



## CloudTest<sup>®</sup> WebUI Testing Tutorial



SOASTA CloudTest® WebUI Testing Tutorial

©2014, SOASTA, Inc. All rights reserved.

The names of actual companies and products mentioned herein may be the trademarks of their respective companies.

This document is for informational purposes only. SOASTA makes no warranties, express or implied, as to the information contained within this document

---

---

## Table of Contents

<b>Why Web UI/Ajax Testing? .....</b>	<b>1</b>
<b>Prerequisites.....</b>	<b>1</b>
Browser Recorder Firefox Add-On.....	1
SOASTA Conductor Prerequisites .....	2
KnowledgeBase and Tutorials.....	2
Using the SOASTA Store .....	3
<b>Testing Your WebUI / Ajax Site.....</b>	<b>4</b>
<b>Introduction to WebUI / Ajax Tests .....</b>	<b>4</b>
<b>Recording Browser User Scenarios.....</b>	<b>6</b>
Recording a WebUI / Ajax Test Clip .....	6
Inspecting the Newly Created Clip .....	14
Browser Action Elements and Properties .....	15
Adding a Text Output .....	18
Adding a Screenshot Output.....	19
Adding an Interval Delay between Each Browser Action.....	20
<b>Playing a Simple WebUI / Ajax Test .....</b>	<b>22</b>
Result Details .....	23
Inspecting Clip Element Outputs .....	24
<b>Advanced Clip Editing .....</b>	<b>26</b>
Adding a Validation on Body Text .....	26
Adding a Validation on an Element .....	27
<b>Analyzing Results .....</b>	<b>30</b>
Other Widgets Useful to WebUI / Ajax Tests .....	34

## Why Web UI/Ajax Testing?

After many years of trials and standards development, Rich Internet Applications (RIAs) based on Asynchronous JavaScript and XML (Ajax) are appearing everywhere. These Ajax-based user interfaces (UIs) combine rich graphics, Browser Actions, and dynamic content using messaging events to deliver an application across the Web that combines the best of the Web and the desktop.

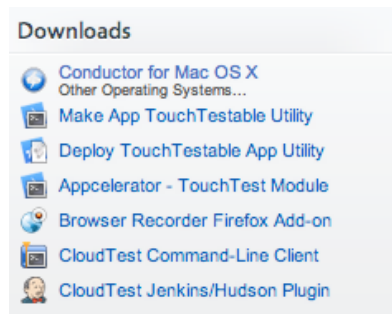
End users receive a richer end user experience while IT organizations receive the ability to support a single application image from the server. Leading software application vendors are moving rapidly to deliver new versions of their products to enable this approach. Customer-facing self-service applications seem to be the market force pushing these new applications forward. Companies save money because customers easily support themselves and are much happier with the service they do receive . . . as long as the applications work and perform.

## Prerequisites

Browser recording requires that both the Firefox Browser and the SOASTA Browser Recorder add-on (for Firefox) be installed on the machine where recording will take place.

## Browser Recorder Firefox Add-On

The SOASTA Browser Recorder is a Firefox Add-On that can be downloaded from SOASTA CloudTest's Central tab, Welcome page, Downloads section.



- See [Installing the Browser Recorder Extension](#) for additional steps.



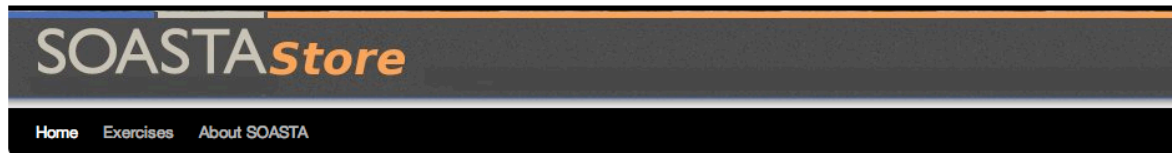
CloudTest provides two types of recording in support of test creation. This tutorial covers Browser Recording, which allows the user to capture user interface actions performed on a given target web site. CloudTest also provides HTTP(S) recording.

- Refer to the [Load Testing Tutorial](#) and to the CloudLink, Knowledge Base's HTTP(S) Recording section for additional HTTP(S) recording instructions.
- Refer to the [KnowledgeBase > Browser Recording section](#) for additional Browser Recording instructions.

## Using the SOASTA Store

This tutorial uses the sample site, [SOASTA Store](#), which is provided as a practice site for SOASTA tutorials.

The SOASTA Store provides some of the most common eCommerce browser actions and user interface features. Keep in mind, it is not necessary to enter real or even complete information since the site is only for practice.



### Scalable, Affordable Testing of Web Apps

#### Using SOASTAStore

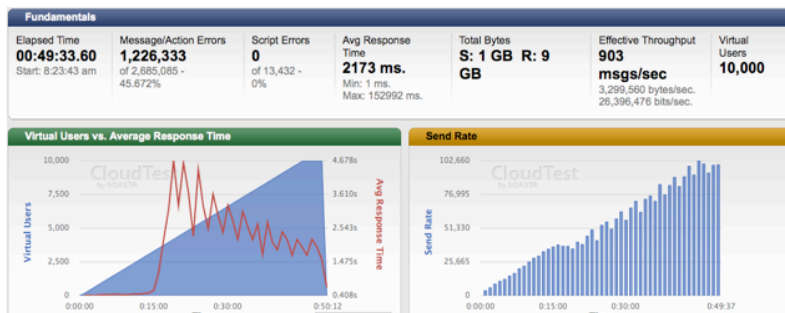
This site is intended to provide a platform for demonstrating SOASTA's CloudTest and for training. It is a very simple product store with a few books, DVDs and CDs for purchase. The site will change over time to enable our Performance Engineers and partners to demonstrate and train on new features in CloudTest.

#### Archives

- [February 2011](#)

#### Meta

- [Register](#)
- [Log in](#)



## Testing Your WebUI / Ajax Site

The SOASTA Browser Recorder Add-on for Firefox automates the creation of real world Web UI/Ajax test cases by recording Browser Actions from the target Web application.

This allows the basics of a Web UI/Ajax test to be created very quickly. The test designer does not need to convert to another format — the test creation process is done in place.

This *Tutorial* presents procedures for the following essential **WebUI / Ajax test** methods:

- Introduction to WebUI / Ajax Tests
- Recording Browser User Scenarios
- Creating a WebUI / Ajax Test
- Composing a WebUI / Ajax Test
- Running and Monitoring a WebUI / Ajax Test
- Analyzing Results

### Introduction to WebUI / Ajax Tests

Browser Recording is performed in the Clip Editor. While recording, each Browser Action is captured (for example, Click, Type, Drag and Drop, etc.) and automatically added to the test clip that is being created.

Typically, a single test clip defines a test case. You can also output page elements to properties for use as parameters in subsequent testing steps, or to the SOASTA CloudTest Results Viewer for help in debugging a test.

As a general guideline, your test should account for all the factors of the application(s) in the environment in which they occur, and include one, or, as many viable test cases as it will take to arrive at a good test composition. In the context of WebUI / Ajax testing, planning will also take into account the following factors:

- **The types of actions users take within the browser**

The test designer will consider the types of user interactions and user flow that make up the best test. Once a test flow is determined for a given UI it can be easily captured. The test designer can move quickly from Browser Recording to defining validations and other test details for those captured Browser Actions.

- **The timing of user actions**

Web UI/Ajax testing presents a special timing challenge because the HTML page is modified asynchronously. This lack of determinism is handled for the tester in SOASTA CloudTest using a comprehensive set of waitFor commands. WaitForElementText, waitForElementPresent, waitForJavascriptCondition and waitForElementVisible are some examples. SOASTA CloudTest's wait commands allow the tester to gain control of test timing targeting a complex Ajax Web application.

- **The validation of tests**

Verifying the behavior of a Web UI/Ajax application is an exercise that requires a great deal of detailed validations. After each Browser Action is recorded, the test designer can add as many validations as needed to each Browser Action by picking from a comprehensive list of built-in Verify commands. These Verify commands are specific to validating web pages (for example, verifyAttributeValue, verifyElementText, verifyJavascriptCondition, verifyElementPresent, verifyTextPresent, etc.). SOASTA's approach allows deep extensive validations to be built easily and quickly.

SOASTA test clips can combine a mix of Browser Actions and HTTP(s) messages. This capability supports the use of Web service calls to obtain data values and to validate the end result of a test. As an example, the test of a Web UI component called *Add Customer* can use this feature to validate that the database contains the correct values after an update has occurred.

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

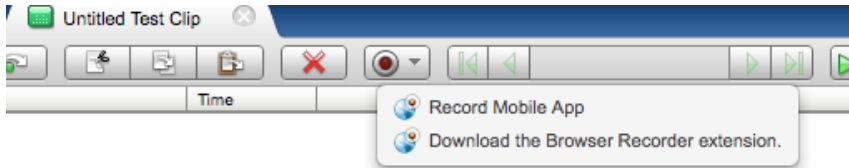
Complex test suites can be easily built based on your test cases. Using SOASTA Conductor, WebUI / Ajax tests can then be run in many instances of multiple browsers (Firefox, Internet Explorer, Safari) on any of the major OS platforms (Windows, Linux, and Mac OS X). A single test can use multiple Conductors and those Conductors can be installed on computers distributed across the Internet. Each Conductor can run multiple browser sessions simultaneously.



## Recording Browser User Scenarios

With the Browser Recorder successfully installed and the WebUI / Ajax target created in the prior steps, we will now use the Browser Recorder add-on to create a new test clip using that WebUI / Ajax target. The following steps **must be** performed using the Firefox browser.

When in Firefox, if the necessary Browser Recorder extension has yet to be installed, an additional drop-down command appears.

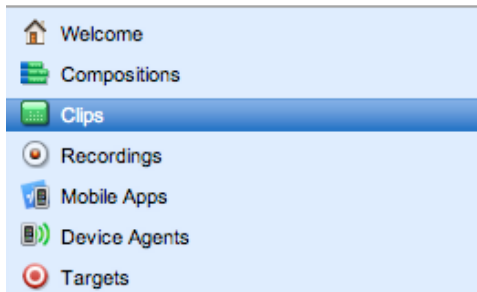


If necessary, choose the Download the Record Browser extension command to install the extension and then restart Firefox before proceeding.

