.
- 4 Capture the screen using the camera button in the command.
- 5 Use your mouse to select a screen region to touch.
- 6 Optionally, specify an x or y offset from the selected region to touch.
- 7 Optionally, define a range of match and mismatch possibilities and see if they match the actual device screen.

NOTE See [Image-Based Reference Points](#) for detailed information on advanced image match settings.

- 8 Optionally, specify *tolerance levels* for variations between the reference image(s) and the actual device screen—you can adjust tolerance settings to include matches and exclude mismatches.
- 9 **Save** advanced settings, **Save** the command.

In the image below, the command selects the name of a field in the Gmail sign in page on an Android browser. However, the command is set to touch (click in) a field that is offset from the center of the selected region.

Figure 12-1 Find and Touch Image

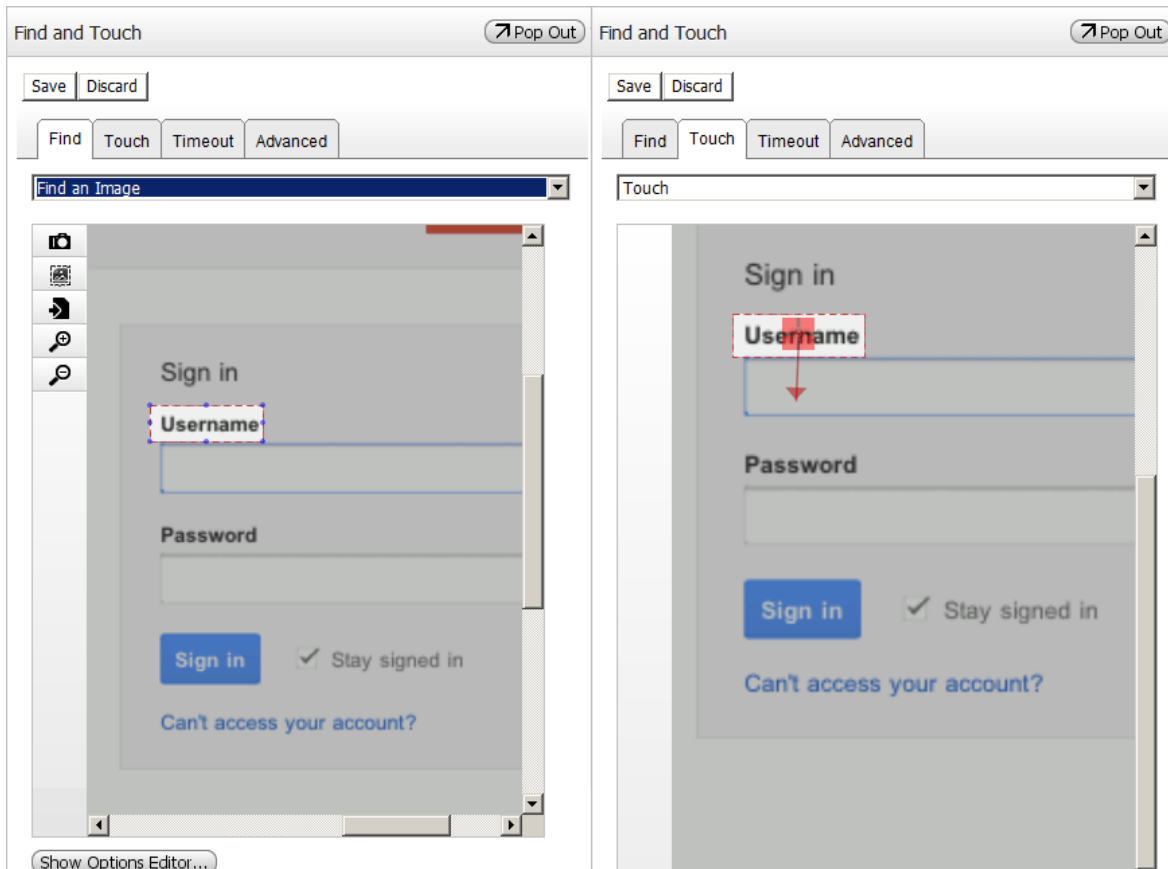











Table 12-1 Find and Touch Command Properties—Image

Tab	Field/Control	Description
Find	Find an Image	Select to search for and touch a screen region.
	Selection controls	Camera icon  – Click to take snapshot of current device screen. Clear Selection  – Click to reset device image area. Import image  – Click to import device image (from DeviceAnywhere test results that were directly captured from the device and uploaded to the web portal). Zoom buttons  – Click to resize captured image of device screen in command.
	Show Options Editor	Opens controls for specifying match and mismatch ranges as well as tolerance levels for variance between reference images and the actual device screen. Allows you to troubleshoot areas of mismatch between reference images by adjusting tolerance sliders.

Tab	Field/Control	Description
Touch	Selection controls	 Click and hold down your mouse in the center of the red box displayed. Then drag your mouse to the offset you want touched  A red arrow indicates the point to be touched. You can also adjust the x and y coordinates captured in the text boxes provided. <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> Touch at (<input type="text" value="-1"/> , <input type="text" value="61"/>) offset from the center of found region. </div>   – Click to resize captured image of device screen.
Advanced	Hold Time	Enter time in milliseconds to touch screen region.
	Comment	Enter an optional comment in the field provided.
	Check Point	Select true to capture the first and last screenshots during command execution, false (default) not to capture any proofs.
Timeout	Wait Time	Time within which the reference image must be found.
	Timeout Action	Specifies what to do if the image is not found: Continue with script – The script does not fail. Return failure immediately – The script fails when image is not matched.
	Main Category	Optional: Select an error category from the drop-down list.
	Error Type	Select an error type from the list displayed – this error type will be triggered when the command times out.
	Severity	Error type severity inherited from error definition in project properties
	Error Code	Error code inherited from error definition in project properties
	Description	Error description inherited from error definition in project properties
	Add Notes	Check to append notes to the error description. Enter notes in the box provided. Click the puzzle piece icon  to pass in a value from a variable.

In the options editor, you can adjust tolerance sliders to troubleshoot mismatches between the reference image and the device screen. Tolerance levels specify the variations between the reference image and the actual device screen. (Refer to [Image-Based Reference Points](#) for details on advanced image settings.)

Figure 12-2 Find and Touch Advanced Settings

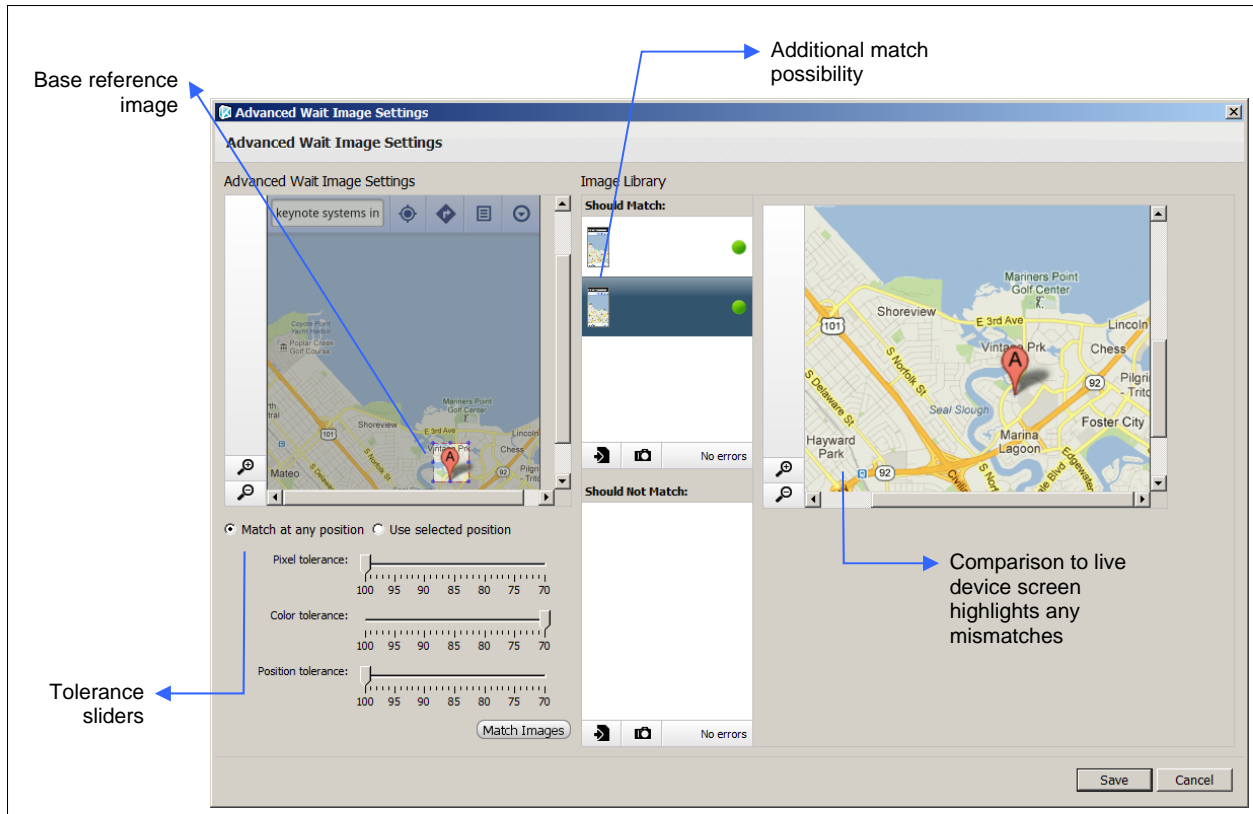









Table 12-2 Find and Touch (Image) Options Editor

Control/Field	Description
 Zoom buttons	Use to enlarge or shrink the device screen display; available for the base reference image as well as comparison image.
 Camera icon	Click to take snapshot of current device screen – the image taken is a candidate for match (Should Match) or mismatch (Should Not Match). The image is automatically compared to the base image and areas of mismatch are highlighted in pink. Images that correctly match or do not match base image are displayed with a green icon  . Images that incorrectly match or do not match the base image are displayed with a red icon  .
 Import image icon	Click to import an image as a candidate for match (Should Match) or mismatch (Should Not Match). The image is automatically compared to the base image and areas of mismatch are highlighted in pink. Images that correctly match or do not match the base image are displayed with a green icon  . Images that incorrectly match or do not match the base image are displayed with a red icon  .
Pixel tolerance slider	Use to set the percentage of pixels to match, with 100 being a strict match.
Color tolerance slider	Use to set the color matching required between the reference image(s) and the device screen, with 100 being a strict match.
Position tolerance slider	Use to set the radius around each pixel’s original position in which to look for the pixel, with 100 being a strict match (or the tightest radius).

Control/Field	Description
Match Images	Click to match images after changing tolerance levels or redefining the reference image area.
Save	Save advanced settings.
Cancel	Click to exit advanced settings without saving changes.

You can right-click any image in the **Should Match** or **Should Not Match** library for additional controls:

- ◆ **Make Base Image**—Replaces base image with selected image.
- ◆ **Should Not Match**—Moves an image from the match to the mismatch library.
- ◆ **Should Match**—Moves an image from the mismatch to the match library.
- ◆ **Remove From Library**—Deletes image from library.

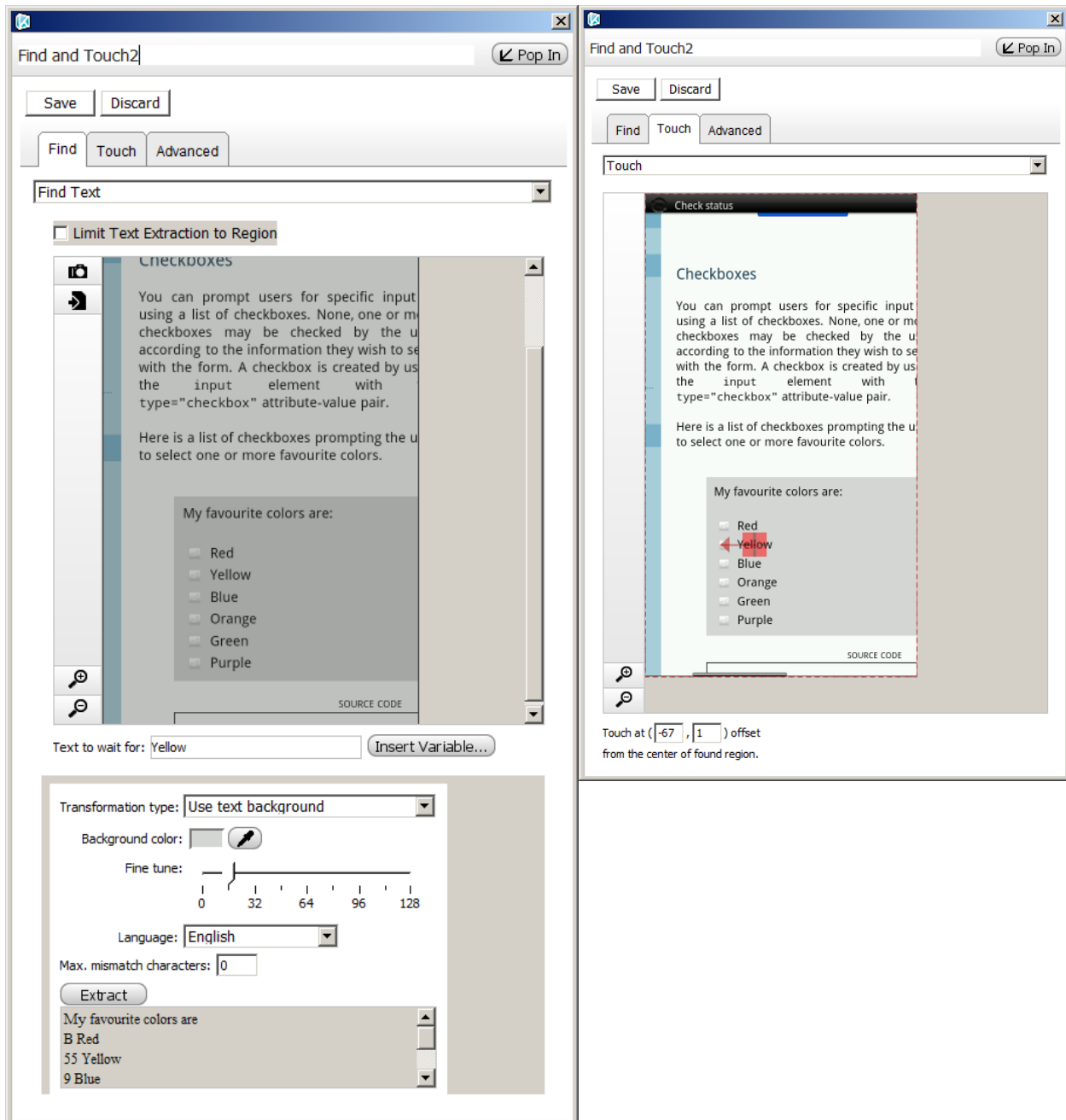
12.1.2 Touching a Text String

The Find and Touch command can search for and touch a text string. In this mode, Find and Touch uses character recognition technology instead of image matching to perform a touch.

You can also touch a screen region at an offset from the chosen text string (e.g., press a button to the left of the text). The area to be touched is defined in terms of the x and/or y offsets from the center of the text string.

In Figure 12-3 below, the command is set up to check a box next to a string of text. The box is defined in terms of the x offset from the center of the text string.

Figure 12-3 Find Text and Touch Text








To use the command to touch a text string:




- 1 Acquire the device and navigate to the appropriate screen.
- 2 Drag the command onto the script canvas.
- 3 Select **Find Text**.
- 4 Capture the screen using the camera button in the command.
- 5 Optionally, define a screen region from which to extract text.

- 6 Choose a transformation method and specify settings.
- 7 **Extract** text.
- 8 Define the text string to wait for.
- 9 Optionally, define a touch offset.
- 10 Save the command.

NOTE See [Text-Based Reference Points](#) for detailed information on text matching.

Table 12-3 Find and Touch Command Properties—Text

Tab	Control/Field	Description
Find	Find Text	Select to search for and touch a string of text.
	Selection Controls	<p>Camera icon  – Click to take snapshot of current device screen.</p> <p>Clear Selection  – Click to reset device image area.</p> <p>Import image  – Click to import device image (from DeviceAnywhere test results that were directly captured from the device and uploaded to the web portal).</p> <p>Zoom buttons  – Click to resize captured image of device screen in command.</p>
	Limit Text Extraction to Region	Check to limit text extraction to a selected screen region. Use your mouse to select the region.
	Language drop-down list	Select a language for the text string.
	Transformation type drop-down list	<p>Use text color – Select to choose text color with eyedropper icon.</p> <p>Use text background – Select to choose text background with eyedropper icon.</p> <p>Convert image to black and white – Select to convert image of device screen to black and white; choose color that separates black from white with eyedropper icon.</p> <p>Adjust contrast – Select to change contrast of device screen image; choose delimiting color with eyedropper icon.</p> <p>See Text Transformations for an explanation.</p>
	Eyedropper 	Click button and hover over image of device screen to choose a color while transforming text. Click image to select a color.
	Fine tune	Adjust to aid text extraction – use after choosing transformation type and color. Set the slider as high as possible in order to eliminate false positive matches. See Text Transformations for details on the tolerance slider for each transformation type.
	Max. mismatch characters	Enter the maximum number of characters that can be mismatched while identifying the Text to Wait For , e.g., if you specify that any two characters can be mismatched in the string “California,” the term “Callfornla” will be considered a match.
	Extract	Click after choosing transformation settings.
	Text to Wait For	Enter a subset of extracted text as the string of text to touch. Click Insert Variable to insert a variable or parameter in place of entering

Tab	Control/Field	Description
		the text string to wait for.
Touch	Selection controls	 Click and hold down your mouse in the center of the red box displayed. Then drag your mouse to the offset you want touched  . A red arrow indicates the point to be touched. You can also adjust the x and y coordinates captured in the text boxes provided. <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> Touch at (<input type="text" value="-67"/> , <input type="text" value="1"/>) offset from the center of found region. </div>  Zoom buttons – Click to resize captured image of device screen in command.
Advanced	Hold Time	Enter time in milliseconds to touch screen region.
	Comment	Enter an optional comment in the field provided.
	Check Point	Select true to capture the first and last screenshots during command execution, false (default) not to capture any proofs.

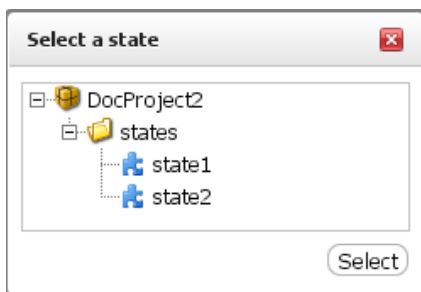
Use the [Timeout tab](#) to define a command completion time and trigger error reporting when the command fails.

12.1.3 Touching a State

Find and Touch can search for and touch a text string or reference image defined in a state. It can also touch a point at an offset from the image or text in the selected state.

To touch an image or text in a state:

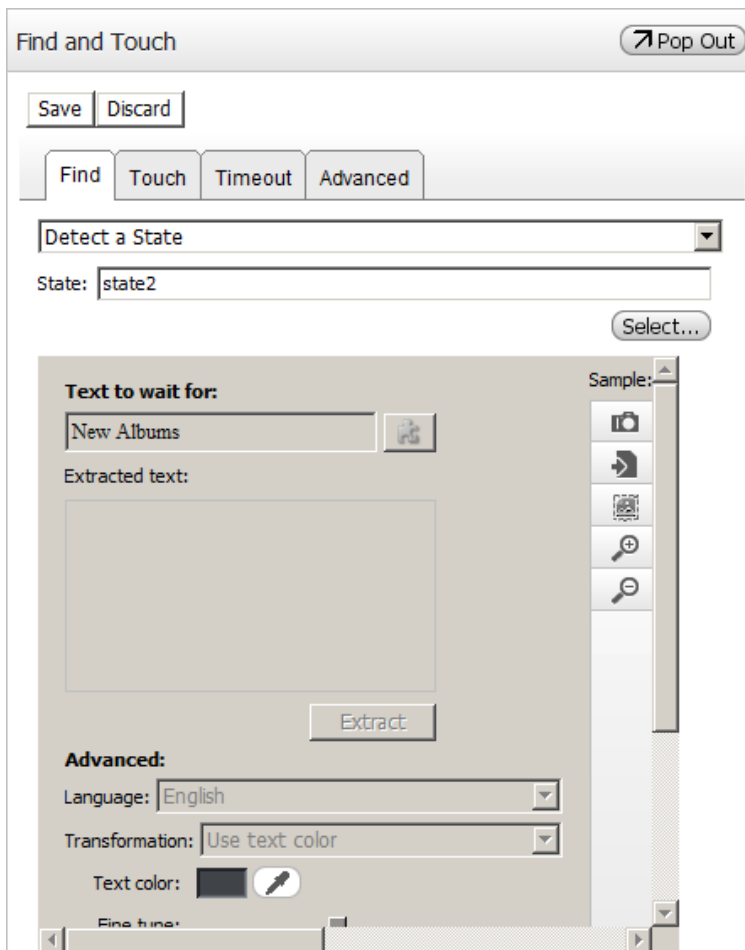
- 1 Drag the command onto the script canvas.
- 2 Select **Detect a State**.
- 3 Click **Select**. The Select a state dialog box lists available states in the current project.



- 4 Select a state from the list of project states and click **Select**.



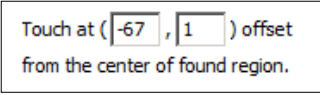




The Find and Touch command displays the selected **State**. You can change your selection at any time in command properties.



- 5 Optionally, specify an x or y offset (in pixels) from the selected region to touch.
- 6 Specify other settings (see the table below) and **Save** the command.

Table 12-4 Find and Touch State Command Properties—State

Tab	Control/Field	Description
Find	Detect a State	Select to search for and touch a state.


Tab	Control/Field	Description
	Select	Click to select a state from the current project. In the dialog box that appears, select a state and click Select .
Touch	Offset controls	Specify x and y offsets (positive or negative) in the text boxes provided.   Zoom buttons   – Click to resize captured image of device screen in command.
Advanced	Hold Time	Enter time in milliseconds to touch screen region.
	Comment	Enter an optional comment in the field provided.
	Check Point	Select true to capture the first and last screenshots during command execution, false (default) not to capture any proofs.
Timeout	Wait Time	Time within which the state must be found.
	Timeout Action	Specifies what to do if the state is not found and matched: Continue with script – The script does not fail. Return failure immediately – The script fails when the state is not matched.
	Main Category	Optional: Select an error category from the drop-down list.
	Error Type	Select an error type from the list displayed – this error type will be triggered when the command times out.
	Severity	Error type severity inherited from error definition in project properties
	Error Code	Error code inherited from error definition in project properties
	Description	Error description inherited from error definition in project properties
	Add Notes	Check to append notes to the error description. Enter notes in the box provided. Click the puzzle piece icon  to pass in a value from a variable.

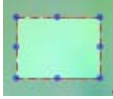
Use the [Timeout tab](#) to define a command completion time and trigger error reporting when the command fails.

