

# RedwoodHQ

2.0 — Last update: 2018/04/30

PrimaTest

# Table of Contents

<b>RedwoodHQ Overview .....</b>	<b>2</b>
Release Notes .....	3
<b>Quick Start Guide .....</b>	<b>6</b>
Quick Start 1: Download, Install, Execute .....	7
Quick Start 2: First Test Case with Modified Action .....	9
Quick Start 3: Scripted Action, Clone Test Case and a new Execution .....	14
Quick Start 4: Multi-User Usage, Git and Execution Analysis .....	19
Quick Start 5: RedwoodHQ Agent .....	24
Quick Start 6: Scripted Test Case, Multithreaded Execution and New Project .....	25
Quick Start 7: Importing TestNG/JUnit and Execution from Script .....	30
<b>F.A.Q. ....</b>	<b>32</b>
<b>Documentation .....</b>	<b>33</b>
Selenium.....	35
Actions .....	38
New, Delete, Clone and Tabs for Action .....	39
Action Details and Selecting Script .....	40
Parameters .....	42
Action Within Action .....	44
Actions Left Pane .....	46
Action History .....	48
Test Cases.....	49
New, Delete, Clone and Tabs for Test Cases.....	50
Test Case Details, Selecting Script, TestNG/JUnit and After State.....	51
Test Cases Left Pane .....	54
Test Case History .....	56
Data Driven Test Case Data .....	57
Action Collection .....	59
Adding Actions to Collection .....	60
Copy and Paste, Move, Remove .....	62
Parameters in Action Collection .....	63
Execution Flow .....	65
Return Value.....	68
Machine Role and Machine Variables .....	69
Scripts .....	71
Automation Script .....	72

Adding/Deleting Jar(s) and Binaries .....	74
Git Source Control .....	76
Search/Replace .....	78
Upload/Delete File/Folder .....	79
Importing TestNG/JUnit .....	80
Execution from Script.....	81
Ivy .....	82
Execution .....	83
Details, Email, Settings, Variables, Totals and Export Results .....	84
Select Machines .....	88
Test Cases in Execution .....	90
Execution Test Details .....	92
Historical Comparison Results (Aggregate Report) .....	97
Data Driven Test Cases in Execution.....	98
Test Sets .....	99
Variables.....	101
Machines .....	103
Continuous Integration .....	106
Elasticsearch with Kibana .....	109
Settings.....	110
Users.....	111
Projects .....	113
Email .....	114
Install Guide.....	115
RedwoodHQ Server.....	116
Update Windows RedwoodHQ Server .....	117
MongoDB: v2 to v3 migration.....	118
Linux or Mac Server Install .....	119
RedwoodHQ Agent (Windows, Linux, Mac).....	120
Update RedwoodHQ Agent.....	122
Configuration (properties.conf).....	123
Migration to another RedwoodHQ instance .....	124
<b>Looking Glass.....</b>	<b>125</b>
Looking Glass Quick Start.....	126
Install Stand Alone Looking Glass .....	127
Looking Glass F.A.Q. ....	128

# RedwoodHQ Overview

---

RedwoodHQ is a **free Open Source test automation framework** that allows multiple users to develop automation code, create readable action/keyword driven test cases and execute them all under a single Web interface.

The framework was created and is maintained by [PrimaTest](#).

Download installer file from [www.RedwoodHQ.com](http://www.RedwoodHQ.com) website and afterwards go to [Quick Start Guide](#) tutorials to begin using RedwoodHQ.

[Documentation](#) section will have full explanation and examples for all RedwoodHQ functional areas and their usages.

[Release Notes](#) will have information on what changes have been made for different versions of RedwoodHQ.

To get community support of the product or just discuss RedwoodHQ and test automation in general visit [Google groups forum](#).

If you wish to fork open source RedwoodHQ project, file a bug report or contribute to it then please visit its [Github page](#).



# Release Notes

## Version 2.5 – 11/13/2016

- Linux Ubuntu RedwoodHQ Server released: [Linux or Mac Install](#)
- OS X (Mac) RedwoodHQ Server released: [Linux or Mac Install](#)
- Elasticsearch with Kibana for more detailed reporting: [Elasticsearch with Kibana](#)



- Fixes and Updates for Python support
- New Totals column in Executions -> [All Executions] to tell in a glance how many passed/failed/not run

[All Executions]				
Search:				Show Locked:
<input type="checkbox"/> Name ▲	Test Set	Status	Totals	
<input type="checkbox"/> <a href="#">Amazon Shopping</a>	Amazon Shopping	Ready To Run	1	0 0
<input type="checkbox"/> <a href="#">Searching Amazon</a>	Search	Ready To Run	8	3 0

- Moved from MongoDB v2.6 to v3.2 to utilize faster and more compact WiredTiger tech
- Multiple bug fixes

## Version 2.4 – 5/1/2016



- Able to do modify test cases to be data driven: [Test Case Data](#)
- User can now export test execution results: [Export as PDF](#)
- After State as an Action Collection: [After State](#)

- Bug and Performance Fixes

## Version 2.3 – 1/13/2016



- By popular demand: RedwoodHQ Agents for Linux and Mac (OS X): [Agent Install](#)
- Able to hook up to external code repository (eg: Github): [Git](#)
- Git commit of individual files, seeing previous versions and doing search: [Git](#)
- History and versioning of Test Cases and Actions: [Test Case History](#)
- A lot of performance improvements (now remote teams can work better with RedwoodHQ server being on another location)
- Test Designer and Test Developer user roles: [Users](#)
- Cloning Projects: [Projects](#)
- Searching and exporting Logs in test details screen: [Execution Details](#)
- Able to do a search in Scripts by using Find functionality: [Scripts Search/Replace](#)
- Bug and Performance Fixes

## Version 2.2 – 2/10/2015



- New Programming Languages: **Python and C#** (if updating from version 2.1 or 2.0 then you will need to create new project to use these new languages)
- All 3 Languages in One Project and Automated Test Case
- Ivy (Maven) Dependency Support
- Change Parameter Order for Actions
- Bug and Performance Fixes

## Version 2.1 – 8/13/2014



- Able to Import TestNG/JUnit tests
- Able to execute TestNG/JUnit code right inside Scripts page
- Upload folder with sub folders and files at one time in Scripts page
- Test Case/Action script picker button to open a script which it is pointing at
- To make Test Details easier to read, stack trace of Action Collection shows only links for code line number where failure happened but full trace can be viewed by clicking on Full Trace button
- 'Show Locked' filter check box for [All Executions] (unchecked by default, easier to navigate through executions which are considered active)
- To the right of the Search field in Test Case Left Pane there is now a number of how many test cases are currently being displayed
- Big update to Documentation
- Multiple Bug Fixes and other Enhancements

## Version 2.0 – 4/16/2014



- First version to be released to the public (previous versions been used internally by various projects)
- Looking Glass is now part of RedwoodHQ
- Easy to use Installer for all Windows OS's
- Initial Documentation and Quick Start Guide

# Quick Start Guide

---

## Getting Started with RedwoodHQ

The following Quick Start guides provide an end-to-end instructions on working with RedwoodHQ and it's main features. This can also be considered as RedwoodHQ Tutorials.

- [Quick Start 1: Download, Install, Execute](#)
- [Quick Start 2: First Test Case with Modified Action](#)
- [Quick Start 3: Scripted Action, Clone Test Case and a new Execution](#)
- [Quick Start 4: Multi-user Usage, Git and Execution Analysis](#)
- [Quick Start 5: RedwoodHQ Agent](#)
- [Quick Start 6: Scripted Test Case, New Project and Multithreaded Execution](#)
- [Quick Start 7: Importing TestNG/JUnit and Execution from Script](#)



# Quick Start 1: Download, Install, Execute

## Quick Start/Tutorial 1 Video

Here is a (**2 minute**) video that goes through all the steps listed below:



## Download

1. Go to: [www.RedwoodHQ.com](http://www.RedwoodHQ.com)
2. Click on Downloads tab
3. Click on OS version of RedwoodHQ you want to download, Windows 64-Bit recommended



If you want to deploy server on **Linux or Mac** environment, please go through the [following instructions](#).

## Install

1. Double click on the RedwoodHQ setup file
2. Go through the installation wizard and select default options
  - In ports section of installer it might ask you for Firewall permission where you click on Allow button (which will validate your port setting)
3. Wait until Installation is over (this might take a bit of time) and then click on Finish button
4. Browser will automatically open to url of RedwoodHQ on localhost (eg: `http://127.0.0.1:3000`)
5. Login with default admin user credentials
  - Username: admin
  - Password: admin



If you get an **error to login as admin user**, then do the following and try to login again:  
Open command prompt, navigate to mongodb bin folder where RedwoodHQ is installed on (eg: `c:\Program Files\RedwoodHQ\vendor\MongoDB\bin`), run the following command:  
`mongo ..\..\..\dbscripts\scripts.txt`

## Execute



Running Execution for the very first time will take a little bit of time for '**Starting Execution**' process, all runs **afterwards will begin fast!**

1. Under 'All Executions' click on 'Amazon Shopping' link
2. Expand 'Set Variables' by clicking on little triangle button next to name
3. For 'Browser' under 'Set Variables' select the browser you want to run the test cases against by clicking on the Value field for it (which should have 'Firefox' by default) and click on drop down arrow to specify other browser if you want (Internet Explorer or Chrome)
4. Under 'Select Machines' check the checkbox for '127.0.0.1' (agent was installed with RedwoodHQ server)
5. Under 'Test Cases' check the checkbox for test case 'Add Star Trek to Cart'
6. Click on 'Run Selected Execution' green arrow button on top (it might take a little bit of time to do initial run since it will be first time when all the scripts will compile)
7. Test case will now run and then when it's done the test case will have status of 'Finished' with result 'Passed'



If **test case has failed** then try: 1. **Re-run the test again** because Amazon website could be introducing roll-out of a new feature which impacted current automation run, and if it still doesn't work 2. You might need to update Selenium or Firefox Geckodriver/Chrome driver, please follow the [instructions here](#).