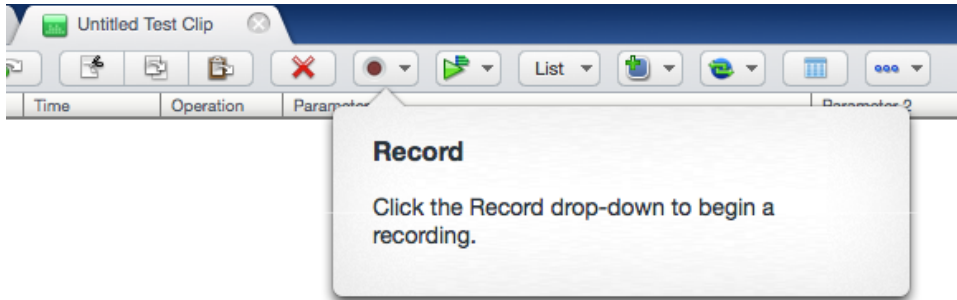
## Recording a WebUI / Ajax Test Clip

The Clip Editor provides the ability to record browser actions inline. Target creation occurs on-the-fly, the underlying targets are accessible when creating or modifying system or custom properties.

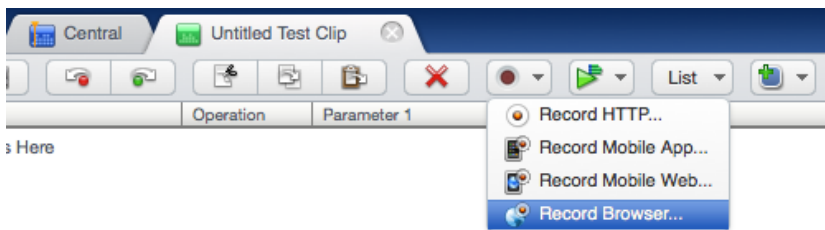
1. Click Central > Clips selected, click New to launch the Clip Editor.



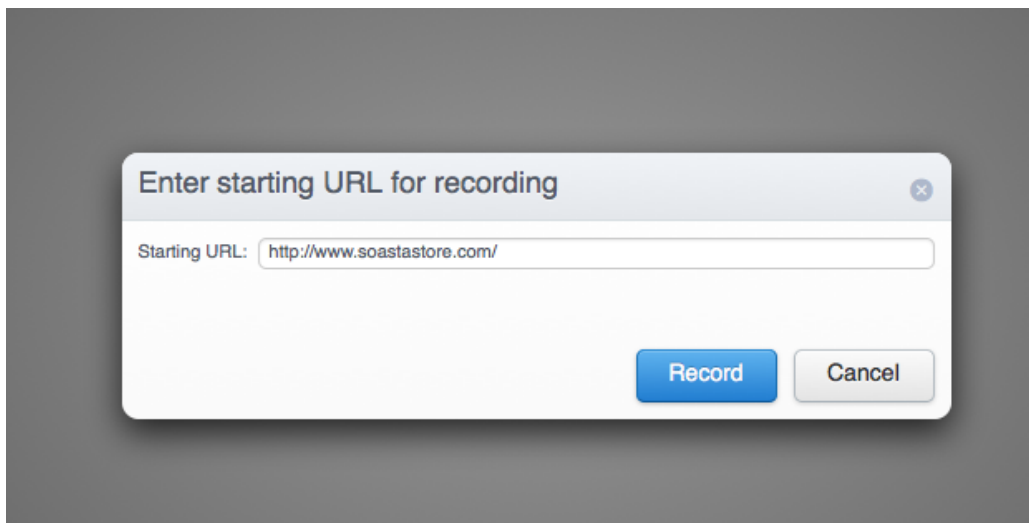
A new *Untitled Test Clip* opens.



2. Click the Record button and then select Record Browser.



The Enter Starting URL for recording box appears for you to enter the target URL.

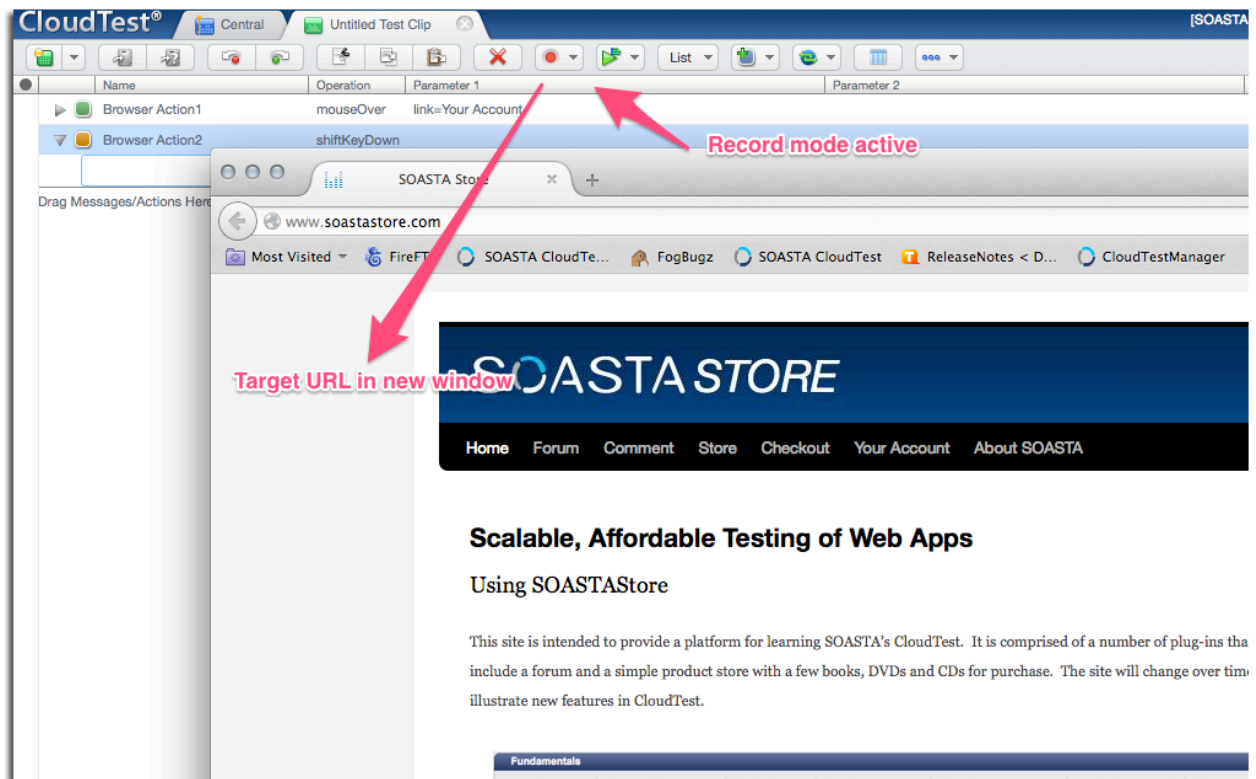


If the URL entered is already in a target, then that target is used. Otherwise, a new WebUI target is created.

**Note:** The shot below shows the Clip Editor in List view. Refer to [List View Mode](#) to learn how to switch between view modes.

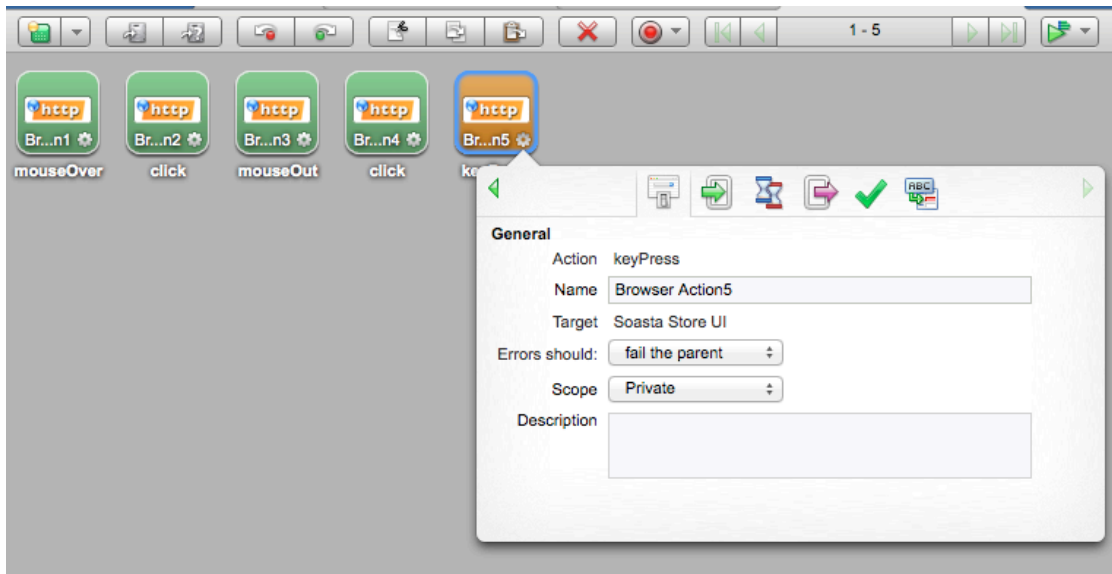
3. Click Record to proceed with your recording in a new window.

When you do so, the target URL opens in a new Firefox window and the Clip Editor, Record button indicates recording is underway (both shown below).



4. Now that we've got the target site up in its own window, we're ready to record. Click the Store link in the top toolbar of the store site. If necessary, use your mouse to arrange the CloudTest and the target site windows so you can see them both.

The actions added to the clip will vary according to your actions. For example, depending on how you handle a mouse, you may get a mouseOver browser action in your clip prior to the click(s). The clip will also vary around user choices such as whether to access an object using the keyboard or mouse.



5. Click three or more of the Add to Cart buttons on the Store page. Additional click actions are added to the workspace.

6. In the Store page, scroll up to the e-Commerce section, which shows the Shopping Cart's contents and then click one of the blue links to view a product detail page for one of your selected items.