12.1.4 Swipe

While you can record a swipe by holding down the Ctrl key and dragging the mouse across the device screen in [record mode](#), you can also program a swipe in Find and Touch. You can select a screen region or text as the swipe start point and then define the swipe trajectory and end point.


To define a swipe in Find and Touch:

- 1 Acquire the device and navigate to the appropriate screen.
- 2 Drag the command onto the script canvas.
- 3 Capture the screen using the camera button  in the command.

- 4 Select a screen region  as the starting point to define a swipe.

NOTE While this step is not necessary, it enables you to set the coordinates of the swipe start point at (0,0). If you do not choose a screen region as a start point, the center of the screen is set to (0,0).

5 In the **Touch** tab, select **Swipe** from the drop-down list.

6 Move the swipe endpoints  to set the length and direction of your swipe. A pink arrow indicates the length and direction of the swipe.



7 If you need to adjust swipe coordinates—the positive or negative offsets from (0,0).

Swipe from (,) to (,) offsets from the center of found region.

8 If you need to, change the swipe **Duration** (in milliseconds) in the **Advanced** tab.

9 **Save** the command.

Find and Touch Pop Out

Save Discard

Find Touch **Advanced**

Swipe



Swipe from (,) to (,) offsets from the center of found region.

12.2 Send Keys

This section describes each tab in the Send Keys command, starting with the default tab in Figure 12-4 and Table 12-5 below. See [Specifying Device Input](#) for details on specifying keystrokes on a device.

Figure 12-4 Send Keys Command Properties

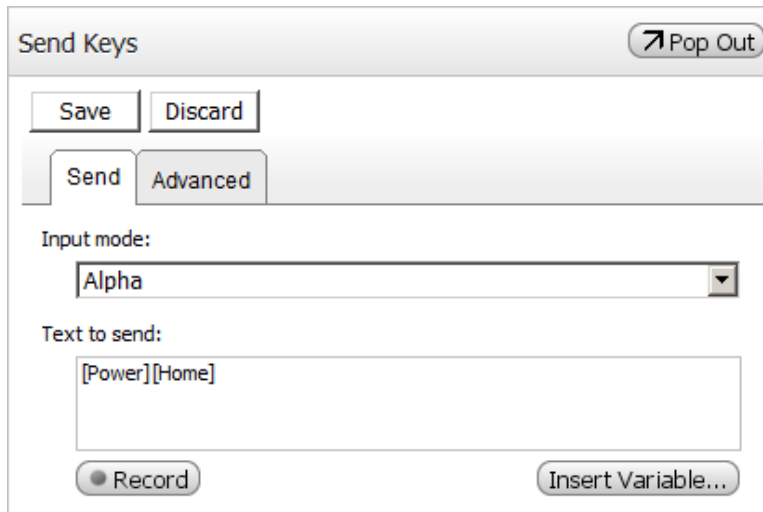


Table 12-5 Send Keys Command Properties

Tab	Control/Field	Description
Send	Text to send	Specify keystrokes (or a parameter in place of keystrokes) to be entered on device keyboards and touchscreens. You can type or paste text into this field, or directly interact with the device and capture the key sequence. Click Insert Variable to insert a variable or parameter in place of keystrokes.
	Input mode	Key mode that the device uses in the field into which text is being entered – Alpha for alphanumeric field such as the body of a text message, Numeric for a numeric field such as a phone number, and Web for the URL field of a browser
	Record/Stop	Click to record/stop recording device interaction (key presses, touches, swipes) in the command. NOTE This is a command-specific recording ability and is different from script-level recording available by clicking Record above the script canvas.
Advanced	Check Point	Select true to capture the first and last screenshots during command execution, false (default) not to capture any proofs.
	Comment	Enter an optional comment in the field provided.
	Delay Time	Enter delay between key presses in milliseconds.
	Hold Time	Enter time in milliseconds to touch each key.

12.3 Play Audio

Play Audio plays an MP3 file or a specified DTMF tone sequence. You can use this command when your script calls another device. Play Audio can also be run concurrently with the next command in the script. So in a two-device test case, you can use Play Audio to play a sound from one device and Wait Event to listen for sound on the other device (see [Examples](#)).

Figure 12-5 Play Audio Command Properties

Table 12-6 Play Audio Command Properties—Default Tab

Control/Field	Sub-Field	Description
Play audio file radio button		Select to play an MP3 file.
	Select Recording	If playing an audio file, click to choose the file.
	Play File	If playing an audio file, click to listen to the chosen file.
Play generated DTMF tone radio button		Select to send DTMF tones as a file.
	Tone Sequence	Enter the tone sequence, e.g., 12345.
	Tone Duration (ms)	Enter the duration of each tone in milliseconds, e.g., 500.
	Play Tone	Click to listen to generated tones.
Press Numeric Keys for DTMF radio button		Select to press device keys. The input is received by the listening device as DTMF tones.
	Tone Sequence	Enter the tone sequence, e.g., 12345.
	Tone Duration	Enter the duration of each tone in milliseconds, e.g., 500.
	Key Mode	Key mode in which keys are pressed, e.g., Alpha or Numeric.
Run Next Command Immediately		Check to run Play Audio concurrently with the next command in the script, e.g., Wait Audio or Wait Event.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.4 Hardware Extension

Depending on the device (platform and type of integration) you are working with, you can perform several hardware operations using the Hardware Extension command.

For example on software-integrated devices:

- ◆ Flip out for Android and accelerometer (orientation) for iPhone supported.
- ◆ Accelerometer and flip open/close operations are not supported on software-integrated BlackBerry.
- ◆ Accelerometer operations are not supported on software-integrated Android.
- ◆ Software-integrated devices cannot be power cycled

For example, on a hybrid iPhone 4 (see Figure 12-6 below), you can connect/turn on or disconnect/turn off the battery, camera light, or data cable. In addition, you can change the orientation of the device (see Figure 12-7 below).

Figure 12-6 Hardware Extension Command Properties for Hybrid iPhone 4

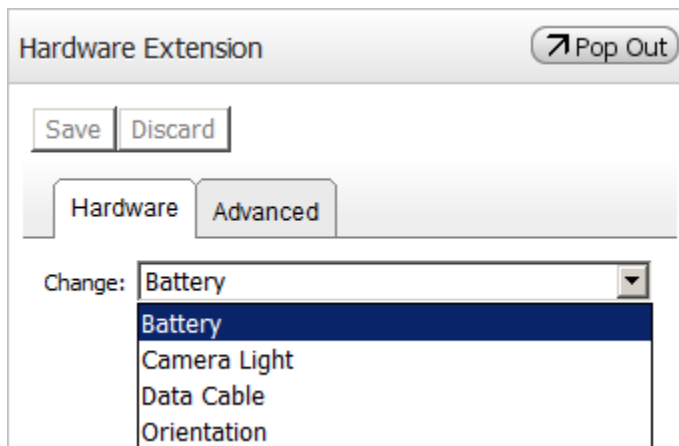


Figure 12-7 Hardware Controls for Device Orientation

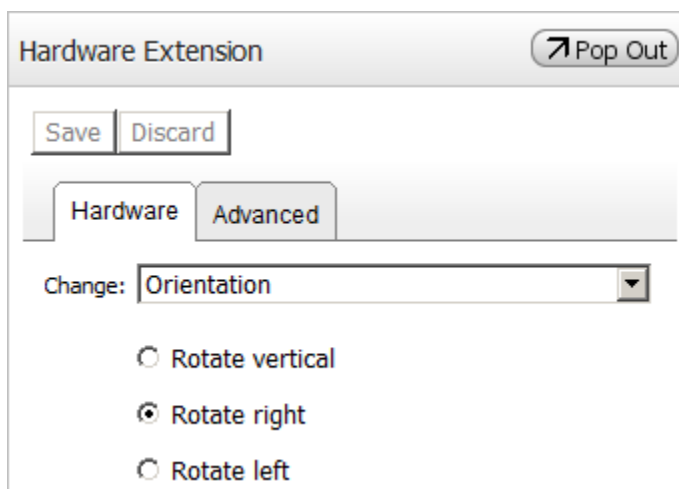


Figure 12-8 Hardware Controls for Device Data Cable

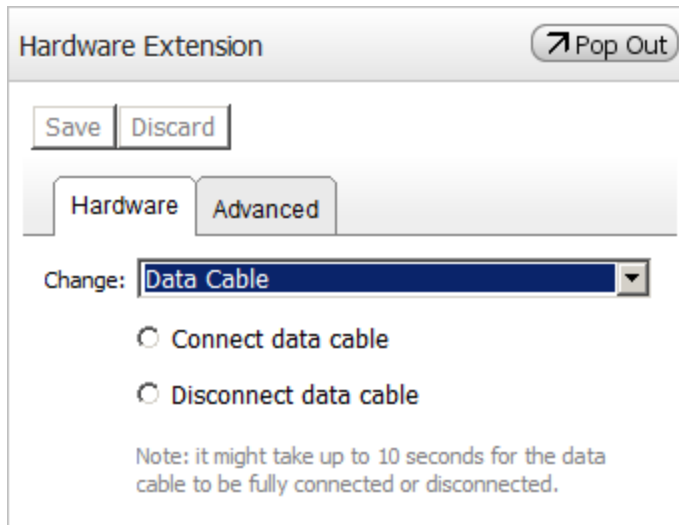


Figure 12-9 Hardware Controls for Power Cable

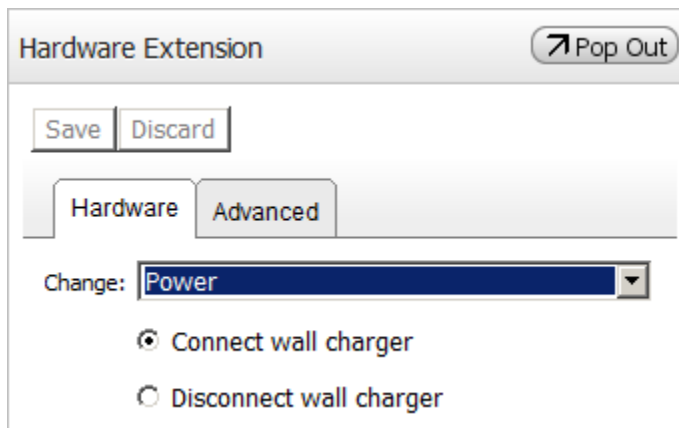


Table 12-7 Hardware Extension Command Properties—Default Tab

Control/Field	Description
Change drop-down list	Select the hardware peripheral you wish to control, e.g., Power, Battery, Orientation, Data Cable, Camera Light . The available list depends on the device. <i>See Figure 12-6.</i>
Peripheral radio buttons	Battery – connect/disconnect Power – connect/disconnect wall charger Camera light – turn on/off Data cable – connect/disconnect Orientation – vertical , horizontal by turning it left or right

In the **Advanced** tab, use the **Checkpoint** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.5 Wait Event

The Wait Event command searches for and uses a combination of reference points as the basis for creating script branches. You can create script branches based on device-specific reference points as well as

device-independent states and web elements. See [Wait Event Command](#) in [Script Logic](#) for steps to add branches in Wait Event.

When you first drag the Wait Event command onto your script canvas, it creates placeholders for two default branches: Tie and Timeout. An additional placeholder is created for each reference point you add to the Wait Event command.

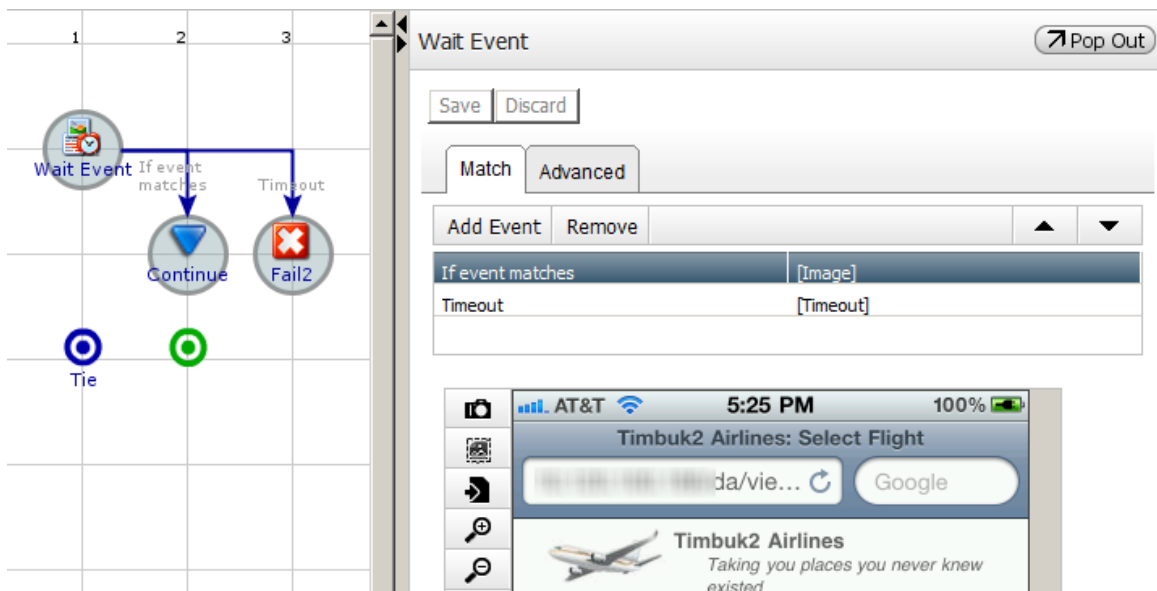
The sections below give a field-by-field description of the Wait Event command for each type of reference point.

12.5.1 Waiting for an Image

You can use the Wait Event command to define an image-based reference point for script verification. An image-based reference point is an area of a device screen that serves as an expected result against which the outcome of a script can be verified. See [Image-Based Reference Points](#) for defining and using reference images.

The high-level flow for defining a reference image in Wait Event:

- 1 Acquire the device and navigate to the screen you would like to use to define your reference image.
- 2 Drag the Wait Event command onto the script canvas. Controls are displayed for a default image-based reference point. The device screen is automatically captured in the command. You can change the type of reference point.
- 3 Optionally, click **Add Logic** to display the default reference point in a branch of its own along with the Tie and Timeout branches.



- 4 Click **Add Event** to add a reference point, and optionally, change the default names of branches.
- 5 Using your mouse, select the screen region you would like to use as a reference image.
- 6 Optionally, define a range of match and mismatch possibilities and see if they match the actual device screen.

NOTE See [Image-Based Reference Points](#) for detailed information on advanced image match settings.

- 7 Optionally, specify *tolerance levels* for variations between the reference image(s) and the actual device screen—you can adjust tolerance settings to include matches and exclude mismatches.
- 8 **Save** advanced settings, **Save** the command.

Figure 12-10 Wait Event—Image

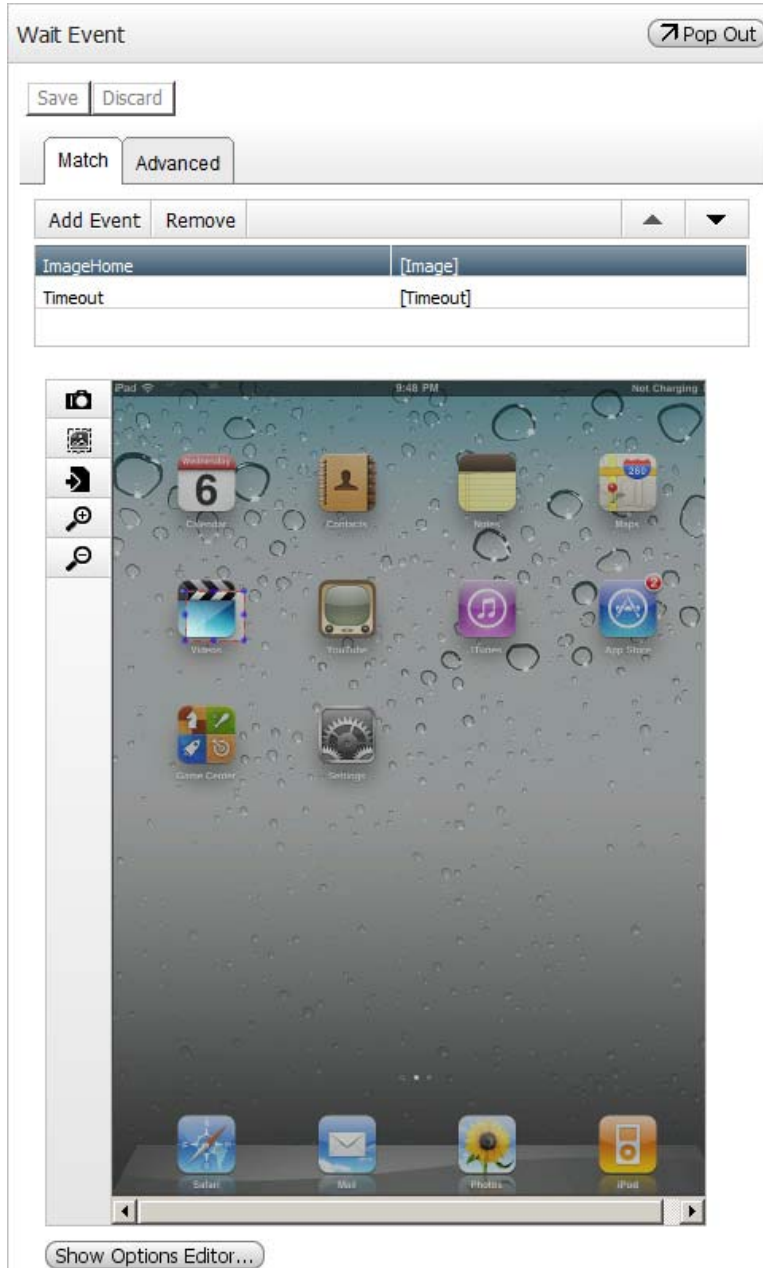





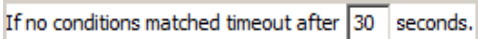


Table 12-8 Wait Event Command Properties—Image

Tab	Field/Control	Description
Match	Add Logic	Click to add Timeout and Tie branches.
	Add Event	Click to add a branch.
	Remove	Click to remove a branch.

Tab	Field/Control	Description
	Branch controls	Up/down  – use to change the order of events.
	Wait Image	Select to use an image as a reference point.
	Selection controls	<p>Camera icon  – Click to take snapshot of current device screen.</p> <p>Clear Selection  – Click to reset device image area.</p> <p>Import image  – Click to import device image (from DeviceAnywhere test results that were directly captured from the device and uploaded to the web portal).</p> <p>Zoom buttons  – Click to resize captured image of device screen in command.</p>
	Show Options Editor	Opens controls for specifying match and mismatch ranges as well as tolerance levels for variance between reference images and the actual device screen. Allows you to troubleshoot areas of mismatch between reference images by adjusting tolerance sliders.
	Timeout	Select and enter the time (in seconds) within which image must be matched. 

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

Use the [Timeout tab](#) to define a command completion time and trigger error reporting when the command fails.

In the options editor, you can adjust tolerance sliders to troubleshoot mismatches between the reference image and the device screen. Tolerance levels specify the variations between the reference image and the actual device screen. (Refer to [Image-Based Reference Points](#) for details on advanced image settings.)

Figure 12-11 Wait Event (Image) Advanced Settings

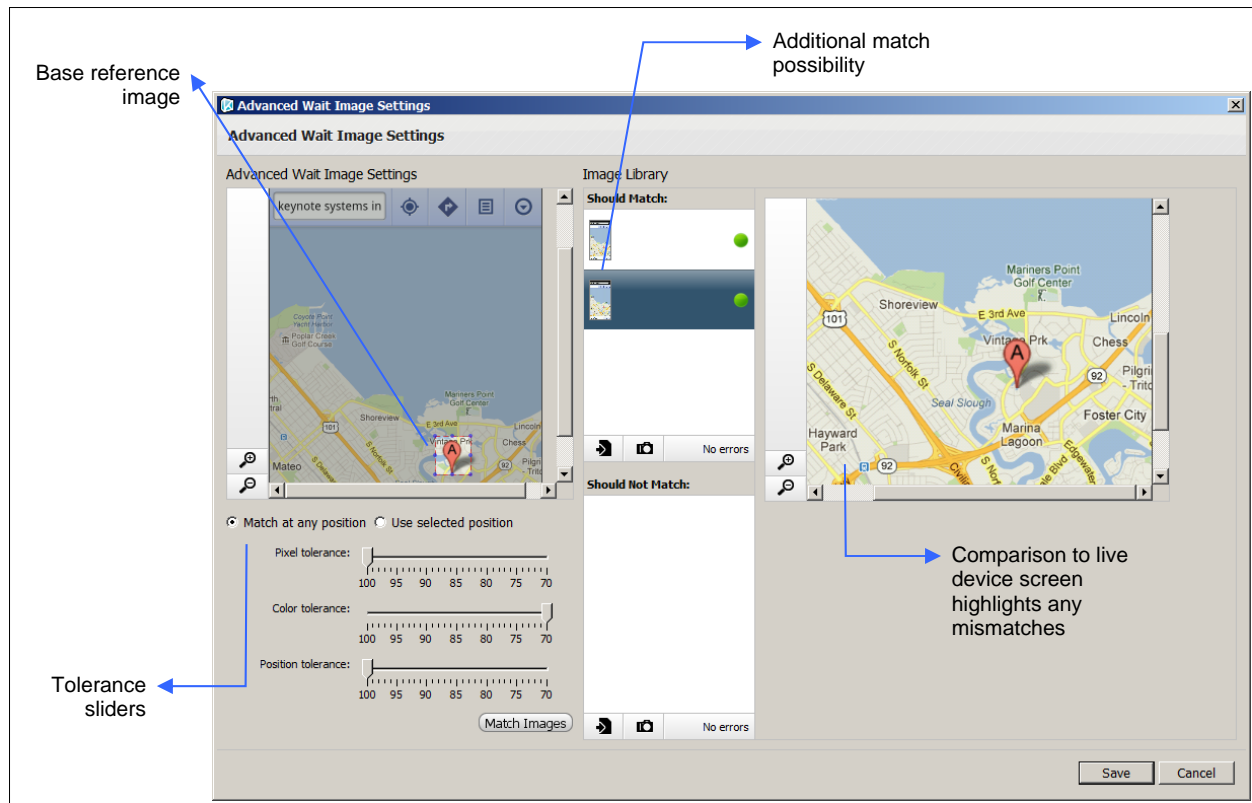









Table 12-9 Wait Event (Image) Options Editor

Control/Field	Description
Zoom buttons 	Use to enlarge or shrink the device screen display; available for the base reference image as well as comparison image.
Camera icon 	Click to take snapshot of current device screen – the image taken is a candidate for match (Should Match) or mismatch (Should Not Match). The image is automatically compared to the base image and areas of mismatch are highlighted in pink. Images that correctly match or do not match the base image are displayed with a green icon  . Images that incorrectly match or do not match the base image are displayed with a red icon  .
Import image icon 	Click to import an image as a candidate for match (Should Match) or mismatch (Should Not Match). The image is automatically compared to the base image and areas of mismatch are highlighted in pink. Images that correctly match or do not match the base image are displayed with a green icon  . Images that incorrectly match or do not match the base image are displayed with a red icon  .
Pixel tolerance slider	Use to set the percentage of pixels to match, with 100 being a strict match.
Color tolerance slider	Use to set the color matching required between the reference image(s) and the device screen, with 100 being a strict match.
Position tolerance slider	Use to set the radius around each pixel's original position in which to look for the pixel, with 100 being a strict match (or the tightest radius).

