Best Practices for Object Scripting in Studio

Release 6.4

DeviceAnywhere Enterprise Monitoring 6.0

Mobile App Monitoring 8.0

DeviceAnywhere Enterprise Automation 6.4

August 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 INTELLECTURAL 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

1 Overview

DeviceAnywhere Enterprise and Mobile App Monitoring have automated scripting capabilities that vastly simplify script creation and management.

- Script recording—you can now record scripts out of the box by interacting directly with a device. And you can always edit a recorded script to add commands or branches.
- Object-level commands—you can work directly with web elements or native objects to create scripts that operate across multiple devices and are not sensitive to changes in your app UI. Object-level commands enable you to create and maintain fewer scripts.

Read on for best practices and information on recording scripts, general tips for testing mobile apps, and specific tips on testing web apps or native apps.

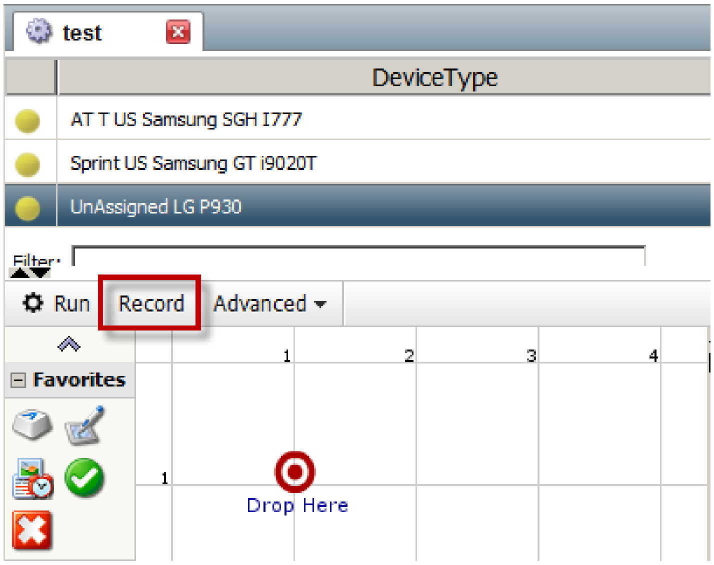
2 Script Recording

Many Ways to Script

With the script recorder and the ability to work with objects, there are now several ways to create automated scripts in Studio:

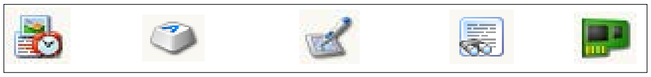
- Record scripts
    - The quickest way to create a script is to record simple, device-specific (or partitioned) implementations.
    - You can record unpartitioned web- or native-object scripts

    NOTE Native object recording on iOS is not available in DAE Monitoring and MAM.

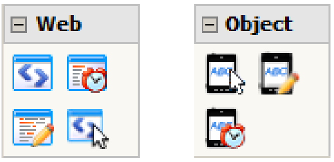


	DeviceType
●	AT T US Samsung SGH I777
●	Sprint US Samsung GT i9020T
●	UnAssigned LG P930

- Use UI-based commands—you can drag and drop commands onto the script canvas to work with the text or images from your device screen. Because of the differences in device UIs, these scripts are generally partitioned by device model. With Studio's visual scripting language, even those with little or no scripting experience can create implementations very quickly.



- Use Web or Object commands—instead of working with the visual content of a device screen, you work with the underlying web elements or native objects. These scripts are unpartitioned and can operate across devices and remain robust through UI changes.



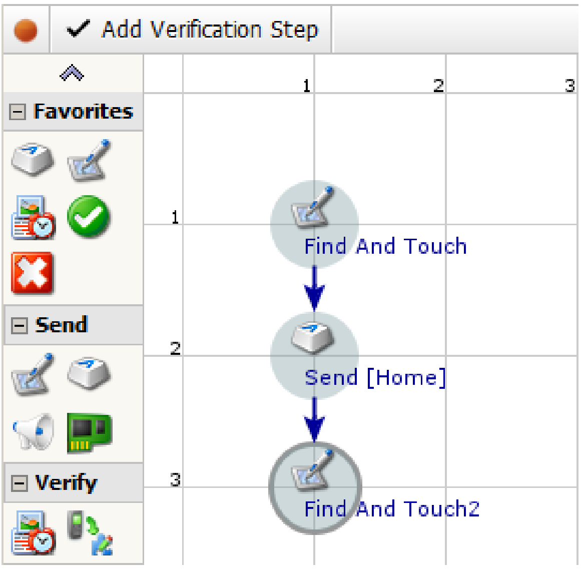
When to Use Recording

If you want to create a simple, single-branch script, recording is the way to go. For example, your script might consist of swiping the device screen and pressing a button. Recording is also useful to create a baseline script that you can modify later. You can always edit recorded scripts to add commands and make them more complex.

Recording creates a script with the following commands:

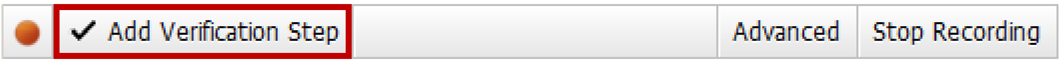
- Find and Touch commands—for touches and swipes
  - Hold the Ctrl key and drag your mouse to swipe the device screen.
- Send Keys commands—for key presses and data entry
- Web Touch and Web Element commands when recording in Web Mode
- Object Touch when recording Object commands

Figure 1 Recording a Script



While recording, you can insert non-recordable commands:

- Click **Add Verification Step** to define text-, image-, or other types of screen verifications.



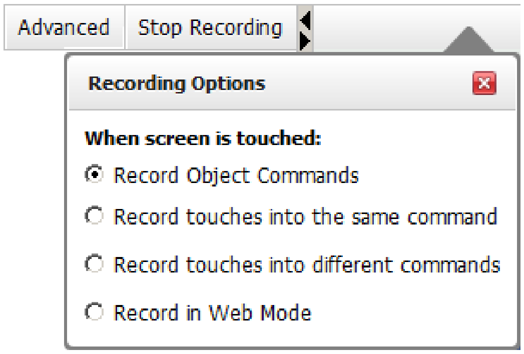
- Drag and drop any other command from the toolbar, e.g., to call a variable.

Recorded scripts are saved automatically.

Choosing a Recording Option

After you have acquired a device and clicked **Record** above the script canvas, you can choose recording options—**Advanced**. You can switch recording options as you record a script.

Figure 2 Recording Options



Recording Options

When screen is touched:

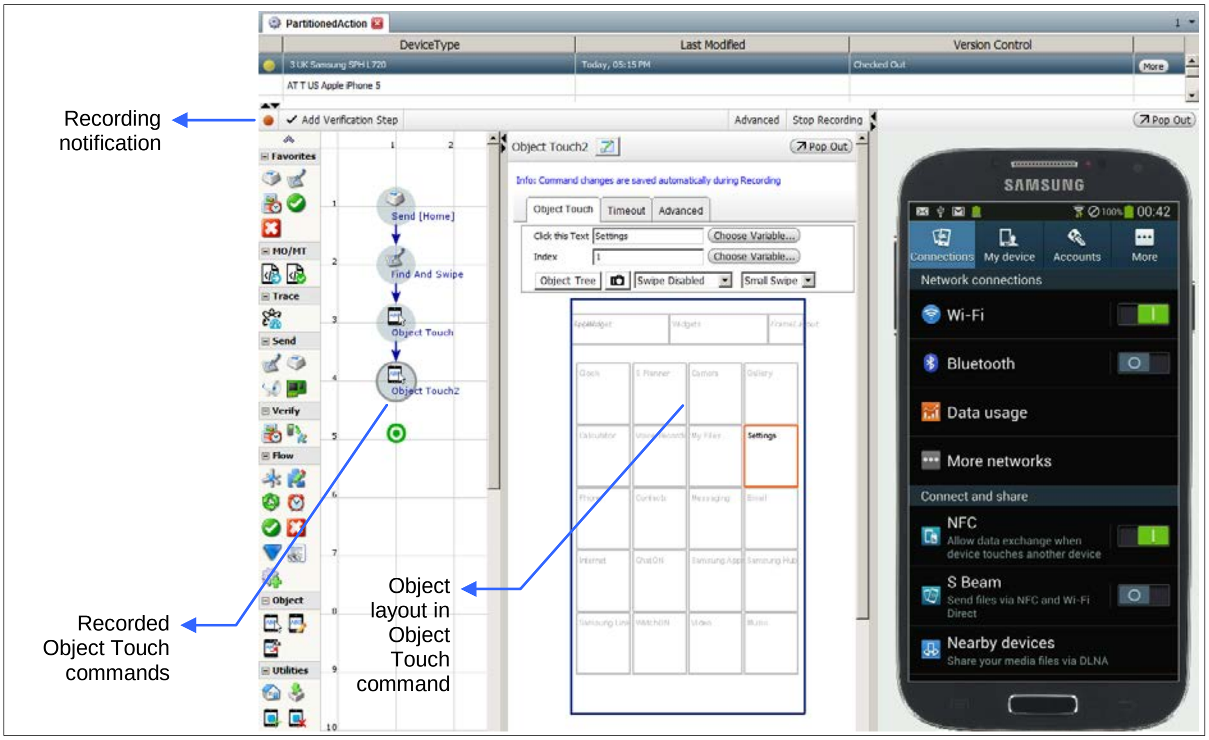
● Record Object Commands

○ Record touches into the same command

○ Record touches into different commands

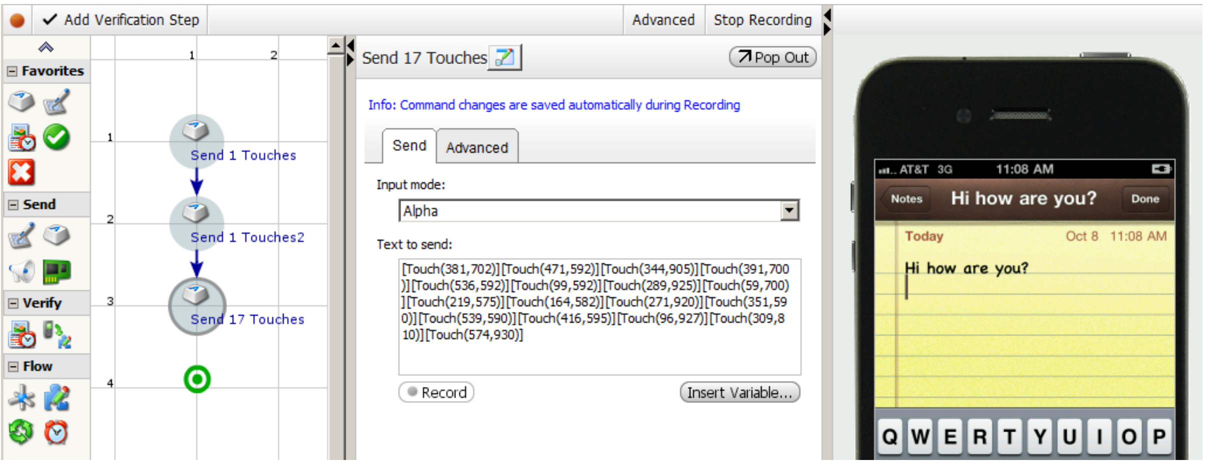
○ Record in Web Mode

- Recording object commands is the default recording mode on devices enabled for object-based scripting. On Android devices, you can interact with any native application; on iOS devices, you can only work with applications uploaded via Studio. When you click on a device, Object Touch commands are captured.



- Record consecutive touches in the same command. Consecutive key presses or screen taps are recorded into a single Send Keys command. If there is delay between presses/taps, they are recorded into separate Send Keys commands. Screen taps are defined in terms of the x and y coordinates, e.g., `[Touch(51,146)]`.

  Use this option when it makes sense to have consecutive key presses in one command, e.g., when entering content into a text message or a note.