## e-Commerce

### Shopping Cart

Product	Qty	Price
<a href="#">The Sixth Man</a>	1	\$9.99
<a href="#">Vegas Moon</a>	1	\$14.99
<a href="#">Jefferson Key</a>	1	\$11.99
<a href="#">Buried Prey</a>	1	\$11.99

7. Click the product photograph to view it in a popup window and then close it.
8. With the product detail page still showing, enter Search text in the provided entry field and then click (or key press) the Search button. For example, in our clip we entered an item's title: "Jefferson Key". Note the search result text for this action as we'll use the word spellbinding later to validate text in the test.

## Jefferson Key

[Jefferson Key](#) One of the most spellbinding and ingenious openings in all of thrillerdom.

Old Price: ~~\$12.99~~  
Price: \$11.99  
You save: \$1.00! (7.70%)  
Shipping: \$0.00

Add To Cart

-----  
Avg. Customer Rating:

★★★★★ (0)

Your Rating:

1 Save



9. Return to the top menu and click Checkout.

## Checkout

### Shopping Cart

Product	Qty	Price	
<a href="#">The Sixth Man</a>	1	\$9.99	▣
<a href="#">Vegas Moon</a>	1	\$14.99	▣
<a href="#">Jefferson Key</a>	1	\$11.99	▣
3 items	Total: \$36.97		
<a href="#">Checkout</a>	Clear cart		

Please review your order

Product	Quantity	Price	Total	
 <a href="#">The Sixth Man</a>	1 <input type="text"/> <input type="button" value="Update"/>	\$9.99	\$9.99	<input type="button" value="Remove"/>
 <a href="#">Vegas Moon</a>	1 <input type="text"/> <input type="button" value="Update"/>	\$14.99	\$14.99	<input type="button" value="Remove"/>

The Checkout page appears. This page has a lot of useful buttons and entry fields. Keep in mind, the information you enter here is practice only and is not actually used to purchase products.

10. Scroll to the heading “Please review your order” and change one or more values.

- Click Update for each numeric change that you make.
- Click Remove for any one of the items in your Shopping Cart. If an item you added gets the message "cannot ship to..." then remove it before proceeding.

11. Scroll down and fill out all of the information in the Join up now section.

**Note:** Since this exercise is solely to illustrate some of the UI features of a typical eCommerce site, it is NOT necessary to actually complete a purchase or to enter any true information. Any errors that occur because of incomplete information can be safely ignored in this context or you can continue on until a successful transaction is complete.

12. Fill out the Your billing/contact details section.

13. Click the Purchase button to complete the transaction.

**SOASTA STORE**

Home Forum Comment Store Checkout Your Account About SOASTA

### Transaction Results

Thank you, your purchase is pending. You will be sent an email once the order clears.

Thank you for purchasing with SOASTA Store, any items to be shipped will be processed as soon as possible, any items that can be downloaded can be downloaded using the links on this page. All prices include tax and postage and packaging where applicable. You ordered these items:

Name	Price	Quantity	Item Total
Vegas Moon	\$14.99	2	\$29.98
Jefferson Key	\$11.99	2	\$23.98
Buried Prey	\$11.99	2	\$23.98

Total Shipping: \$0.00  
Total: \$77.94

Search

**Archives**

**Meta**

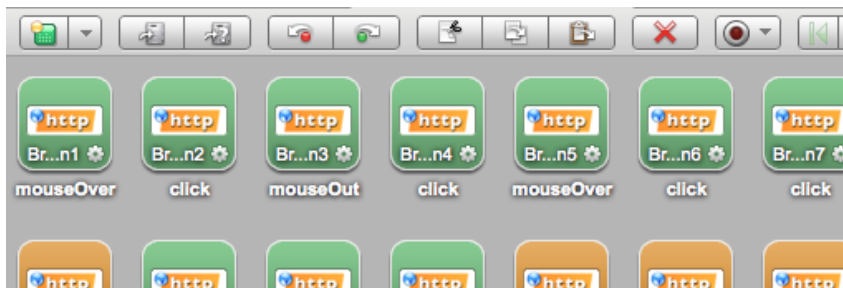
- Site Admin
- Log out

**About You**

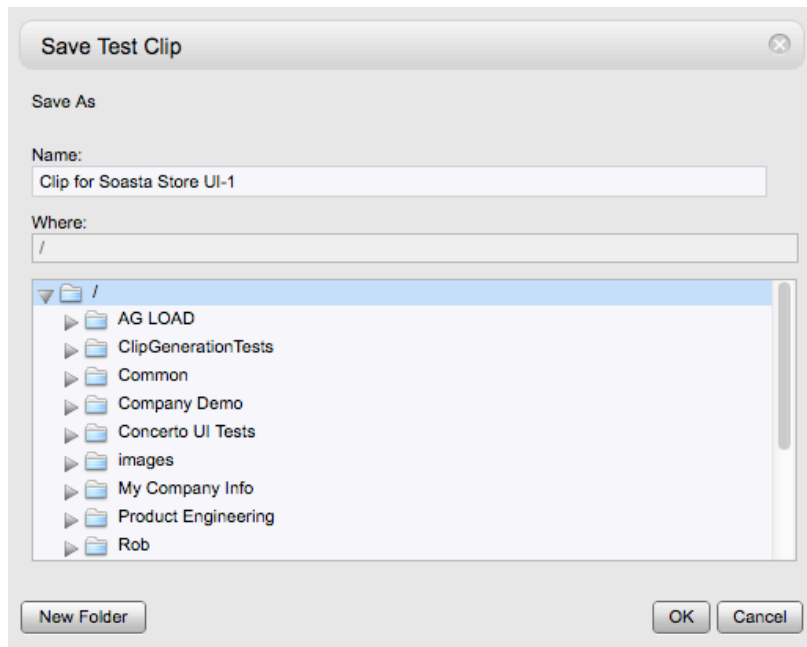
- IP: 74.132.248.80
- Location: US

The Transaction Results page shown above corresponded to Browser Action25 in our example clip.

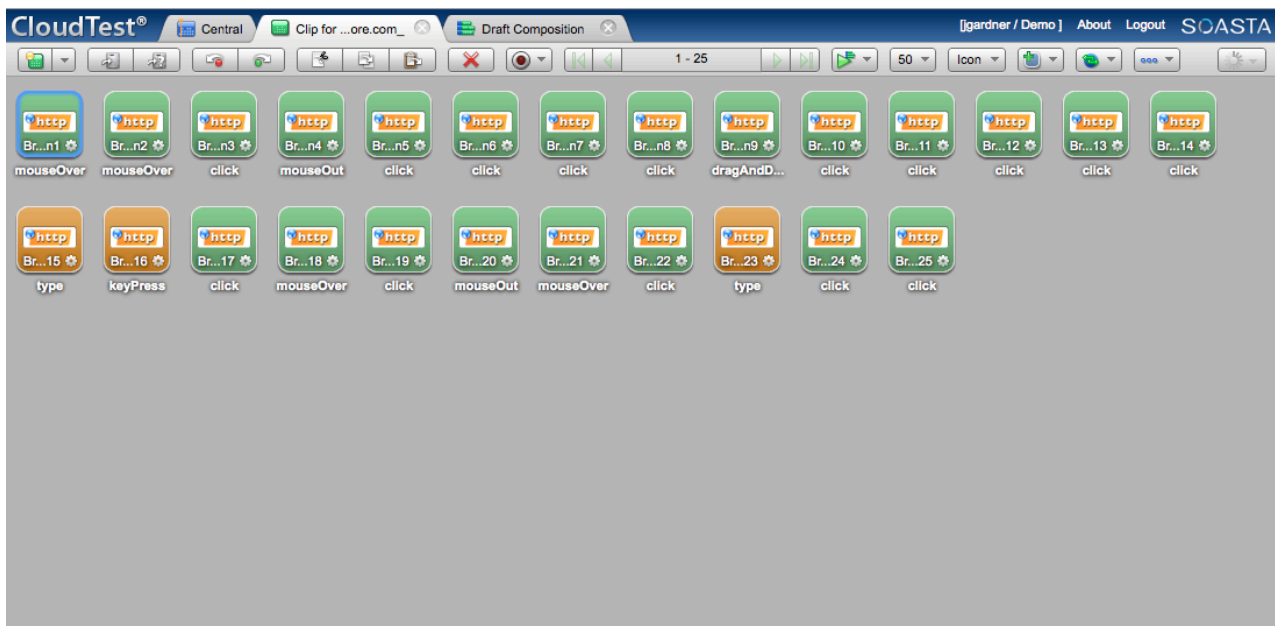
14. In the Clip Editor, click Record a second time icon to stop the recording.



15. Click Save on the Clip Editor toolbar.  
The Save Test Clip box appears.
16. Enter a name for the test clip (or accept the default one) and then click OK.



The newly created clip is shown in Icon view below.



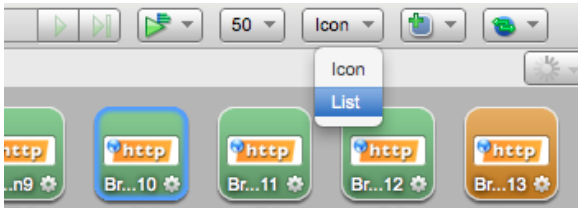
The captured Browser Actions are the result of the user actions you performed during recording. Your specific recording will vary slightly in appearance, depending on the actions taken to complete the above steps.



## Inspecting the Newly Created Clip

Before proceeding with test creation, switch to List view to inspect the newly created test clip. List view provides visual access to more clip element details by presenting them in a tabular list.

1. Click the drop down Icon/List button on the toolbar and then select List.



The List view is probably the most useful view of a test during the editing process because it shows Parameters (6<sup>th</sup> column right).