Control/Field	Description
Match Images	Click to match images after changing tolerance levels or redefining the reference image area.
Save	Save advanced settings.
Cancel	Click to exit advanced settings without saving changes.

You can right-click any image in the **Should Match** or **Should Not Match** library for additional controls:

- ◆ **Make Base Image**—Replaces base image with selected image.
- ◆ **Should Not Match**—Moves an image from the match to the mismatch library.
- ◆ **Should Match**—Moves an image from the mismatch to the match library.
- ◆ **Remove From Library**—Deletes image from library.

12.5.2 Waiting for Text

You can set a string of text as a text-based reference point in the Wait Event command. Keynote uses optical character recognition (OCR) technology to verify that text on the device screen matches the reference string. See [Text-Based Reference Points](#) for a detailed discussion on defining and using text-based reference points.

To use Wait Event for a text-based reference point:

- 1 Acquire the device and navigate to the screen you would like to use to define your reference image.
- 2 Drag the Wait Event command onto the script canvas. Controls are displayed for a default image-based reference point.
- 3 Change the type of reference point to text based (**Wait Text**).
- 4 Optionally, click **Add Logic** to display the default reference point in a branch of its own along with the Tie and Timeout branches.
- 5 Optionally, click **Add Event** for additional reference points.
- 6 Optionally, define a screen region from which to extract text.
- 7 *Transform* text from the device screen to facilitate extraction. See [Text Transformations](#) for detailed instructions on performing text transformations.
- 8 **Extract** text.
- 9 Enter a subset of the extracted text to define your reference string.
- 10 **Save** the command.

Figure 12-12 Wait Event—Text

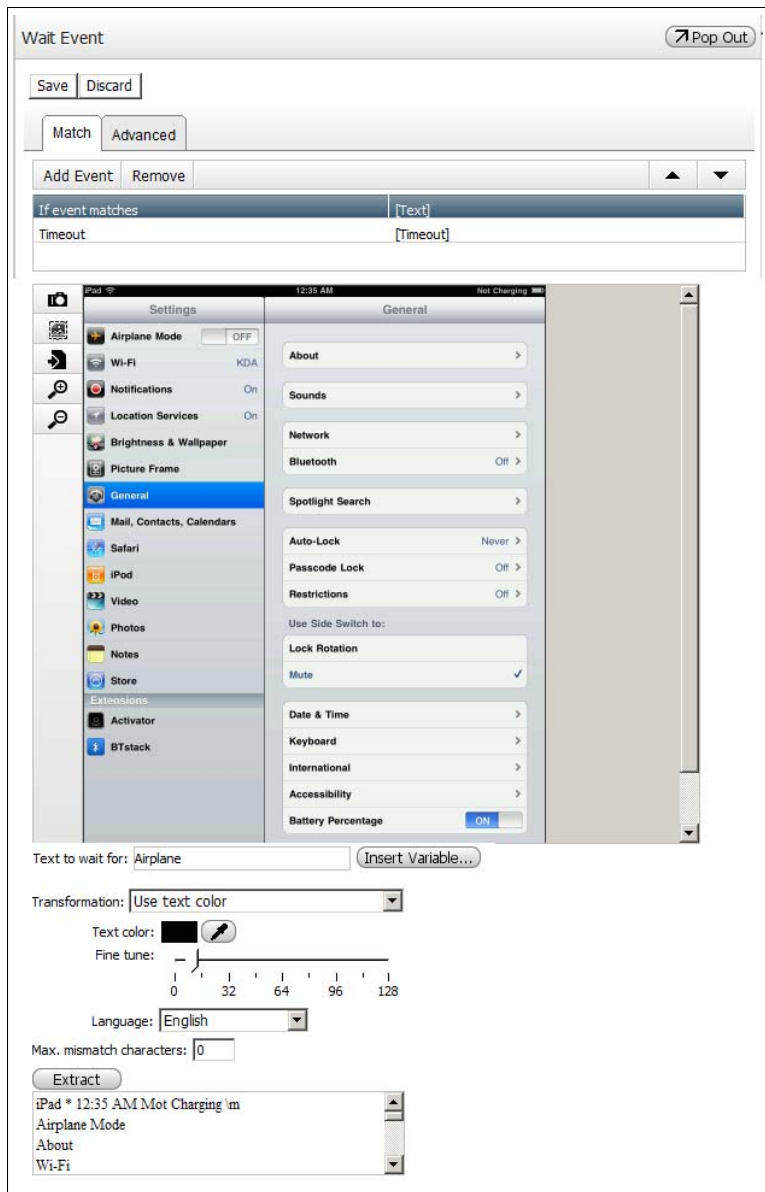








Table 12-10 Wait Event Command Properties—Text

Tab	Control/Field	Description
Match	Add Logic	Click to add Timeout and Tie branches.
	Add Event	Click to add a branch.
	Remove	Click to remove a branch.
	Branch controls	Up/down  – use to change the order of events.
	Wait Text	Select to wait for a string of text.
	Selection controls	Camera icon  – Click to take snapshot of current device screen. Clear Selection  – Click to reset device image area.

Tab	Control/Field	Description
		<p>Import image  – Click to import device image (from DeviceAnywhere test results that were directly captured from the device and uploaded to the web portal).</p> <p>Zoom buttons  – Click to resize captured image of device screen in command.</p>
	Language drop-down list	Select a language for the text string.
	Transformation type drop-down list	<p>Use text color – Select to choose text color with eyedropper icon.</p> <p>Use text background – Select to choose text background with eyedropper icon.</p> <p>Convert image to black and white – Select to convert image of device screen to black and white; choose color that separates black from white with eyedropper icon.</p> <p>Adjust contrast – Select to change contrast of device screen image; choose delimiting color with eyedropper icon.</p> <p>See Text Transformations for an explanation.</p>
	Eyedropper 	Click button and hover over image of device screen to choose a color while transforming text. Click image to select a color.
	Fine tune	Adjust to aid text extraction – use after choosing transformation type and color. Set the slider as high as possible in order to eliminate false positive matches. See Text Transformations for details on the tolerance slider for each transformation type.
	Max. mismatch characters	Enter the maximum number of characters that can be mismatched while identifying the Text to Wait For , e.g., if you specify that any two characters can be mismatched in the string “California,” the term “Callfornla” will be considered a match.
	Extract	Click after choosing transformation settings.
	Text to Wait For	Enter a subset of extracted text as the string of text to wait for. Click Insert Variable to insert a variable or parameter in place of entering the text string to wait for.
	Timeout	<p>Select and enter the time (in seconds) within which text must be matched.</p> <p>If no conditions matched timeout after <input type="text" value="30"/> seconds.</p>

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

Use the [Timeout tab](#) to define a command completion time and trigger error reporting when the command fails.


12.5.3 Waiting for Audio

You can use the Wait Event command to have your script detect device audio for verification. For instance, you can set an audio reference point to verify that a multimedia file from a web site has been successfully played. You can also have your script detect a DTMF tone sequence played to the device.

Figure 12-13 Wait Event—Audio

The screenshot shows the 'Wait Event2' dialog box with the 'Advanced' tab selected. At the top, there are 'Save' and 'Discard' buttons. Below them are 'Match' and 'Advanced' tabs. A toolbar contains 'Add Event', 'Remove', and up/down arrow buttons. A table lists the event as 'Sound' with a value of '[Audio]' and a 'Timeout' field with a value of '[Timeout]'. The 'Detect any sound' radio button is selected, with a volume sensitivity slider set to 'High'. A note below the slider reads: 'Note: use lower volume sensitivity to filter out ambient noise.' The 'Detect DTMF tone' radio button is unselected, with an empty 'Tone Sequence' text box and a 'Play Tone' button below it.

Table 12-11 Wait Event Command Properties—Audio

Tab	Control/Field	Description
Match	Add Logic	Click to add Timeout and Tie branches.
	Add Event	Click to add a branch.
	Remove	Click to remove a branch.
	Branch controls	Up/down  —use to change the order of events.
	Wait Audio	Select to wait for audio.
	Detect any sound radio button	Select to detect any device audio.
	Volume sensitivity slider	Move the slider to the Low end if you expect device audio to be compromised by interference or static.
	Detect DTMF tone radio button	Select to detect a sequence of DTMF tones.
	Tone Sequence	If you have chosen to wait for DTMF tones, enter the sequence of numbers.
	Play Tone	If you have chosen to wait for DTMF tones, click to listen to the number sequence.
	Timeout	Select and enter the time (in seconds) within which audio must be detected. If no conditions matched timeout after <input type="text" value="30"/> seconds.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

Use the [Timeout tab](#) to define a command completion time and trigger error reporting when the command fails.

12.5.4 Waiting for a State

The Wait Event command can call a previously defined state for script verification. At runtime, the implementation for the device on which the script is being run is automatically loaded.

Figure 12-14 Wait Event—State

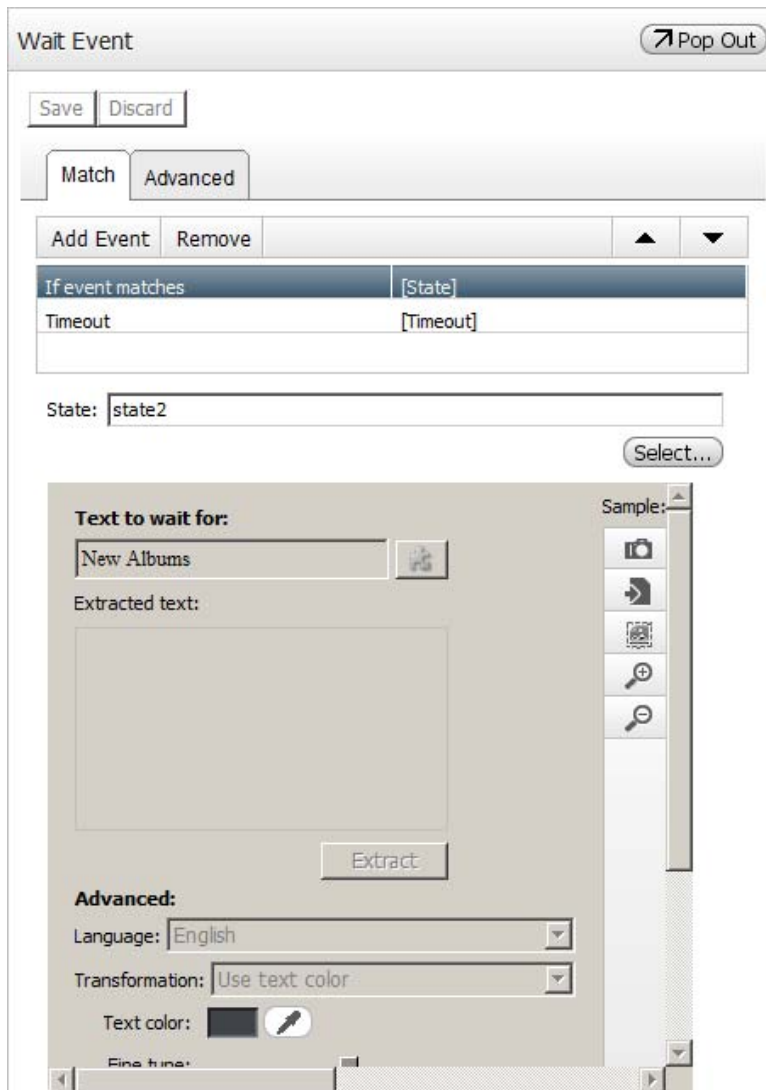



Table 12-12 Wait Event Command Properties—State

Tab	Control/Field	Description
Match	Add Logic	Click to add Timeout and Tie branches.
	Add Event	Click to add a branch.
	Remove	Click to remove a branch.
	Branch controls	Up/down  –use to change the order of events.

Tab	Control/Field	Description
	Wait State	Select to wait for a reference point in a state.
	Select	Click to choose a previously created state. Then choose the state and click Select in the dialog box that appears. The state is listed in the State field.
	Timeout	Select and enter the time (in seconds) within which state must be detected. If no conditions matched timeout after <input type="text" value="30"/> seconds.

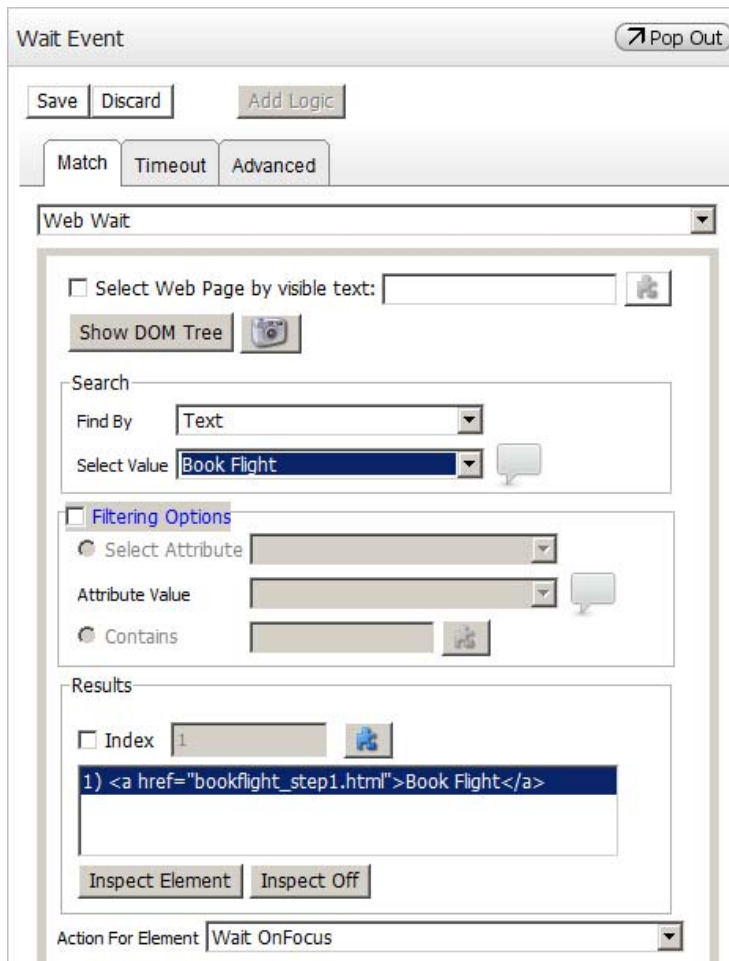
In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

Use the [Timeout tab](#) to define a command completion time and trigger error reporting when the command fails.

12.5.5 Waiting for a Web Element

Wait Event enables you to use an element as a reference point to verify a sequence of web page or application interactions. The command waits for an element to appear on the page, to come into focus or blur, or to become visible or hidden based on previous actions. Web element-based reference points can be used in unpartitioned scripts. See [Web Elements](#) in [Working with Commands](#) for more information.



Figure 12-15 Wait Event—Web

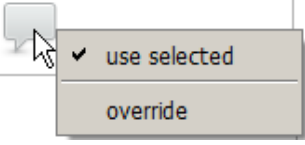





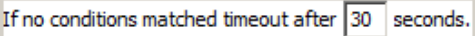


To use Wait Event for a web element-based reference point:

- 1 Acquire the device and navigate to the web page you would like to use to define your reference point.
- 2 Drag the Wait Event command onto the script canvas. Controls are displayed for a default image-based reference point.
- 3 Change the type of reference point to web element based (**Web Wait**). The DOM is automatically loaded to display command properties. If necessary, click the camera icon to load the DOM.
- 4 Optionally, click **Add Logic** to display the default reference point in a branch of its own along with the Tie and Timeout branches.
- 5 Optionally, click **Add Event** for additional reference points.
- 6 Using filters provided, search for and select a web element (see [Searching for Elements](#) for details).
- 7 Select an action to perform, e.g., wait for the element to load (see [Choosing an Action for a Selected Element](#) for details).
- 8 **Save** the command.

Table 12-13 Wait Event Command Properties—Web Element

Tab	Control/Field	Description
Match	Add Logic	Click to add Timeout and Tie branches.
	Add Event	Click to add a branch.
	Remove	Click to remove a branch.
	Branch controls	Up/down  —use to change the order of events.
	Web Wait	Select to wait for a web element.
	Select web page by visible text	Check and enter unique text to identify a web page from multiple sessions or a multi-page application.
	Camera icon 	Click to refresh the DOM, e.g., if you have navigated to a different web page on the device.
	Show DOM Tree	Click to view page markup in a DOM in a separate window (see Figure 12-49 above).
	Find By	Drop-down list to select a search criterion to locate an element: Name —search by the name attribute of an element. ID —search by the id attribute of an element. Text —search by the text between opening and closing tags of an element. Tag —search by the tag applied to an element. XPath —search by the path expression identifying the element in the DOM. CSS Selector —search by the CSS Selector location of the element.
	Select Value	Drop-down list with values matching the search criterion when searching by Name , ID , Text , or Tag When searching by XPath or CSS Selector , you can enter a value in the field. When searching by XPath , you can copy the path expression of an element from the DOM viewer.

Tab	Control/Field	Description
	Override value controls 	Available for entering a value when searching by Name , ID , Text , or Tag . Hover to view options: By default, the field is set to use selected value from the Select Value drop-down list. Click override to enter a value of your own or to pass in the value contained in a variable.
	Puzzle piece icon 	Click to pass in the value contained in a variable. Available when you override a value from the Select Value drop-down list. Also available when you search for an element by XPath or CSS Selector .
	Evaluate	Available when searching by CSS Selector . Click after entering a CSS Selector location to view search results.
	Filtering Options	Check to apply additional filters to search results (listed in the Results pane).
	Select Attribute radio button	Click to perform additional filtering by attribute.
	Attribute Value	Drop-down list with values matching the attribute selected Hover over the override button  to override the attribute value. You can enter a value or pass in the value contained in a variable by clicking the puzzle piece icon  .
	Contains radio button	Click to perform additional filtering by text between element opening and closing tags. Enter a value in the field provided or pass in the value contained in a variable by clicking the puzzle piece icon  .
	Index	Click to perform additional filtering by the index number of search results listed in the Results pane. Enter a number in the field provided or pass in the value contained in a variable by clicking the puzzle piece icon  .
	Inspect Element	Click after selecting an element from the Results pane to highlight it on the device screen.
	Inspect Off	Turn inspect mode off.
	Action for Element	Drop-down list to specify interaction with a chosen element: Wait Hidden – confirms that the selected element is not displayed on the page, even in the portion not visible on the device screen. Wait OnBlur – checks that a selected element loses focus, e.g., after a form field is deselected. Wait Exists – checks to find the selected element on the page. Wait OnFocus – checks that a selected element comes into focus, e.g., after a form field is selected. Wait Visible – checks that the selected element is displayed on the page, even if the screen must be scrolled in order to view it.
	Timeout	Select and enter the time (in seconds) within which web element must be matched. 

See the command reference for [Web Element](#) for a description of controls available in the DOM window.

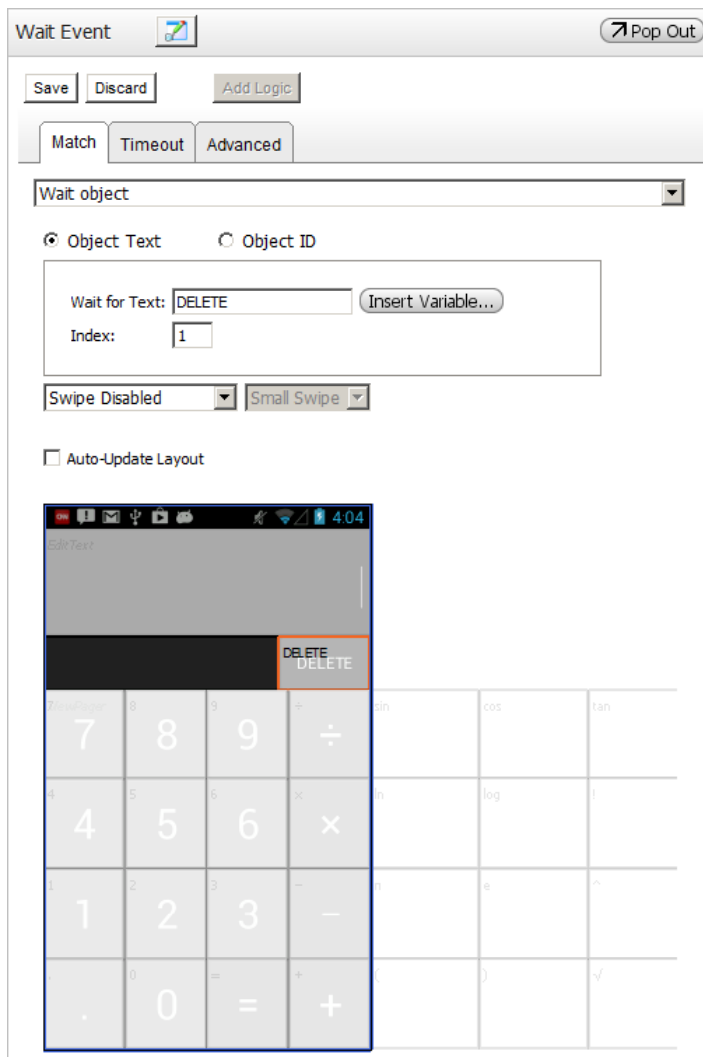
In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

Use the [Timeout tab](#) to define a command completion time and trigger error reporting when the command fails.

12.5.6 Waiting for an Object

Wait Event enables you to use a native object as a reference point to verify a sequence of native application interactions. Native object-based reference points can be used in unpartitioned scripts. See [Native Objects in Commands](#) in [Working with Commands](#) for more information.

Figure 12-16 Wait Event—Object





To use Wait Event for object-based reference point:

- 1 Acquire the device and navigate to the native application page you would like to use to define your reference point.
- 2 Drag the Wait Event command onto the script canvas. Controls are displayed for a default image-based reference point.

- 3 Change the type of reference point to web element based (**Wait object**). The object layout is automatically displayed in command properties.
- 4 Optionally, click **Add Logic** to display the reference point in a branch of its own along with the Tie and Timeout branches.
- 5 Optionally, click **Add Event** for additional reference points.
- 6 Opt to wait for **Object Text** or identify an object by **Object ID**.
- 7 Select an object.
- 8 If necessary, implement a swipe so that the object is displayed on the device screen at run time.
- 9 If necessary, select an object from several displayed by specifying its **Index** number.
- 10 **Save** the command.