Send 17 Touches

Info: Command changes are saved automatically during Recording

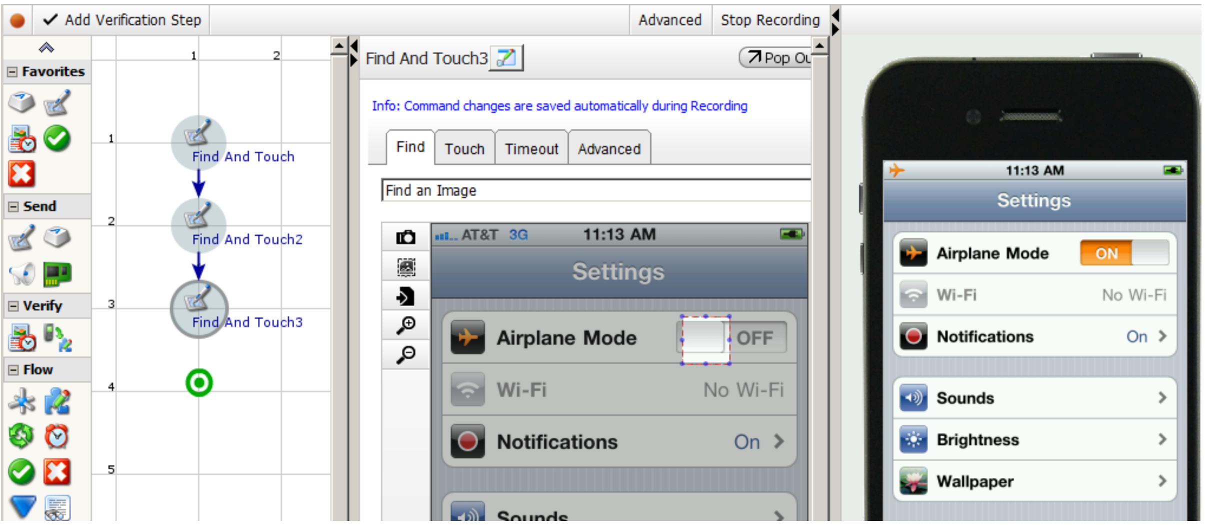
Input mode: Alpha

Text to send:

```
[Touch(381,702)][Touch(471,592)][Touch(344,905)][Touch(391,700
)][Touch(536,592)][Touch(99,592)][Touch(289,925)][Touch(59,700)
][Touch(219,575)][Touch(164,582)][Touch(271,920)][Touch(351,59
0)][Touch(539,590)][Touch(416,595)][Touch(96,927)][Touch(309,8
10)][Touch(574,930)]
```

- Record consecutive touches in separate commands (default for devices without object-level scripting enabled). 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.

  Use this mode when it's important to have visual information in your command, e.g., when moving from one application page to another, or when enabling or disabling important settings.

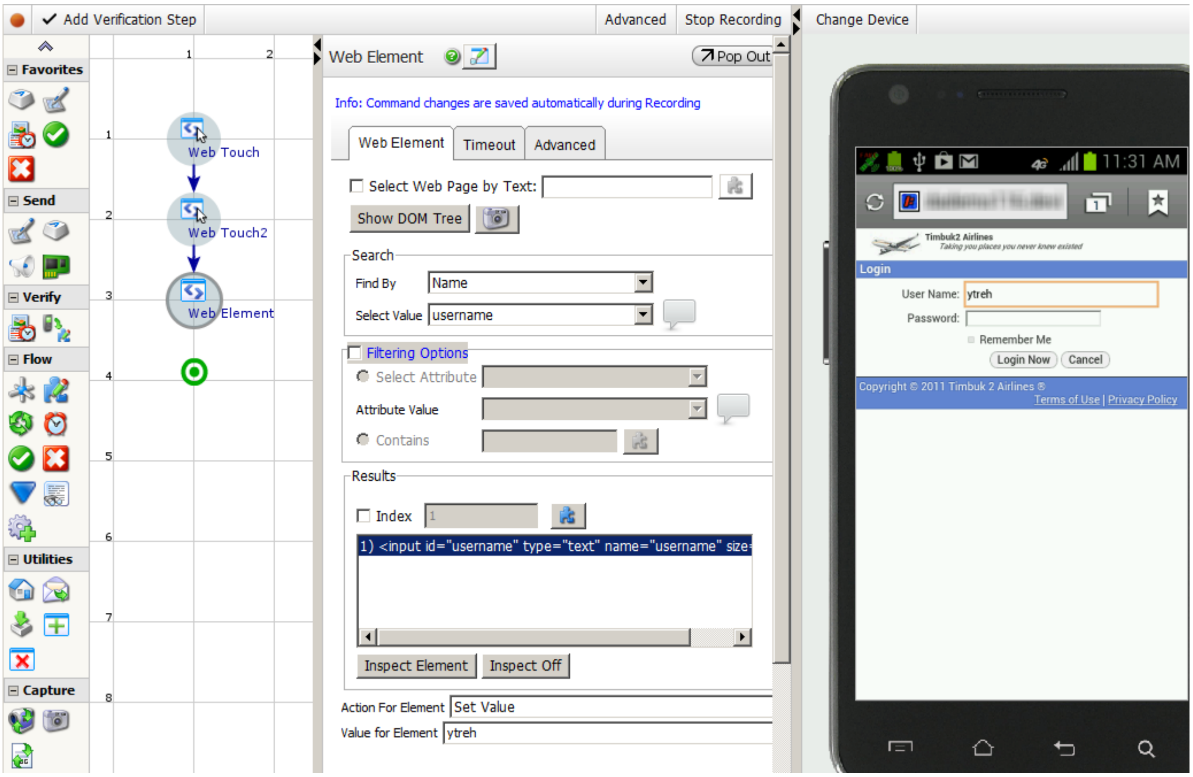


Find And Touch3

Info: Command changes are saved automatically during Recording

Find an Image

- **Record in Web Mode**—use to capture Web commands when interacting with a web application.



Info: Command changes are saved automatically during Recording

Find By: Name

Select Value: username

1) <input id="username" type="text" name="username" size

Action For Element: Set Value

Value for Element: ytreh

User Name: ytreh

You will need to launch your browser before recording in Web Mode. Web Mode captures Web Element and Web Touch commands inside a web page or application.

As Web commands interact directly with web elements, it is best not to mix Web commands and UI-based commands in the same script—see General Tips for Testing Mobile Apps below.

Recording Tips

- When recording, as when writing a script, think about repeatability—know the path you wish to take on the device and record only that. Remember, once you click **Record**, any device interaction is recorded into your script. Any extraneous clicks recorded will interfere with expected results.
- Start recording from a known place such as the device home screen. Then after recording, run your script from the same place, e.g., the home screen.
- Before recording, have the values that you want to enter into fields handy.
- If recording Web commands, make sure that your devices meet the requirements for doing so (review the Studio Scripting Guide).

3 General Tips for Testing Mobile Apps

Comparing Visual and Object-Level Scripts

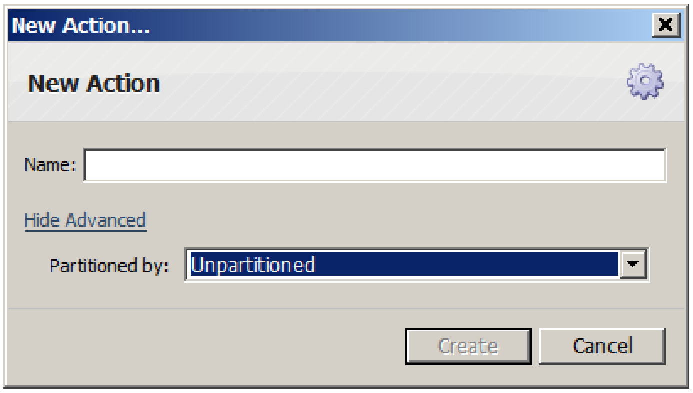
Table 1 Visual vs. Object-Level Scripts

	Visual Scripts	Object-Level Scripts
Commands	Use UI-based commands in any application.	Use Web commands for enabled web applications. Use Object commands for any native application on enabled Android devices; use on native applications uploaded via Studio on iOS.
UI Changes	Impacted by UI changes; image or text matching commands will need to be updated.	Reliable across UI changes.
Partitioned/ Unpartitioned	Typically requires device-specific actions with implementations per device model to account for UI differences.	Web element-based scripts operate across all devices. Native object-based scripts operate across devices in the same OS family.
Verifications	Script verification based on UI (device image or text).	Verification based on finding elements/objects.

Making the Most of Unpartitioned Scripts

With web or native object-level commands, instead of working with the visual content of a device screen, you work with the underlying web elements or native objects. Depending on how you set up your applications, underlying elements stay relatively stable across changes to the UI. In turn, you can set up unpartitioned scripts that work across devices. Web element-based scripts operate across all devices. Native object-based scripts operate across devices in the same OS family. You need not partition a script to account for differences in how your application is displayed on different devices.

Figure 3 Creating an Unpartitioned Action



New Action...

New Action

Name:

Partitioned by: Unpartitioned

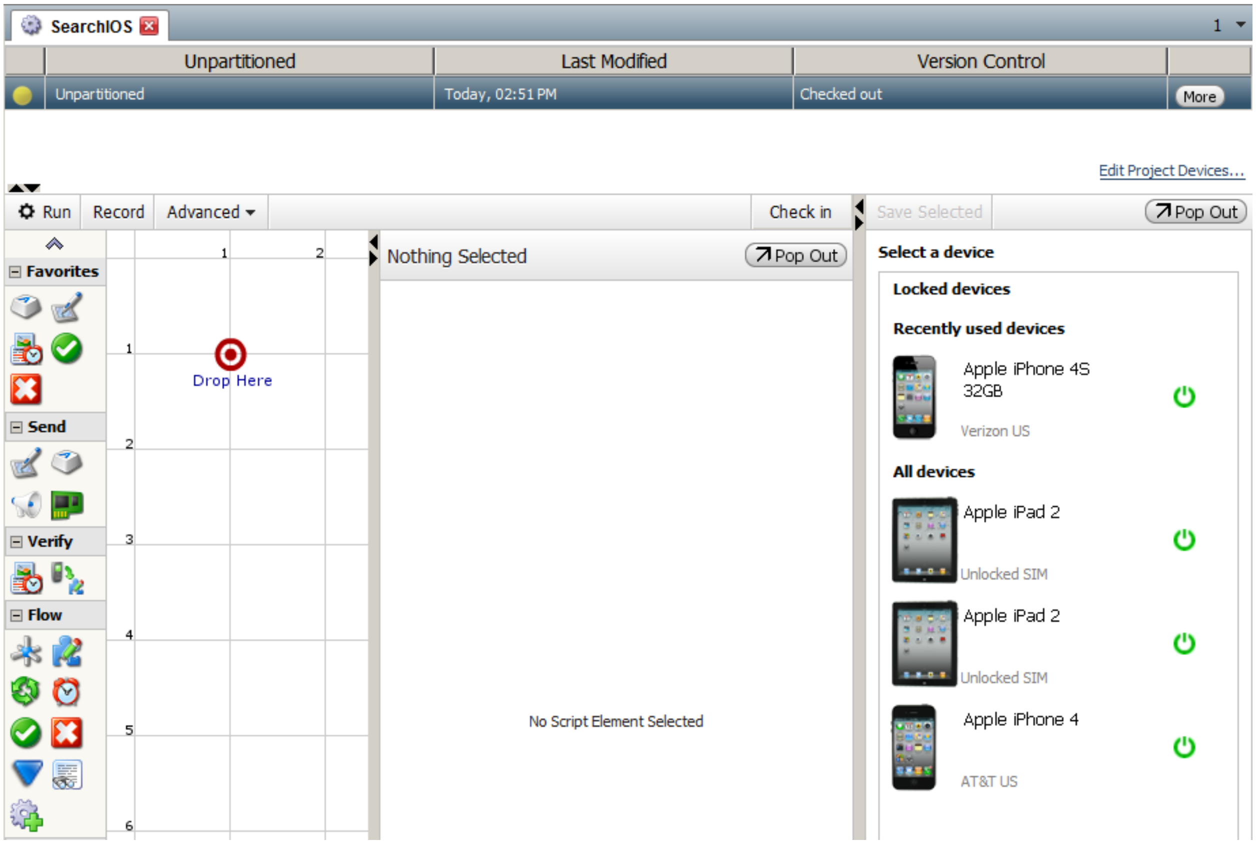
NOTE Once a script is set up as partitioned or unpartitioned, it cannot be reversed.