**There are 2 ways to access RedwoodHQ for multiple users:**

1. Open browser and navigate to IP address of the machine + port number where RedwoodHQ Server is installed on (eg: <http://192.1.23.113:3000>).
2. Where RedwoodHQ Server is installed there is a 'Open RedwoodHQ' shortcut created under RedwoodHQ Shortcut Folder (Start menu).
3. More info in [Quick Start 4](#) guide.

# Quick Start 2: First Test Case with Modified Action

---

## Quick Start/Tutorial 2 Video

Here is a **(5 minute)** video that goes through all the steps listed bellow:



## Identify New Test Case

Lets create our own Amazon test case and make it more interesting by modifying existing action step:

1. Open Browser and navigate to [www.amazon.com](http://www.amazon.com)
2. **(modify action)** Select 'Books' for search department drop down and search for 'Dune paperback'
3. Select first item that appeared by index
4. Add the item to cart
5. Verify item was added to cart

## Modify Action

We already have an amazon action created called 'Search Amazon' but what is missing is selecting a department from drop down next to the text field before clicking on Go. Since this department drop down selection logic belongs to search event we'll be modifying 'Search Amazon' action instead of creating a new one.

1. Click on Actions tab
2. In left pane double click on 'Search Amazon' action, which can be found in multiple ways:
  - Type in word 'search' in a text field above drop down with value 'ALL' to filter out
  - On bottom of left pane click on 'Actions Tree' and expand 'amazon' or 'search' tag to see the action under them
3. Add a new parameter to opened 'Search Amazon' action in the right pane:
  - Click on 'Add Parameter' button
  - Type in 'department' for name field and click on Update button

4. Because we are doing a simple selection from a drop down, we are going to use 'Select Item In Element' selenium action that comes with RedwoodHQ:
  - From left pane find 'Select Item In Element' action (it's under selenium tag)
  - Left click and drag and drop 'Select Item In Element' action between two 'Send Keys to Element' actions on the right pane
5. Now we need to point the 'Select Item In Element' action against the object it will work with and we'll use 'Looking Glass' tool to help us out with that:
  - Click on 'Looking Glass' button ( white square with red circle)
  - In a Looking Glass window select the browser you want to use and click on Open button
  - In a new browser navigate to [www.amazon.com](http://www.amazon.com)
  - In a Looking Glass window select the looking glass button, do a mouse over the word 'All' next to search text field in amazon website and left click
  - Copy the xpath that appeared (which should look like this: `//*[@id='searchDropDownBox']` ) and close Looking Glass window
6. Paste the xpath value you copied from Looking Glass in to 'ID' field for 'Select Item In Element' action
7. Click on 'ID Type' field, click on drop down arrow for it and select 'XPath'
8. Now you need specify a value that will be used this 'Select Item In Element' action, since users will be specifying this as a parameter value you should click on 'Visible Text' field and from drop down select parameter variable `${ department }` (this means it will use this variable as a value and since it's a parameter it will take that value from user input from a test case)
9. Since selecting department should be the first thing you do, click on the up green arrow for 'Select Item In Element' to move it up and make it as a first steps that this action executes

10. Click on Save and you should see your action looking like this:

**Action Details**

Name: Search Amazon

Description:

Action Type: ☐ Script ☒ Action Collection

Status: Automated

Tags: amazon search

**Add Parameter**

Name	Parameter Type	Possible Values
search for	String	
department	String	

**Action Collection**

Action Name	Execution Flow	Parameter Name	Parameter Value	Return Value	Machine Role
1 Select Item in Element	Record Error Stop Test Case	ID	//*[@id= 'searchDropDownBox']		Default
		ID Type	XPath		
		Visible Text	\$(department)		
		Value	<NULL>		
		Index	<NULL>		
2 Send Keys to Element	Record Error Stop Test Case	ID	twotabsearchtextbox		Default
		ID Type	ID		
		Key	<NULL>		
		Text	\$(search for)		
3 Click on Element	Record Error Stop Test Case	ID	nav-submit-input		Default
		ID Type	Class Name		

## Create New Test Case

Now we have all the actions we need to make our test case automated.

1. Click on Test Cases tab
2. Click on 'New Test Case' button
3. Type in 'Add Dune to Cart' for name field
4. Keep 'Action Collection' radio button selected
5. For 'Tags' field, type in 'amazon' and press enter
6. In the left pane expand 'Actions Tree' and drag and drop the following actions in to right pane one by one to generate the logical flow of our test case
  - 'Open Browser' (selenium tag)
  - 'Search Amazon' (amazon tag)
  - 'Select Item' (amazon tag)
  - 'Add to Cart' (amazon tag)
  - 'Verify Item Added to Cart' (amazon tag)

7. Alternatively you could have began typing the name of action in 'Search for action' text field in the right pane 'Action Collection' part, select it with and click on 'Add Action' green button to the left to add that action to the bottom of the list.
8. Now lets set the values for the actions. You can type in the value you want and press enter which will move cursor to the next field and next action.
  - 'Open Browser':  
URL = www.amazon.com  
Browser Type = select \${ Browser } from drop down arrow
  - 'Search Amazon'  
search for = Children of Dune  
department = Books
  - 'Select Item'  
Item Index = 0
9. Now that we are happy with our test case, for 'Status' drop down (bellow Description text field) select value 'Automated'
10. Click on Save button and you should see your test case looking something like this:

**Add Dune to Cart**

Test Case Details

Name: Add Dune to Cart

Description: Segoe UI

Status: Automated

Tags: amazon

Test Case Type: ☐ Script ☒ Action Collection

After State:

Action Collection

Action Name	Execution Flow	Parameter Name	Parameter Value	Return Value	Machine Role
1 Open Browser	Record Error Stop Test Case	URL	www.amazon.com		Default
		Browser Type	\${Browser}		
2 Search Amazon	Record Error Stop Test Case	search for	Dune paperback		Default
		department	Books		
3 Select Item	Record Error Stop Test Case	Item Index	0		Default
4 Add to Cart	Record Error Stop Test Case				Default
5 Verify Item Added to Cart	Record Error Stop Test Case				Default

## Modify Test Set

Test case is ready for execution so lets add it to existing test set.

1. Click on Execution tab
2. In left pane click on 'Test Sets'
3. Click on Edit button for 'Amazon Shopping' test set (or double click on it)
4. Expand 'amazon' tag and check the checkbox for 'Add Dune to cart' (or uncheck and then check 'amazon' check box which will now check all test cases which belong to that tag)
5. Click on Save and click on Yes in message box

## Execute

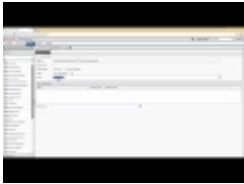
1. In left pane click on 'Executions'
2. Under 'All Executions' click on 'Amazon Shopping' link
3. Expand 'Set Variables' by clicking on little triangle button next to name
4. For 'Browser' under 'Set Variables' select the browser you want to run the test cases against by clicking on the Value field for it (which should have 'Firefox' by default) and click on drop down arrow to specify other browser (Internet Explorer or Chrome)
5. Under 'Select Machines' check the check box for '127.0.0.1' (agent was installed with RedwoodHQ server)
6. Under 'Test Cases' check the check box for our new test case 'Add Dune to Cart'
7. Click on 'Run Selected Execution' green arrow button on top
8. Test case will now run and then when it's done the test case will have status of 'Finished' with result 'Passed'

# Quick Start 3: Scripted Action, Clone Test Case and a new Execution

---

## Quick Start/Tutorial 3 Video

Here is a (**5 minute**) video that goes through all the steps listed below:



## Identify Another New Test Case

Lets generate another new test case where we will have to put a scripted Action to achieve our testing objective.

1. Open Browser and navigate to [www.amazon.com](http://www.amazon.com)
2. Search for 'Star Wars blu ray'
3. (**new action**) Check the '1970 – 1979' check box for 'Movie & TV Show Release Decade'
4. Select first item that appeared by index
5. Add the item to cart
6. Verify item was added to cart

## Add New Action

In order to work with 'check box' controls for 'Movie & TV Show Release Decade' we'll need to create a new action. The actual controls in amazon are not check boxes (they are links that render differently based on click) and because of that this is a good opportunity to show how to create a fully scripted action.

**First, lets create a script for our new action:**

1. Click on Scripts tab
2. Expand 'src' folder in left pane
3. Right click on 'actions' folder and select New -> Java Action
4. Type in 'CheckMovieTVReleaseDecade.java' and click OK



5. Type in the code to handle the check box part of amazon page (copy/paste the code example bellow to replace the template code that appears):

```
package actions;
import actions.selenium.Browser;
import org.openqa.selenium.*;
import org.openqa.selenium.support.ui.Select;
import java.util.*;
public class CheckMovieTVReleaseDecade {
    public void run(HashMap<String, Object> params) {
        WebElement element =
Browser.Driver.findElement(By.xpath("//*[@id='leftNavContainer']/SPAN[text()='Date range value']+\"\"[/../../INPUT]));
        if ((Boolean)params.get("Check") &&
(element.getAttribute("value").contains("false"))){
            element.click();
        }
        else if (!(Boolean)params.get("Check") &&
element.getAttribute("value").contains("true")){
            element.click();
        }
        try{Thread.sleep(4000);}catch(InterruptedException e){}
    }
}
```

6. Click on Build Scripts button (grey button with downward arrow) to make sure it compiles correctly and there are no errors
7. Click on Save and then **click on Git Push** (computer with green arrow button) and then OK to push the code we just created to main branch. We will discuss more about Git in [Quick Start 4](#).