Table 12-14 Wait Event Command Properties—Object

Tab	Control/Field	Description
Match	Add Logic	Click to add Timeout and Tie branches.
	Add Event	Click to add a branch.
	Remove	Click to remove a branch.
	Branch controls	Up/down  – use to change the order of events.
	Wait object	Select to wait for an object.
	Object Text radio button	Select to specify an object to wait for by its text.
	Wait for Text	Text of the object to wait for – automatically populated when you select an object with text. You can also enter a value or click Insert Variable to pass in the value contained in a variable.
	Index	If more than one object matches the text entered, select a specific object by entering its index number.
	Object ID radio button	Select to specify an object to wait for by its ID.
	Swipe controls	Swipe Disabled – default; no swipe is implemented. Swipe Down – swipes the device screen as if you were dragging your finger down over the screen. Swipe Up – swipes the device screen as if you were dragging your finger up over the screen. Swipe Left – swipes the device screen as if you were dragging your finger left over the screen. Swipe Right – swipes the device screen as if you were dragging your finger right over the screen.
	Swipe size controls	Small Swipe – swipes approximately one item at a time. Big Swipe – swipes approximately one screen at a time.
	Auto-Update Layout	Checked by default; updates the object layout when you navigate to another device screen.
	Timeout	Select and enter the time (in seconds) within which the object must be found. 

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

Use the [Timeout tab](#) to define a command completion time and trigger error reporting when the command fails.

12.6 Extract Text

The Extract Text command allows you to extract a text string from a device screen and store it in a parameter or variable for use later. Text is extracted using character recognition technology. You can define your text string as preceding and/or following a given occurrence of a term, as a fixed number of characters following a given occurrence of a term, or as a regular expression. See [Extract Text Command](#) in [Parameters and Variables](#) for a detailed description.

NOTES Be sure to use a [variable or parameter](#) of the appropriate type for the data you are extracting.

Variables containing a data set can only be called using the [Loop command](#)—see [Working with Data Sets](#).

Table 12-15 Extract Text—OCR Settings Tab






Control/Field	Description
Selection Controls	Camera icon  – Click to take snapshot of current device screen. Clear Selection  – Click to reset device image area. Import image  – Click to import device image (from DeviceAnywhere test results that were directly captured from the device and uploaded to the web portal). Zoom buttons  – Click to resize captured image of device screen in command.
Limit Text Extraction to Region	Check to limit text extraction to a selected screen region. Use your mouse to select the region.
Language drop-down list	Select a language for the text string.
Transformation type drop-down list	Use text color – Select to choose text color with eyedropper icon. Use text background – Select to choose text background with eyedropper icon. Convert image to black and white – Select to convert image of device screen to black and white; choose color that separates black from white with eyedropper icon. Adjust contrast – Select to change contrast of device screen image; choose delimiting color with eyedropper icon. See Text Transformations for an explanation.
Eyedropper 	Click button and hover over image of device screen to choose a color while transforming text. Click image to select a color.
Fine tune	Adjust to aid text extraction—use after choosing transformation type and color. Set the slider as high as possible in order to eliminate false positive matches. See Text Transformations for details on the tolerance slider for each transformation type.
Max. mismatch characters	Enter the maximum number of characters that can be mismatched while identifying the Text to Wait For , e.g., if you specify that any two characters can be mismatched in the string “California,” the term “Callfornla” will be considered a match.
Extract	Click after choosing transformation settings.

Figure 12-17 Extract Text—OCR Settings Tab



Define your text string in the **Text Extraction Rules** tab.

Figure 12-18 Extract Text—Text Extraction Options Tab

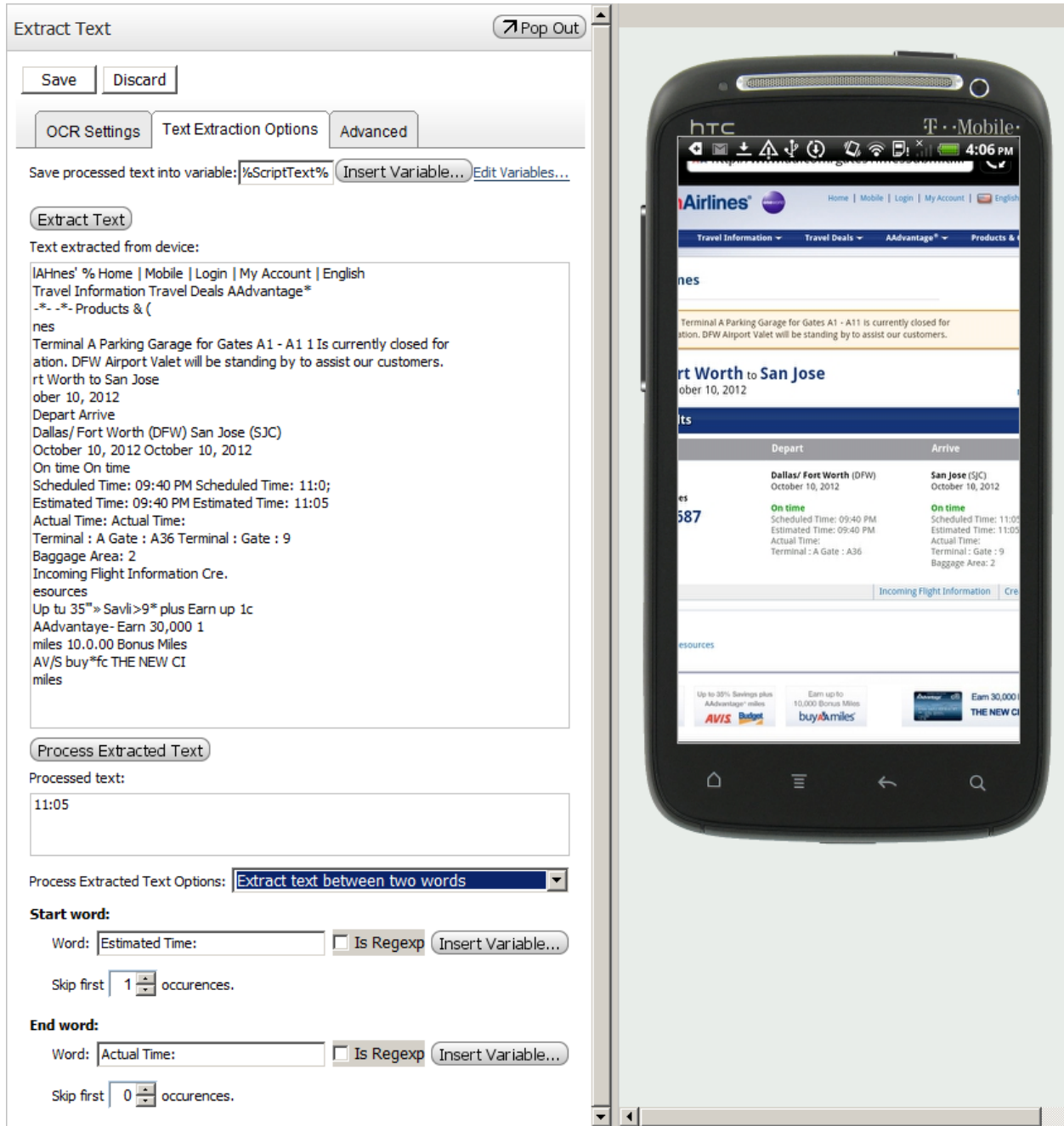


Table 12-16 Extract Text—Text Extraction Rules Tab

Control/Field	Sub-Field	Description
Insert Variable		Click to choose a variable to store extracted text in. The variable is then listed in Save processed text into variable .
Extract Text		Extract text based on transformation settings defined in OCR Settings tab.

Control/Field	Sub-Field	Description
Process extracted text: Leave text as is		All extracted text is stored in the chosen variable.
Process extracted text: Extract text between two words		Select to define terms preceding and/or following the text you want to extract.
	Start Word	Enter the preceding term. You can also click Insert Variable to use the contents of a parameter/variable as the preceding term.
	Is Regexp	Check to use regular expressions to define the preceding term.
	Skip first x occurrences	Enter the number of occurrences of the preceding term to skip from the top of the screen.
	End Word	Enter the subsequent term. You can also click Insert Variable to use the contents of a parameter/variable as the subsequent term.
	Is Regexp	Check to use regular expressions to define the subsequent term.
	Skip first x occurrences	Enter the number of occurrences of the subsequent term to skip from the top of the screen.
Process extracted text: Extract characters after a word		Select to define the text string you want as a fixed number of characters after a given occurrence of a word/phrase.
	Include x characters	Enter the number of characters to extract.
	Word	Enter the preceding term. You can also click Insert Variable to use the contents of a parameter/variable as the preceding term.
	Is Regexp	Check to use regular expressions to define the preceding term.
	Skip first x occurrences	Enter the number of occurrences of the preceding term to skip from the top of the screen.
Process extracted text: Use regular expression		Select to define the text string to be extracted using regular expressions in the field provided. You can also click Insert Variable to use the contents of a parameter/variable as the regular expression.
Process Extracted Text		Click to view text string that will be stored in the selected variable.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.7 Branch

The Branch command allows you to create script branches based on the value of variables or parameters. You must set a branch condition, i.e., associate the branch with the value of one or more variables or parameters. The variable or parameter value(s) must be set earlier in the script.

When you first drag the Branch command onto your script canvas, it creates a placeholder for a single branch, for which you have to specify the branch condition. It also creates the Tie and Default placeholders for additional script logic. To use the command:

- 1 Drag the Branch command onto the script canvas.
- 2 Add a branch condition(s) for the existing branch (**Add Expression**). For each expression, choose a variable/parameter and specify the value.

- 3 Create additional branches as required (**Add Branch**).
- 4 Provide branch conditions for all branches you have created.

See [Branch Command](#) in [Script Logic](#) for detailed instructions on using this command and the order in which to fill out fields.

Figure 12-19 Branch Command

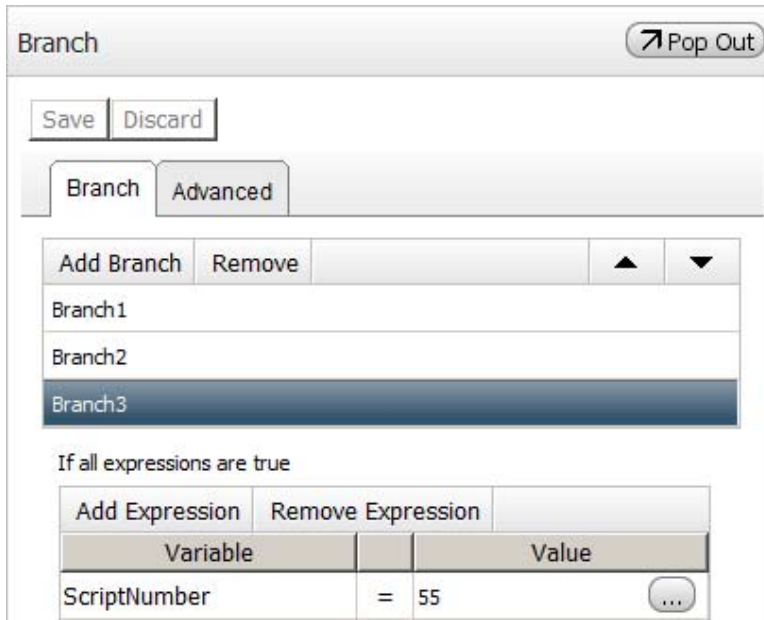
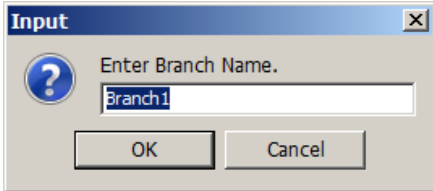





Table 12-17 Branch Command Properties—Main Tab

Control/Field	Description
Add Branch	Click to create a branch. You can double-click the default name and enter a new name in the dialog box that appears. 
Branch controls	Up/down  – use to change the order of branches.
Remove	Click to delete a selected branch.
Add Expression	Click to create a placeholder for a branch condition.
Variable	After adding an expression, double-click in this field to select a variable. Click  to select a variable or parameter.
Value	After adding an expression and selecting a variable, double-click in this field to specify a value as a branch condition. You can also click the  button to select a value contained in a variable. The variable must already have been set to this value earlier in the script.
Remove Expression	Click to remove a selected branch condition.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.8 Set Variable

[Variables](#) allow you to store values in your script to be used as the basis for script logic (e.g., in the [Loop](#) or [Branch](#) commands). You can set different values for variables at different points in your script by using the Set Variable command. (See also [Setting Variable Values in the Set Variable Command](#).)

Figure 12-20 Set Variable

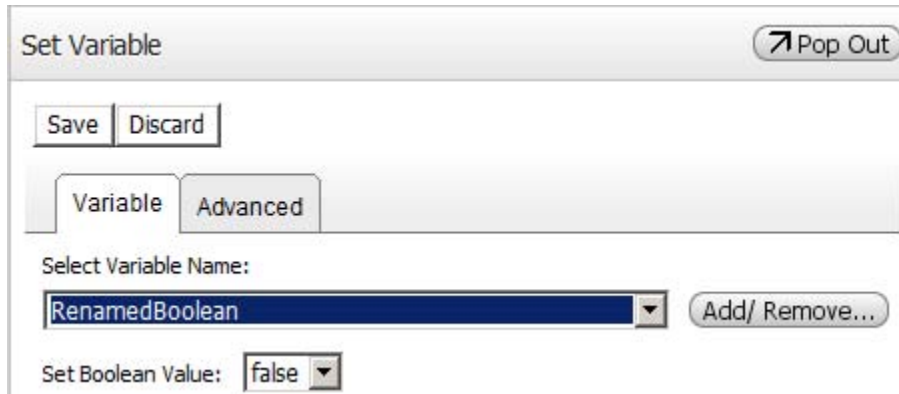


Table 12-18 Set Variable Command Properties

Control/Field	Description
Variable to set	Select a variable from the drop-down list. IMPORTANT Do <i>not</i> choose a data set column. Data set values cannot be set using the Set Variable command.
Add/Remove	Click to create a variable/parameter and then set the value in Set Variable.
Variable value controls	Boolean – if your variable type is Boolean, choose true or false . String – if your variable type is Text, enter an alphanumeric value in the field provided. You can click Insert Variable to pass in the value from another parameter or variable. Number – if your variable type is Number, enter a numeric value in the field provided. You can click Insert Variable to pass in the value from another parameter or variable.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.9 Loop

The Loop command allows the script to continue in a loop over a set of commands for a fixed number of times or until certain [conditions](#) are met. You can also loop over the records of a [data set](#).

When you drag the Loop command onto your script canvas, it creates placeholders for two branches: Loop (for commands within the loop) and Tie (for commands after exiting the loop).

12.9.1 Looping Based on Loop Conditions

You can loop for a fixed number of iterations or based on the value of a variable(s). The variable or parameter value(s) must be set earlier in the script and then reset within the loop.

See [Loops](#) in [Script Logic](#) for detailed instructions on creating and populating a loop.

The high-level flow for creating a Loop based on loop conditions:

- ◆ To have your loop iterate for a fixed number of times, select **Iterate fixed number of times** and enter a loop count.
- ◆ To set a loop condition based on the value of parameters or variables (all expressions must no longer be true for the script to exit the loop at runtime):
 - a Select **Iterate while all expressions are true**.
 - b Click **Add Expression** to select a variable/parameter as a loop condition.
 - c Select a **Variable** and enter its **Value**.
 - d Optionally, enter a **Max Loop Count**.

Figure 12-21 Looping for a Fixed Number of Iterations

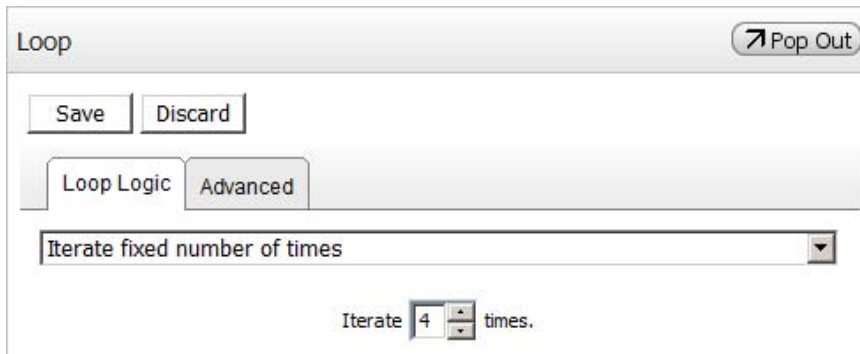


Figure 12-22 Looping Based on the Value of Variables

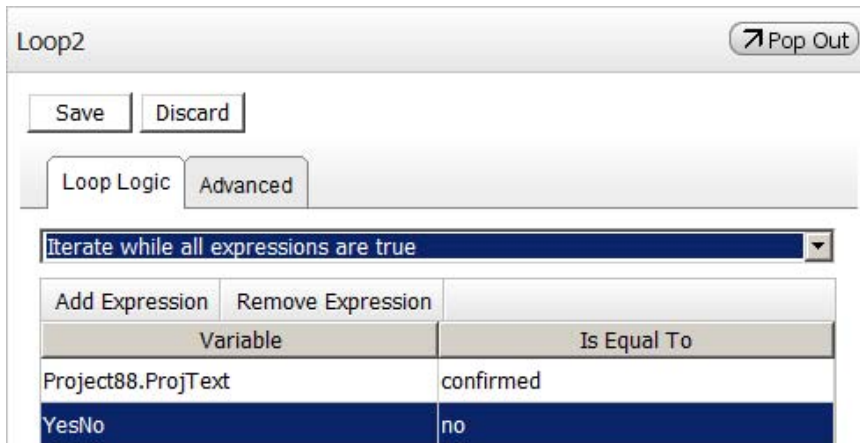





Table 12-19 Loop—Iterations and Loop Conditions

Tab	Control/Field	Description
Loop Logic	Iterate fixed number of times	Select to have your loop iterate for a fixed number of times over a set of commands. Enter a numeric value in the field provided. Iterate  times.

Tab	Control/Field	Description
	Iterate while all expressions are true	Select to base your loop on the value of one or more parameters/variables.
	Add Expression	Click to bring up Select Variable dialog box and choose a previously defined variable/parameter. The variable is then listed in the command.
	Variable	After adding an expression, double-click in this field to select a variable. Click  to select a variable or parameter.
	Value	After adding an expression and selecting a variable, double-click in this field to specify a value as a loop condition. You can also click  to select a value contained in a variable. The variable must already have been set to this value earlier in the script.
	Remove Expression	Click to delete the selected variable from the list of expressions.
Advanced	Check Point	Select true to capture the first and last screenshots during command execution, false (default) not to capture any proofs.
	Comment	Enter an optional comment in the field provided.
	Max Loop Count	Enter a maximum number of iterations. Not available when looping for a fixed number of iterations or over data set records.

12.9.2 Looping Over a Data Set

The Loop command allows you to create a loop for commands that loop iteratively through the values in a data set. The data set is previously created and stored in a script or global variable. Within the loop, for commands that interact with the data set, you must specify the fields (columns) that they interact with.

To use the command:

- 1 Create a variable or parameter associated with a data set.
- 2 Drag the Loop command onto the script canvas.
- 3 Select a variable containing a data set (see Figure 12-23 below).
- 4 Select data set records to loop over.
- 5 Drag commands to populate the loop placeholder.
- 6 For commands that interact with the data set, select the field (column) of data they interact with (see the Send Keys command in Figure 12-24 below—the field name is appended to the variable name displayed within percent signs).
- 7 Use the Tie branch for commands after exiting the loop.

See [Working with Data Sets](#) in [Parameters and Variables](#) for detailed instructions on using this command. Also see [Examples](#) for a script that implements a data set.

Figure 12-23 Loop Command—Data Set Records

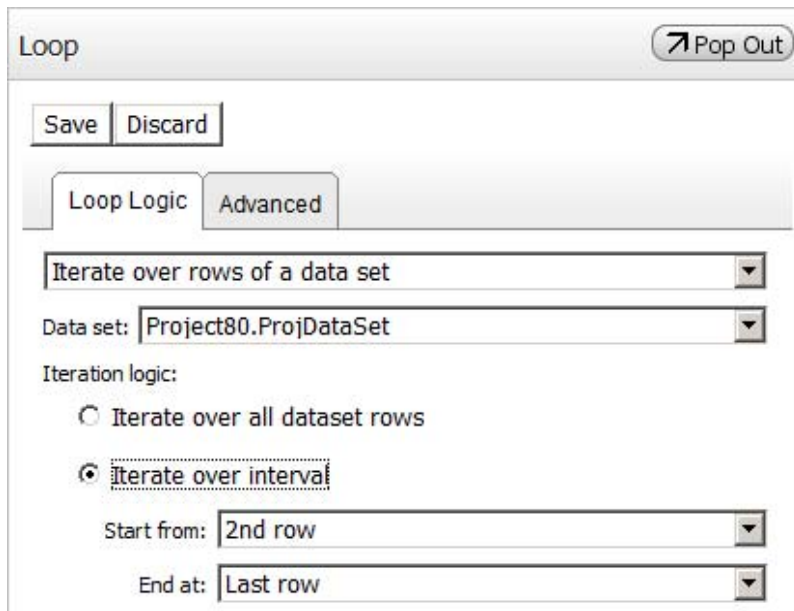


Figure 12-24 Data Set Field in Send Keys

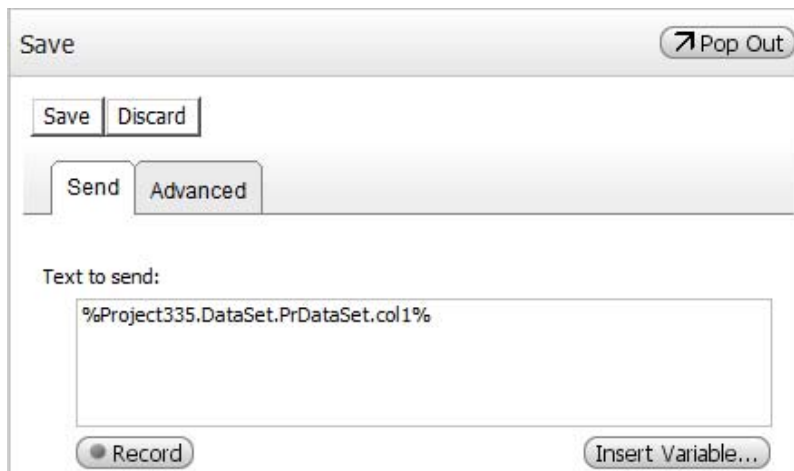


Table 12-20 Loop Logic Tab—Data Set

Control/Field	Sub-Field	Description
Iterate over rows of a data set		Select to loop over selected records of a data set.
Data set		Select a script or project data set from the drop-down list. Project variables are prefixed with the project ID, e.g., <code>Project80.ProjDataSet</code> .
	Iterate over all data set rows radio button	Select to have commands in the loop iterate over all records.
	Iterate over interval radio button	Select to have commands in the loop iterate over a specified range of records.
	Start from	Choose a start record from the drop-down list, e.g., First row, 2nd row, 3rd row

Control/Field	Sub-Field	Description
	End at	Choose an end record from the drop-down list, e.g., First row, 2nd row, ... Last row.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.10 Wait

The Wait command inserts a pause for a specified amount of time in the test script.