If your application behaves differently on say, iPhones and Android devices, create separate projects for scripts and devices for each platform. You could then create unpartitioned scripts in each project to work across all platform devices.

The image below shows a new, unpartitioned script that can work across all iOS devices shown in the right pane. You can select any device to interact with it in order to write or record your script.

NOTE As object-level commands interact directly with web or native elements, it is best not to mix them with UI-based commands in the same script.

Figure 4 Unpartitioned Script Canvas



Accounting for Pop-Ups and Other One-Time Application Behavior

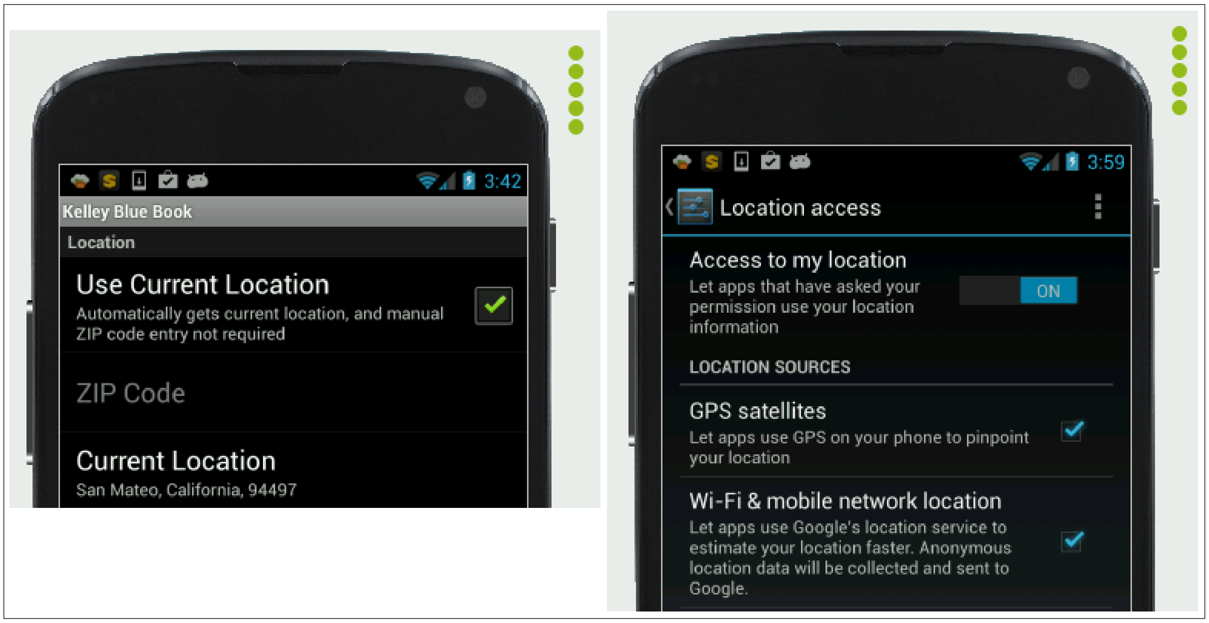
Applications and devices display one-time behaviors that are often not scripted for. You must account for these behaviors and the points at which they occur in your scripts and reference points.

For example, the first time you log in to a banking application, you might be required to sign an agreement. Or when you navigate to the store locator in an application, it asks to use your current location. These screens only appear again when you reset the device or application, reinstall the application, clear the browser cache, change application location settings, or change a device's use of GPS data.

The images below show the location sharing screen of the Kelly Blue Book application and the GPS settings on an Android device. Any change to these settings can cause a change in how your web or native application behaves. Before you run a script, check that these settings have not changed, or build scripts that check such device settings.

If these settings are likely to change, you should create branches for the one-time application screens that appear, e.g., asking you to set a location or sign a user agreement.

Figure 5 Settings That Change Application Behavior



3:42

Kelley Blue Book

Location

Use Current Location ☑
Automatically gets current location, and manual ZIP code entry not required

ZIP Code

Current Location
San Mateo, California, 94497

3:59

Location access

Access to my location ON
Let apps that have asked your permission use your location information

LOCATION SOURCES

GPS satellites ☑
Let apps use GPS on your phone to pinpoint your location

Wi-Fi & mobile network location ☑
Let apps use Google's location service to estimate your location faster. Anonymous location data will be collected and sent to Google.

Using Object vs. Image or Text Verification

You can use different types of reference points to verify script outcomes. Each kind of reference point has unique benefits and can serve different uses.

- Image- or text-based reference points allow you to choose text or an image from the device screen that must be matched at runtime. As these reference points are UI based, they must be defined separately for each device in your project.
- Web element- or native object-based reference points use underlying structures to verify a script outcome on an application page. These reference points can be defined for all devices in a project (web) or for all devices on a platform (native object).

Use UI-based reference points to check the actual appearance of something, for example:

- Company branding
- To check that the correct image is displayed
- The color and size of an image
- Actual text (with a strict margin of error) that appears on the screen

NOTE As UI-based reference points are device specific, be sure to use them only in partitioned scripts (action implementations), not combined with other Web and Object commands.

Use object-based reference points when the appearance or content of an element doesn’t matter so much, for example:

- To verify that you are on an application page by the presence/absence of an element
- To verify a certain outcome by the presence/absence of an element on a page

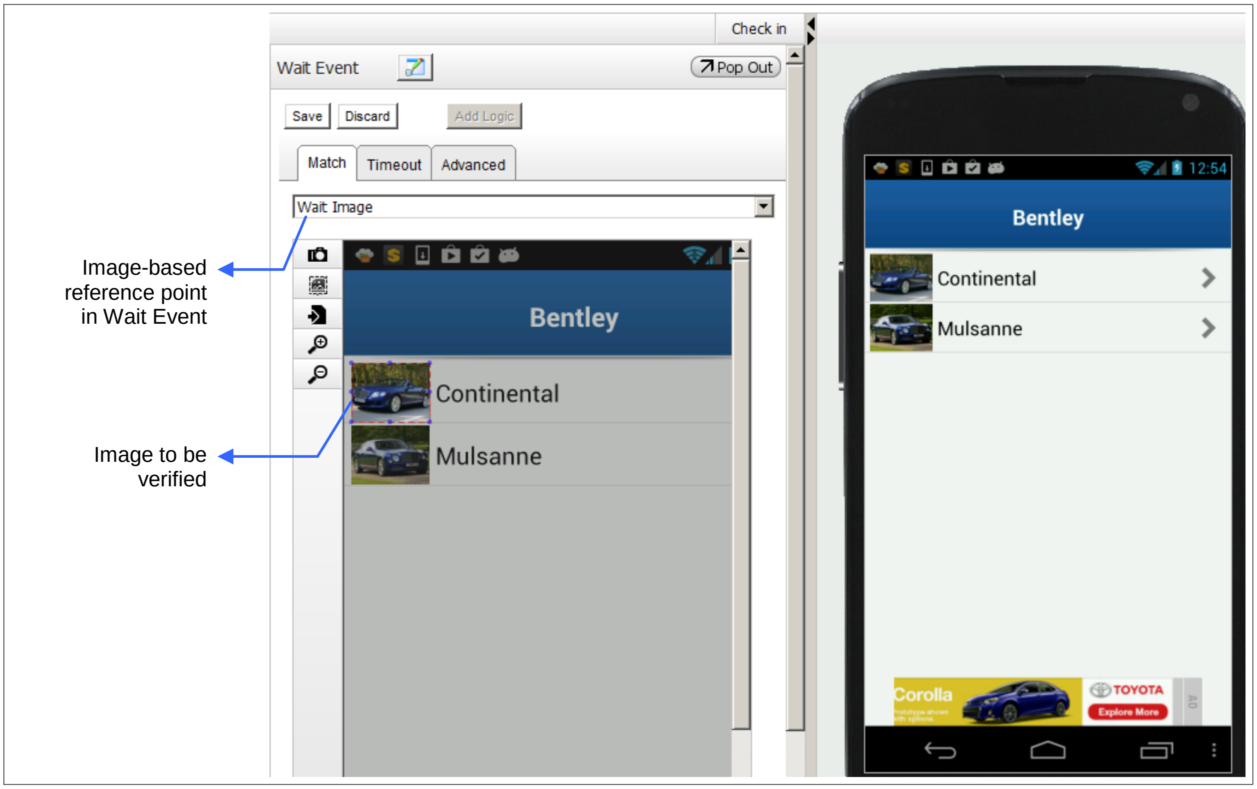
The table below compares visual and object-level reference points so you can choose which to use based on your needs:

Table 2 Visual vs. Object-Level Reference Points

	Visual Reference Points	**Object-Level Reference Points**
Based on	Pixel-to-pixel matching of a screen area or text extracted using character recognition	Web element or native object Select any web element as a reference point. Select a native object that has associated text as a reference point.
Commands	Wait Event and states	Web Wait (web elements) Wait Event and states (native objects)
UI Changes	Impacted by UI changes.	Reliable across UI changes.
Use for	Verifying exact display of an image or text	Verifying existence of an object or element on a page but not its appearance
Partitioned/ Unpartitioned	Use in partitioned scripts. Or define in a state for all project devices and call from an unpartitioned script.	Can use in both partitioned and unpartitioned scripts.

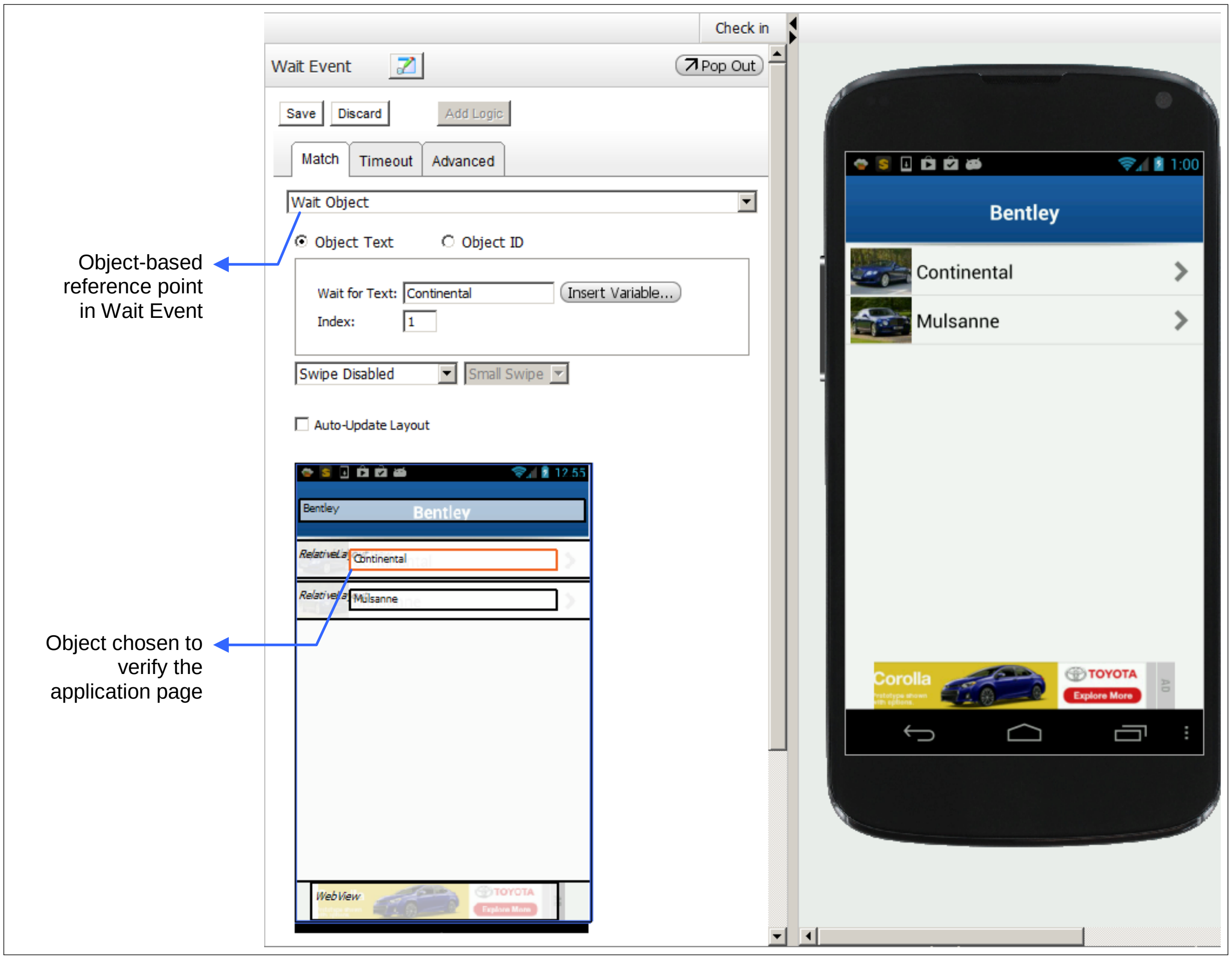
The images below show different reference points for the same page of a native application. In the image-based reference point below, the scripter wants to verify that the image of a Bentley Continental is matched exactly at runtime.