✿ Because the checkbox we see on amazon page is actually an image, what the above code does is checks whether image for a given date range that user provides is 'selected' or 'unselected' ('Looking Glass' was used to see property value of the object) and based on that it clicks on it. So if user specified Boolean value of 'true' for 'Check' parameter in a test case, the code will check whether 'Date range value' image is 'selected' or 'unselected' and clicks based on that.

**Now lets create an Action definition to be used by Test Case(s):**

1. Click on Actions tab
2. Click on New Action button
3. Type in 'Check/Uncheck the Movie & TV Show Release Decade checkbox' for name field
4. Keep 'Script' selected since this is going to be a scripted action
5. For 'Tags' field, type in 'amazon' and press enter
6. Add 2 parameters with the name exactly as they appear in the code we developed:
  - Name first one 'Check' and specify Parameter Type for it as 'Boolean'
  - Name second one 'Date range value'
7. For 'Select Script' field point to our class method we just created by typing in:
  - actions.CheckMovieTVReleaseDecade.run
  - Alternatively you could have used script picker (a looking glass icon button to the right of Select Script field) to select the method.
8. Now that we are happy with our action, for 'Status' drop down (bellow Description text field) select value 'Automated'
9. Click on Save button and you should see your new action looking like this:

The screenshot shows the configuration form for a new action named 'Check/Uncheck the Movie & TV Show Release Decade checkbox'. The form includes the following fields and sections:

- Name:** Check/Uncheck the Movie & TV Show Release Decade checkbox
- Description:** (Empty text field)
- Action Type:** ☒ Script ☐ Action Collection
- Status:** Automated (dropdown menu)
- Tags:** amazon (tag with close icon)
- Parameters Table:**

Name	Parameter Type	Possible Values	
Check	Boolean		
Date range value	String		
- Select Script:** CheckMovieTVReleaseDecade.run (with a script picker icon)

## Clone Test Case

Since this test case looks very similar to the 'Add Star Trek to Cart' lets clone it and add the new action step we just created.

1. Click on Test Cases tab
2. Double click on 'Add Star Trek to Cart' test case to open it
3. Click on Clone button (to the right of Delete one)
4. Type in name 'Add Star Wars to Cart' and click OK

5. In a left pane click on Actions or Actions Tree and locate our new action called 'Check/Uncheck the Movie & TV Show Release Decade checkbox'
6. Drag and drop our new action between Search Amazon and Select Item actions
7. Type in the following values for the new action:
  - Check = TRUE
  - Date range value = 1970 – 1979
8. For 'Search Amazon' action modify it to the following values:
  - search for = Star Wars Blu Ray
9. Since we are expecting to have a lot of test cases to be executed, lets make this more compatible with that idea by specifying 'After State' (expand top portion of 'Test Case Details' by clicking on little triangle button first) by typing in 'Close Current Window' in that field and selecting it when it appears and press Enter to add it to After State action collection (or just drag and drop it there). What will happen is whatever action specified in that field for that test case will get executed after test case has finished running (and in our case it will close current browser window so if we execute 100 test cases we wouldn't have 100 browsers opened at the end)
10. Click on Save and you should see your test case looking like this:

The screenshot displays the PrimaTest interface for configuring a test case. The top bar shows two tabs: 'Add Star Trek to Cart' and 'Add Star Wars to Cart' (active). Below the tabs, the 'Test Case Details' section is expanded, showing the following fields:

- Name:** Add Star Wars to Cart
- Description:** Segoe UI
- Status:** Automated
- Tags:** amazon
- Test Case Type:** Script (selected), Action Collection
- After State:** Close Current Window

Below the details, the 'Action Collection' section is expanded, showing a list of actions in a table. The table has columns for Action Name, Execution Flow, Parameter Name, Parameter Value, Return Value, and Machine Role. The actions are as follows:

Action Name	Execution Flow	Parameter Name	Parameter Value	Return Value	Machine Role
1 Open Browser	Record Error Stop Test Case	URL	www.amazon.com		Default
2 Search Amazon	Record Error Stop Test Case	search for	Star Wars Blu Ray		Default
		department	<NULL>		
3 Check/Uncheck the Movie & TV Show Release Decade checkbox	Record Error Stop Test Case	Check	TRUE		Default
		Date range value	1970 – 1979		
4 Select Item	Record Error Stop Test Case	Item Index	0		Default
5 Add to Cart	Record Error Stop Test Case				Default
6 Verify Item Added to Cart	Record Error Stop Test Case				Default

## New Execution

Lets create a new Execution and use the existing Test Set where you will specify a new test case that was just created.

1. Click on Execution tab
2. In left pane click on 'Test Sets'
3. Click on Edit button for 'Amazon Shopping' test set (or double click on it)
4. Expand 'amazon' tag and check the checkbox for 'Add Star Wars to Cart' (or uncheck and then check 'amazon' check box which will now check all test cases which belong to that tag)
5. Click on Save and click on Yes in message box
6. In left pane click on 'Executions'
7. Under 'All Executions' click on 'New Execution' button
8. Type in Name 'New Amazon Execution' and for Test Set set it to 'Amazon Shopping'
9. Expand 'Settings' by clicking on little triangle button next to name
10. Check the checkbox for 'No After State' so that browser wont close for this run and we can see it open after test is done for now (for official test case runs you want this unchecked)
11. Click on Save button
12. Expand 'Set Variables' by clicking on little triangle button next to name
13. For 'Browser' under 'Set Variables' select the browser you want to run the test cases against by clicking on the Value field for it (which should have 'Firefox' by default) and click on drop down arrow to specify other browser (Internet Explorer or Chrome)
14. Under 'Select Machines' check the check box for '127.0.0.1' (agent was installed with RedwoodHQ server)
15. Under 'Test Cases' check the check box for our new test case 'Add Star Wars to Cart'
16. Click on 'Run Selected Execution' green arrow button on top
17. Test case will now run and then when it's done the test case will have status of 'Finished' with result 'Passed'

# Quick Start 4: Multi-User Usage, Git and Execution Analysis



For this exercise it is assumed that [Quick Start 3](#) was done.







## RedwoodHQ is a Server

Since RedwoodHQ is a **multi-user** test automation framework **server** it is assumed that multiple users will be able to login and work in it. As mentioned before, you should have only **1** RedwoodHQ server up and running while everyone connects to that IP address through their browsers (so no need to install anything for other users). Don't be afraid for your data if this is just your test environment and you are planning to move to a more permanent one later on, just follow these [Migration](#) steps when you are ready to do so (if you are planning on keeping anything you create).

## Add New User

In order for another person to use RedwoodHQ let's create a new user first:

1. Click on Settings tab
2. Click on Add User button
3. Type in the following values:
  - User ID = test
  - First/Last Name = Test User
  - Email = test@test.com
  - Password = test
  - Repeat Password = test
4. Click on Submit button and you will see user 'test' appearing with the 'Status' red next to it (because that user is not logged in) like this:

Add User   Search: <input type="text"/>				
Status	User ID	First/Last Name	Tags	
	admin	Administrator		 
	test	Test User		 

## Login as a New User

Now that new user is created, try to login with on either same machine as a current one or from a different machine altogether to simulate other user accessing RedwoodHQ.

**If you just want to login on the same machine as where you are currently logged in as user admin then do this:**

1. In top right hand corner click on word 'admin' and click on Logout
2. Click on Yes on prompt
3. In the Login page type in:
  - Username = test
  - Password = test
4. Click on Login button and you will get logged in as a test user

**If you want to login from a different machine then do this:**

1. Open browser and type in the ip address of RedwoodHQ server machine + port number (typically 3000) that was specified during install (eg: <http://192.168.44.554:3000>). If you don't know the IP address then just follow [these steps](#) on RedwoodHQ server machine
2. In the Login page type in:
  - Username = test
  - Password = test
3. Click on Login button and you will get logged in as a test user

## Code and Git Repository

Each user has their own code they work with that is regulated by internal Git repository, so if my user test does any type of modification to the code and doesn't 'push it' then other users will not see his changes. This is done so when a user works on their code, they are not concerned that they will interfere with other users code and executions and only push when they feel confident to do so. User also has an option when to Pull the code from main branch that was pushed by other users. Here is the scenario of how this works:

1. As a logged in user 'test', click on Scripts tab
2. Expand 'src' and 'actions' folders in left pane and double click on 'CheckMovieTVReleaseDecade.java' file
3. Modify the code by adding a print statement and making xpath of element invalid and adding a new java import to handle println, so your code should look like this:

```

package actions;
import static java.lang.System.out;
import actions.selenium.Browser;
import org.openqa.selenium.*;
import org.openqa.selenium.support.ui.Select;
import java.util.*;
public class CheckMovieTVReleaseDecade {
    public void run(HashMap<String, Object> params) {
        out.println("This is a hello from user test!");
        WebElement element =
Browser.Driver.findElement(By.xpath("//invalidxpath//img[@alt=\""+params.get("Date
range value")+"\""]));
        if((Boolean)params.get("Check") &&
(element.getAttribute("src").contains("checkbox_unselected"))){
            element.click();
        }
        else if (!(Boolean)params.get("Check")&&
element.getAttribute("src").contains("checkbox_selected")){
            element.click();
        }
        try{Thread.sleep(4000);}catch(InterruptedException e){}
    }
}

```

4. Click on Build Scripts button (which will automatically save the script) and **DO NOT** click on Git Push yet
5. Go to Execution 'New Amazon Execution' and run the 'Add Star Wars to Cart' test case (if you are not sure how, follow these steps in [Quick Start 3](#))

## Execution Analysis

Because we have purposely modified the script to fail by specifying invalid xpath, we see that the test case has failed and now it's a good opportunity to see how we can analyze the results of it.