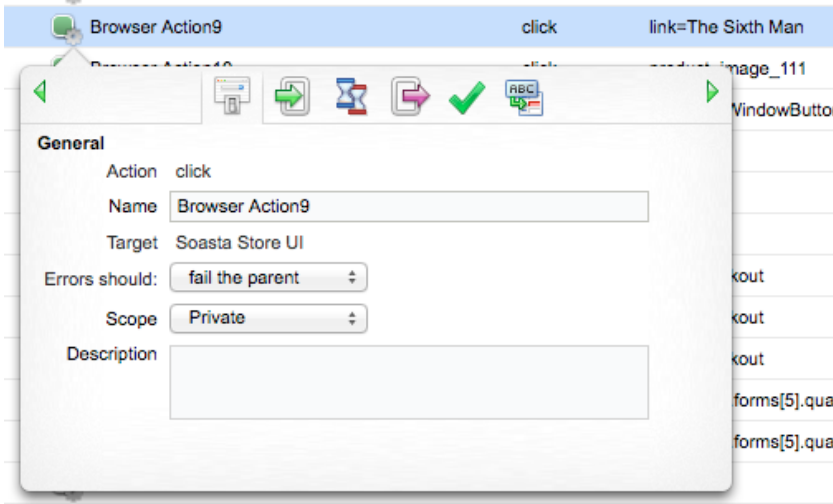
A screenshot of the CloudTest interface showing the List view of a test clip. The interface includes a toolbar at the top with various icons and a dropdown menu set to 'List'. Below the toolbar is a table with columns for Name, Operation, Parameter 1, and Target Name. The table contains 21 rows of browser actions, with 'Browser Action10' highlighted in blue.

Name	Operation	Parameter 1	Target Name
Browser Action1	mouseOver	link=Store	Soasta Store UI
Browser Action2	click	link=Store	Soasta Store UI
Browser Action3	mouseOut	link=Store	Soasta Store UI
Browser Action4	click	product_111_submit_button	Soasta Store UI
Browser Action5	mouseOver	link=Store	Soasta Store UI
Browser Action6	click	product_112_submit_button	Soasta Store UI
Browser Action7	click	product_116_submit_button	Soasta Store UI
Browser Action8	click	product_118_submit_button	Soasta Store UI
Browser Action9	click	link=The Sixth Man	Soasta Store UI
Browser Action10	click	product_image_111	Soasta Store UI
Browser Action11	click	TB_closeWindowButton	Soasta Store UI
Browser Action12	click	s	Soasta Store UI
Browser Action13	type	s	Soasta Store UI
Browser Action14	keyPress	s	Soasta Store UI
Browser Action15	mouseOver	link=Checkout	Soasta Store UI
Browser Action16	click	link=Checkout	Soasta Store UI
Browser Action17	mouseOut	link=Checkout	Soasta Store UI
Browser Action18	click	document.forms[5].quantity	Soasta Store UI
Browser Action19	type	document.forms[5].quantity	Soasta Store UI
Browser Action20	click	submit	Soasta Store UI
Browser Action21	click	document.forms[12].submit	Soasta Store UI

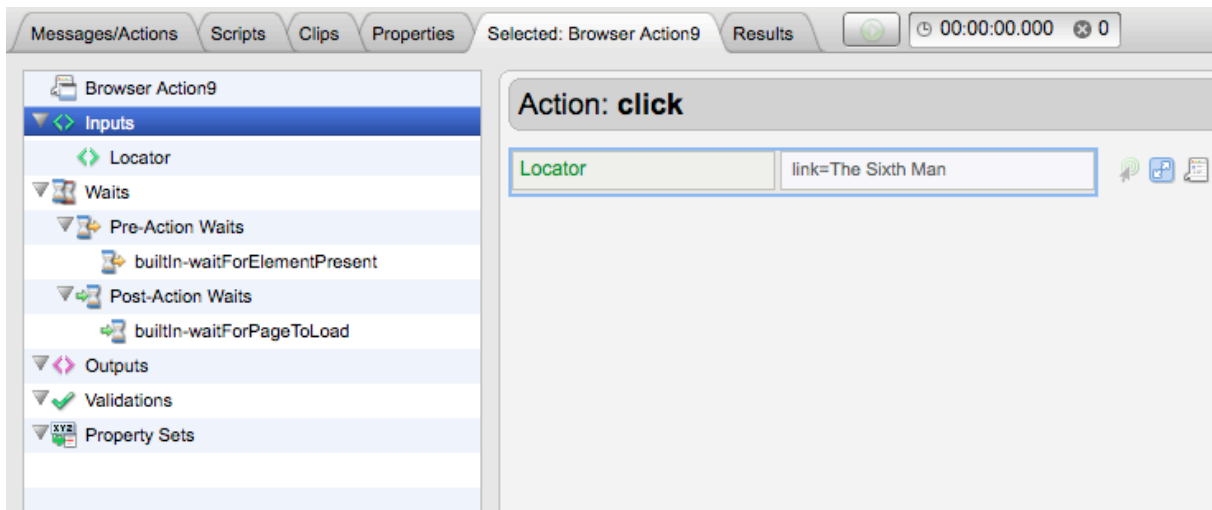
## Browser Action Elements and Properties

Browser action details can be viewed either by clicking the Gear icon on the browser name to show the Info Window or by double-clicking to open an action in the lower panel Action Editor. When the Info Window is open the lower panel is minimized, and when the Action Editor is open, the Info Window goes away.

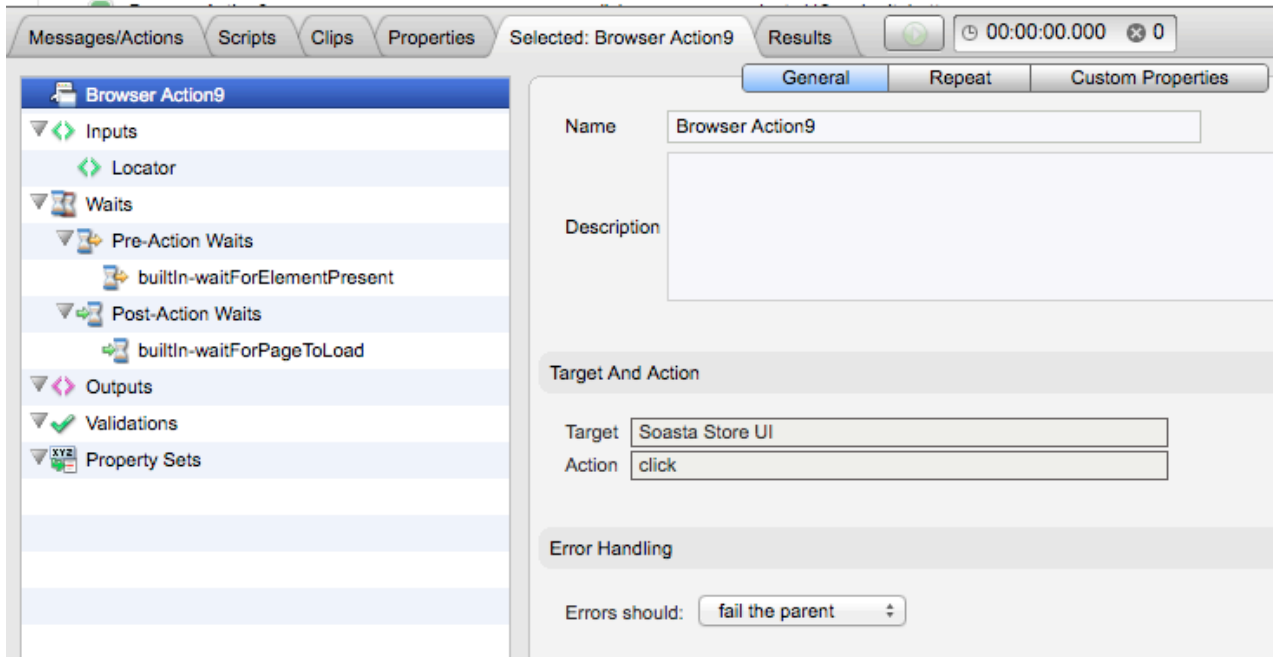
- Whether in Icon or List view, you can examine elements and properties for any Browser Action by selecting clicking its Gear icon. When you do so, the Info Window tab displays selection details.



- You can also examine elements and properties for any Browser Action by selecting it in the workspace above and then double-clicking it. When you do so, the <Selected: Browser Action> tab displays the contents of the selection in its own pane.




The Info Window and Action Editor nodes are: General (top-node above), Inputs, Waits, Outputs, Validations, and Property Sets. While the *Selected: Browser Action* tab active, the browser action-level nodes are shown in the tree on the left.



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

#### Examine and edit Browser Action properties for any given selection:

1. In the *Selected: Browser Action 12* tab, familiarize yourself with the available Browser Action elements and properties.


-  **General** (including Repeat and Custom Properties tabs in the lower panel only)

#### **Inputs** (Locator, Coordinate String)

Locators are unique characteristics that identify a specific element or object on a web page. Locators come in many forms, including links, IDs such as those defined within CSS, and XPath expressions. For example, a locator can be captured as a Document Object Model (DOM) location such as in an XPath statement, or by its use of HTML elements, CSS classes, or JavaScript.

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

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

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

-  **Validations**

Validations determine whether a Browser Action succeeded or failed. Browser action validations range from simple true/false conditions such as “verifyAlertNotPresent,” which verifies that the Browser Action did not cause an alert to appear; to more complex conditions such as “verifyTableCellText,” which verifies that a particular cell in a given table on the page contains the specified text. A single Browser Action can have an unlimited number of validations. Any validation failures will be exposed in the Results Dashboard.

-  **Property Sets**

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

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

- *Custom Properties*

Custom Properties are user-defined properties that are available to all clip elements, including Browser Actions. Custom properties can be thought of as backdoors that allow access to portions of the object model more easily. For example, you can capture a *sessionID* using a custom property that can be passed to a global property and used across tests.

- *System Properties*

System Properties are available to all clip elements, including Browser Actions. SOASTA CloudTest defines system properties. For example, a Browser Action target has system properties for browser driving; while a test clip has system properties such as name, repeat timing, label, and more.