Figure 6 Verifying an Image in a Native Application



In object-level reference point below, an object with associated text ("Continental") is used to verify the same application page.

Figure 7 Verifying an Object in a Native Application



Object-based reference point in Wait Event

Object chosen to verify the application page

Wait Event

Wait Object

● Object Text ○ Object ID

Wait for Text: Continental

Index: 1

Swipe Disabled

☐ Auto-Update Layout

4 Tips for Testing Web Apps

Benefits of Using Web Commands

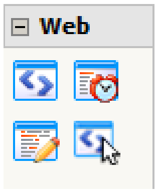
Commands in the **Web** category offer the most direct and robust way to interact with web content on Android, iOS, and BlackBerry devices.

With Web commands, you can create unpartitioned scripts, work with elements that are not visible on the device screen, fill out and submit entire forms quickly, and open and close the native browser from anywhere on the device.

- Web Element—Selects an element from the web page and performs an appropriate action.
- Web Wait—Verifies a web element.
- Web Form—Fills values in fields and submits a web form.

- Web Touch—Finds and touches/clicks a web element.
- Additionally, the Browser Open and Close All Browser Sessions utilities enable you to open and close native browser sessions.

Figure 8 Web Commands



Opening and Closing Browsers Quickly

Use the Browser Open and Close All Browser Sessions shortcut commands in the **Utilities** category to speed up mobile web testing.

Browser Open—Launches the native device browser from anywhere on the device and directly navigates to a desired URL.

Close All Browser Sessions—Closes all open native browser sessions from anywhere on the device.

Both commands also give you the option of clearing browser cache when you open or close the native browser. Browser Open and Close All Browser Sessions work on Android and iOS devices and require the DeviceAnywhere Agent.

To close Android browser sessions, use:

- The Close All Browser Sessions command (check **Clear browser cache**) on Android 4.0.x devices
- The Open Browser command with **Clear browser cache** checked (this closes any open native browser sessions and opens a fresh session at the URL specified)

You can also open and close native and non-native device browsers in these other ways:

- Use the Launch App and Close App commands on Android devices (used for native applications).
- Use the Find and Touch command.

NOTES These other methods to open/close browsers do not automatically clear the cache or register the device MCD with the DeviceAnywhere DOM Server for web testing.

Find and Touch is a UI-based command and should not be combined with object-level commands in an unpartitioned script.

Pointers for Selecting the Correct Web Element

In order to interact with a web element, you must first search for it and select it in a Web command.

Studio Web commands make it easy to select the right type of element.

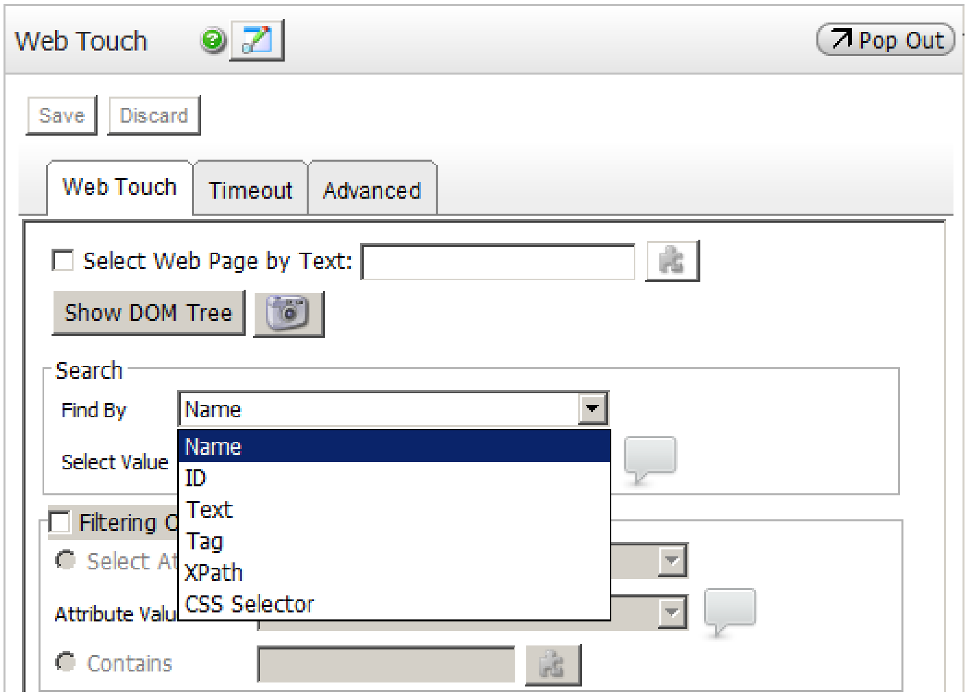
- For example, only clickable elements are presented in Web Touch and only `<form>` elements in Web Form. Web Wait and Web Element allow you to choose and work with any element.

- Additionally, you can select any element shown in a Web command, even those not visible on the device screen. You do not have to implement a swipe to work with an element that only appears when you scroll the page.

Web commands allow you to search for an element by any of these criteria:

- **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

Figure 9 Search Criteria for Web Elements



Whenever possible, search for an element by **Name**, **ID**, **Text**, and **Tag**. 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.

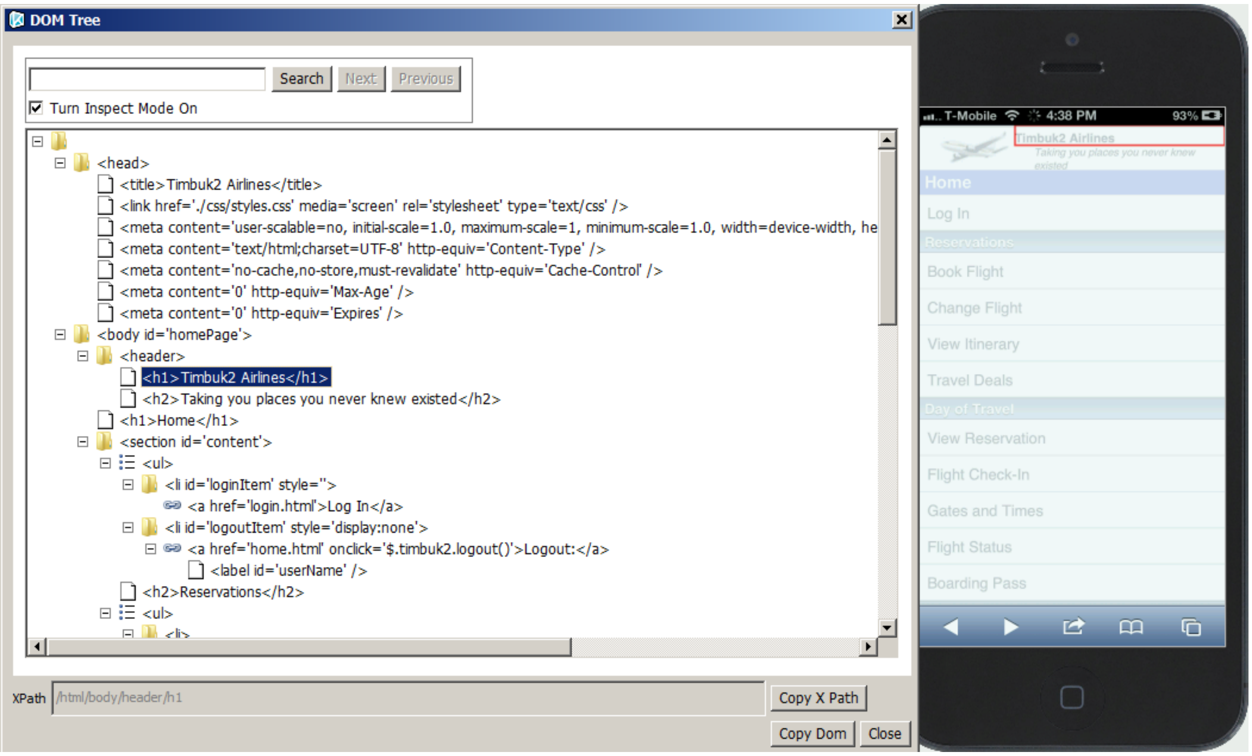
If you cannot find an element using a particular search criterion, try another.

The web page DOM viewer is another powerful tool that helps you locate an element and its path expression.

When you launch the DOM viewer in a command (**Show DOM Tree**), you can enable inspect mode (**Turn Inspect Mode On**.) When you click on the device, the corresponding element is highlighted on the device screen as well as the DOM tree. Likewise, when you select an element in the tree, it is also highlighted on the screen, helping you locate the element you want.

The **XPath** location of any selected element is also displayed in the DOM viewer.

Figure 10 Inspect Mode in the DOM Viewer



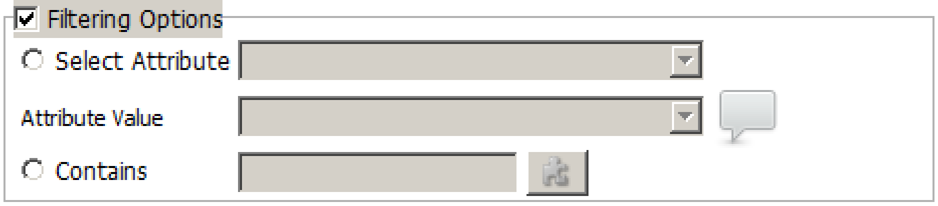
```
<head>
    <title>Timbuk2 Airlines</title>
    <link href='./css/styles.css' media='screen' rel='stylesheet' type='text/css' />
    <meta content='user-scalable=no, initial-scale=1.0, maximum-scale=1, minimum-scale=1.0, width=device-width, he
    <meta content='text/html;charset=UTF-8' http-equiv='Content-Type' />
    <meta content='no-cache,no-store,must-revalidate' http-equiv='Cache-Control' />
    <meta content='0' http-equiv='Max-Age' />
    <meta content='0' http-equiv='Expires' />
<body id='homePage'>
    <header>
        <h1>Timbuk2 Airlines</h1>
        <h2>Taking you places you never knew existed</h2>
    <h1>Home</h1>
    <section id='content'>
        <ul>
            <li id='loginItem' style=''>
                <a href='login.html'>Log In</a>
            <li id='logoutItem' style='display:none'>
                <a href='home.html' onclick='$.timbuk2.logout()'>Logout:</a>
                    <label id='userName' />
        <h2>Reservations</h2>
        <ul>
            <li>
```

XPath /html/body/header/h1

Tools for Narrowing Down Element Search Results

If you have many search results and need to narrow them down, you can:

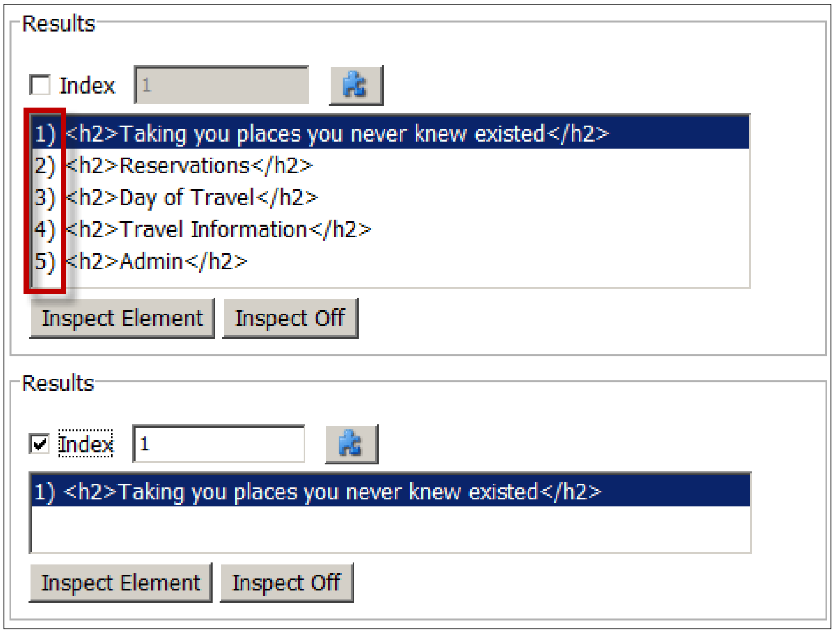
- Use the additional search filters in **Filtering Options**.