1. While still in 'New Amazon Execution' click on 'Add Star Wars to Cart' link under Test Cases section to open Test Details for it
2. In this section we can see all the actions that have passed, failed or not ran and their appropriate values

3. Since 'Check/Uncheck the Movie & TV Show Release Decade checkbox' action has failed, it has expanded and contains screen shot, error and a stack trace
4. Bellow Results there is a Logs section that shows all the print statements that were made by your code, in our case you should see 'This is a hello from user test!' print statement
5. Click on 'View' button to have new tab open to show you a screenshot at which point the action has failed
6. Close the screenshot tab and click on stack trace link  
'actions.selenium.CheckMovieTVReleaseDecade.run(CheckMovieTVReleaseDecade.java:10)' which now goes to exact line failure on Scripts tab so you can fix it accordingly

## Different User Different Result

Since user 'test' did not 'push' his code in, the other user 'admin' will not be affected by it. The only way user admin can get this code if user 'test' does a 'Push' and then user 'admin' does a 'Pull'. Lets see this in action now.

1. Login as a user 'admin'
2. Go to Execution 'New Amazon Execution'
3. You can see in that execution that the test case 'Add Star Wars to Cart' has failed because user 'test' has ran it with his code
4. Now run this same 'Add Star Wars to Cart' test case and observe the results
5. The test case has passed because user 'admin' doesn't have the code which user 'test' does
6. Click on 'Add Star Wars to Cart' link in execution to open Test Details
7. All action steps are shown as Passed and we see that that under Logs there is nothing (because the print statement is also in users 'test' code)
8. Login as a user 'test'
9. Open 'CheckMovieTVReleaseDecade.java' modify the xpath code to be valid again but keep the print statement so your code will look like this:

```
package actions;
import static java.lang.System.out;
import actions.selenium.Browser;
import org.openqa.selenium.*;
import org.openqa.selenium.support.ui.Select;
import java.util.*;
public class CheckMovieTVReleaseDecade {
    public void run(HashMap<String, Object> params) {
        out.println("This is a hello from user test!");
        WebElement element =
```



```
Browser.Driver.findElement(By.xpath("//img[@alt=\""+params.get("Date
range value")+"\""]));
    if((Boolean)params.get("Check") &&
(element.getAttribute("src").contains("checkbox_unselected"))){
        element.click();
    }
    else if (!(Boolean)params.get("Check")&&
element.getAttribute("src").contains("checkbox_selected")){
        element.click();
    }
    try{Thread.sleep(4000);}catch(InterruptedException e){}
}
}
```

10. Click on Build Scripts button and then on Git Push button
11. Login as a user 'admin'
12. Go to Scripts tab and open CheckMovieTVReleaseDecade.java
13. Observe that print statement is not in the script yet
14. Click on Git Pull button (computer with red arrow)
15. Now the print statement that user 'test' pushed appeared and any future executions by user 'admin' will print that statement in the log when this action code is used

# Quick Start 5: RedwoodHQ Agent

---

## What is RedwoodHQ Agent for?

RedwoodHQ Agent is primarily used for execution of test automation and it is compatible with any Windows, Linux and Mac OS X. Once the agent is installed on the machine, it will self register on RedwoodHQ server and users will be able to select it under 'Machines' option for execution. Also, if a user wants to use optional 'Looking Glass' feature then they would need to install Agent on their own machine as well. Here are the steps to install the agent.

1. Go to the computer where you want to install the agent on (not RedwoodHQ Server one since it already has Agent installed)
2. Open browser and type in the ip address + port number of machine where RedwoodHQ is installed on (eg: <http://192.1.23.113:3000>)
3. Click on 'Download Agent' button and save the agent setup file
4. Double click on the setup file and click on Next on installer screen until Agent Port part
5. On the installer Agent Port where it's asking you for 'RedwoodHQ Server IP/Host Name', enter just the IP address of where the RedwoodHQ server is installed on and click on Next buttons till installer begins to install the files
6. After installer finishes, navigate to RedwoodHQ and login
7. Click on Execution tab and click on 'Machines' in left pane
8. You should see a new machine with IP address of where you installed it on
9. Click on edit (pencil) button for the machine ip where you just installed agent on and you can specify the Tag and Description for it to identify it:
  - In Description type in 'This machine is for Regression Execution Only'
  - In the Tags field type in 'Regression' and press Enter
  - Click on Update button
10. Go to 'Amazon Shopping' execution and run 'Add Star Trek to Cart' test case against a new Machine by selecting it in 'Machines' section



For instructions to install **Linux** or **Mac (OS X)** agent please go here: [Agent Install](#)

# Quick Start 6: Scripted Test Case, Multithreaded Execution and New Project

---

## Scripted Test Case

You can bypass the action driven approach completely and just have a test case that points directly at a script. We'll use 'Looking Glass' record playback feature to quickly create a very simple test case for us but otherwise the code can be as simple or as complex as you want using Java/Groovy/Python/C# language and other libraries (more info on setting your own Jar(s) can be found [here](#)).

### Create Script

1. Click on Scripts tab
2. Expand 'src' folder in left pane
3. Right click on 'src' folder and select New -> Folder
4. Type in 'TestCases' and click OK
5. Now right click on 'TestCases' folder and select New -> Java Action
6. Type in 'VerifyIndianaJonesLinkBluRay.java' and click OK
7. Click on 'Looking Glass' button ( white square with red circle)
8. Select Browsers -> and click on a browser you want to use for recording (eg Chrome) and click on Open button
9. Click on 'Code' tab in Looking Glass
10. Once the browser opens click on Record button (white square with red circle) and do the following steps:
  - In a new browser navigate to [www.amazon.com](http://www.amazon.com)
  - Type in 'Indiana Jones Blu Ray' in search text field
  - Click on Go button
  - Click on Record button again in Looking Glass to stop the recording
11. Now that we have this simple test code, lets make it more interesting by validating that text of the first link is the one we are looking for:
  - Click on the Inspector tab
  - Click on looking glass button to get object properties
  - Do mouse over the first link that appears (which is: 'Indiana Jones: The Complete Adventures') and click on it
  - Now copy the Xpath value we have for this element
  - Click on Code tab and using Xpath value we have put an assert statement to validate that elements text value, so the final code in Looking Glass will look like this:

```

driver.get("http://www.amazon.com/");
driver.findElement(By.xpath("//*[@id='twotabsearchtextbox']")).clear();
driver.findElement(By.xpath("//*[@id='twotabsearchtextbox']")).sendKeys("indiana
jones blu ray");
driver.findElement(By.xpath("//*[@id='nav-search']/FORM[1]/DIV[2]/DIV[1]/INPUT[1]"));
org.junit.Assert.assertEquals(driver.findElement(By.xpath("//*[@id='result_0']/DIV[1]
Jones and the Temple of Doom"));

```

12. Click on Run button (green arrow) in Looking Glass to do a playback of the code you have and it should go through without any problems or errors
13. Now that we have a working code we can copy paste it in to our VerifyIndianaJonesLinkBluRay.java test case in RedwoodHQ under run method and also add code to open the browser to work with (Firefox in this example) and set timeout to 30 seconds for elements, so the entire thing will look like this:

```

package TestCases;
import java.util.*;
import org.openqa.selenium.*;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
class VerifyIndianaJonesLinkBluRay{
    public void run(HashMap<String, Object> params){
        WebDriver driver = new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(30,
java.util.concurrent.TimeUnit.SECONDS);
        driver.get("http://www.amazon.com/");

        driver.findElement(By.xpath("//*[@id='twotabsearchtextbox']")).clear();

        driver.findElement(By.xpath("//*[@id='twotabsearchtextbox']")).sendKeys("indiana
jones blu ray");

        driver.findElement(By.xpath("//*[@id='nav-search']/FORM[1]/DIV[2]/DIV[1]/INPUT[1]"));

        org.junit.Assert.assertEquals(driver.findElement(By.xpath("//*[@id='result_0']/DIV[1]
Jones and the Temple of Doom"));
    }
}

```

14. Click on Build button and it should compile successfully

✿ If you want to do web based automation you don't have to create your own scripts, you can still use the RedwoodHQ Selenium pre-made actions like the ones that have been used in 'Add Star Trek to Cart' sample test case. *You can also mix and match code and actions for a test case which are utilizing different libraries: REST/SOAP, CLI, Selenium for Web, etc.*

## Create Test Case

1. Click on Test Cases tab
2. Click on 'New Test Case' button
3. Type in the following values for the new test case:
  - Name = Verify Indiana Jones Blu Ray search link text
  - Status = Automated
  - Test Case Type = Script
  - Select Script = TestCases.VerifyIndianaJonesLinkBluRay.run
4. Click on Save button and your test case should look like this:

The screenshot shows the 'Test Case Details' form for a test case named 'Verify Indiana Jones Blu Ray search link text'. The form includes the following fields and options:

- Name:** Verify Indiana Jones Blu Ray search link text
- Description:** A text area with a rich text editor toolbar above it, showing the text 'Segoe UI'.
- Status:** Automated (selected from a dropdown menu)
- Tags:** An empty text field.
- Test Case Type:** Script (selected with a radio button), Action Collection (unselected with a radio button)
- Select Script:** TestCases.VerifyIndianaJonesLinkBluRay.run (text input with a search icon on the right)

## Update Test Set

1. Click on Execution tab
2. In the left pane click on 'Test Sets'

3. Open Amazon Shopping test set, add a new test case to it and save it

## Multithreaded Execution

Selenium supports multi browser execution on the same machine so lets update one of our machines to be able to handle multiple threads and then execute our new scripted test case.