Figure 12-25 Wait Command Properties

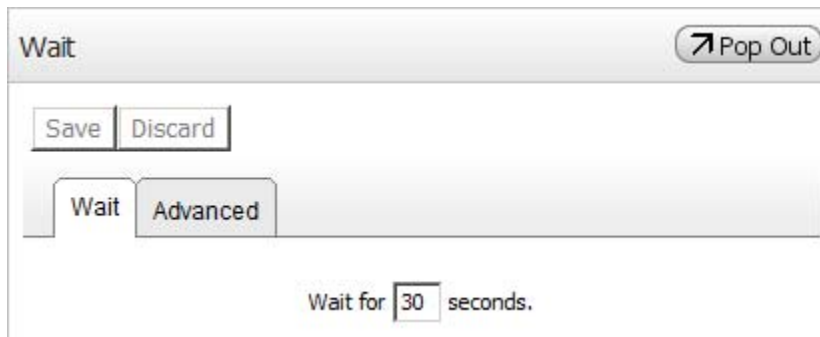


Table 12-21 Wait Command Properties

Control/Field	Description
Wait for x seconds	Enter number of seconds to wait.

You can enter a **Comment** in the **Advanced** tab.

12.11 Success

Use the Success command to explicitly terminate your script successfully at a certain point, e.g., at the end of a branch.

Figure 12-26 Success Command Properties

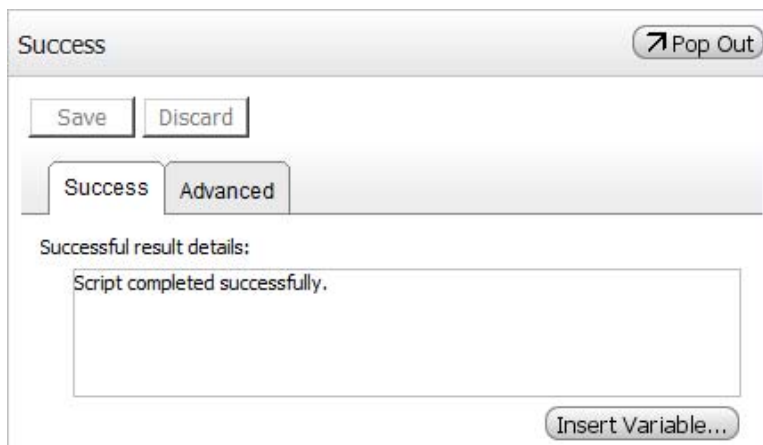


Table 12-22 Success Command Properties—Main Tab

Control/Field	Description
Successful result details	If desired, overwrite the default termination message for display in test results. Click Insert Variable to include the contents of a parameter or variable in the termination message.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.12 Fail

Use the Fail command to explicitly terminate your script unsuccessfully at a certain point, e.g., at the end of a branch. You can choose to display error messaging, and optionally, append the default error definition.

Figure 12-27 Fail Messaging

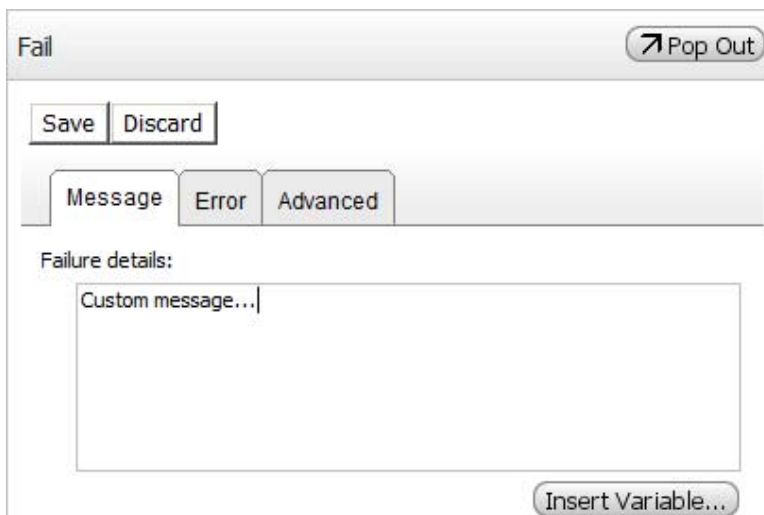


Figure 12-28 Fail Error Messaging

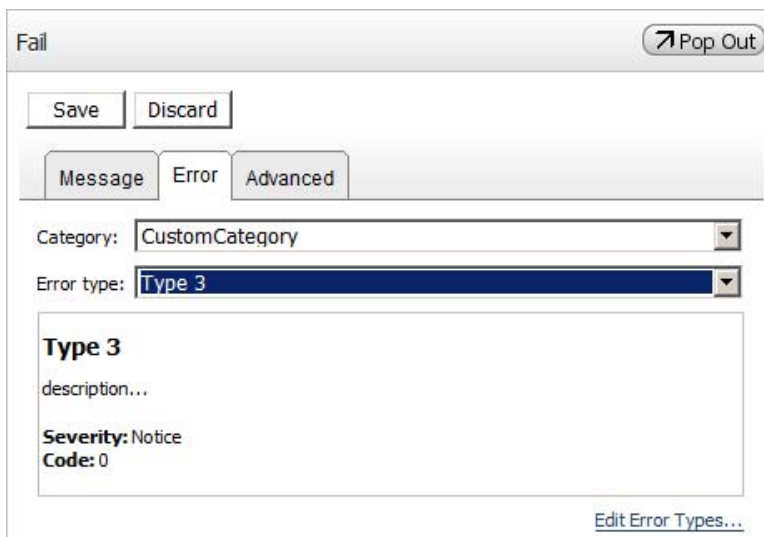


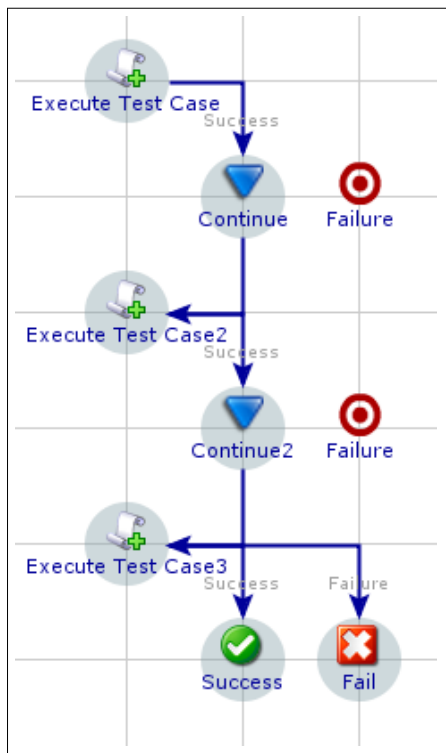
Table 12-23 Fail Command Properties

Tab	Control/Field	Description
Message	Failure details	Enter a termination message for display in test results. Click Insert Variable to insert the contents of a variable/parameter into your note.
Error	Category	Select an error category from the drop-down list provided.
	Error type	Select an error type from the list displayed for the selected category. This error will be triggered when the script encounters the Fail command.
	Edit Error Types	Alternatively, click to define a new error type in the Error Types tab of project properties .

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.13 Continue

You can use the Continue command to tie a branch with no other commands to others or back to the main flow of the script, as shown in the figure below.



The Continue command has no editable properties. The command defaults to the **Advanced** tab, where you can use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

Figure 12-29 Continue Command



12.14 Navigate To

The Navigate To command presses a key (or key sequence) and then waits for a reference point. It repeats this procedure for a fixed number of times or until the reference point is found, whichever comes first. The Navigate To command creates Success and Failure branches based on whether the reference point has been found. There is also a default branch, Tie, to continue with the script outside the Success and Failure branches.

To use the command:

- 1 Acquire a device and navigate to the screen you wish to use as a reference point.
- 2 Drag the Navigate To command onto the script canvas (see Figure 12-30 below).
- 3 Enter a key or key sequence. The command will press this key (sequence) and wait for a reference point up to a specified number of times.
- 4 Select a reference point to wait for (you can choose to define a device-specific, text- or image-based reference point or you can select a previously defined state).
- 5 Define a device-specific, text- or image-based reference point or choose a reference point in a state.

See [Navigate To Command](#) in [Script Logic](#) for detailed instructions on using this command.

Figure 12-30 Navigate To Command Properties—Logic Tab

Navigate To Pop Out

Save Discard

Logic Condition Advanced

Input mode:
Alpha

Key Sequence:
[Return]

Record Insert Variable...

Navigate to logic:
Send key sequence while condition is not matched.
Repeat no more than times.
Timeout after seconds if condition can not be matched.

Table 12-24 Navigate To Command Properties—Logic and Advanced Tabs

Tab	Control/Field	Description
Logic	Input mode	Key mode that the device uses in the field into which text is being entered— Alpha for alphanumeric field such as the body of a text message, Numeric for a numeric field such as a phone number, and Web for the URL field of a browser
	Key Sequence	Enter a key or key sequence. Enter the names of navigation keys in square brackets, e.g., [Home]. Click Insert Variable to pass in the key sequence from a variable/parameter.
	Record/Stop	Click to record/stop recording device interaction (key presses, touches, swipes) in the command. NOTE This is a command-specific recording ability and is different from script-level recording available by clicking Record above the script canvas.
	Repeat no more than x times	Number of times to press the key (or key sequence) and then wait for a reference point
	Timeout after x seconds	Time (in seconds) within which the reference point must be found
Advanced	Advanced Wait Image Settings	<i>Only</i> if using an image-based reference point: Click Show Options Editor for advanced options (see Using an Image-Based Reference Point below).
	Check Point	Select true to capture the first and last screenshots during command execution, false (default) not to capture any proofs.
	Comment	Enter an optional comment in the field provided.
	Delay Time	Enter delay between key presses in milliseconds.
	Hold Time	Enter hold time in milliseconds for each key.

12.14.1 Using an Image-Based Reference Point

You can perform several iterations of pressing a key sequence and waiting to find an image-based reference point in the Navigate To command. You define the reference point in the **Condition** tab.

Table 12-25 Navigate To Condition Tab—Image





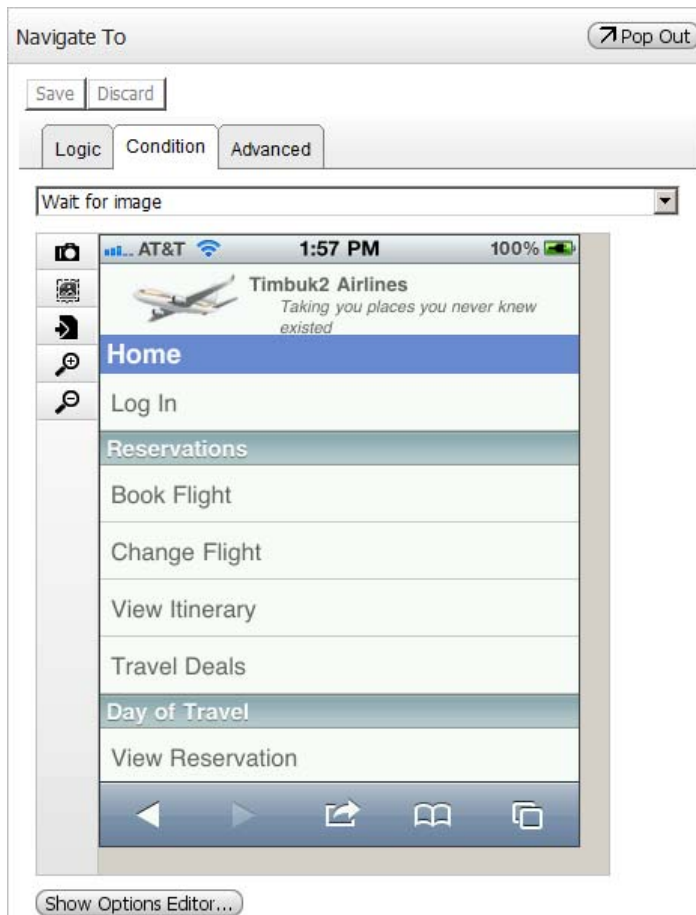
Control/Field	Description
Wait for image	Select to define a device-specific, image-based reference point that the command must verify.
Selection controls	Camera icon  – Click to take snapshot of current device screen. Clear Selection  – Click to reset device image area. Import image  – Click to import device image (from DeviceAnywhere test results that were directly captured from the device and uploaded to the web portal). Zoom buttons  – Click to resize captured image of device screen in command.
Show Options Editor	Opens controls for specifying match and mismatch ranges as well as tolerance levels for variance between reference images and the actual device screen. Allows you to troubleshoot areas of mismatch between reference images by adjusting tolerance sliders.

Figure 12-31 Navigate To—Image



In the options editor, you can adjust tolerance sliders to troubleshoot mismatches between the reference image and the device screen. (Refer to [Image-Based Reference Points](#) for details on advanced image settings.)

Figure 12-32 Navigate To (Image) Advanced Settings

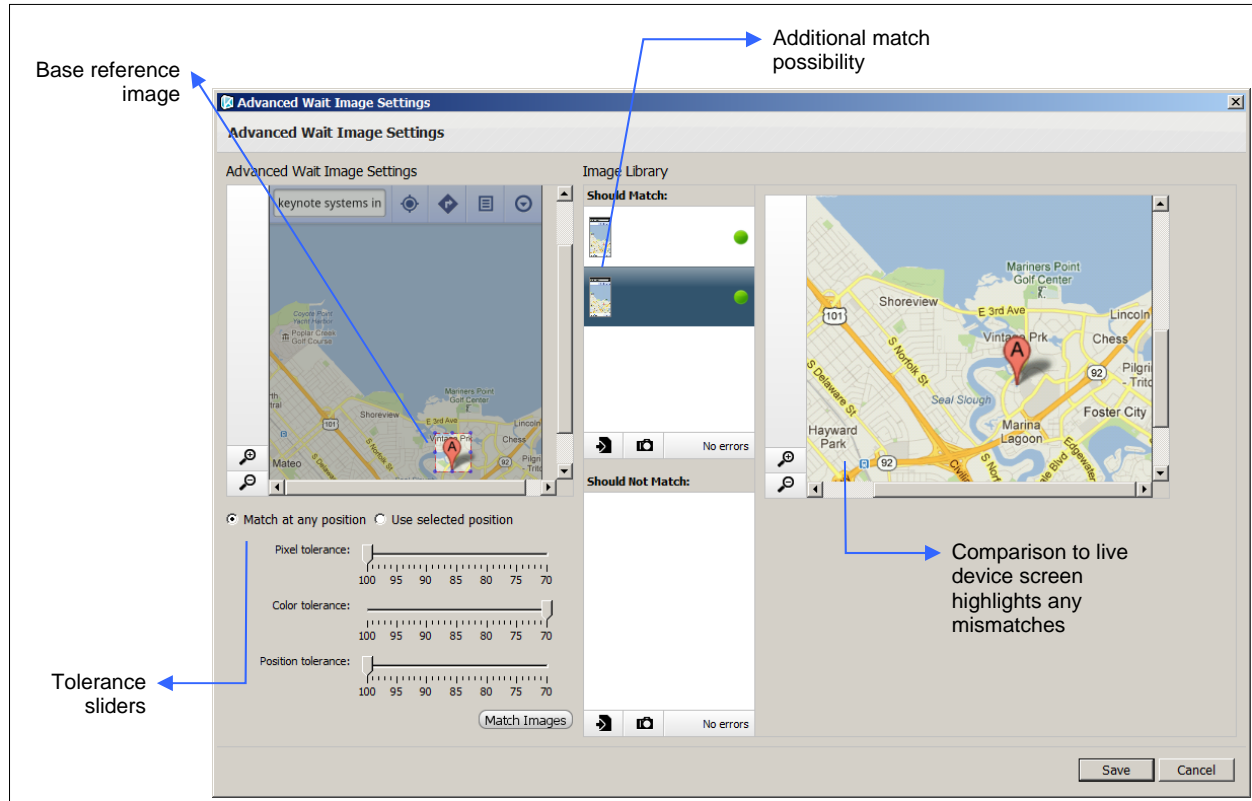









Table 12-26 Navigate To (Image) Options Editor

Control/Field	Description
Zoom buttons 	Use to enlarge or shrink the device screen display; available for the base reference image as well as comparison image.
Camera icon 	Click to take snapshot of current device screen—the image taken is a candidate for match (Should Match) or mismatch (Should Not Match). The image is automatically compared to the base image and areas of mismatch are highlighted in pink. Images that correctly match or do not match the base image are displayed with a green icon  . Images that incorrectly match or do not match the base image are displayed with a red icon  .
Import image icon 	Click to import an image as a candidate for match or mismatch. The image is automatically compared to the base image and areas of mismatch are highlighted in pink. Images that correctly match or do not match the base image are displayed with a green icon  . Images that incorrectly match or do not match the base image are displayed with a red icon  .
Pixel tolerance slider	Use to set the percentage of pixels to match, with 100 being a strict match.

Control/Field	Description
Color tolerance slider	Use to set the color matching required between the reference image(s) and the device screen, with 100 being a strict match.
Position tolerance slider	Use to set the radius around each pixel's original position in which to look for the pixel, with 100 being a strict match (or the tightest radius).
Match Images	Click to match images after changing tolerance levels or redefining the reference image area.
Save	Save advanced settings.
Cancel	Click to exit advanced settings without saving changes.






You can right-click any image in the **Should Match** or **Should Not Match** library for additional controls:

- ◆ **Make Base Image**—Replaces base image with selected image.
- ◆ **Should Not Match**—Moves an image from the match to the mismatch library.
- ◆ **Should Match**—Moves an image from the mismatch to the match library.
- ◆ **Remove From Library**—Deletes image from library.

12.14.2 Using a Text-Based Reference Point

You can perform several iterations of pressing a key sequence and waiting to find a text-based reference point in the Navigate To command. You define the reference point in the **Condition** tab.

Table 12-27 Navigate To Condition Tab—Text

Control/Field	Description
Wait for text	Select to define a text-based reference point that the command must verify.
Selection controls	Camera icon  —Click to take snapshot of current device screen. Clear Selection  —Click to reset device image area. Import image  —Click to import device image (from DeviceAnywhere test results that were directly captured from the device and uploaded to the web portal). Zoom buttons  —Click to resize captured image of device screen in command.
Limit Text Extraction to Region	Check to limit text extraction to a selected screen region. Use your mouse to select the region.
Language drop-down list	Select a language for the text string.
Transformation type drop-down list	Use text color —Select to choose text color with eyedropper icon. Use text background —Select to choose text background with eyedropper icon. Convert image to black and white —Select to convert image of device screen to black and white; choose color that separates black from white with eyedropper icon. Adjust contrast —Select to change contrast of device screen image; choose delimiting color with eyedropper icon. See Text Transformations for an explanation.
Eyedropper 	Click button and hover over image of device screen to choose a color while transforming text. Click image to select a color.

Control/Field	Description
Fine tune	Adjust to aid text extraction—use after choosing transformation type and color. Set the slider as high as possible in order to eliminate false positive matches. See Text Transformations for details on the tolerance slider for each transformation type.
Max. mismatch characters	Enter the maximum number of characters that can be mismatched while identifying the Text to Wait For , e.g., if you specify that any two characters can be mismatched in the string “California,” the term “Callfornla” will be considered a match.
Extract	Click after choosing transformation settings.

Figure 12-33 Navigate To—Text



12.14.3 Using a State

You can perform several iterations of pressing a key sequence and waiting to find a state in the Navigate To command. You define the reference point in the **Condition** tab.

Figure 12-34 Navigate To—State

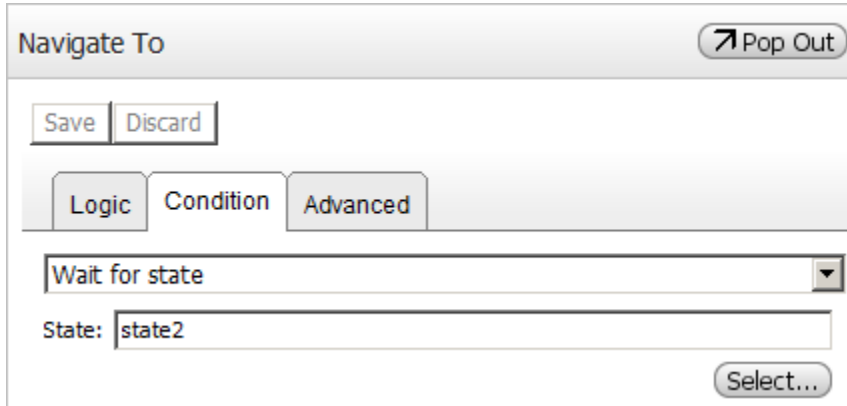


Table 12-28 Navigate To Condition Tab—State

Control/Field	Description
Wait for state	Select to choose a state that the command must wait for.
Select	Click to choose a previously created state. Then choose the state and click Select in the dialog box that appears. The state is listed in the State field.

12.15 Execute Action

Use the Execute Action command to call a previously defined action. Execute Action offers several options for specifying [proofs](#). These proof settings are separate from proof settings for individual commands in the action being called.

Figure 12-35 Execute Action Command Properties

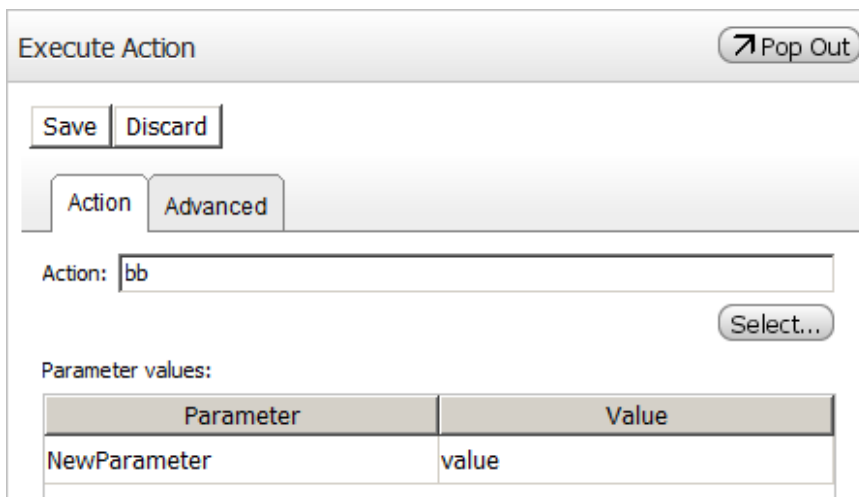

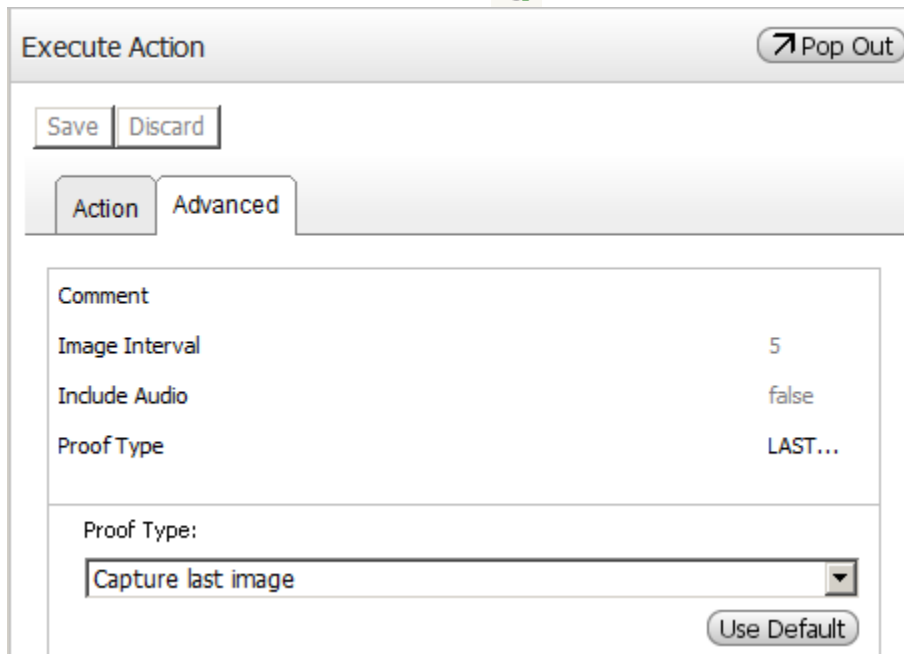


Figure 12-36 Proof Settings in Execute  Action


Execute Action Pop Out

Save Discard

Action Advanced

Comment

Image Interval	5
Include Audio	false
Proof Type	LAST...