- Select a search result by **Index** number.

  Note that the index number is not intrinsic to 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 log in to a web page. An element's position in search results can also change from one device to another or when you change device orientation. Source code can also be altered to change the relative position of an element in the markup.

  The images below show search results before and after filtering by index number.



Results

☐ Index: 1

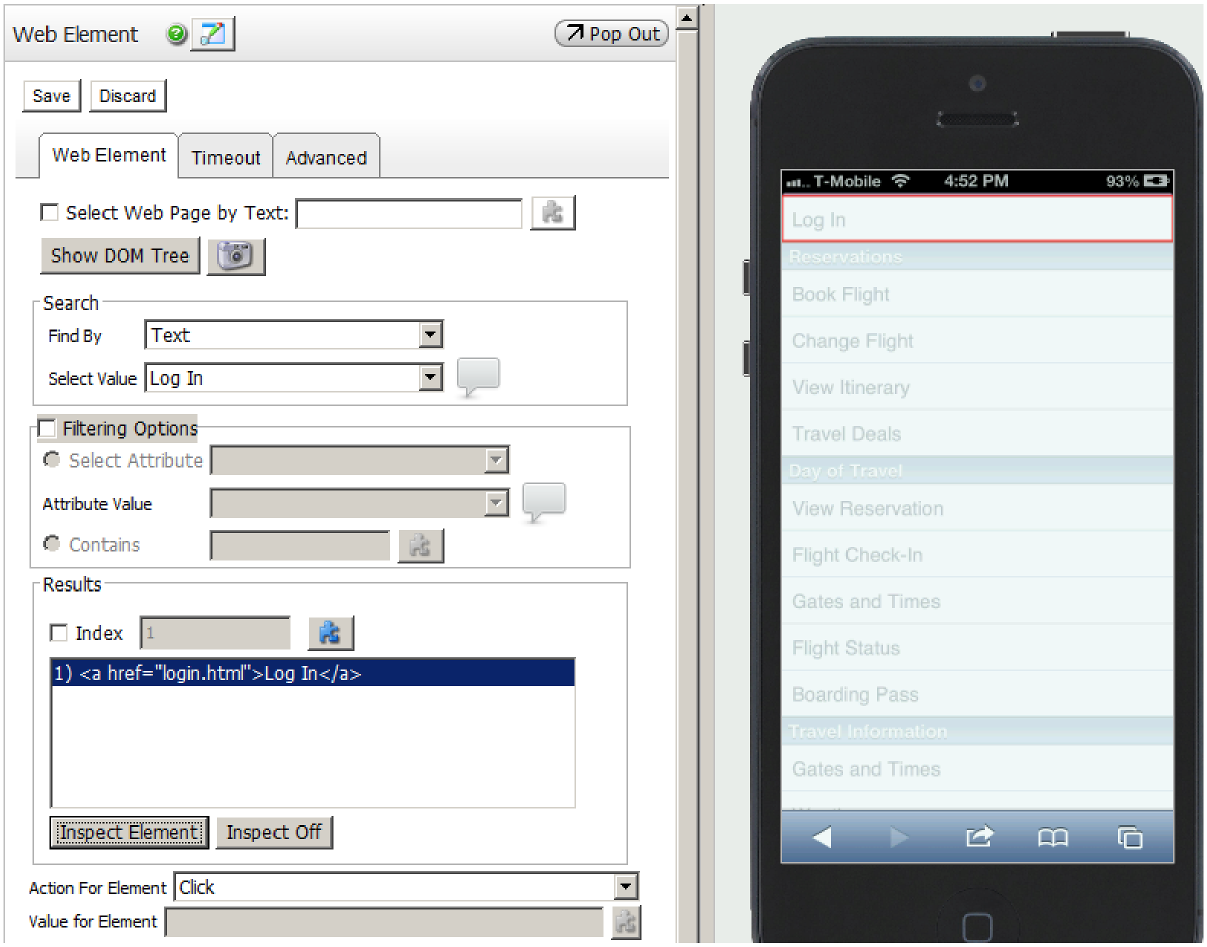
1) <h2>Taking you places you never knew existed</h2>
2) <h2>Reservations</h2>
3) <h2>Day of Travel</h2>
4) <h2>Travel Information</h2>
5) <h2>Admin</h2>

Results

☑ Index: 1

1) <h2>Taking you places you never knew existed</h2>

- Select a search result and then click **Inspect Element**. The element is highlighted on the device screen **Inspect Element** is a powerful tool that simultaneously highlights an element in a command and on the device screen, helping you locate the element you want.



☐ Select Web Page by Text:

Search

Find By: Text

Select Value: Log In

☐ Filtering Options

○ Select Attribute

Attribute Value

○ Contains

Results

☐ Index: 1

1) <a href="login.html">Log In</a>

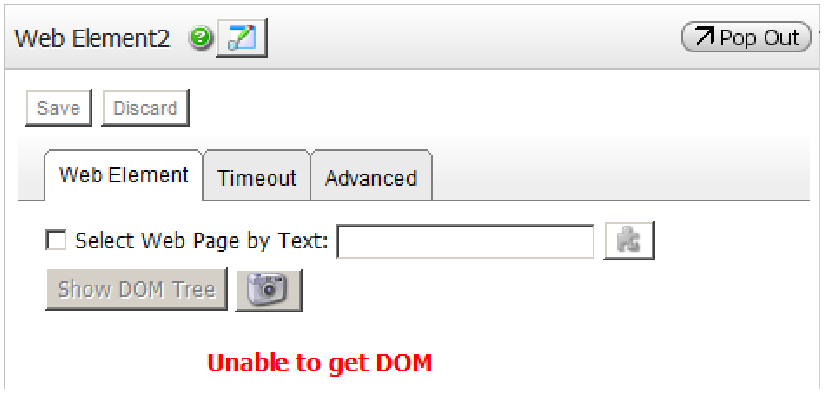
Action For Element: Click

Value for Element:

Troubleshooting Empty Web Commands

If you open a Web command and it does not fill out with information, you will see the following message:

Figure 11 Unable to Process Web Command



Unable to get DOM

There are several things you could check to troubleshoot this issue:

- If you have a private, on-premise environment, check that your DOM Server is configured correctly.
- Check that your device meets all the requirements for web testing.
- Check that your device is acquired.
- Check that your device is on a test web page with the required JavaScript tag.
- Check that your device MCD is registered with the DOM Server.

Options for Setting the Device MCD for Testing Web Apps

To use Web commands on a device, you must register the device MCD with the DOM Server at the start of a device session. The best practice is to register the device MCD automatically using the Browser Open command.

You might still need to register the MCD manually, e.g., if you are using a non-native device browser. Navigate to the registration page (`http://<DOMServerAddress>/da/views/mcd.html`) and enter the MCD. You must do so from each device browser you plan to use.

In on-premise environments, you have the option to be prompted if the device MCD is not automatically detected (i.e., when using the Browser Open command). This option must be enabled when you install the DOM Server.

The prompt to enter an MCD will be seen whenever the device cannot be registered automatically, even by manual testers in your environment. To avoid this, you can install separate systems for manual and automated testing. Also ensure that you script for this prompt in your automated scripts. And if you are not able to test your hybrid application after enabling this prompt, contact Keynote Support.

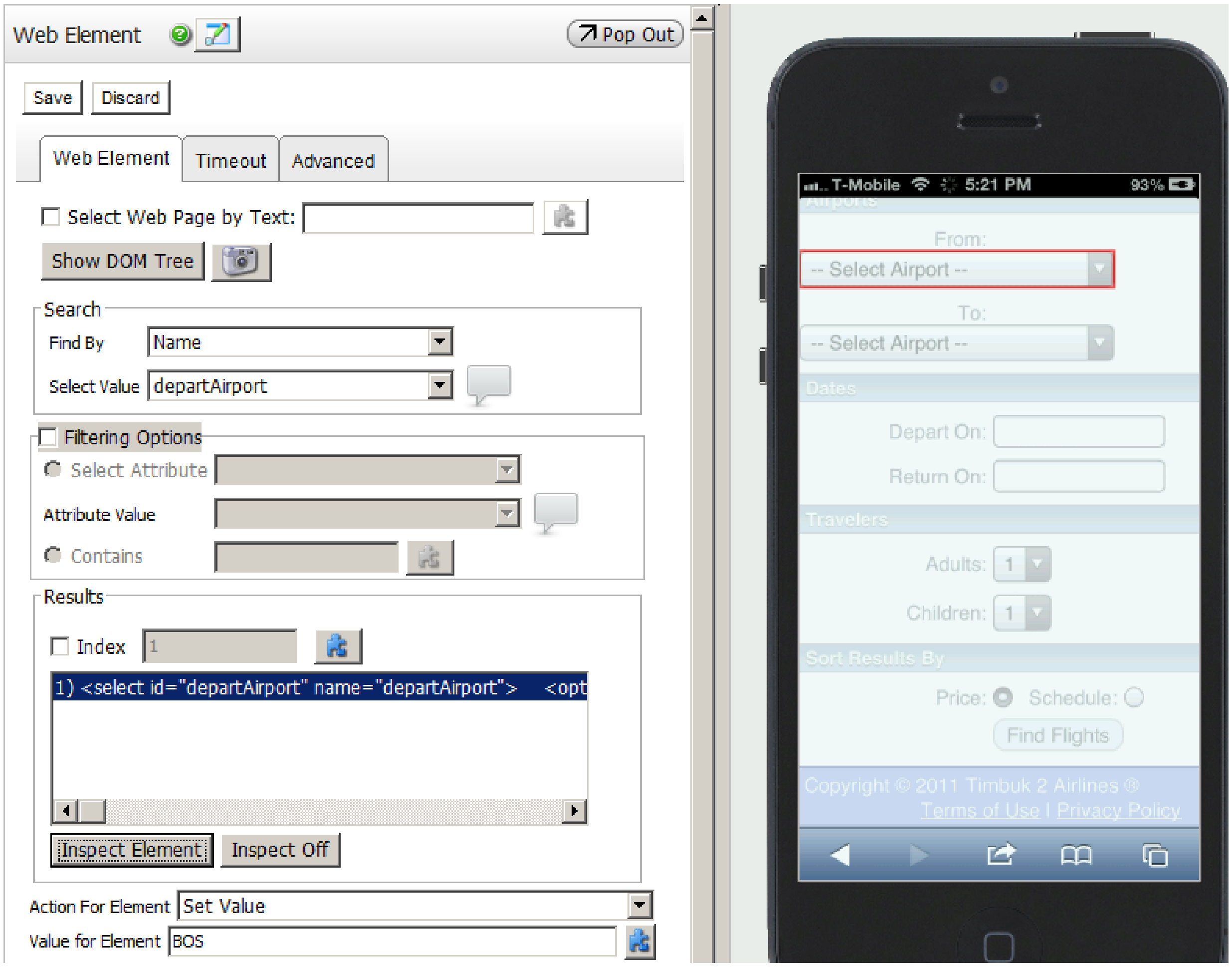
Being prompted to enter a device MCD is not recommended in production systems.