### Increase Thread Count

1. Click on Execution tab
2. In the left pane click on 'Machines'
3. Click on 'Edit' (pencil) button for '127.0.0.1' to edit this machine values
4. For 'Max Threads' field type in value of '2' and click on 'Update'
5. Now this machine can have 2 parallel test cases executed on it and you can easily specify the thread count even higher if you think the machine can handle it (based on type of test cases you execute)



Multithreading is also a very useful for things like **API testing** where each thread has a low memory/cpu cost and it's encouraged to run these test cases in parallel to fully utilize the machine resources.

### Execute Parallel Test Cases

1. In left pane click on 'Executions'
2. Under 'All Executions' click on 'Amazon Shopping' link
3. Under 'Select Machines' check the check box for '127.0.0.1' (agent was installed with RedwoodHQ server)
4. For 'Threads' column for the machine type in value of '2' (so we'll use 2 out of maximum threads of 2)
5. Under 'Test Cases' check the check box for our new test case 'Verify Indiana Jones Blu Ray search link text' AND check the existing one 'Add Star Trek to Cart'
6. Click on 'Run Selected Execution' green arrow button on top
7. Test case will now run in parallel and then when it's done both test cases will have status of 'Finished' with result 'Passed'

## New Project

RedwoodHQ can have multiple projects to work with and each of them would have it's own set of test cases and scripts. This allows for users to create an official project to work with (rather than a Sample one) and/or have different projects for different teams/applications to work in.

1. Click on Settings tab
2. In the left pane click on 'Projects'
3. Click on 'Add Project' button
4. Type in Project Name as 'Sample 2' and for Project Template select 'Selenium' (if you select Default then an empty project with no samples or selenium libraries will be created) and click OK
5. **Wait for 5 minutes** so back-end can generate appropriate project data
6. In a top right corner select 'Sample 2' from 'Choose Project' drop down and click on Yes
7. You are now in a new project where any Execution/Test Cases/Script changes will be specific to this project alone and not impact other ones

# Quick Start 7: Importing TestNG/JUnit and Execution from Script

---

## Importing TestNG/JUnit

If you have existing TestNG/JUnit you can now import them and use RedwoodHQ for their execution and results tracking. This can be a very useful feature since the framework allows for execution and results tracking of ANY Java/GROOVY code (API, REST/SOAP, etc.) or tool (Selenium, Appium, etc.).

### Upload TestNG/JUnit Files

Before Importing test cases we first need to upload all the script files which contain all the code. In this example we'll be using **sample TestNG test cases** which can be downloaded here: [AmazonTests](#) After you download the sample zip file make sure to unzip it in to any folder you want afterwards.

1. Click on Scripts tab
2. Expand 'src' folder in left pane
3. Right click on 'src' folder, select New -> Upload Directories
4. On a new popup screen navigate to folder where you have unzipped sample test cases to
5. Left click on 'AmazonTests' folder and click on Open button
6. Click 'OK' button for 'Files have been uploaded.' message box
7. All files and folders are now uploaded in to RedwoodHQ
8. Click on 'Git Push' button so that all users will be able to use these scripts

✿ If you are using any additional jars which are needed to execute the automated test cases (something that doesn't come by default with java and/or Selenium) then you just upload them to 'External Libraries' portion of Scripts (as individual jars or entire folder of them).

### Import TestNG/JUnit Test Cases

Now that you have uploaded your script files we need to make sure that Test Case entries exist for each @Test method so they can be executed as part of Execution.

1. Click on 'Import Test Cases' button
2. On a 'Select Test Cases to Import' popup check the check-boxes for all tests and click on OK
3. Click 'OK' button for 'Imported 3 test cases.'



4. Click on Test Cases tab
5. You will now see all the test cases that were imported from newly uploaded files with names generated based on this pattern (which can be modified after import): `<Package>.<ClassName>.<MethodName>`
6. Create a new test set and execution to run the new test cases

## Execution from Script page

TestNG and JUnit tests can also be executed right in the Scripts IDE page that will run the script right on current client machine (make sure you have Agent installed on it). This can be very useful for creating/doing modifications to a script and then running a quick validation of it without running through Execution.

1. Click on Scripts tab
2. Expand 'src' folder in left pane
3. Expand 'AmazonTests' folder
4. Double click on 'ProductSearching.java'
5. Click on 'Run TestNG/JUnit Test Case in opened script' button (green play arrow)
6. Check the check-box for 'AmazonTests.ProductSearching.LordofTheRingsSearchSelenium' test (only one can be selected at a time)
7. Click on 'OK' button
8. Test case runs through and the following result appears on the bottom 'Output' pane:  
 Starting Test...  
 [TestNG] Running: Command line suite  
 ===== Command line suite Total tests run: 1,  
 Failures: 0, Skips: 0 =====  
 Test Passed



Another way to point at a TestNG/JUnit test case without doing an Import is to create a new Test Case in 'Test Cases' tab, select appropriate Test Case Type (TestNG or JUnit) and point at the @Test method you want using script picker.

# F.A.Q.

---

## I have issues with Selenium not working in RedwoodHQ, what do I do?

There is a documentation section that deals with some general [Selenium tips](#).

## Does every user needs to install RedwoodHQ in order to use it?

**No**, you just need to have **ONE** RedwoodHQ Server installed, all other users just open their browsers (IE, Chrome, Firefox, Safari) on any OS and point to the IP address of that computer + RedwoodHQ port number (which is 3000 by default) **without** any additional installation required.

**For example:** 173.22.344.12:3000

## Why do I need RedwoodHQ Agent installed?

RedwoodHQ Agent is used for two things:

1. You need to install the Agent on the machine where you are planning to execute any automation from RedwoodHQ.
2. Agent is also used by feature called 'Looking Glass' (in RedwoodHQ) .

## I want to migrate all RedwoodHQ data from one machine (eg: My personal PC/ Laptop) to another (eg: Which will now be official RedwoodHQ Server), how do I do that?

It's a very easy process, just follow these [Migration](#) steps and you should be good to go in no time.

## Can I really just upload any Java jar/Python/C# and binary I want and use it for my code like I would on any other IDE?

Yep, just follow [these steps](#).

## I have more questions, where do I get answers?

To get community support of the product or just discuss RedwoodHQ and test automation in general visit [Google groups forum](#).

# Documentation

---

## Documentation

This documentation provides a detailed description and examples of all features and how they function.

If you haven't done so already then please go through [Quick Start Guide](#) since it's the best way to start and get to know main features of RedwoodHQ.

If you still have questions, please do not hesitate to visit our [Forum](#).

## Main Topics

- [Selenium](#)  
Information on how Selenium works in RedwoodHQ
- [Actions](#)  
Creation and usage of actions in Action Driven test automation development.
- [Test Cases](#)  
Creation and usage of Test Cases as Scripts, TestNG/JUnit or part of Action Driven development.
- [Action Collection](#)  
How to build up a collection of actions and specify various values to drive their execution.
- [Scripts](#)  
Code development, working with various Jars/Automation Tools, Git source control, Importing TestNG/JUnit tests and executing right in a Scripts page.
- [Execution](#)  
Executing test cases, analyzing their results and comparing between different executions.
- [Test Sets](#)  
Defining test sets to be used by executions.
- [Variables](#)  
Setting variables to be used in Action Collections or Scripts.

- [Machines](#)  
Configuring and setting up machines for Execution.
- [Continuous Integration](#)  
The explanation of how Jenkins/Team City runs RedwoodHQ automatically.
- [Settings](#)  
Setting up Users, Projects and Email.
- [Install Guide](#)  
Details of installing and updating RedwoodHQ Server and Agent.
- [Migration](#)  
Migrating all of RedwoodHQ data and code from one machine to another one.

# Selenium

---

## RedwoodHQ and Selenium

RedwoodHQ can work with almost any test automation tool (Selenium, Appium, Silk, etc.) and code (Java, Groovy, Python, C#). All these tools can be added via Jars, Binaries or Ivy dependency: [Jars/Binaries](#) and [Ivy](#)

In this part of documentation we will go over some of the most common questions related to **Selenium** implementation in RedwoodHQ.

### Updating to different version of Selenium

Since version 2.5, RedwoodHQ Java/Selenium project is configured to use Selenium version 3.2 and all the binaries associated with that version. As browser versions progress forward so does Selenium in which case you might be required to update it in order for automation to run successfully.