- *Global Properties*

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

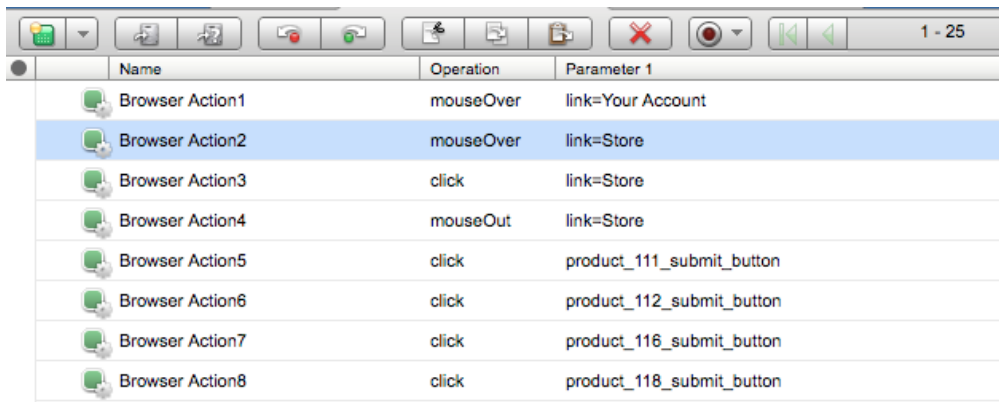
**TIP:** Refer to [Property Sets in a Browser or App Action](#) for more information on property sets.

## Adding a Text Output

In the following steps, we will add an output to each of the browser actions in our clip that correspond to the Add Cart buttons we clicked above. This output will capture the body text of the page so we can find interesting things to validate.

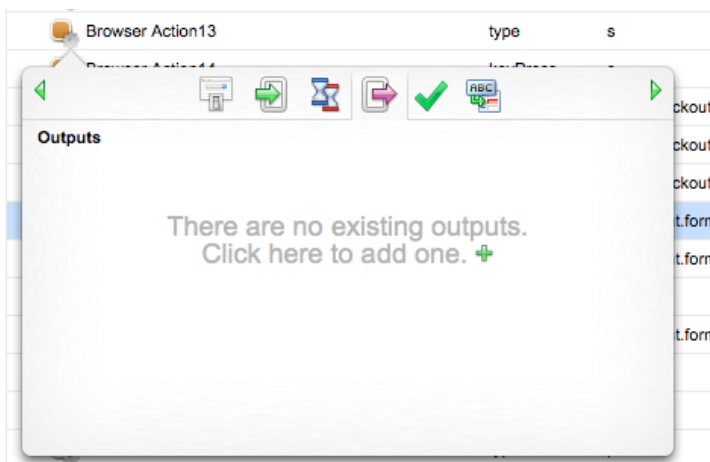
Typically, in the early stages of test building an output is used for informational purposes. However, the content of outputs can also be stored and used in a custom property set for later use.

1. In the Clip Editor list view, identify the Add Cart buttons by their locators, which include the `product_xxx_submit_button` format (shown below in Browser Actions 5-8).

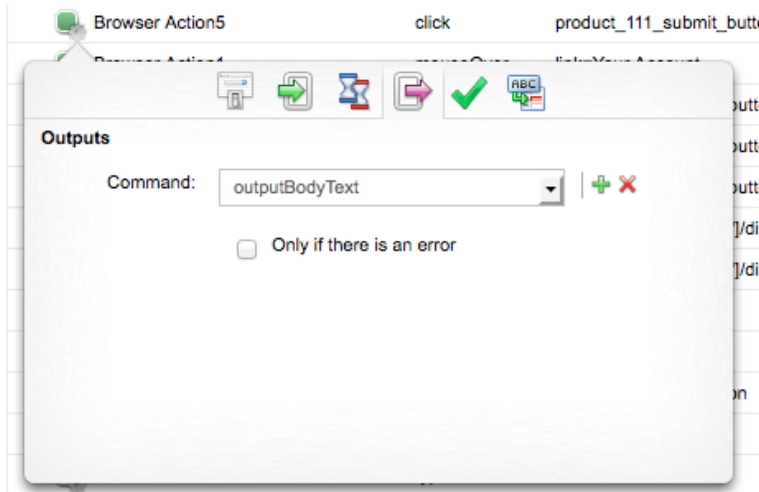


Name	Operation	Parameter 1
Browser Action1	mouseOver	link=Your Account
Browser Action2	mouseOver	link=Store
Browser Action3	click	link=Store
Browser Action4	mouseOut	link=Store
Browser Action5	click	product_111_submit_button
Browser Action6	click	product_112_submit_button
Browser Action7	click	product_116_submit_button
Browser Action8	click	product_118_submit_button

2. Click the Left Arrow in the Info Window to return to the type action identified above.
3. Click the Output tab.



4. Click the green Plus icon. A new output is added to the Outputs list.
5. Click the Command list drop-down, and then scroll and select `outputBodyText`.



Each page in the sequence included some text about the shopping cart's contents, starting with "Your shopping cart is empty" and then "You just added "The Sixth Man" to your shopping cart," and so forth. While we don't really need the output on the browser action if we already know what to validate—adding key outputs can serve as a helpful primer for a page or any of its elements that have text we'd like to validate.

## Adding a Screenshot Output

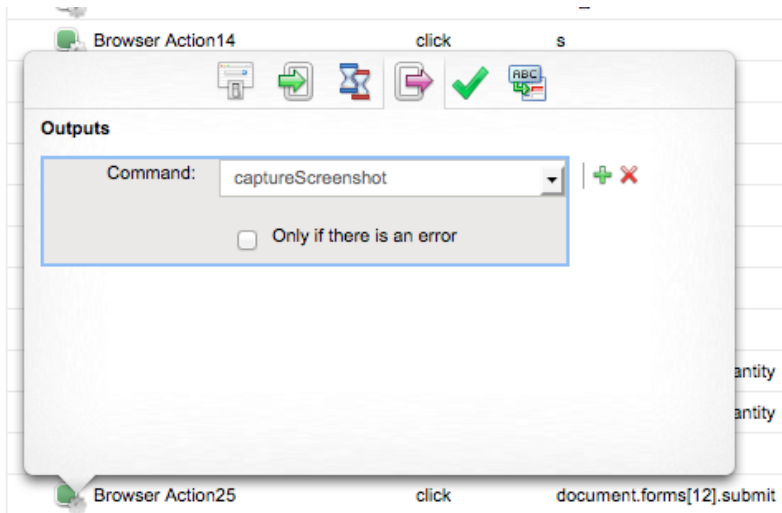
We can also use the `captureScreenshot` output for helpful reminders of which action corresponds to which page or object on a page. In the steps below, we'll get a screenshot of the last action in our test clip to find something on it to validate.

1. In the Clip Editor list view, locate the final browser action in your clip and open its Info Window.

In our example clip, the final page is the transaction page and corresponds to Browser Action25.

2. Click the Output tab.
3. Click the green Plus icon. A new output is added to the Outputs list.

4. Click the Command list drop-down, and then scroll to select *captureScreenshot*.

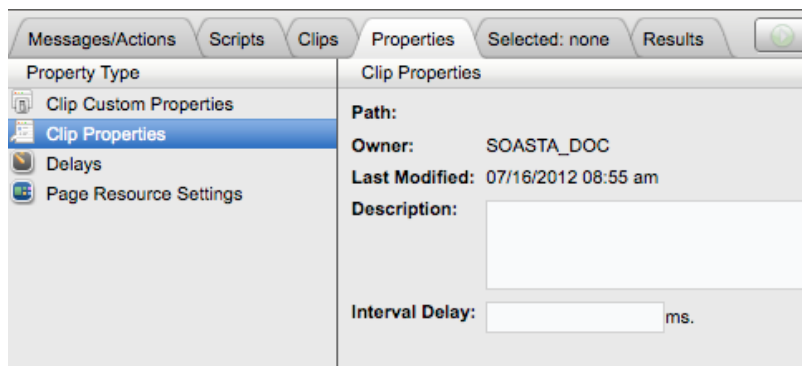


5. Click Save on the Clip Editor toolbar.

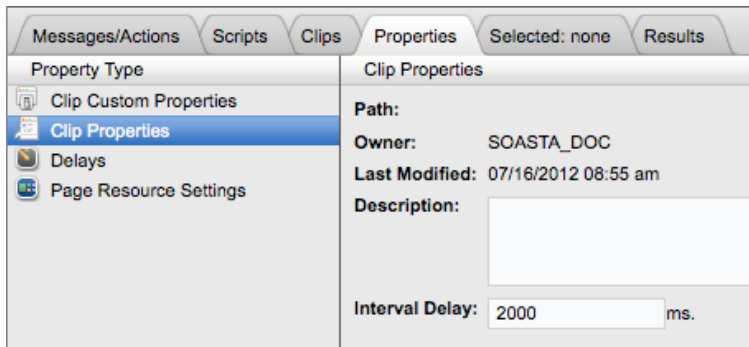
## Adding an Interval Delay between Each Browser Action

In the following steps, we will add a delay to the test clip for use when debugging the test. This type of delay may not be useful when running automated tests, such as those tests that are part of nightly builds; however, it can be useful to impose some delays to make the test viewable during the test editing phase.

1. Click the Properties tab in the sub-pane and then select the Clip tab at the top of the pane (the Clip tab may already be visible if properties are already open from the prior exercise).
2. In the Property Type list, click Clip Properties.



3. In the Clip Properties area on the right, enter an Interval Delay in the given field. For example, 2000 ms. Entering 2000 adds a two second gap between each action in the given test clip.



4. Click Save on the Clip Editor toolbar.



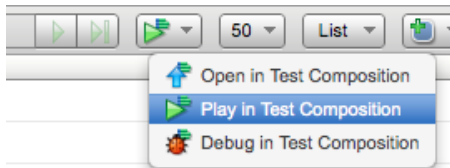
## Playing a Simple WebUI / Ajax Test

Now that we've added a few outputs to the browser actions in the Clip Editor, we are ready to play this in a test composition and get a better idea what we have to validate. You can start your desktop client Conductor before or during the steps below.

1. Start the Conductor that you used to record this clip. Since we just recorded this clip we know which Conductor to start.

**Note:** If your SOASTA Conductor is not started when you click Play (or select the Play in Test Composition command), you will be prompted at that time to start it. The Warning box that appears names the Conductor to start. Click Continue once you have started the named Conductor on your desktop client.