One-Step Values for Web Elements

When you work with Web commands, you can select an element and choose a value for it in one go. In other words, you don't need to have one command for setting focus in a field and another for entering a value. You can set a value for an element in the Web Element or Web Form commands.

The image below shows how to select a field and set a value for it in a single Web Element command.

Figure 12 Setting Element Value in Web Element



Find By: Name

Select Value: departAirport

1) <select id="departAirport" name="departAirport"> <opt

Action For Element: Set Value

Value for Element: BOS

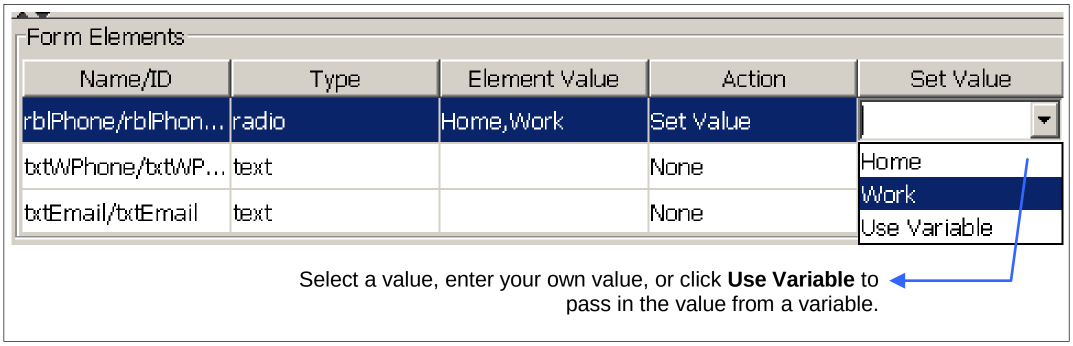
One-Step Form Entry and Submission

The Web Form command allows you to select a form, set values for all fields (including fields not visible on the device screen), and submit the form.

When you select a form (`<form>` element) in the Web Form command, all fields in the form are displayed, and you can set a value for each of them.

The image below shows how to enter data into several form fields in the Web Form command.

Figure 13 Entering Data in Form Fields—Web Form



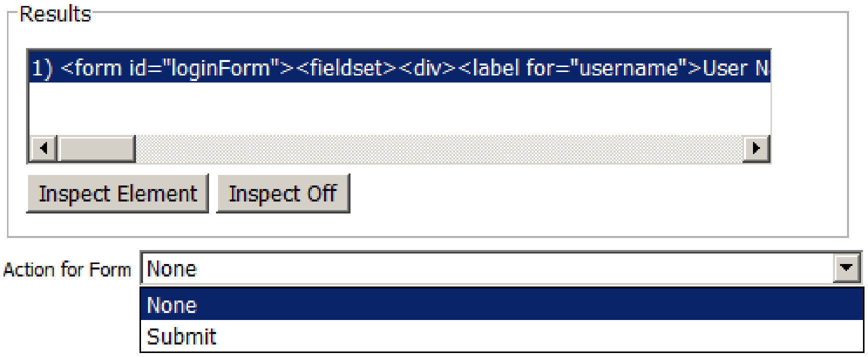
Form Elements

Name/ID	Type	Element Value	Action	Set Value
rblPhone/rblPhon...	radio	Home,Work	Set Value	
txtWPhone/txtWP...	text		None	
txtEmail/txtEmail	text		None	

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

Submit buttons for the form as a whole are *not* listed with other form elements. To click a submit button after filling out form fields, choose the **Submit** action for the form as a whole.

Figure 14 Actions for a Form in Web Form



Results

1) <form id="loginForm"><fieldset><div><label for="username">User N

Action for Form: None

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.

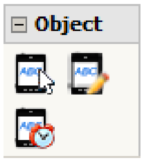
5 Best Practices for Testing Native Apps

Benefits of Using Object Commands

Commands in the **Object** category enable you to create unpartitioned scripts for native applications on Android (4.0.4+) and iOS (7.1+) devices. Devices must be configured for this feature to work—please contact Keynote Support.

As Object commands work directly with native application objects, your scripts operate across device models and are not sensitive to application UI changes. You also have powerful tools to launch or close native application sessions from anywhere on the device.

Figure 15 Object Commands



- Launch App—Open a native application.
- Close App—Close a native application.
- Object Touch—Find and click an object (identified by associated text).
- Object Edit—Enter data into an object.
- Object Extract Text—Extract text from an object to a variable.
- To use a native object for script verification, select **Wait Object** in a state or the Wait Event command.

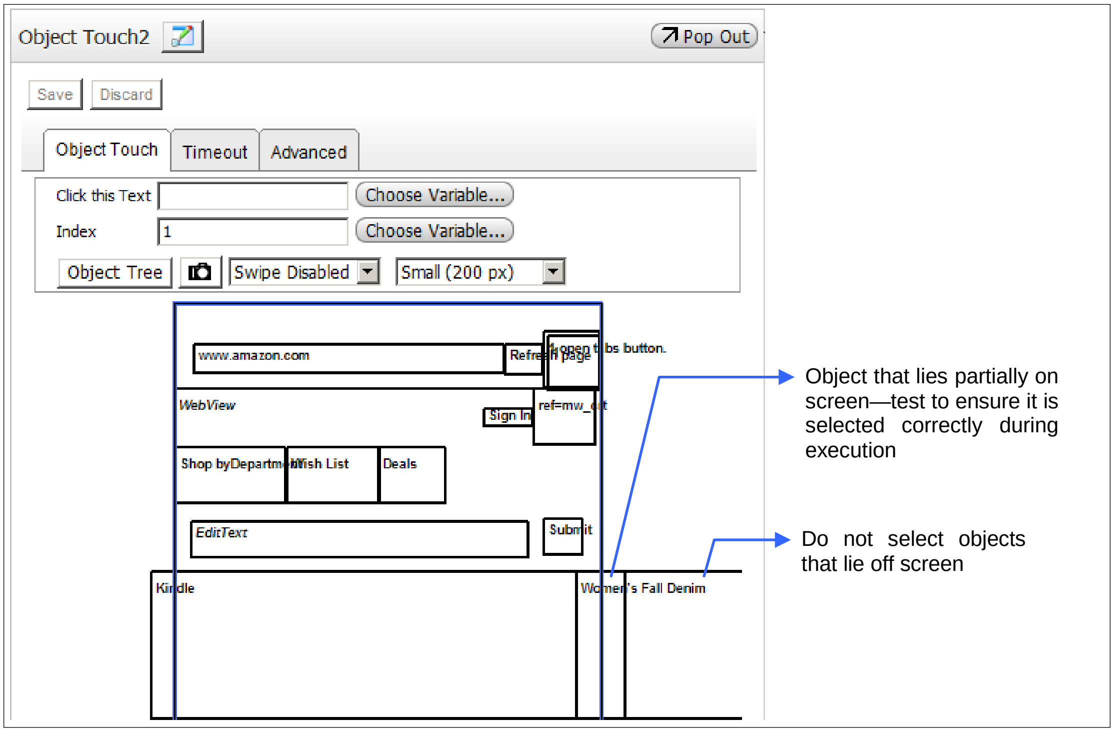
Pointers for Selecting the Correct Native Object

In order to interact with a native object, you must first select it in an Object command. Unlike [Web commands](), Object commands display the visual layout of native objects on an application page, with a slight difference—only objects visible on the device screen are displayed. You can only select and work with an object shown in the layout.

Objects That Are Partially Inside the Layout

Some objects in the layout can appear to be outside the edges of the device screen. The object you select must at least be partially contained within the layout or it will not be correctly selected during run time.

Figure 16 Selecting a Native Object



Click this Text
Index 1
Swipe Disabled
Small (200 px)

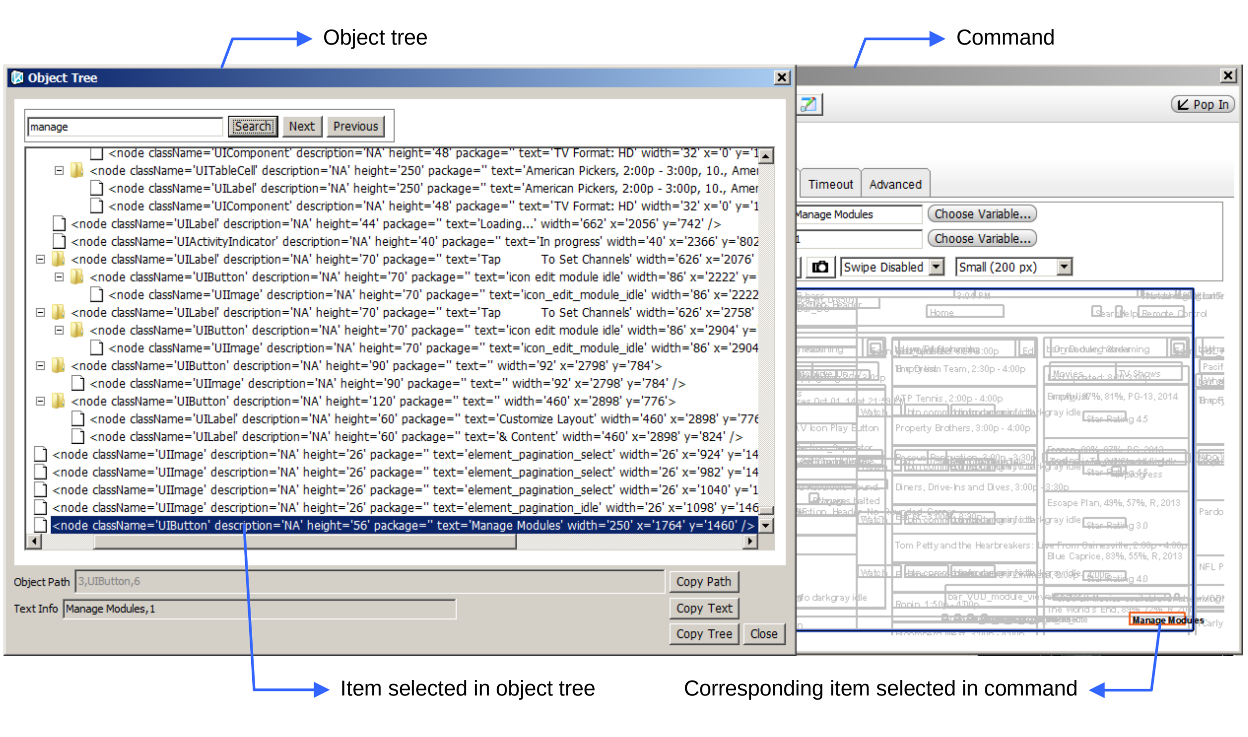
Object that lies partially on screen—test to ensure it is selected correctly during execution

Do not select objects that lie off screen

Using the Object Tree

When scripting for applications with a dense object layout in many layers, launch the **Object Tree** to search for and identify an object to work with. As you search for object text and select a node in the tree, the corresponding object is highlighted in the command.