1. Identify to which version of Selenium you want to update. You can view latest version in official [Selenium website](#)
2. Go to Scripts and on left pane open Ivy.xml (to configure dependency to the new version)
3. Update all 'selenium-java' entries versions to equal to the one you need. For example if you need to configure to version 3.3 you will have the following dependencies:  
"org.seleniumhq.selenium" name="selenium-java" rev="3.3.0"  
"org.seleniumhq.selenium" name="selenium-server" rev="3.3.0"  
"org.seleniumhq.selenium" name="selenium-support" rev="3.3.0"  
Or if you need to always get latest version of selenium then just have 'rev' equal to "latest.release"
4. Do a build to make sure latest dependency works and then push your changes so that other users will have them as well
5. If you also need to **update the binaries** which are used by Selenium (like Chrome driver) then you can go to [Selenium website](#) and scroll down to see what binaries are there that is required
6. Download the binaries you need
7. Go to Scripts and on left pane expand 'bin' directory
8. Delete the binary you want to replace (for example chromedriver.exe)
9. Right click on 'bin' directory, New -> Upload and select the driver you are replacing with (eg: chromedriver.exe)

## Setting up Selenium to point at a as specific version of the Browser to test on

One of the things that is problematic for test automation is when Browsers update. It can cause a whole range of issues from user requiring to update to latest selenium version and drivers to something simply failing. A simple example is a case with Chrome 57 where .maximize() method (or any browser resizing) started to give exceptions which required the browser start up to be maximized by default. In a nutshell, it is always good to control the update of Browser when you need to, rather than when they auto-update. Here are the two ways to do it:

### 1. Stop Autoupdate

One of the more simpler ways is to stop auto-update for the browsers. That way you will always have the version that you currently have on your computer/agent box and nothing new will all of a sudden appear to wreck your day.

Firefox: <https://www.technipages.com/enable-disable-automatic-updates-in-firefox>

Chrome (more difficult): <http://www.wikihow.com/Completely-Disable-Google-Chrome-Update>

The only problem with this solution is that if someone else will start using automation (or you spin up another agent) then they will by default get whatever browser they currently have on it which could be a version you are not supporting. That's why a more preferable is the second solution:

### 2. Point at a specific version of a Browser

You can point Selenium at a specific binary of a browser to start up which can be a version that you need. That way you will always point at a specific version of the browser and can even deploy other specific versions to test your automation code with. Here is Chrome as an example.

- Download and install older version of Chrome (which will work in parallel with your current Chrome). There is no official website that does that unfortunately so I had to do it from this website: <https://www.slimjet.com/chrome/google-chrome-old-version.php>
- Modify/Add ChromeOptions to use that binary instead of default one (actions\selenium\Browser.groovy), here is a code example (assuming that the version of the chrome was 56 and deployed to autobrowsers directory under C: or root if mac):

```
DesiredCapabilities capabilities = DesiredCapabilities.chrome()
ChromeOptions options = new ChromeOptions()
if(os.contains("mac")){options.setBinary("/autobrowsers/Chrome.app/
Contents/MacOS/Google Chrome")}
else{options.setBinary("c:\\autobrowsers\\chrome64_56.0.2924.87\\chrome.exe")}
capabilities.setCapability(ChromeOptions.CAPABILITY, options)
Driver = new RemoteWebDriver(service.getUrl(),capabilities)
```

- Run the automation and it will now use this binary instead of your currently installed version of chrome

## Selenium Helper Actions in RedwoodHQ

RedwoodHQ comes (by default) with Selenium Java helper scripts and actions. They are purely optional and are used to get you going with selenium from day one. User has a complete control over that code and can always modify it to suite their needs. The same actions can be called out in Test Cases/Action Collections or even by other Java/Groovy code.

RedwoodHQ does not include helper selenium code in C# or Python but all selenium code in these languages will work identically like in a stand alone IDE. So the tutorials for selenium in these languages should work without any problems (examples: [Python Selenium](#) and [C# Selenium](#))

# Actions

---

## What is an Action?

Action is a keyword that describes what event it is about to do. A collection of actions is used by a test case (unless test case is pointing directly at a script) where parameter values can be specified in order to drive that action event. This in turn makes creation of automated test cases easier since they are more readable and easier to maintain.

Typically an Action is an event or activity performed in the tested product that has business value, for example, adding an item to the basket in an online store. In other frameworks this is similar to 'Step definition' (eg: Cucumber) or 'Keywords' (eg: Robot Framework).

Action Driven approach facilitates parallel work on creating automated test cases. Stakeholders of testing can create actions that users will perform in the tested product. Then, engineers can write scripts to automate these actions and, at the same time create test cases including them as well as not yet automated actions.

An Action can either point directly at a script that will get executed (and pass the values which were specified in a test case) or be an action collection and utilize Action Within Action approach.

## Topics

- [New, Delete, Clone and Tabs for Action](#)
- [Action Details and Selecting Script](#)
- [Parameters](#)
- [Action Within Action, using action collection](#)
- [Actions Left Pane](#)
- [Action Collection Part of the Screen](#)
- [Action History](#)



# New, Delete, Clone and Tabs for Action

## New, Delete, Clone



### New and Save

Clicking on 'New Action' button will open a new [New Action] tab. Click on 'Save' button to save it.

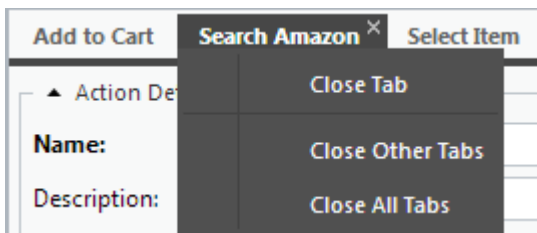
### Delete

To delete an action you have to open it first and then click on Delete button which will prompt you with 'Delete Confirmation' message.

### Clone

To clone an action and all of its values you have to open it first and then click on Clone button which will open to specify new Action name and click OK.

## Tabs



User can open multiple action tabs and easily navigate between them by clicking on their names or moving them around by holding left click and moving left or right.

Right clicking on any tab opens up the menu to:

*Close Tab* will close the tab that was right clicked on.

*Close Other Tabs* will close all other tabs except for this one.

*Close All Tabs* will close all tabs including the one that was right clicked on.



Going to a different application area (eg: Execution) will **NOT** close any open tabs or unsaved work.

# Action Details and Selecting Script

The screenshot shows the 'Action Details' form with the following fields:

- Name:** Search Amazon
- Description:** Sets a value in to 'Search' text field and clicks on Go button
- Action Type:** ☐ Script ☒ Action Collection
- Status:** Automated (dropdown menu)
- Tags:** amazon search

## Name

**(required)** It is recommended to keep your name very readable, concise and for anyone to be able to understand what this action is trying to accomplish.

## Description

A description should explain in more details what action will do especially if there are multiple steps inside the action that are trying to accomplish an automated task. Description text will also show up on the Left Pane when user will do a mouse over the action name.

## Action Type

Selects the type of action this is going to be:

**Script** – Selected by default and means that this Action will pass parameter values to a script code that will be executed.

**Action Collection** – This means that Action will execute other in it in order to accomplish a task. This is typically done in order to use secondary actions (like clicks, types, etc.) in order to accomplish action objective. More information how setup Action Collection with actions can be found [here](#).

## Status

This will determine the current state of an action and whether the test case that uses it should be executed or not.

**To be Automated** – Action definition or the code is not yet done, it can still be used in test case creation but if you try to run the execution that points to the test case that uses this action then RedwoodHQ will report an error saying that action is not in Automated state.


**Automated** – This Action is fully ready to be used in execution so the test case that uses it will run.

**Needs Maintenance** – Similar to ‘To be Automated’ but what it means is that action worked before and now no longer does so it requires maintenance and if you try to run the execution that points to the test case that uses this action then RedwoodHQ will report an error saying that action is not in Automated state.

## Tags

You can provide one or more Tags to associate action (typically to a functional area) by typing the name (no space) and pressing Enter key. This will also mean that this action can be found more easier by search as well as Action Tree tab on Left Pane where it will appear under the tags it belongs to.

## Select Script



If Action is a type of ‘Script’ then Select Script field will appear bellow Action Details. You tie Action to a method of a class that will be used by execution by specifying full path to it:

`<package>.<class>.<method>` (for example: `actions.general.Wait.run`) There are 2 ways of pointing to a method:







### Manually Entering

You can enter a full path manually by simply typing in there.

### Script Picker

By clicking on looking glass button of Select Script you can navigate to the method that you want to be used. By clicking on the up arrow button to the right of ‘Select Script’ one will go to Scripts tab and open that script.

# Parameters

+ Add Parameter			
Name	Parameter Type	Possible Values	
My string parameter	String	First possible value,2nd possible	 
My boolean parameter	Boolean		 
My array of string parameter	Array of String		 

Parameters allows user to specify values when creating Test Cases (or Action Collections in Actions as described more [here](#)) which will then be passed to the action code for execution. For more information on how to specify values for parameter in Action Collection and various parameter types please go [here](#).

## Add Parameters

Clicking on 'Add Parameter' button will add a new line to Parameters grid where you can specify:

### Name

Any name for the parameter. Make sure to keep the name readable so that test case designer user would understand what kind of values they need/can specify in order for this action to work properly.

### Parameter Type

*String* (can specify any type of value)

*Boolean* (TRUE or FALSE only)

*Array of String* (multiple string values)

### Possible Values

You can help user to choose values from a drop down in Action Collection for a parameter by specifying them as possible values. Enter any string value you want and press enter and if you want to give more values then set the cursor to the right of previous value and repeat the process. Possible Values can **only be used by String types**.

## Edit and Delete Parameter

### Edit

Click on Pencil button for parameter you want to edit or double click on it. Specify whatever value is needed and then click on Update button or Cancel if you don't want to use the changes.

**Delete**

Click on Delete button for parameter you want to permanently remove from this action.