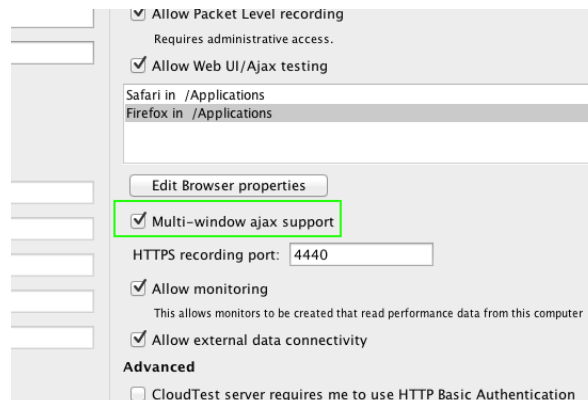
2. To play this clip in a new WebUI / Ajax test composition, click the Use in Composition drop-down and then select Play in Test Composition.



When you do so, the Composition Editor appears with a draft test composition and begins to play in the Composition Editor, Play tab, which displays the default Result Details dashboard.

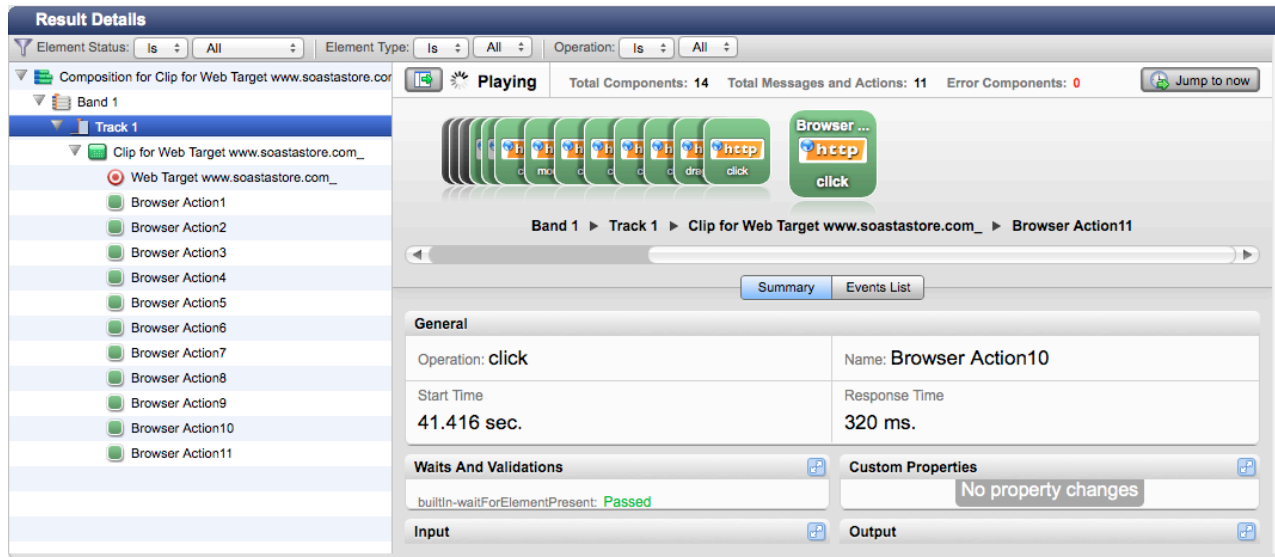
**Note:** Since the SOASTA Store is only for practice use, don't become overly attached to getting no errors in this example scenario. If you experienced a Registration/Login error on playback and you're not in a SOASTA class where the Admin can reset your login for you, simply ignore the error.

In some cases, Firefox users report issues with the Checkout link, in which case ensure that Multi-window Ajax support is checked in your Conductor's Preferences.



## Result Details

The Results Details chart is the first place test results are presented, including in this case its outputs, and whether or not the test composition passed or failed.



Result Details uses a Cover Flow to display the test composition's message stream, as well as a Navigation Tree. Items stream into both unless something is clicked. Once you've clicked an element of the test—you can click Jump to now to return to stream mode.

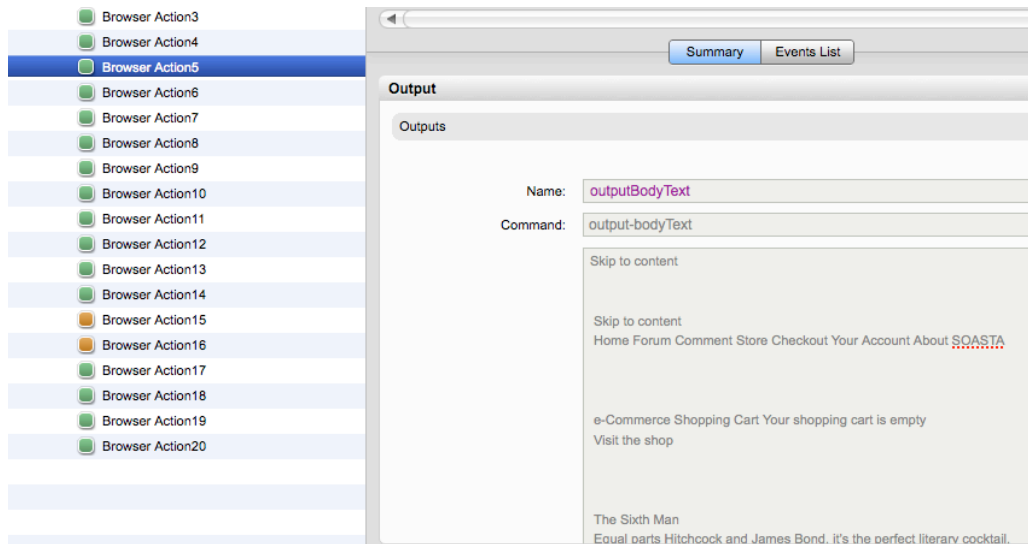
- Click any object in the Cover Flow at the top to center it and display its details and play statistics in the panes below.
- Use the scrollbar to browse the flow. Select any item to show its low-level details. However, we can easily specify a different Conductor at any time by editing the target's Conductor setting (preferably before we click Play).

To open the target from within the Clip Editor, click the Messages/Actions tab in the lower panel, and then double-click the Included Target in the Included Targets list. You can also locate the target by name in the Central > Targets list, of course.

## Inspecting Clip Element Outputs

In the basic edits above we added text outputs for each of the Add Cart clicks as well as a screenshot output for the last browser action in the test clip. Now, let's use the following steps to inspect those outputs in this result.

1. In the Navigation Tree (on the left), select Browser Action 5.
2. Click the Maximize icon on the Outputs section.

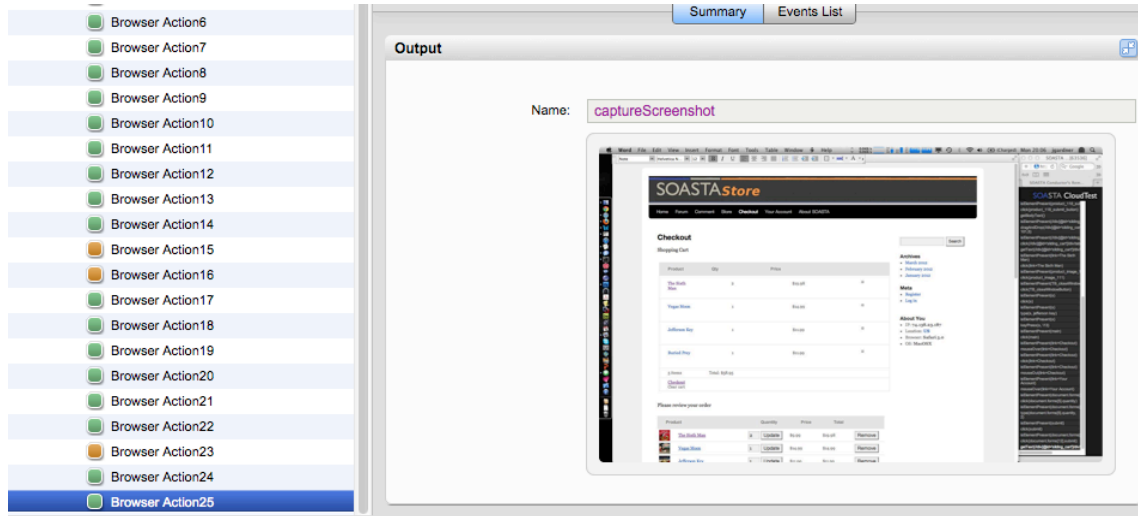


Note the text, "Your shopping cart is empty."

3. Do the same for Browser Action6 (and repeat for each of the other actions to which you added a body text output). Note that Browser Action6's bodyTextOutput includes the text, "You just added "The Sixth Man" to your cart."

**TIP:** In our example clip, the remaining body text outputs are: “You just added "Vegas Moon" to your cart,” “You just added "Jefferson Key" to your cart,” and finally, “You just added "Buried Prey" to your cart.”

4. Do the same for Browser Action25 (or whatever your last action is named).



5. Click the screenshot to pop it to full size. In the screenshot above, we noted that the final in the Total field was \$58.95.

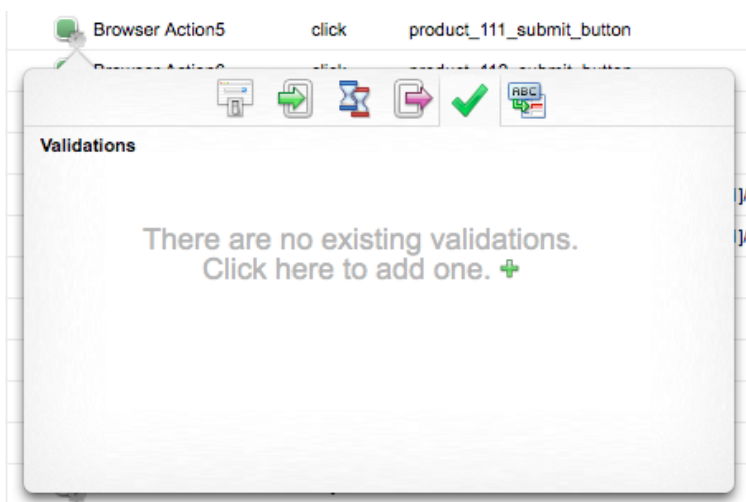
## Advanced Clip Editing

In the advanced steps below, we'll use the information we learned from the outputs of the first play of our new test to add some validations.