Figure 17 Object Tree Launched from Object Command



Search: manage

```
    <node className='UIComponent' description='NA' height='48' package='' text='TV Format: HD' width='32' x='0' y='1
  <node className='UITableCell' description='NA' height='250' package='' text='American Pickers, 2:00p - 3:00p, 10., Amer
    <node className='UILabel' description='NA' height='250' package='' text='American Pickers, 2:00p - 3:00p, 10., Amer
    <node className='UIComponent' description='NA' height='48' package='' text='TV Format: HD' width='32' x='0' y='1
<node className='UILabel' description='NA' height='44' package='' text='Loading...' width='662' x='2056' y='742' />
<node className='UIActivityIndicator' description='NA' height='40' package='' text='In progress' width='40' x='2366' y='802
<node className='UILabel' description='NA' height='70' package='' text='Tap         To Set Channels' width='626' x='2076'
  <node className='UIButton' description='NA' height='70' package='' text='icon edit module idle' width='86' x='2222' y=
    <node className='UIImage' description='NA' height='70' package='' text='icon_edit_module_idle' width='86' x='2222
<node className='UILabel' description='NA' height='70' package='' text='Tap         To Set Channels' width='626' x='2758'
  <node className='UIButton' description='NA' height='70' package='' text='icon edit module idle' width='86' x='2904' y=
    <node className='UIImage' description='NA' height='70' package='' text='icon_edit_module_idle' width='86' x='2904
<node className='UIButton' description='NA' height='90' package='' text='' width='92' x='2798' y='784'>
  <node className='UIImage' description='NA' height='90' package='' text='' width='92' x='2798' y='784' />
<node className='UIButton' description='NA' height='120' package='' text='' width='460' x='2898' y='776'>
  <node className='UILabel' description='NA' height='60' package='' text='Customize Layout' width='460' x='2898' y='776
  <node className='UILabel' description='NA' height='60' package='' text='& Content' width='460' x='2898' y='824' />
<node className='UIImage' description='NA' height='26' package='' text='element_pagination_select' width='26' x='924' y='14
<node className='UIImage' description='NA' height='26' package='' text='element_pagination_select' width='26' x='982' y='14
<node className='UIImage' description='NA' height='26' package='' text='element_pagination_select' width='26' x='1040' y='1
<node className='UIImage' description='NA' height='26' package='' text='element_pagination_idle' width='26' x='1098' y='146
<node className='UIButton' description='NA' height='56' package='' text='Manage Modules' width='250' x='1764' y='1460' />
```

Object Path: 3,UIButton,6

Text Info: Manage Modules,1

Command: Manage Modules; 1; Swipe Disabled; Small (200 px)

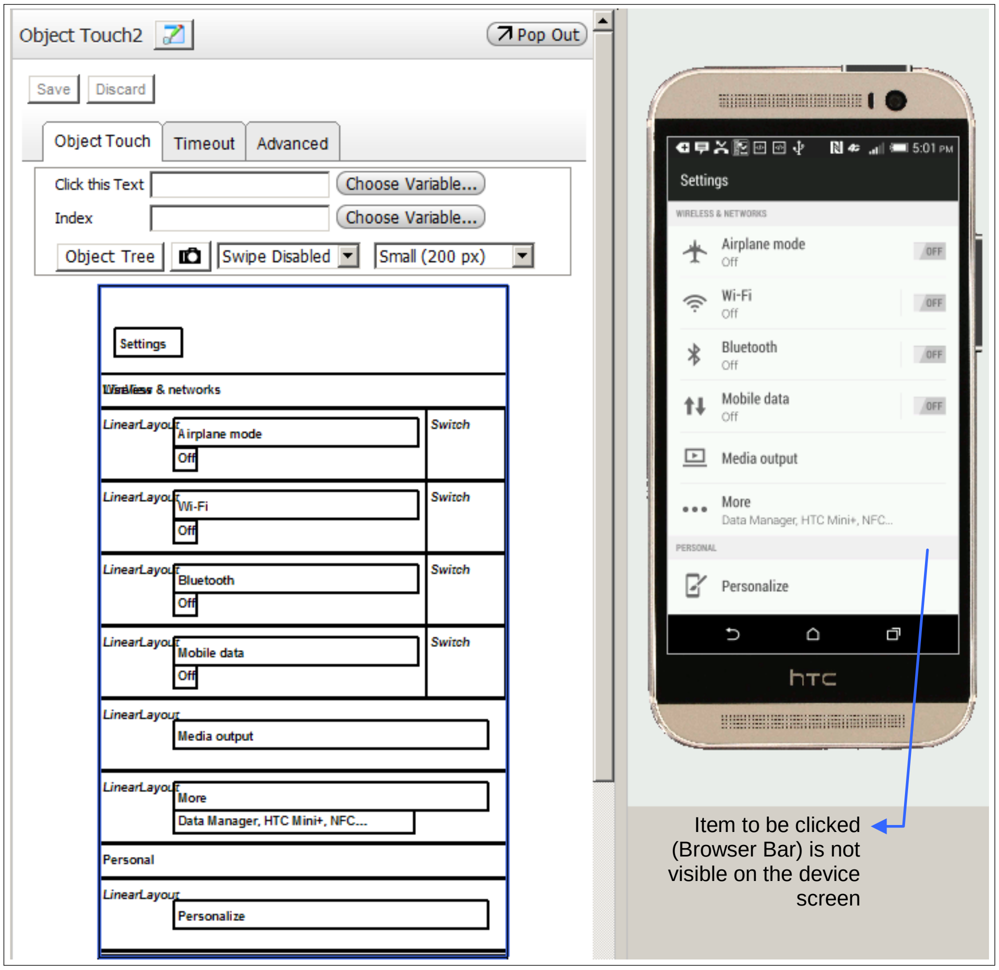
Selecting an Object by Text vs.

- In the Object Touch command (see Figure 18 and Figure 19 below), you select an object by specifying associated text. You can also pass in this value from a variable.
- 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 all displayed. You can then use the **Index** number to indicate which of the displayed objects you want to select.

Scrolling to Make an Object Visible

You can work with an object that becomes visible after you scroll up/down or left/right. On the page shown below, you need to swipe the screen up before you can see the Android setting “Browser Bar.”

Figure 18 Before Scrolling



Item to be clicked (Browser Bar) is not visible on the device screen

In a command, you must implement a swipe so that the object you want becomes visible. Choose the same swipe direction as your finger would move if you were physically controlling the device:

1 Swipe the device till the object is visible.

2 Update object layout 📷 in your command.

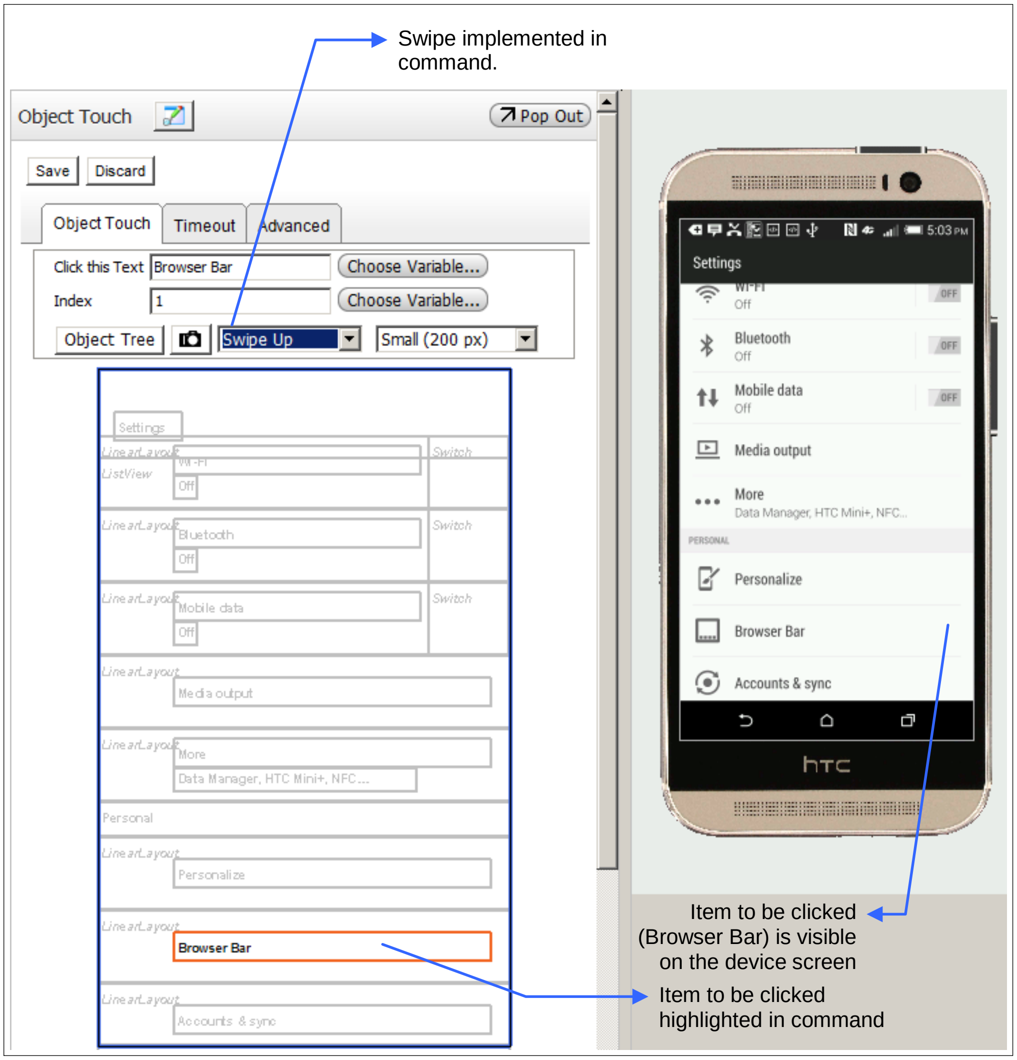
3 Select the object in the command.

4 Implement a swipe in the command so that the device is correctly swiped at runtime to bring the object into view.

5 You can change the default **Timeout** value.

At run time, the screen is scrolled until the object is found or until the command times out.

Figure 19 After Scrolling



Swipe implemented in command.

Object Touch

Click this Text: Browser Bar

Index: 1

Swipe Up

Small (200 px)

Item to be clicked (Browser Bar) is visible on the device screen

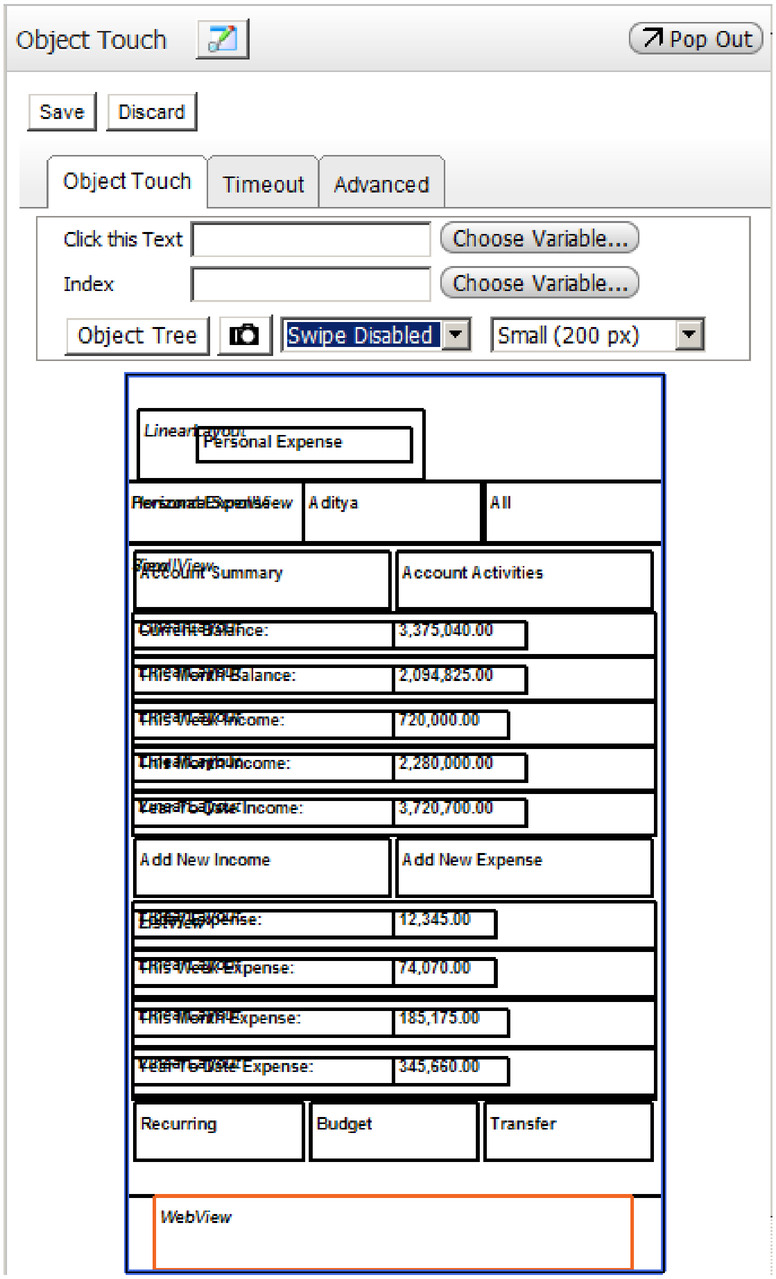
Item to be clicked highlighted in command

Web Frames

Object commands cannot be used on nested web pages within a native application. These web frames are usually marked as **Web View** in an Object command (see Figure 20 below).