# Action Within Action

## Action Collection in Action

When 'Action Collection' type is selected for the action what that means is that it will execute the action steps which are added to its action collection rather than specific script. This powerful feature allows to create another abstract layer to avoid step duplication and make sure test case that uses that action remains readable (more details on how to work in Action Collection screen can be found [here](#)). The parameters that are created for this action are used as variables that can be passed as data values to other actions which are in action collection. Bellow are some of the recommended type of actions within actions that can be created.

## Examples on how to best utilize this feature:

### Using multiple key-worded actions to make a single readable one

▲ Action Details

**Name:**

Search Amazon

**Description:**

**Action Type:**

☐ Script ☒ Action Collection

**Status:**

Automated ▼

**Tags:**

amazon × search ×

⊕ Add Parameter

Name	Parameter Type	Possible Values
search for	String	

▲ Action Collection

⊕ Search for Action using name or tags ▼

	Action Name	Execution Flow	Parameter Name	Parameter Value	Return Value	Machine Role	
▲ 1	<a href="#">Send Keys to Element</a>	Record Error Stop Test Case	ID	twotabsearchtextbox		Default	↑ ↓ ×
			ID Type	ID			
			Key	<NULL>			
			Text	\${search for}			
▲ 2	<a href="#">Click on Element</a>	Record Error Stop Test Case	ID	nav-submit-input		Default	↑ ↓ ×
			ID Type	Class Name			

We want test case designer to be able to do a search on amazon page by having an action that types in a value and clicks on Go button and we want make sure this action is easy and fast to use.

- Since the only input from user is to specify a text for the search field we have created only one string type parameter 'search for'.
- We have added two actions that use Selenium code: 'Send Keys to Element' (to set the text for Search text field) and 'Click on Element' (to click on Go button)
- Using Looking Glass we ID and ID Types that the controls use
- For 'Text' parameter I have put the name of the parameter as a variable (a value that will be entered on a test case level) `${ search for }`
- Now a test case designer can use this new 'Search Amazon' action on a test case level making it instantly readable and without worrying what code is being used to do clicking and text typing and for what attributes

Add Star Trek to Cart

Test Case Details

Action Collection

Search for Action using name or tags

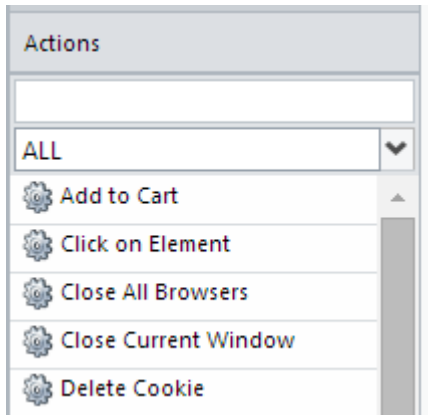
	Action Name	Execution Flow	Parameter Name	Parameter Value
1	<a href="#">Open Browser</a>	Record Error Stop Test Case	URL	www.amazon.com
			Browser Type	\${Browser}
2	<a href="#">Search Amazon</a>	Record Error Stop Test Case	search for	Star Trek Blu Ray

### Multiple Actions accomplishing specific objective

# Actions Left Pane

---

## Actions



Left Pane allows user to see, search and access Actions and to be used by Action Collections.

### Opening Action

Double clicking on the action will open it on the right pane.

### Search

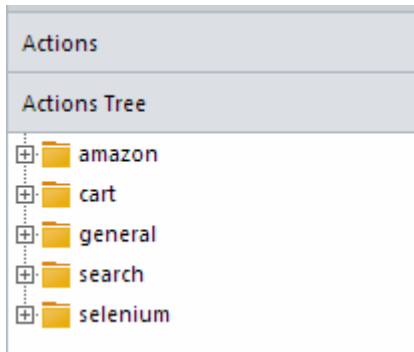
In this field you can type in action name or tag it belongs to and it will only display actions that match typed in value.

### Filter by Status

Selecting any other Status than 'ALL' will only show actions that have that value selected (eg: 'To be Automated') and additionally you can do a Search within it as well. This feature is useful for to see what Actions you need to work on or are in a working state.



## Actions Tree







Clicking on 'Actions Tree' on Left Pane will bring up actions by Tags which they belong to. An action can have multiple Tags so it can appear multiple times.

# Action History

---

## Action History

▲ History			
Date ▼	UserID	Previous Version	
01/13 08:28:13	Dmitri	<a href="#">View Action</a>	
01/13 08:27:38	Denis	<a href="#">View Action</a>	
01/13 08:27:23	Denis	<a href="#">View Action</a>	
01/13 08:27:09	Denis	<a href="#">View Action</a>	

User is able to not only view the history of the action as it was modified but also be able to roll back to the previous version of it.

### Viewing Previous Version

Clicking on 'View Action' link for 'Previous Version' column will open a new tab that will show how the action looked like when it was modified on that date by the user. No changes can be done in that new tab and it is only for view purposes only.

### Rolling Back to Previous Version

Clicking on green arrow icon to the version user wants to roll back to would prompt a confirmation message and then roll back to that specific version of the action.

# Test Cases

---

## What is a Test Case?

A test case is an object which gets executed by an Execution. It can utilize an Action Driven approach and use action collection to create the steps to execute or can be pointed directly at a test case script (either class method or TestNG/JUnit @Test).

## Topics

- [New, Delete, Clone and Tabs for Test Cases](#)
- [Test Case Details, Selecting Script, TestNG/JUnit and After State](#)
- [Test Cases Left Pane](#)
- [Action Collection Part of the Screen](#)
- [Test Case History](#)
- [Data Driven Test Case Data](#)

# New, Delete, Clone and Tabs for Test Cases

## New, Delete, Clone



### New and Save

Clicking on 'New Test Case' button will open a new [New Test Case] tab. Click on 'Save' button to save it.

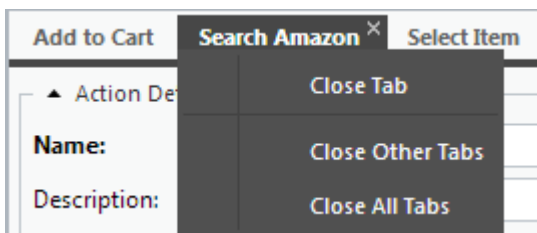
### Delete

To delete a Test Case you have to open it first and then click on Delete button which will prompt you with 'Delete Confirmation' message.

### Clone

To clone a Test Case and all of it's values you have to open it first and then click on Clone button which will open to specify new Test Case name and click OK.

## Tabs



User can open multiple action tabs and easily navigate between them by clicking on their names or moving them around by holding left click and moving left or right.

Right clicking on any tab opens up the menu to:

*Close Tab* will close the tab that was right clicked on.

*Close Other Tabs* will close all other tabs except for this one.

*Close All Tabs* will close all tabs including the one that was right clicked on.



Going to a different application area (eg: Execution) will **NOT** close any open tabs or unsaved work.

# Test Case Details, Selecting Script, TestNG/JUnit and After State

---

**Test Case Details**

**Name:** Add Star Trek to Cart

**Description:** Goes to Amazon, searches for Star Trek, goes to details and add's it to cart.

**Status:** Automated

**Tags:** amazon

**Test Case Type:** ☐ Junit ☐ TestNG ☐ Script ☒ Action Collection

**After State:** Close Current Window

## Name

**(required)** A name of a test case should not only be able to be readable but should also align with the naming convention you have established in your team.

## Description

A description should explain in more details what test case will do especially if test case is scripted to avoid forcing others trying to parse the script.

## Status

This will determine the current state of a test case and whether it should be executed or not.

**To be Automated** – Test Case action steps or the code is not yet done, if you try to run the execution that points to this test case then RedwoodHQ will report an error saying that test case is not in Automated state.

**Automated** – This Test Case is fully ready to be used in execution.

**Needs Maintenance** – Similar to 'To be Automated' but what it means is that Test Case worked before and

now no longer does so it requires maintenance and if you try to run the execution then RedwoodHQ will report an error saying that test case is not in Automated state.

## Tags

You can provide one or more Tags to associate test case (typically to a functional area) by typing the name (no space) and pressing Enter key. This will also mean that this test case can be found more easier by search as well as Test Case Tree tab on Left Pane where it will appear under the tags it belongs to.

## Test Case Type

Selects the type of test case this is going to be:

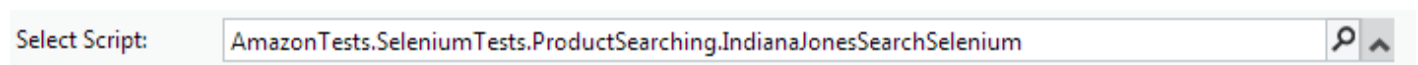
**Script** – Means that this Test Case will directly execute a class method that its script picker is pointing at (bypassing Action Driven framework).

**Action Collection** – Selected by default and means that Test Case will execute actions in the order they are put in Action Collection part of the screen. More information how setup Action Collection with actions can be found [here](#).

**TestNG** – Test Case will execute TestNG Test method (and all the TestNG logic that goes with it) that its script picker is pointing at (bypassing Action Driven framework).

**Junit** – Test Case will execute Junit Test method (and all the Junit logic that goes with it) that its script picker is pointing at (bypassing Action Driven framework).

## Select Script



If Test Case is a type of 'Script/TestNG/Junit' then Select Script field will appear bellow Action Details. You tie Test Case to a method of a class that will be used by execution by specifying full path to it:

`<package>.<class>.<method>` (for example: `actions.general.Wait.run`) There are 2 ways of pointing to a method:

### Manually Entering

You can enter a full path manually by simply typing in there.