### Adding a Validation on Body Text

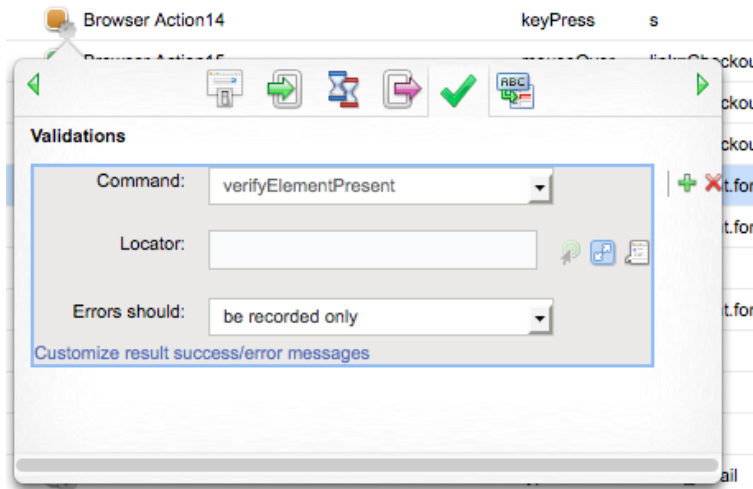
In the following steps, we will add a validation on the body text we learned about from the output above. For each browser action that corresponds to an Add Cart click in the test clip above, we will .

1. Open Browser Action5 in the Info Window and then click the Validations  tab.

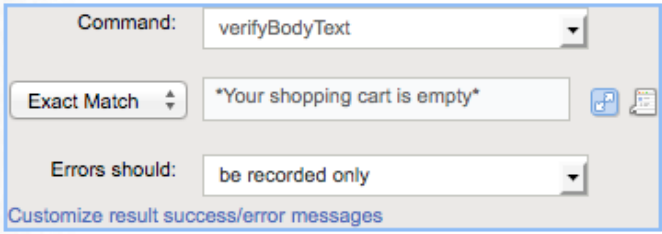


2. Click the green Plus sign.

A new validation form is added to the Info Window.

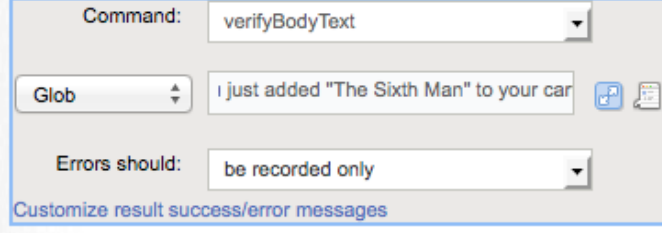


3. In the Command list, click the drop-down and select *verifyBodyText*. This Command verifies the specified text is on the rendered page.



The screenshot shows a configuration panel for a validation command. The 'Command' dropdown is set to 'verifyBodyText'. The 'Exact Match' dropdown is selected. The pattern field contains the text '\*Your shopping cart is empty\*'. The 'Errors should:' dropdown is set to 'be recorded only'. A link at the bottom reads 'Customize result success/error messages'. To the right of the panel are green plus and red minus icons.

4. In the pattern field, select Glob and enter *\*Your shopping cart is empty\**. Leave the default *be recorded only* set in the Failure action field.
5. For the sake of comparison, we will also add a *verifyBodyText* validation to Browser Action6 using *\*You just added "The Sixth Man" to your cart\**.




The screenshot shows the same configuration panel as above, but the 'Exact Match' dropdown is now set to 'Glob' and the pattern field contains the text 'I just added "The Sixth Man" to your car'. The 'Errors should:' dropdown remains 'be recorded only'. The 'Customize result success/error messages' link and the plus/minus icons are also present.

6. Finally, we will validate the total on the final page of the clip, which in our case is Browser Action25. The value to validate was \$58.95.
6. Click Save on the Clip Editor toolbar.

## Adding a Validation on an Element

Validations on page elements require that an exact ID or XPath be provided for a given element. The Locator Finder is available to determine an ID for any action that has a locator parameter, and also for any Wait, Validation, Output, or Property Set that has a locator parameter.

1. Open the Info Window for Browser Action25.
2. In the Info Window, Validations tab click the green Plus sign  to add the validation form.

In the Command drop-down, select *verifyElementText*.

3. Click the Locator field.

The Locator Finder (to the right of the text field) will appear in a disabled state initially.

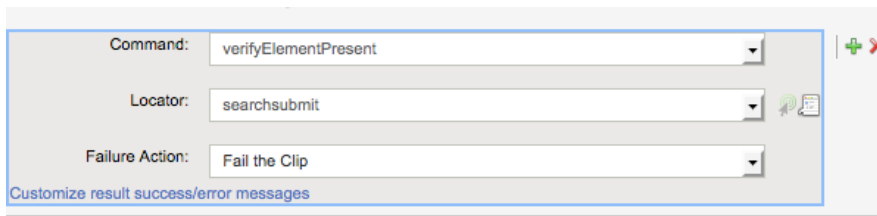


1. Click the Locator Finder to enable it. The target web site opens in a new window (if it was not already open).

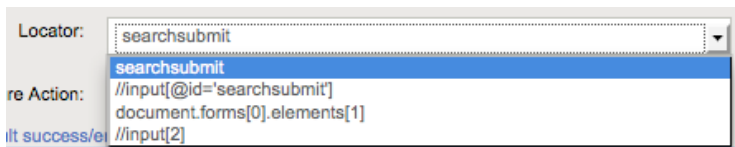


While the Locator Tool is enabled, moving the mouse over the Web page being tested will highlight elements in the page with an orange border.

2. Navigate to the Total field (it doesn't have to be the page showing the final total) and hover the mouse until the orange border snaps-to and then click to select it.. The Locator field in the Clip Editor is populated with the most likely XPath locator for the given selection. If more than one option was detected, the Locator field drop-down appears. The Locator icon returns to its inactive state.

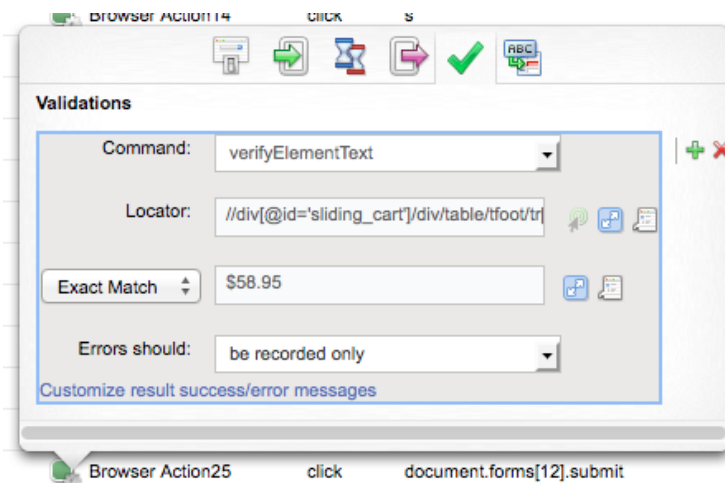


If it appears, click the Locator drop-down to inspect other possible values. In most cases, the Locator will select the right XPath. However, if the validation fails when it shouldn't, it may be necessary to select an alternate value from among those the Locator populates.



In the case of Browser Action25, only one value was detected in our example clip:

```
//div[@id='sliding_cart']/div/table/tfoot/tr[1]/td[2]/span
```



3. Click Save on the Clip Editor toolbar.
4. In the Draft Composition you created earlier, click Play a second time. Or, if it is no longer available, use the Play in Composition command a second time.



## Analyzing Results

When play completes, the test is once again rated on the basis of its own criteria (the default settings are all still in effect).

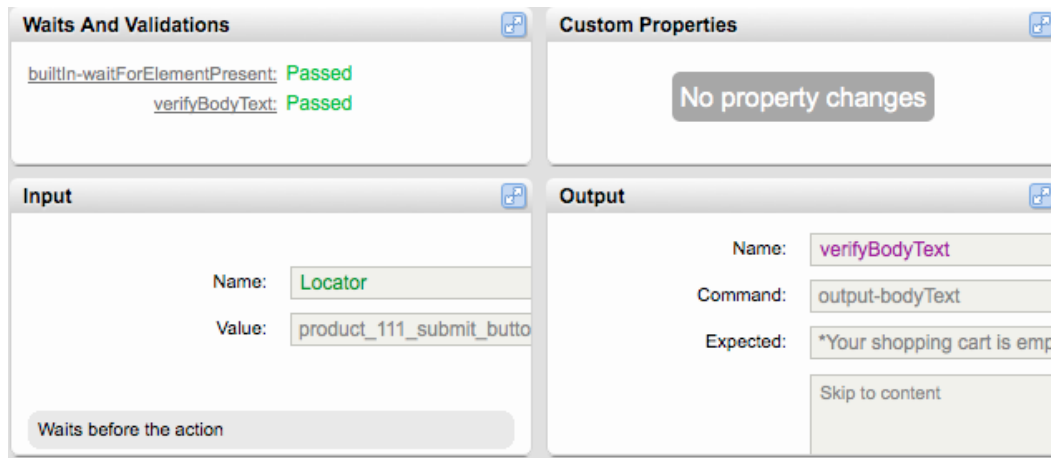
1. Once it appears, click Browser Action5 to examine the validation we added to body text.

The screenshot shows the 'Result Details' window in TestComplete. The left pane shows a test composition tree with 'Track 1' expanded to show 'Browser Action5' selected. The main area displays a 'Playing' status with a progress bar and a sequence of action icons. Below this, the 'General' section shows 'Operation: click' and 'Name: Browser Acti'. The 'Start Time' is 49.609 sec. and the 'Response Time' is 290 ms. The 'Waits And Validations' section shows 'builtin-waitForElementPresent: Passed' and 'verifyBodyText: Passed'. The 'Custom Properties' section is empty, showing 'No prop'. The 'Input' and 'Output' sections are also visible at the bottom.