However, you can use Object commands to work with web elements when you use the Chrome browser on Android devices. You can test any web page without adding a JavaScript tag to the page markup when using Chrome on Android 4.2 devices.

Figure 20 Object Commands Cannot Work on Nested Web Content



Swipe Disabled

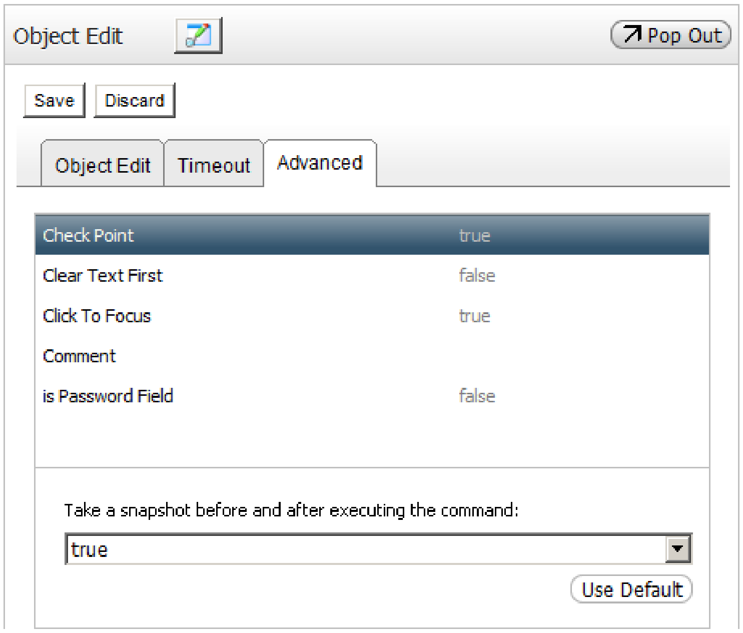
Small (200 px)

One-Step Values for Native Objects

You can use the Object Edit command to enter data into a native object. When you select an object field:

- Studio sets focus in the field automatically. (You can change this using the **Click To Focus** setting in the **Advanced** tab.)
- You can also clear existing text automatically using the **Clear Text First** setting in the **Advanced** tab.
- If you need to clear the contents of an encrypted field (e.g., for a password or SSN) before data entry, set **is Password Field** to **true**.

Figure 21 Options for Object Data Entry



Check Point	true
Clear Text First	false
Click To Focus	true
Comment	
is Password Field	false

Take a snapshot before and after executing the command: true

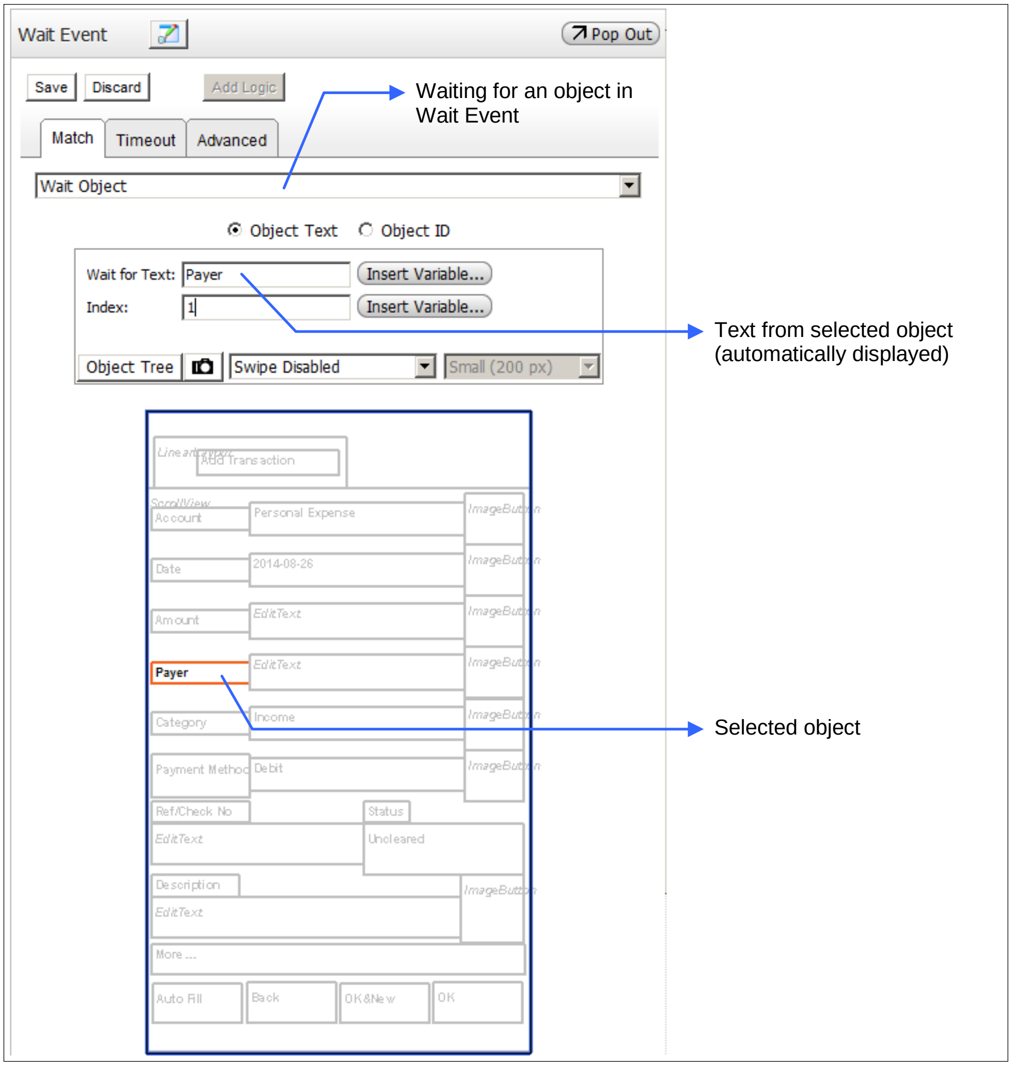
Implementing Native Object Verification

Use the **Wait Object** option in a state or the Wait Event command to implement object-based script verification. You can select an object to wait for by its associated text or by its ID.

- When selecting an object, choose an object that uniquely identifies the application page.
- Whenever possible, select **Object Text** over **Object ID**. Object ID includes an index number of the object's relative position on the application page. This value is not an intrinsic characteristic of an object and can change over time.

The image below shows selecting an object by its associated text. If many objects have the same text, you can choose a specific object by selecting its **Index** number. Whenever possible, choose object text that is unique to an object; the index number is not an intrinsic property of an object and can change over time.

Figure 22 Waiting for an Object



Wait Object

● Object Text ○ Object ID

Wait for Text: Payer

Index: 1

Swipe Disabled

Small (200 px)

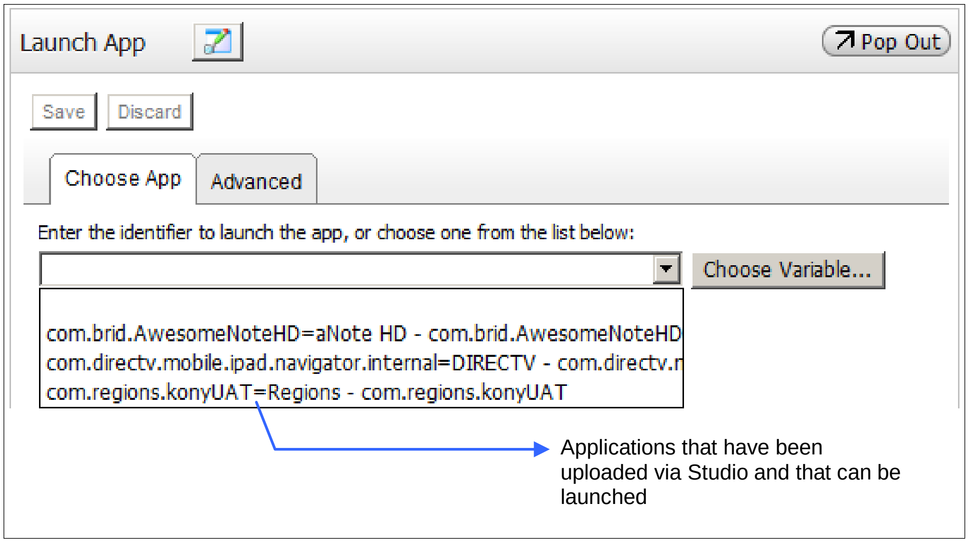
Waiting for an object in Wait Event

Text from selected object (automatically displayed)

Selected object

Best Practice iOS Object Scripting

- iOS Object scripting in the 6.4 release of DAE Automation is not compatible with the prior beta release of iOS Object scripting—you must recapture device screens in Object commands in your existing scripts.
- In order to test an application using object-based scripts on iOS devices, you must launch it using the Launch App command. Only applications uploaded via Studio are listed for selection.



Enter the identifier to launch the app, or choose one from the list below:

com.brid.AwesomeNoteHD=aNote HD - com.brid.AwesomeNoteHD
com.directv.mobile.ipad.navigator.internal=DIRECTV - com.directv.r
com.regions.konyUAT=Regions - com.regions.konyUAT

Applications that have been uploaded via Studio and that can be launched

- In order to prevent memory issues that cause schedules to be interrupted keep iOS object scripts short so that each run is less than 30 minutes. As a best practice, begin each test case with the Launch App command. If a test case exceeds 30 minutes, launch your application using Launch App within the test case every 30 minutes. You can insert Launch App at various points in the test case script to implement this.

  This implies that your application is launched several times in a test cycle, depending on the number of test cases it contains and the length of each test case.

  If you set up your test cases following this guidance, your scheduled test cycles can be the length you want and will run uninterrupted. In this sample scenario of testing a banking application below, a test cycle run on a device consists of:

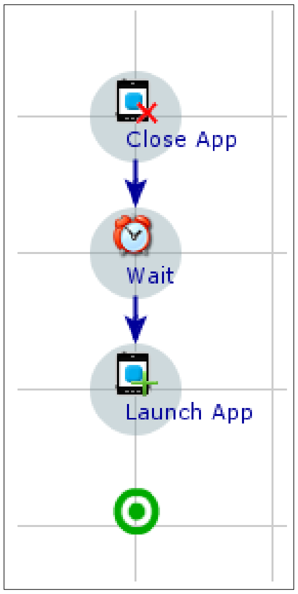
  - Test Case 1 (10 minutes)
    - Launch App, followed by
    - Testing checking account balance
  - Test Case 2 (20 minutes)
    - Launch App, followed by
    - Transferring money between accounts
  - Test Case 3 (1 hour)
    - Launch App 1, followed by
    - Checking tax statements
    - Launch App 2 (30 minutes into the test case), followed by
    - Making a payment

Best Practice Opening Scripts for Every Object Test Case

To start off an object test case, especially a scheduled test case:

- Begin your test case by checking if the device is asleep, and if required, restart the device—many Android devices have sleep timers that cannot be disabled.
- To start using your application in a known state:
  - Reset the application using Close App.
  - Insert a brief Wait time.
  - Open the application using Launch App.

  This clears data from the previous application session and opens it on the home page.


Close App
Wait
Launch App