Proof Type:

Capture last image

Use Default

Table 12-29 Execute Action Command Properties

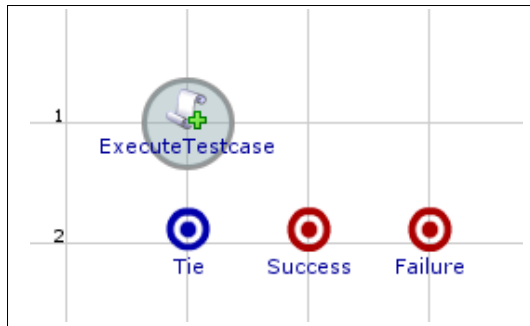
Tab	Field/Control	Description
Action	Select	Click to choose an action from the list of project actions.
	Parameter column	This section is populated only if the action being called has any associated parameters. You can specify the value of these parameters in the Value column. The parameter name is not editable.
	Value column	Double-click and enter a value for the action parameter. You can also click Insert Variable to pass in the value contained in another parameter or variable (see Setting Parameter Values).
Advanced	Comment	Enter an optional comment in the field provided.
	Image Interval	<i>Only if capturing images at fixed intervals, specify the interval in seconds.</i>
	Include Audio	<i>Only if capturing video, select true to include audio information.</i>
	Proof Type	<p>Do not capture anything—select not to capture any proofs.</p> <p>Capture last image—capture the last device screen of the command.</p> <p>Capture first and last images—capture the first and last screens of the command.</p> <p>Capture images at fixed interval—select to capture images at regular intervals.</p> <p>Capture whole page—capture a scrolling image of the page displayed on the device screen.</p> <p>Capture video—capture device vide in MPEG.</p>

12.16 Execute Test Case

Use the Execute Test Case command to call a previously defined test case from a test cycle.

When you first drag the Execute Test Case command onto your script canvas, it creates placeholders for three default branches: Tie, Success, and Failure.

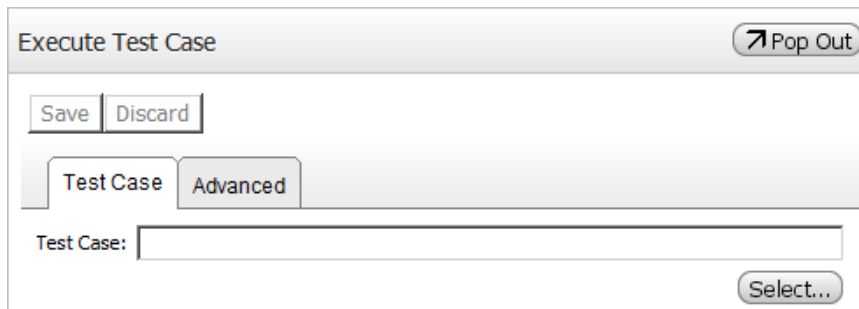
Figure 12-37 Branches in Execute Test Case



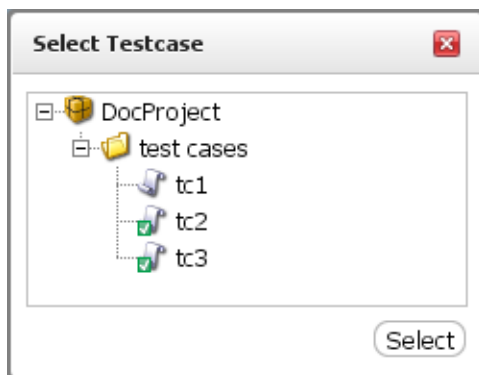
You can drop commands in the Success branch to define script action if the test case is executed successfully. Likewise, you can drop commands in the Failure branch to define script action if the test case is unsuccessful. When done defining the Success and Failure branches, you can use the Tie branch to tie branches together and continue with the script. For example, you might want to set a variable to one value in the Success branch and to another value in the Failure branch, after which you want both branches to follow the same command sequence. Use the Tie branch to define this command sequence and continue with the script.

To use the command:

- 1 Drag the Execute Test Case command onto the script canvas.



- 2 Click **Select**. This brings up a dialog box with a list of project test cases to choose from.




- 3 Select a test case from the list and click **Select**. The name of the selected test case is now displayed in the Execute Test Case command.
- 4 Drag commands onto the Tie, Success, and Failure branches to complete your test cycle script.

Figure 12-38 Execute Test Case Command Properties

Parameter	Value
TParam	889335

Table 12-30 Execute Test Case – Test Case Tab

Control/Field	Description
Select	Click to choose a previously created test case from the project.
Parameter	This section is populated only if the test case being called has any associated parameters. You can specify the value of these parameters in the Value column. The parameter name is not editable.
Value	Double-click and enter a value for the test case parameter. You can also click  to pass in the value contained in another parameter or variable (see Setting Parameter Values).

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.17 Execute Cleanup Action

In a test cycle, use the Execute Cleanup Action command to call a project action to perform cleanup operations on the test device. The action called does not impact test results. For example, you might want to use the Execute Cleanup Action command to call an action that resets the device before the next test case is run on it.

NOTE While the **Checkpoint** control is enabled in Execute Cleanup Action, proofs from the command are *not* inserted into test results. Any proofs from the action being called are also *not* inserted into test results.

Figure 12-39 Execute Cleanup Action Command Properties

Table 12-31 Execute Cleanup Action Command Properties

Control/Field	Description
Select	Click to choose a previously created action from your project.

You can enter a **Comment** in the **Advanced** tab.

12.18 Reset

The Reset command disconnects and reconnects the battery, then powers on the device. There are no editable properties in this command; when you open the command, it defaults to the **Advanced** tab, where you can enter a **Comment**.

Figure 12-40 Reset Command Properties



12.19 Load Application

This command uploads an application from the DeviceAnywhere application repository or your file system onto an Android, BlackBerry, BREW, Brew MP, iOS, or Windows Phone 7 device onto which applications can be side loaded. The device must be connected by data cable to the Ensemble Server. Check the device in the [Test Center](#) view of DeviceAnywhere Studio to ensure that application upload functionality is available for it.

Application delivery methods vary by device platform. Available delivery methods are:

- ◆ Data cable—This delivery method automatically installs the application on the device.
- ◆ SMS—An SMS message containing a link to the application in the DeviceAnywhere repository is sent to the device. This requires a modem—please contact your Keynote Solutions Consultant.

To use the command, drag it onto the script canvas.

Use one of these methods to select an application:

- ◆ Select from the list of applications available in the DeviceAnywhere repository (see Figure 12-41)—each application is listed with information on the mobile platform (e.g., Android), size, and filename.

You can pass the name of the application to upload from a variable or parameter. You would do this, for instance, if you wanted to specify a different application to upload for each script run. Click **Insert variable** to select the variable—the name is displayed enclosed in percent (%) signs in the **Select Application** field.

- ◆ Select an application file from your file system.

NOTE Ensure that the application you select is appropriate for the device.

Figure 12-41 Load Application Command Properties

Table 12-32 Load Application Command Properties

Field/Control	Sub-Control	Description
Source: Upload application from MyApps		Select to choose an application already uploaded to the DeviceAnywhere repository.
	Name	Select the name of the application from the drop-down list that appears. All applications uploaded to the DeviceAnywhere repository are listed in the command.
	Type	Automatically filled when you select an application—do not change.
	Version	Automatically filled when you select an application.
	Upload type radio buttons	<p>Upload application via data cable—Directly installs the application on the device.</p> <p>Upload application via SMS—Sends an SMS message containing the application URL to the device. You must click on the URL to download the application.</p> <p>NOTES Available delivery methods depend on the device platform. SMS delivery requires a modem—contact your Keynote Solutions Consultant.</p>
Source: Upload application from a file		Select to choose an application file from your file system.
	Type	Select the application type (device platform).

Field/Control	Sub-Control	Description
	Path to application	Click Browse to navigate your file system and select the application file.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.20 Browser Open

The Browser Open web utility enables you to start a native browser session on a specified web page from anywhere on the device. Additionally, you can opt to close all previous native browser sessions and clear the browser cache before opening the specified page.

NOTE Please review the [Requirements](#) for using Web commands.

Figure 12-42 Browser Open

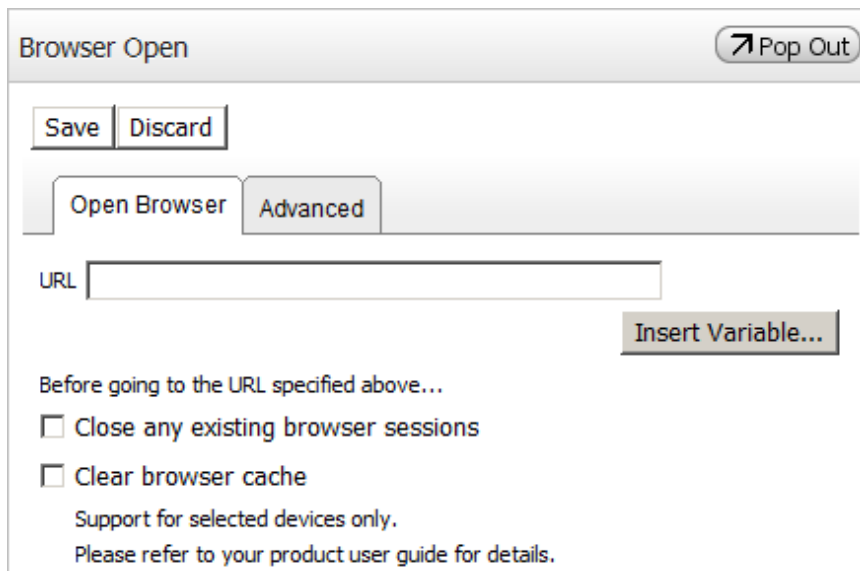


Table 12-33 Browser Open Command Properties

Control/Field	Description
URL	Enter the address to navigate to. Click Insert Variable to pass in the value from a variable.
Close any existing browser sessions	Check to close all prior browser sessions before opening the native browser at the page specified.
Clear browser cache	Check to clear the browser cache before opening the requested session.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.21 Close All Browser Sessions

The Close All Browser Sessions web utility closes all open sessions of the device native browser and optionally, clears the browser cache.

NOTE Please review the [Requirements](#) for using Web commands.

Figure 12-43 Close All Browser Sessions

Table 12-34 Browser Close Command Properties

Control/Field	Description
Clear browser cache	Check to clear the browser cache before closing all open native browser sessions.

In the **Advanced** tab, use the **Checkpoint** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.22 Toggle Recording

You can use the Toggle Recording command to start/stop capturing device images and video for insertion into test results. Use a pair of these commands in your script to indicate start and stop points for capturing device video/snapshots (See [Commands in the Capture Category](#) in [Proofs](#).)

Figure 12-44 Toggle Recording Command Properties

NOTE This command is currently not operational in Mobile App Monitoring.

Table 12-35 Toggle Recording Command Properties

Control/Field	Sub-Field	Description
Start Video Recording radio button		Select to start capturing device video. Insert the command in your script where you wish to start capturing proofs.
	Include Audio	Check to include device audio in the video file.
Start Image Capturing radio button		Select to capture images at regular intervals. Insert the command in your script where you wish to start capturing proofs.
	Capture every x seconds	Enter a number for the interval (in seconds) at which to capture images.
Stop		Select to stop capturing video or images. Insert in your script at the point at which you wish to stop capturing proofs.

You can enter a **Comment** in the **Advanced** tab.

12.23 Capture from Device

Use Capture from Device to capture snapshots or video of the current device screen. Script execution is halted until capture using this command is completed. Use this command to capture automatic and dynamic changes to the device screen as when streaming video. See also [Commands in the Capture Category](#) in [Proofs](#).

Figure 12-45 Capture from Device—Image Capture

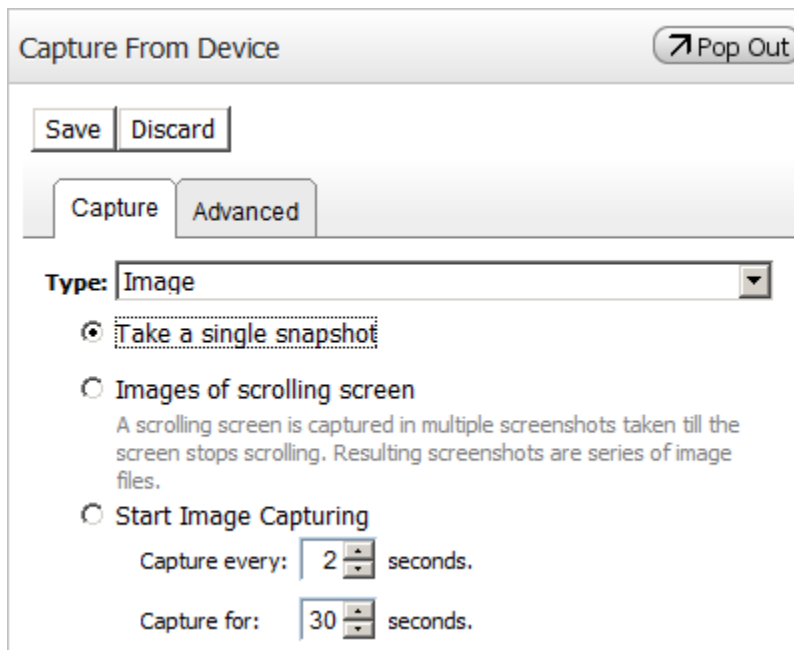


Figure 12-46 Capture from Device—Video Capture

Table 12-36 Capture from Device Command Properties

Control/Field	Sub-Field	Description
Type: Image		Select to capture images at regular intervals.
	Take a single snapshot radio button	Select to capture the current device screen.
	Images of scrolling screen radio button	Select to capture images as the device screen is scrolled (e.g., of a web page that is longer than the device screen).
	Start Image Capturing radio button	Select to capture images at regular intervals for a fixed length of time.
	Capture every x seconds	Enter an interval (in seconds) at which to capture images.
	Capture for x seconds	Enter duration (in seconds) for which to capture images.
Type: Video		Select to capture device video in MPEG format.
	Capture for x seconds	Enter duration (in seconds) for which to capture video.
	Include Audio	Check to include device audio in the video file.

You can enter a **Comment** in the **Advanced** tab.

12.24 Toggle Extract Log

Toggle Extract Log indicates start and end points in your script for capturing the device log, which is displayed in test results. In your script, use a pair of these commands to indicate start and stop points for capturing the device log. (See also [Commands in the Capture Category](#) in [Proofs](#).)

Figure 12-47 Toggle Extract Log—Start and Stop Settings

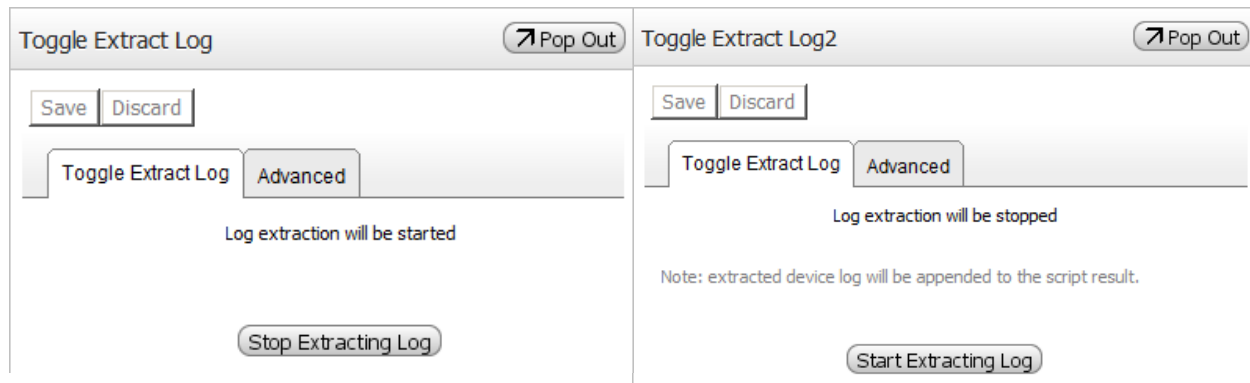


Table 12-37 Toggle Extract Log Command Properties

Control/Field	Description
Stop Extracting Log	Click in the second instance of the command in your script to indicate the stop point for capturing device log. (Simply drag the command into your script to indicate the start point for capturing device log.)
Start Extracting Log	Appears when you have set a command to stop extracting the device log.

12.25 Web Element

The Web Element command enables you to select an element from a web page and perform an appropriate action. Supported actions include extracting a value to a variable and setting a value (e.g., in a text box). Review the [prerequisites](#) for using Web commands.

You must have your device acquired and pointed to the test web page in order to use this command. Web commands can only act on web elements. (Refer to [Web Elements](#) in [Working with Commands](#) for step-by-step instructions on using Web commands.)

Figure 12-48 Web Element

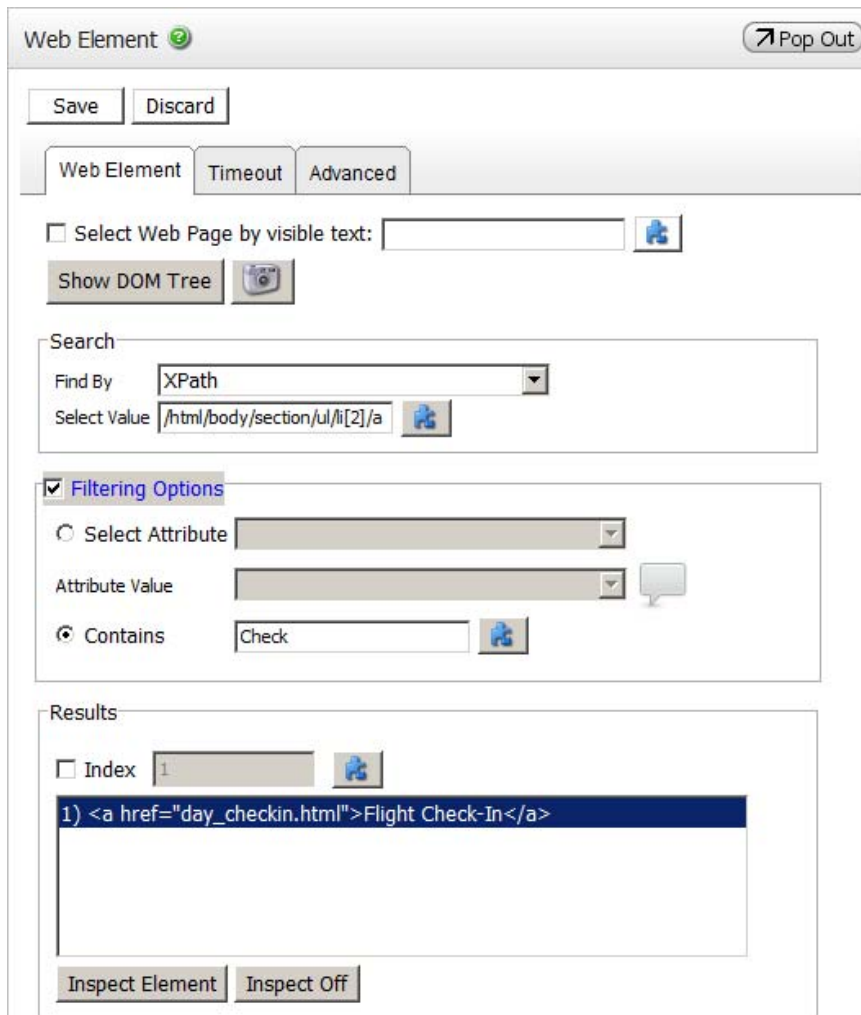

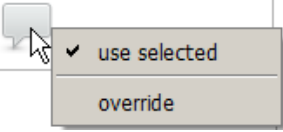









Table 12-38 Web Element Command Properties—Web Element Tab

Control/Field	Sub-Field	Description
Camera icon 		Click to refresh the DOM, e.g., if you have navigated to a different web page on the device.
Show DOM Tree		Click to view page markup in a DOM in a separate window (see Figure 12-49 below).
Find By		Drop-down list to select a search criterion to locate an element: Name – search by the name attribute of an element. ID – search by the id attribute of an element. Text – search by the text between opening and closing tags of an element. Tag – search by the tag applied to an element. XPath – search by the path expression identifying the element in the DOM. CSS Selector – search by the CSS Selector location of the element.

Control/Field	Sub-Field	Description
Select Value		Drop-down list with values matching the search criterion when searching by Name, ID, Text, or Tag . When searching by XPath or CSS Selector , you can enter a value in the field. When searching by XPath , you can copy the path expression of an element from the DOM viewer.
	Override value controls 	Available for entering a value when searching by Name, ID, Text, or Tag . Hover to view options: By default, the field is set to use selected value from the Select Value drop-down list. Click override to enter a value of your own or to pass in the value contained in a variable.
	Puzzle piece icon 	Click to pass in the value contained in a variable. Available when you override a value from the Select Value drop-down list. Also available when you search for an element by XPath or CSS Selector .
	Evaluate	Available when searching by CSS Selector . Click after entering a CSS Selector location to view search results.
Filtering Options		Check to apply additional filters to search results (listed in the Results pane).
	Attribute radio button	Click to perform additional filtering by attribute.
	Attribute Value	Drop-down list with values matching the attribute selected Hover over the override button  to override the attribute value. You can enter a value or pass in the value contained in a variable by clicking the puzzle piece icon  .
	Contains radio button	Click to perform additional filtering by text between element opening and closing tags. Enter a value in the field provided or pass in the value contained in a variable by clicking the puzzle piece icon  .
Index		Click to perform additional filtering by the index number of search results listed in the Results pane. Enter a number in the field provided or pass in the value contained in a variable by clicking the puzzle piece icon  .
Inspect Element		Click after selecting an element from the Results pane to highlight it on the device screen.
Inspect Off		Turn inspect mode off.
Action for Element		Drop-down list to specify interaction with a chosen element: Click – clicks selected element. Get Count – counts the elements matching the search criteria specified. Get Value – extracts text between opening and closing tags or the value entered in a field. Set Value – allows you to specify a value for a selected element, e.g., a form field. Submit – clicks a button of type submit – ensure that you choose the correct element for this action.