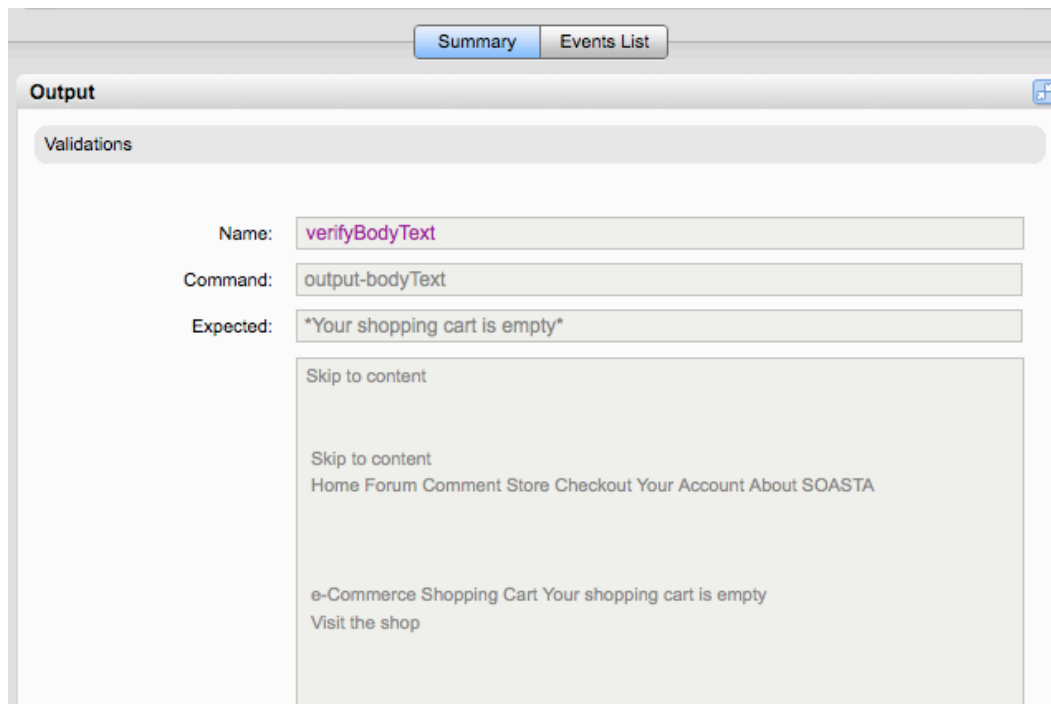
The early returns look good for Browser Action5 as the Waits and Validations section shows that verifyBodyText passed.

2. With Browser Action5, still selected (or re-selected if the test completed play and jumped to the final actions) click the verifyBodyText link.

When you do so, the Output field scrolls to reveal the verifyBodyText details.



3. Click the Output section Maximize icon.



- Click the Events List tab to see an event view on this browser action that includes its validation(s).

Event(s)				
Event	Time	Level	Event Code	Description
33	47603	Verbose	Transport: brend	<b>Browser Action "Browser Action5" completed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for Web Target www.soastastore.com_" Target: "Web Target www.soastastore.com_" Browser Action: "Browser Action5" ▶Details:
34	47604	Info	Validation: vstart	<b>Starting validation "verifyBodyText".</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for Web Target www.soastastore.com_" Target: "Web Target www.soastastore.com_" Browser Action: "Browser Action5"
35	47605	Verbose	Validation: vcpass	<b>Validation of response body passed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for Web Target www.soastastore.com_" Target: "Web Target www.soastastore.com_" Browser Action: "Browser Action5" ▶Details:
36	47605	Info	Validation: vpass	<b>Validation verifyBodyText passed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for Web Target www.soastastore.com_" Target: "Web Target www.soastastore.com_" Browser Action: "Browser Action5"
37	47605	Info	Browser Action: sent	<b>Browser Action completed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for Web Target www.soastastore.com_" Target: "Web Target www.soastastore.com_" Browser Action: "Browser Action5"

- Next, click Browser Action6 and inspect the same Events List details. This time expand the Details arrow to view additional data.

Event(s)				
Event	Time	Level	Event Code	Description
				Band: "Band 1" Track: "Track 1" Clip: "Clip for Web Target www.soastastore.com_" Target: "Web Target www.soastastore.com_" Browser Action: "Browser Action6" ▶Details:
42	49900	Info	Validation: vstart	<b>Starting validation "verifyBodyText".</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for Web Target www.soastastore.com_" Target: "Web Target www.soastastore.com_" Browser Action: "Browser Action6"
43	49901	Verbose	Validation: vcpass	<b>Validation of response body passed.</b> Band: "Band 1" Track: "Track 1" Clip: "Clip for Web Target www.soastastore.com_" Target: "Web Target www.soastastore.com_" Browser Action: "Browser Action6" ▼Details:  Glob expression match passed: Expression: *You just added "The Sixth Man" to your cart* Value: Skip to content  Skip to content Home Forum Comment Store Checkout Your Account About SOASTA

- Next, click Browser Action25 and maximize its Output section to verify that the expected and observed value in the Total field were the same.

The screenshot displays a test runner interface. At the top, a green button labeled 'Browser ...' with an 'http' icon and the word 'click' is visible. Below it, a breadcrumb trail reads 'Band 1 > Track 1 > Clip for Web Target www.soastastore.com\_ > Browser Action25'. A horizontal scrollbar is present. Below the scrollbar are two buttons: 'Summary' (highlighted in blue) and 'Events List'. The main section is titled 'Output' and contains a table with the following data:

Name:	verifyElementText
Command:	output-elementText
Locator:	//div[@id='sliding_cart']/div/table/tfoot/tr[1]/td[2]/span
Expected:	\$58.95
Observed:	\$58.95

At the bottom of the 'Output' section, there is a grey bar labeled 'Outputs'.

## Other Widgets Useful to WebUI / Ajax Tests

SOASTA CloudTest offers a comprehensive set of analytic widgets. The following are particularly relevant to WebUI testing:

- The Composition Analysis widget renders the “Component Hierarchy” for your test composition.

Composition Analysis						
Component Hierarchy	Avg Duration	Min Duration	Max Duration	Bytes Sent	Bytes Received	Avg. Resp.
▼ Composition	0.000 s	0.000 s	0.000 s	0	0	0.000 s
▼ Band 1	97.494 s	97.494 s	97.494 s	0	0	0.605 s
▼ Track 1	97.486 s	97.486 s	97.486 s	0	0	0.605 s
▼ Clip for Web Target www.s...	97.485 s	97.485 s	97.485 s	0	0	0.605 s
Browser Action1	1.123 s	1.123 s	1.123 s	0	0	1.123 s
Browser Action2	1.876 s	1.876 s	1.876 s	0	0	1.876 s
Browser Action3	2.021 s	2.021 s	2.021 s	0	0	2.021 s
Browser Action4	1.749 s	1.749 s	1.749 s	0	0	1.749 s
Browser Action5	0.421 s	0.421 s	0.421 s	0	0	0.421 s
Browser Action6	0.290 s	0.290 s	0.290 s	0	0	0.290 s
Browser Action7	0.226 s	0.226 s	0.226 s	0	0	0.226 s
Browser Action8	0.286 s	0.286 s	0.286 s	0	0	0.286 s
Browser Action9	1.235 s	1.235 s	1.235 s	0	0	1.235 s
Browser Action10	0.143 s	0.143 s	0.143 s	0	0	0.143 s
Browser Action11	0.210 s	0.210 s	0.210 s	0	0	0.210 s
Browser Action12	0.644 s	0.644 s	0.644 s	0	0	0.644 s
Browser Action13	0.358 s	0.358 s	0.358 s	0	0	0.358 s
Browser Action14	0.402 s	0.402 s	0.402 s	0	0	0.402 s
Browser Action15	0.251 s	0.251 s	0.251 s	0	0	0.251 s
Browser Action16	0.320 s	0.320 s	0.320 s	0	0	0.320 s
Browser Action17	0.419 s	0.419 s	0.419 s	0	0	0.419 s
Browser Action18	0.400 s	0.400 s	0.400 s	0	0	0.400 s

Show:  Clips  Collections  Messages and Actions

- The Event Log widget presents Component details that can be sorted by Events (Time, Component, Source, and Level) and to display Event Levels by type (All, Errors, Statistics, and Informational). This shows the very low level details of each step that occurred during the test.

**Event Log**

Summary **Composition completed.**  
 Start Time **Mon Jul 16 17:49:05 PDT 2012**  
 End Time **Mon Jul 16 17:50:44 PDT 2012**  
 Total Error Components **0**  
 Total Events **160**  
 Total Error Events **0**

Sort events by:  
 Event Level ▾  
 Event Display Level:  
 All Events ▾

Page 1 of 2

Statistics	Count	Time	Component	Details
	46	49901	Browser Action: stats	<p>Browser Action performed at time: 47,182.            Total time: 421 ms            Execution time: 421 ms.</p> <p>Time executing browser action(s): 199 ms.            Time processing outputs: 24 ms.            Time processing property sets: 0 ms.            Time processing validation outputs: 39 ms.            Pre-action waits:              Command output-isElementPresent took 0 ms.            Post-action waits:              none            Conductor processing time: 200 ms.</p> <p><b>Browser Action statistics.</b>            Band: "Band 1" Track: "Track 1" Clip: "Clip for Web Target www.soastastore.com_" Target: "Web Target www.soastastore.com_" Browser Action: "Browser Action6"            ▾Details:</p> <p>Browser Action performed at time: 49,609.            Total time: 291 ms            Execution time: 290 ms.</p> <p>Time executing browser action(s): 112 ms.            Time processing outputs: 35 ms.            Time processing property sets: 0 ms.            Time processing validation outputs: 30 ms.            Pre-action waits:              Command output-isElementPresent took 0 ms.            Post-action waits:              none            Conductor processing time: 115 ms.</p>

See [Creating a New Dashboard](#) and [Adding a Widget to a Dashboard](#) for more information about adding and configuring additional dashboards and widgets for your test.

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