Control/Field	Sub-Field	Description
Value for Element		<p>If you choose the Set Value action, enter a value in the field provided or click the puzzle piece icon  to pass in the value from a variable.</p> <p>If you choose the Get Count or Get Value actions, click the puzzle piece icon  to select a variable to store the value extracted.</p>

You can click **Show DOM Tree** to view the page DOM in a separate window.

Figure 12-49 DOM Tree from the Web Element Command

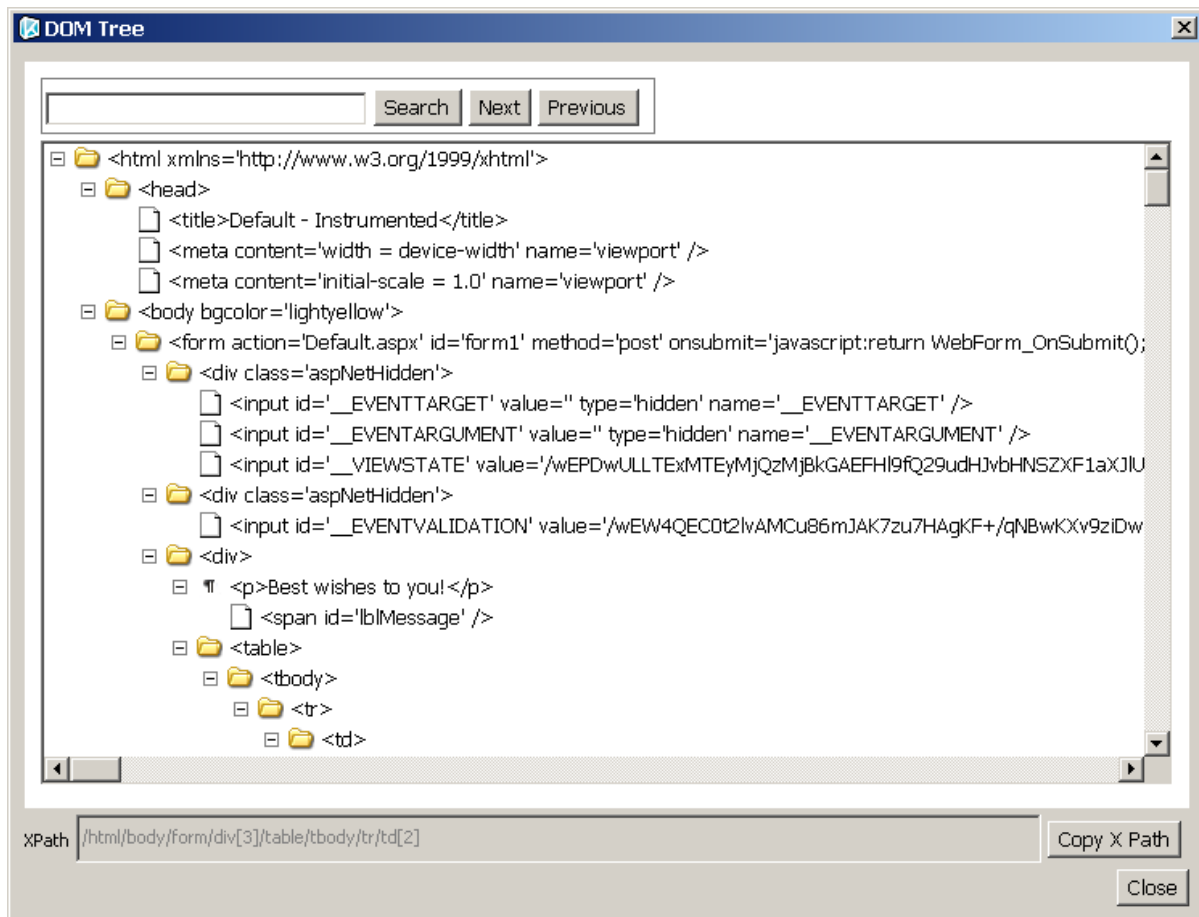


Table 12-39 DOM Tree Window Controls

Control/Field	Description
Search	Enter a search string in the field provided and click to view all matching results highlighted in the DOM. This field is case sensitive.
Next	Click to highlight the next result when there are multiple search results.
Previous	Click to highlight the previous result when there are multiple search results.
XPath	Non-editable field – displays the path expression of a selected node.
Copy XPath	Click to copy the path expression to the Select Value field in the command (when searching for an element by XPath).

Control/Field	Description
Close	Click to exit the DOM viewer and ensure that the device does not remain in inspect mode.


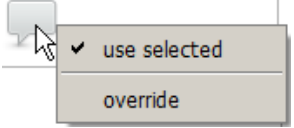

Use the [Timeout tab](#) to define script action if the command fails.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.26 Web Wait

Web Wait enables you to use an element as a Web-based [reference point](#) to verify a sequence of web interactions. The command waits for an element to appear on the page, to come into focus or blur, or to become visible or hidden based on previous steps.

Table 12-40 Web Wait Command Properties—Web Wait Tab

Control/Field	Sub-Field	Description
Camera icon 		Click to refresh the DOM, e.g., if you have navigated to a different web page on the device.
Show DOM Tree		Click to view page markup in a DOM in a separate window (see Figure 12-49).
Find By		Drop-down list to select a search criterion to locate an element: Name —search by the name attribute of an element. ID —search by the id attribute of an element. Text —search by the text between opening and closing tags of an element. Tag —search by the tag applied to an element. XPath —search by the path expression identifying the element in the DOM. CSS Selector —search by the CSS Selector location of the element.
Select Value		Drop-down list with values matching the search criterion when searching by Name , ID , Text , or Tag When searching by XPath or CSS Selector , you can enter a value in the field. When searching by XPath , you can copy the path expression of an element from the DOM viewer.
	Override value controls 	Available for entering a value when searching by Name , ID , Text , or Tag . Hover to view options: By default, the field is set to use selected value from the Select Value drop-down list. Click override to enter a value of your own or to pass in the value contained in a variable.
	Puzzle piece icon 	Click to pass in the value contained in a variable. Available when you override a value from the Select Value drop-down list. Also available when you search for an element by XPath or CSS Selector .
	Evaluate	Available when searching by CSS Selector . Click after entering a CSS Selector location to view search results.
Filtering Options		Check to apply additional filters to search results (listed in the Results pane).
	Attribute radio button	Click to perform additional filtering by attribute.





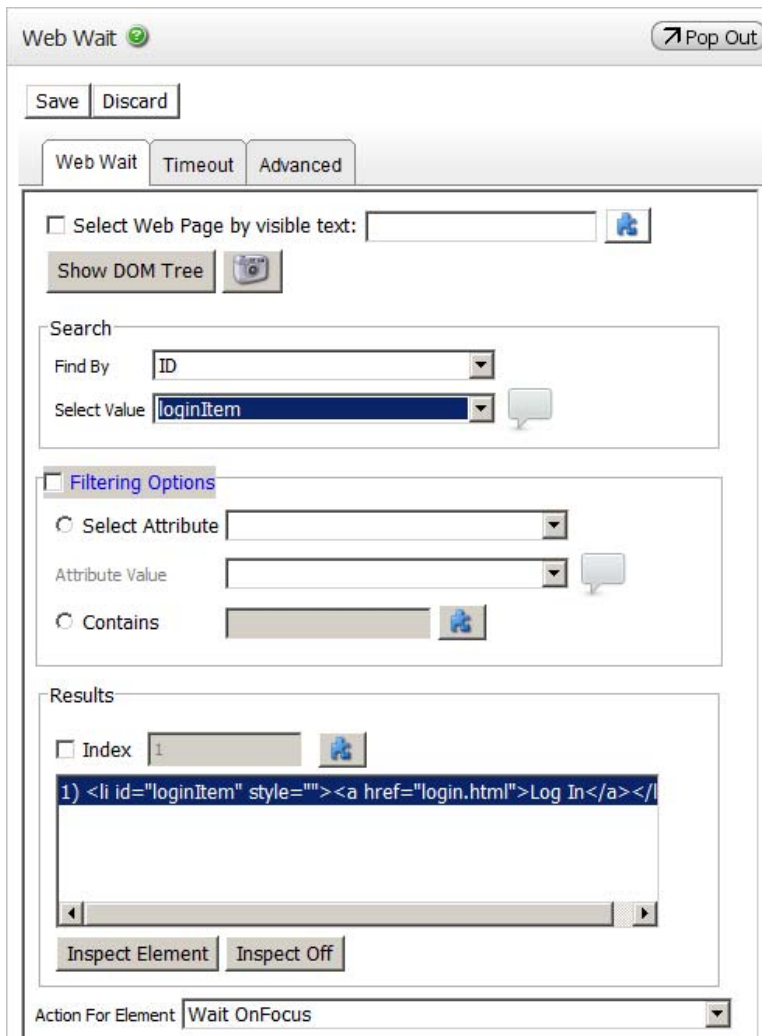
Control/Field	Sub-Field	Description
	Attribute Value	Drop-down list with values matching the attribute selected Hover over the override button  to override the attribute value. You can enter a value or pass in the value contained in a variable by clicking the puzzle piece icon  .
	Contains radio button	Click to perform additional filtering by text between element opening and closing tags. Enter a value in the field provided or pass in the value contained in a variable by clicking the puzzle piece icon  .
Index		Click to perform additional filtering by the index number of search results listed in the Results pane. Enter a number in the field provided or pass in the value contained in a variable by clicking the puzzle piece icon  .
Inspect Element		Click after selecting an element from the Results pane to highlight it on the device screen.
Inspect Off		Turn inspect mode off.
Action for Element		Drop-down list to specify interaction with a chosen element: Wait Hidden – confirms that the selected element is not displayed on the page, even in the portion not visible on the device screen. Wait OnFocus – checks that a selected element comes into focus, e.g., after a form field is selected. Wait Visible – checks that the selected element is displayed on the page, even if the screen must be scrolled in order to view it. Wait OnBlur – checks that a selected element loses focus, e.g., after a form field is deselected. Wait Exists – checks to find the selected element on the page.

Figure 12-50 Web Wait



You can click **Show DOM Tree** to view the page DOM in a separate window. See the command reference for the [Web Element](#) command for a description of the controls available in this window.

Use the [Timeout tab](#) to define script action if the command fails.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.27 Web Form

The Web Form command enables you to search for a form and then specify an appropriate action/value for each form field before submitting the form.

Figure 12-51 Web Form

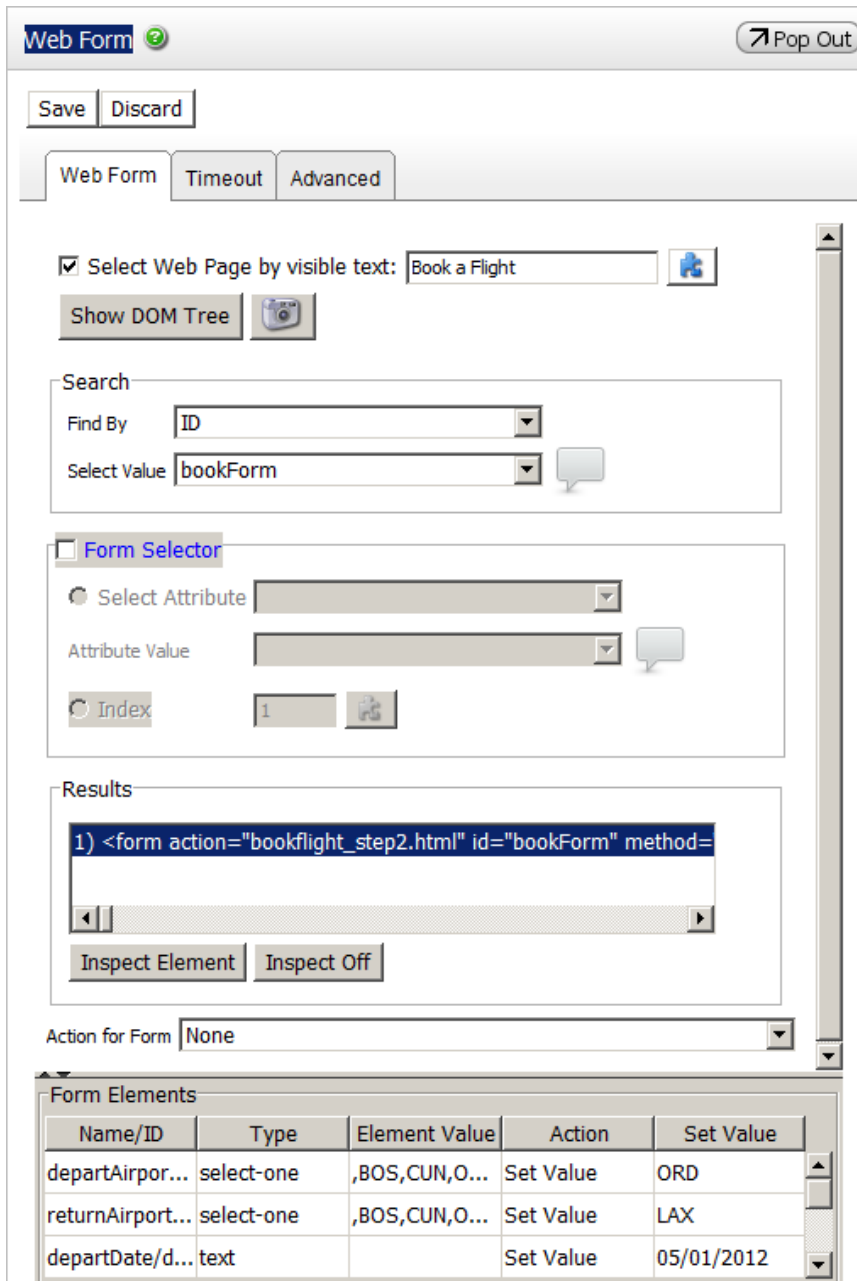

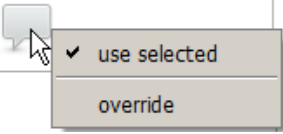






Table 12-41 Web Form Command Properties – Web Form Tab

Control/Field	Sub-Field	Description
Camera icon 		Click to refresh the DOM, e.g., if you have navigated to a different web page on the device.

Control/Field	Sub-Field	Description
Show DOM Tree		Click to view page markup in a DOM in a separate window (see Figure 12-49).
Find By		Drop-down list to select a search criterion to locate a form element: Name – search by the name attribute of the element. ID – search by the id attribute of the element. Text – search by the text between opening and closing tags of the element. Tag – search by the tag applied to the element. XPath – search by the path expression identifying the element in the DOM. CSS Selector – search by the CSS Selector location of the element.
Select Value		Drop-down list with values matching the search criterion when searching by Name, ID, Text, or Tag When searching by XPath or CSS Selector , you can enter a value in the field. When searching by XPath , you can copy the path expression of an element from the DOM viewer.
	Override value controls 	Available for entering a value when searching by Name, ID, Text, or Tag . Hover to view options: By default, the field is set to use selected value from the Select Value drop-down list. Click override to enter a value of your own or to pass in the value contained in a variable.
	Puzzle piece icon 	Click to pass in the value contained in a variable. Available when you override a value from the Select Value drop-down list. Also available when you search for an element by XPath or CSS Selector .
	Evaluate	Available when searching by CSS Selector . Click after entering a CSS Selector location to view search results.
Filtering Options		Check to apply additional filters to search results (listed in the Results pane).
	Attribute radio button	Click to perform additional filtering by attribute.
	Attribute Value	Drop-down list with values matching the attribute selected Hover over the override button  to override the attribute value. You can enter a value or pass in the value contained in a variable by clicking the puzzle piece icon  .
	Index radio button	Click to perform additional filtering by the index number of search results listed in the Results pane. Enter a number in the field provided or pass in the value contained in a variable by clicking the puzzle piece icon  .
Inspect Element		Click after selecting a form element from the Results pane to highlight it on the device screen.
Inspect Off		Turn inspect mode off.
Form Elements pane		Select form fields in turn to work with them.

Control/Field	Sub-Field	Description
	Action column	<p>Click in the column to select an action for the field:</p> <p>None – no action to be performed on the element</p> <p>Set Value – select to specify a value for a field by choosing from available options, entering a value, or passing it in from a variable as appropriate.</p> <p>Checked – select to mark a check box or radio button.</p> <p>Unchecked – select to uncheck a check box or radio button.</p> <p>NOTE Buttons of type submit are not listed with form elements. To click this button, choose the Submit action for the form as a whole.</p> <p>Any other type of button at the bottom of a form is listed with form elements. Do not choose any action for this button. Instead, use the Web Touch command after Web Form to click it. This ensures that the button is clicked after all fields have been filled out.</p>
	Set Value column	<p>Enter a value or select from available options. You can also click Use Variable to pass in a value from a variable.</p>
Action for Form		<p>Select an option to perform an action on the form as a whole after fields have been filled out:</p> <p>None – no action to be performed on the form as a whole</p> <p>Submit – select to click a button of type submit after the form has been filled out.</p> <p>NOTE If the button at the bottom of a form is <i>not</i> of type submit, use the Web Touch command to click it. This ensures that the button is clicked after all fields have been completed.</p>

You can click **Show DOM Tree** to view the page DOM in a separate window. See the command reference for the [Web Element](#) command for a description of the controls available in this window.

Use the **Timeout** tab to define script action if the command fails.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.28 Web Touch

The Web Touch command enables you to search for and specify an element to be clicked on a web page at run time.

Figure 12-52 Web Touch

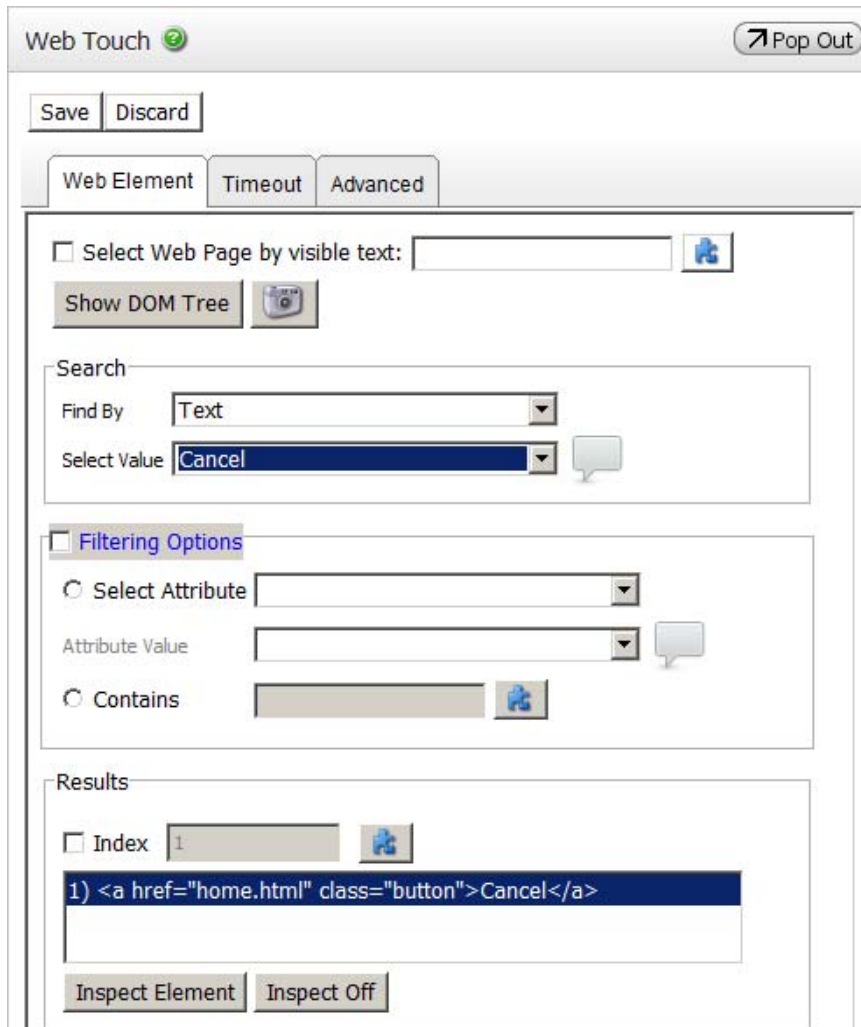

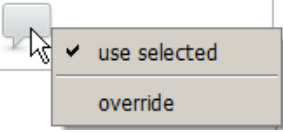







Table 12-42 Web Touch Command Properties – Web Element Tab

Control/Field	Sub-Field	Description
Camera icon 		Click to refresh the DOM, e.g., if you have navigated to a different web page on the device.
Show DOM Tree		Click to view page markup in a DOM in a separate window (see Figure 12-49 below).
Find By		Drop-down list to select a search criterion to locate an element: Name – search by the name attribute of an element. ID – search by the id attribute of an element. Text – search by the text between opening and closing tags of an element. Tag – search by the tag applied to an element. XPath – search by the path expression identifying the element in the DOM. CSS Selector – search by the CSS Selector location of the element.

Control/Field	Sub-Field	Description
Select Value		Drop-down list with values matching the search criterion when searching by Name, ID, Text, or Tag . When searching by XPath or CSS Selector , you can enter a value in the field. When searching by XPath , you can copy the path expression of an element from the DOM viewer.
	Override value controls 	Available for entering a value when searching by Name, ID, Text, or Tag . Hover to view options: By default, the field is set to use selected value from the Select Value drop-down list. Click override to enter a value of your own or to pass in the value contained in a variable.
	Puzzle piece icon 	Click to pass in the value contained in a variable. Available when you override a value from the Select Value drop-down list. Also available when you search for an element by XPath or CSS Selector .
	Evaluate	Available when searching by CSS Selector . Click after entering a CSS Selector location to view search results.
Filtering Options		Check to apply additional filters to search results (listed in the Results pane).
	Attribute radio button	Click to perform additional filtering by attribute.
	Attribute Value	Drop-down list with values matching the attribute selected Hover over the override button  to override the attribute value. You can enter a value or pass in the value contained in a variable by clicking the puzzle piece icon  .
	Contains radio button	Click to perform additional filtering by text between element opening and closing tags. Enter a value in the field provided or pass in the value contained in a variable by clicking the puzzle piece icon  .
Index		Click to perform additional filtering by the index number of search results listed in the Results pane. Enter a number in the field provided or pass in the value contained in a variable by clicking the puzzle piece icon  .
Inspect Element		Click after selecting an element from the Results pane to highlight it on the device screen.
Inspect Off		Turn inspect mode off.

You can click **Show DOM Tree** to view the page DOM in a separate window. See the command reference for the [Web Element](#) command for a description of the controls available in this window.

Use the [Timeout tab](#) to define script action if the command fails. In the **Advanced** tab, use the **Checkpoint** control to specify [proofs](#) for display in test results or enter a **Comment**.

12.29 Launch App

Use this command to select and open a native application from anywhere on the device.

NOTE Please review the [Requirements](#) for using Object commands.

Figure 12-53 Launch App

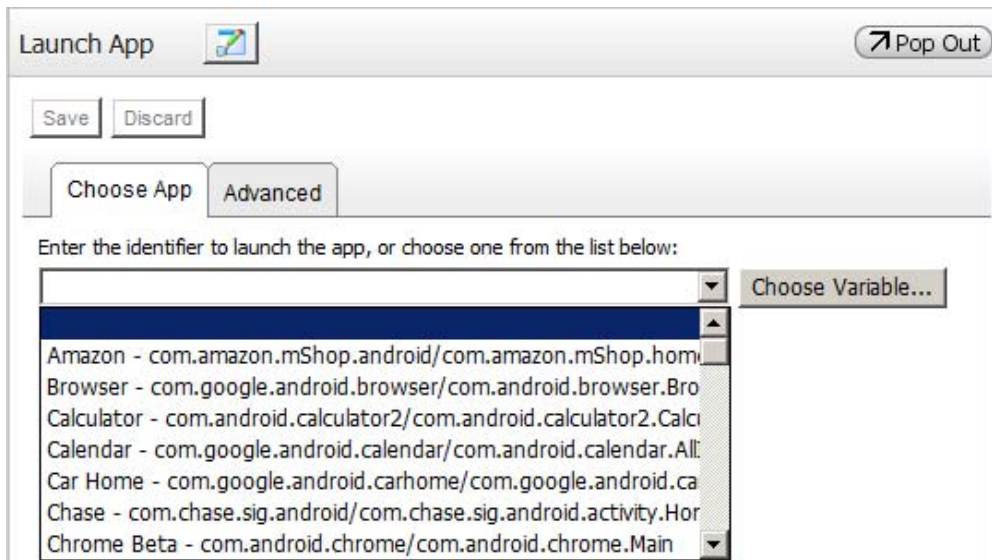


Table 12-43 Launch App Command Properties

Control/Field	Description
Choose App drop-down list	Select an application to open. NOTE All applications are listed. Be sure to choose a native application.
Choose Variable	Pass in the value of the application to open from a variable.

In the **Advanced** tab, you can specify [proofs](#), a command timeout, or enter a **Comment**.

12.30 Close App

Use this command to select and close a native application from anywhere on the device.

NOTE Please review the [Requirements](#) for using Object commands.

Figure 12-54 Close App

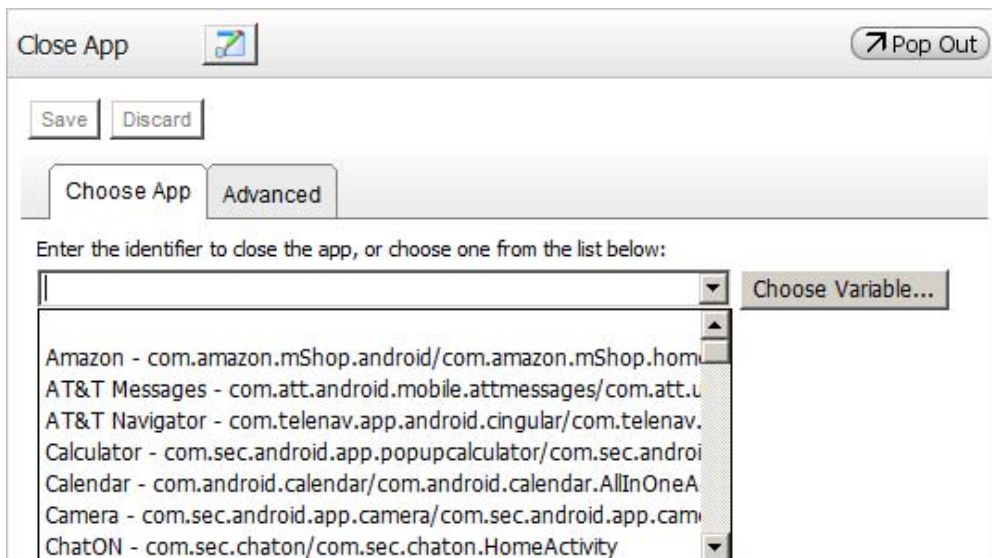


Table 12-44 Close App Command Properties

Control/Field	Description
Choose App drop-down list	Select an application to close. NOTE All applications are listed. Be sure to choose a native application.
Choose Variable	Pass in the value of the application to open from a variable.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for test results, define a command completion **Timeout**, or enter a **Comment**.

12.31 Object Touch

Object Touch enables you to choose a native object to be touched or clicked at run time. See [Native Objects in Commands](#) in [Working with Commands](#) for more information.

Figure 12-55 Object Touch

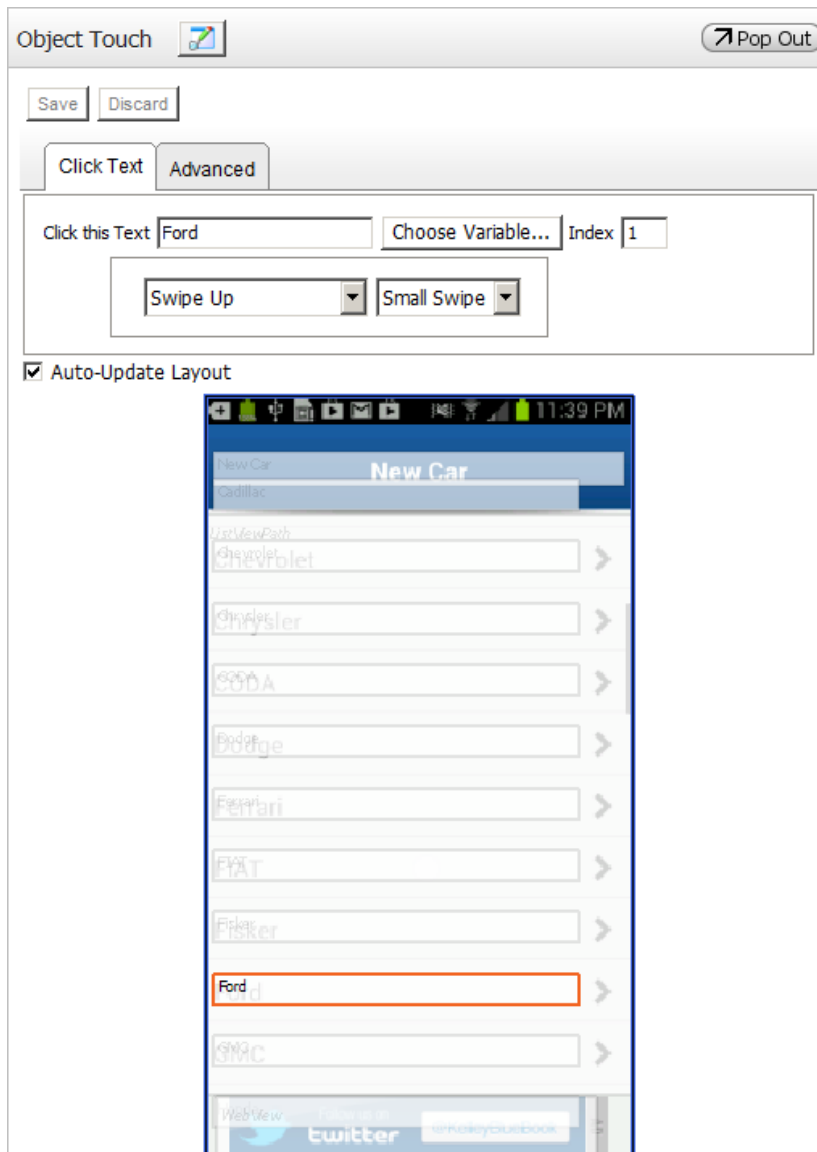


Table 12-45 Object Touch Command Properties

Control/Field	Description
Click this Text	Object text to click – automatically populated when you select an object with text. You can also enter a value or click Choose Variable to pass in the value contained in a variable.
Index	If more than one object matches the text entered, select a specific object by entering its index number.
Swipe controls	<p>Swipe Disabled – default; no swipe is implemented.</p> <p>Swipe Down – swipes the device screen as if you were dragging your finger down over the screen.</p> <p>Swipe Up – swipes the device screen as if you were dragging your finger up over the screen.</p> <p>Swipe Left – swipes the device screen as if you were dragging your finger left over the screen.</p> <p>Swipe Right – swipes the device screen as if you were dragging your finger right over the screen.</p>
Swipe size controls	<p>Small Swipe – swipes approximately one item at a time.</p> <p>Big Swipe – swipes approximately one screen at a time.</p>
Auto-Update Layout	Checked by default; updates the object layout when you navigate to another device screen.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for test results, define a command completion **Timeout**, or enter a **Comment**.

12.32 Object Edit

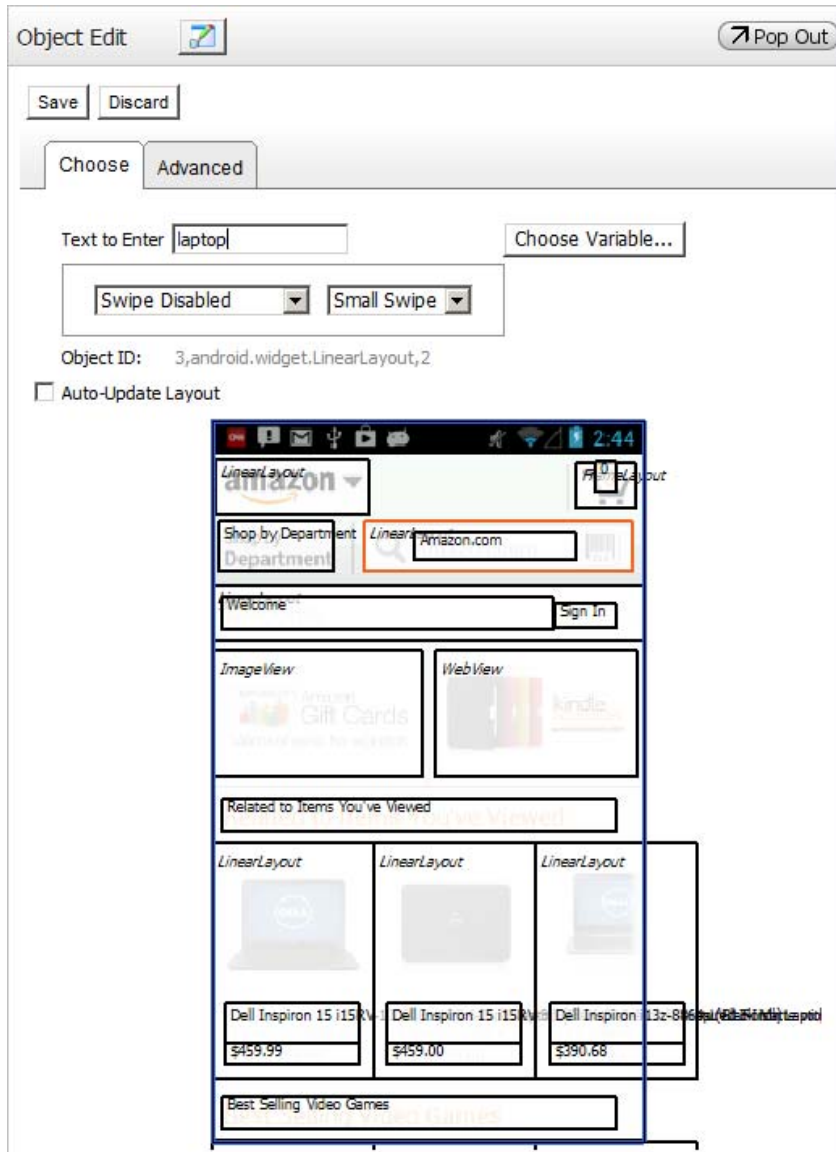
Object Edit enables you to enter a value into a native object field at run time. See [Native Objects in Commands](#) in [Working with Commands](#) for more information.

Table 12-46 Object Edit Command Properties

Tab	Control/Field	Description
Choose	Text to Enter	Specify a value to enter in an object field or click Choose Variable to pass in the value contained in a variable.
	Swipe controls	<p>Swipe Disabled – default; no swipe is implemented.</p> <p>Swipe Down – swipes the device screen as if you were dragging your finger down over the screen.</p> <p>Swipe Up – swipes the device screen as if you were dragging your finger up over the screen.</p> <p>Swipe Left – swipes the device screen as if you were dragging your finger left over the screen.</p> <p>Swipe Right – swipes the device screen as if you were dragging your finger right over the screen.</p>
	Swipe size controls	<p>Small Swipe – swipes approximately one item at a time.</p> <p>Big Swipe – swipes approximately one screen at a time.</p>
	Auto-Update Layout	Checked by default; updates the object layout when you navigate to another device screen.
Advanced	Check Point	Specify proofs for test results
	Clear Text First	Select true to clear the field before entering data in it (false by default).

Tab	Control/Field	Description
	Click To Focus	Select true (default) to set focus in the field before entering data.
	Comment	Enter a comment.
	Timeout Seconds	Enter a maximum command completion time.
	is Password Field	Select true if the field is for password entry and data must be encrypted.

Figure 12-56 Object Edit



12.33 Object Extract Text

Object Extract Text enables you to store text extracted from a selected object in a variable. See [Native Objects in Commands](#) in [Working with Commands](#) for more information.

Figure 12-57 Object Touch

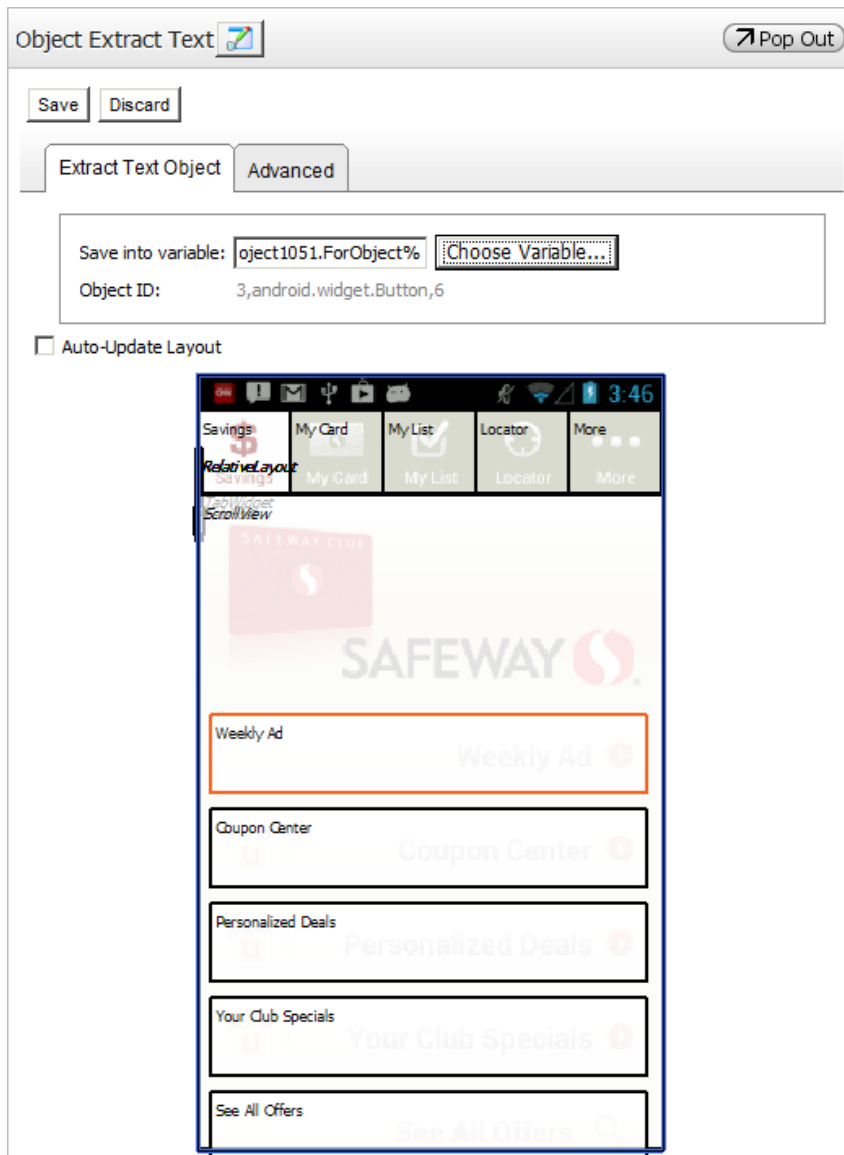


Table 12-47 Object Touch Command Properties

Control/Field	Description
Save into Variable	Click Choose Variable to select a variable in which to store text extracted from the object. The variable name is displayed in this field.
Auto-Update Layout	Checked by default; updates the object layout when you navigate to another device screen.

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for test results, define a command completion **Timeout**, or enter a **Comment**.

12.34 Timer

The Timer command in Mobile App Monitoring enables you to apply named timers to portions of your script so they may be tracked and monitored. You can then specify performance criteria for these script sections in MyKeynote. You can also generate alarms based on performance criteria.

Use a pair of Timer commands to mark out the script portion you wish to track. If wrapping Timer commands around Wait Event, be sure to use a stop timer in every branch other than the Timeout branch.

NOTE The corresponding Toggle Transaction command for DAE Monitoring is described in [DAE Monitoring Best Practice Workflow](#).

Figure 58 Timer

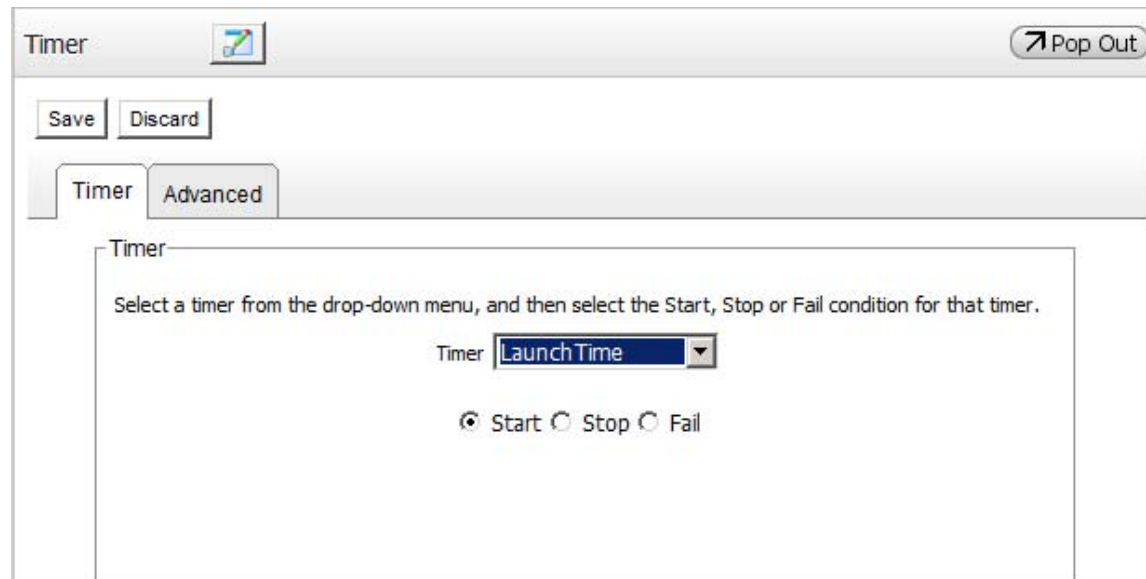


Table 12-48 Timer Command Properties

Control/Field	Description
Timer	Select a named timer from your project.
Timer type radio buttons	<p>Start – select if placing the command at the start of a script section you want to track.</p> <p>Stop – select if placing the command at the end of a script section you want to track.</p> <p>Fail – do not use.</p>

In the **Advanced** tab, use the **Check Point** control to specify [proofs](#) for test results or enter a **Comment**.

Glossary

The following terms are used throughout this document:

action—See [Concepts and Test Scripting Workflow](#) and [Actions](#).

automated test case—See [Concepts and Test Scripting Workflow](#) and [Automated Test Cases](#).

automated test cycle—See [Concepts](#) and [Automated Test Cycles](#).

device-independent—description of test scripts or scripting commands that are valid for all project devices: The Wait, Success, and Fail commands are examples of device-independent commands. Test cases are ideally constructed as device-independent scripts with calls to previously defined actions and states. Actions and states are device-independent in that they are defined for all project devices. However, they consist of device-specific implementations for each device.

device-specific—description of test scripts or scripting commands that capture screens or interactions specific to a device: The Send Keys command can be used to specify a key sequence that is valid for a specific device. Action and state implementations are device-specific scripts that allow you to account for differences in device interfaces.

implementation—See [Concepts and Test Scripting Workflow](#), [Implementing an Action](#), and [Implementing a State](#).

Java state—Java states can be used to poll an external application (such as an application that interacts with a handset) to ensure the application is in the correct state before continuing with a script.

manual test case—A manual test case is created in the Test Case Manager view and is made up of one or more test steps or step groups. Manual test cases require human intervention (vs. automatic execution) to run test steps and are executed as part of a test cycle. A manual test case can be partially or fully automated by associating its test steps with automated actions.

manual test cycle—A manual test cycle is created in the Test Case Manager view and consists of manual or automatic test cases assigned to run on a subset of project devices. As you execute a manual test cycle, you can track completion status and view results.

parameter—See [Concepts](#) and [Parameters and Variables](#).

project—See [Concepts](#) and [Projects](#).

proof—Proofs are device screenshots, video, or log files captured during test runs and displayed in test results. Proofs can be captured manually (while executing manual test cases) or automatically (for automated scripts). *See also* [Proofs](#).

state—A state is a device condition represented by reference audio, text, or image taken from the device screen and is used for script verification. *See also* [Concepts](#) and [States](#).

step group—A step group is a collection of reusable manual test steps and is relevant only to manual test cases.

test step—A test step is a component of a manual test case and represents a series of device interactions. Test step specifications include instructions for the manual tester and the proof required. A test step can also reference a step group. The automated counterpart of a test step is an action.

unpartitioned script—An unpartitioned script is one that works across all project devices. You would create an unpartitioned action if your project consisted of like devices that did not require separate implementations. Web and Object commands also enable you to create unpartitioned actions that work across devices—see [Web Elements](#) and [Native Objects](#).

variable—A variable allows you to store values in your script to be used as the basis for script logic (e.g., in the Loop or Branch commands). Variable values cannot be specified at runtime. *See also* [Parameters and Variables](#).

visual scripting command—The visual scripting environment of the Test Automation view allows you to drag and drop commands from a toolbar onto the script canvas. Visual scripting commands perform various functions on a device, implement script logic, and enable you call other scripts. Available commands depend on whether you are creating an action, test case, or test cycle. Commands are discussed in detail in [Working with Commands](#) and the [Command Reference](#).

web element-based scripting—Visual commands in the Web category enable direct interaction with elements in a web page, web application, or hybrid application. Web commands give you the ability to create unpartitioned scripts that operate across support device models and OS versions.

native object-based scripting—Commands in the Object category enable you to interact directly with native objects in native or hybrid applications. Using Object commands enables you to create unpartitioned scripts that operate across all project devices.