### Script Picker

By clicking on looking glass button of Select Script you can navigate to the method that you want to be used. By clicking on the up arrow button to the right of 'Select Script' one will go to Scripts tab and open that script.

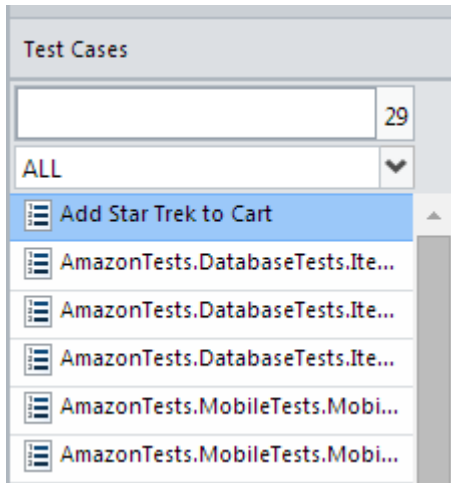
## After State

If you want to execute an action after the test case finishes in execution (doesn't matter if it passes or fails) then you can specify it in an After State (eg: Closing browser after test is done so it would not interfere with the next test case). After State is an action collection where you can specify multiple actions to be executed after Test Case finishes (more information on Action Collection can be found [here](#)).

# Test Cases Left Pane

---

## Test Cases



Left Pane allows user to see, search and access Test Cases.

### Opening Test Case

Double clicking on the test case will open it on the right pane.

### Search

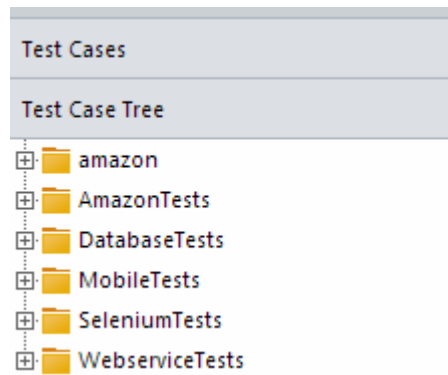
In this field you can type in test case name or tag it belongs to and it will only display test case that match typed in value. To the right of the Search field there is a number of how many test cases are currently being displayed (so if no search is done then it means that this number represents ALL test cases in the system right now).

### Filter by Status

Selecting any other Status than 'ALL' will only show test cases that have that value selected (eg: 'To be Automated') and additionally you can do a Search within it as well. This feature is useful for to see what Test Cases you need to work on or are in a working state.



## Test Case Tree



Clicking on 'Test Case Tree' on Left Pane will bring up test cases by Tags which they belong to. A test case can have multiple Tags so it can appear multiple times.





## Actions/Actions Tree

You can also see and navigate through actions on left pane for Test Cases tab so you can use them for [Action Collection](#). Go [here](#) to get more information how to navigate left pane actions, otherwise the only difference is that double clicking on them will *NOT* open them (since we are in a Test Cases tab).

# Test Case History

---

## Test Case History

▲ History			
Date ▼	UserID	Previous Version	
01/13 08:28:13	Dmitri	<a href="#">View Action</a>	
01/13 08:27:38	Denis	<a href="#">View Action</a>	
01/13 08:27:23	Denis	<a href="#">View Action</a>	
01/13 08:27:09	Denis	<a href="#">View Action</a>	

User is able to not only view the history of the test case as it was modified but also be able to roll back to the previous version of it.

### Viewing Previous Version

Clicking on 'View Test Case' link for 'Previous Version' column will open a new tab that will show how the test case looked like when it was modified on that date by the user. No changes can be done in that new tab and it is only for view purposes only.

### Rolling Back to Previous Version

Clicking on green arrow icon to the version user wants to roll back to would prompt a confirmation message and then roll back to that specific version of the test case.

# Data Driven Test Case Data

## What is Data Driven test case and how is it used in automation

Data-driven test case iterates through the same steps but changes specific values each time. A simple example of that is a failed Login test case which tests for various ways user should have a valid error message on failed attempts. Using that example we would have 3 actions: Open Browser, Login and Verify Error Message. Since these 3 steps are always the same the only thing that changes for them is the data to cause various type of Login attempts, and that is when data driven component comes in to play. Instead of creating a new test case for each type of failure condition user would specify a different data for the step (eg: Login with username but not password, Login with invalid username, Login with valid username but invalid password, etc.) and this test case will iterate through all of it (effectively becoming multiple test cases).

The screenshot displays the PrimaTest interface. The top section, 'Action Collection', lists five actions: 'Open Browser', 'Search Amazon', 'Select Item', 'Add to Cart', and 'Verify Item Added to Cart'. Each action has a 'Record Error Stop Test Case' checkbox and a 'Parameter Value' field. The 'Open Browser' action has parameters for 'URL' (www.amazon.com) and 'Browser Type' (\$Browser). The 'Search Amazon' action has a 'search for' parameter with value \$(TCData.Movies). The 'Select Item' action has an 'Item Index' parameter with value \$(TCData.Select.Item). The bottom section, 'Test Case Data', is a table with columns 'Movies' and 'Select Item'. It contains three rows of data: 'Star Wars blu' (1), 'Star Trek blu' (1), and 'Dune blu' (3). Each row has a red 'X' icon in the rightmost column.

Action Name	Execution Flow	Parameter Name	Parameter Value
1 Open Browser	Record Error Stop Test Case	URL	www.amazon.com
		Browser Type	\$(Browser)
2 Search Amazon	Record Error Stop Test Case	search for	\$(TCData.Movies)
3 Select Item	Record Error Stop Test Case	Item Index	\$(TCData.Select.Item)
4 Add to Cart	Record Error Stop Test Case		
5 Verify Item Added to Cart	Record Error Stop Test Case		

Movies	Select Item
1 Star Wars blu	1
2 Star Trek blu	1
3 Dune blu	3

## Test Case Data

Opening Test Case and scrolling below Action Collection user will see a section called 'Test Case Data'. Clicking to expand it will bring options to make this test cases in to a data driven one.

### Adding Columns and Rows

**Add Column** – Will add a column which will be used as a data point reference in a Action Collection portion of a test case.

**Add Row** – Adds a Row. Each row will be an iteration for the test case execution.

### Entering Value for a Row

In each String field user can specify a value they would like to be used for a column for that specific iteration. Each iteration is a horizontal line of values that will be used together in that run. In the screenshot example a first iteration will execute with a search value of 'Star Wars blu' and it will select item by index as '1'. Second iteration will use 'Star Trek blu' and '1' and third iteration will be 'Dune blu' and '3'.

### Specifying Test Case Data value in a Action Collection

To point test case Action to use a value from Test Case Data you would need to point at it similarly to specifying variable value but calling out TCDATA class first and then the name of the Column (eg: \${TCDATA.Name of column}). That value can be entered manually or be selected as a drop down value (which gets auto-generated after creating a Column). Using an above screenshot example we have created a column name 'Movies' which we want to use in our Search Amazon action, so for parameter 'search for' we have specified \${TCDATA.Movies} and now it will use the values from that column.

### Deleting Rows and Columns

Delete Row – Click on the red delete button next to a row that user wants to delete.

Delete Column – Right click on column name user wants to delete and select Delete Column.

### Data Driven Test Case in Execution

Every time ANY type of modification is made in Test Case Data user is required to click on Data Refresh button in Execution in order to use new values (even if execution was closed and opened again). Please go [here](#) to see how Data Driven test case is used in execution.

# Action Collection

Action Collection							
Search for Action using name or tags							
	Action Name	Execution Flow	Parameter Name	Parameter Value	Return Value	Machine Role	
1	<a href="#">Open Browser</a>	Record Error Stop Test Case	URL	www.amazon.com		Default	↑ ↓ ×
			Browser Type	\$(Browser)			
2	<a href="#">Search Amazon</a>	Record Error Stop Test Case	search for	Star Trek Blu Ray		Default	↑ ↓ ×
3	<a href="#">Select Item</a>	Record Error Stop Test Case	Item Index	0		Default	↑ ↓ ×
4	<a href="#">Add to Cart</a>	Record Error Stop Test Case				Default	↑ ↓ ×
5	<a href="#">Verify Item Added to Cart</a>	Record Error Stop Test Case				Default	↑ ↓ ×

## Action Collection

Action Collection is a collection of steps (actions) that user adds in order to create a flow to correspond to what they want automation to do. All steps in 'Action Collection' get executed one after the other in numbered order that they appear on the screen.

## Topics

- [Adding Actions to Collection](#)
- [Copy and Paste, Move, Remove](#)
- [Parameters in Action Collection](#)
- [Execution Flow](#)
- [Return Value](#)
- [Machine Role and Machine Variables](#)

# Adding Actions to Collection

---

## Left Pane

One way to add/insert actions to Action Collection is drag and dropping them through Left Pane.

### Adding

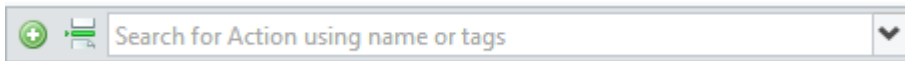
1. In Left Pane expand Actions/Actions Tree
2. Left click on the action you want to add/insert and hold the mouse button
3. Drag over to the Action Collection grey line area and release the button (the mouse pointer will turn in to green plus icon meaning you can drop the action there)

### Inserting

1. In Left Pane expand Actions/Actions Tree
2. Left click on the action you want to add/insert and hold the mouse button
3. Drag and drop over the action in Action Collection screen and it will be inserted below it (or over the grey line area between the actions)

## Typing in Action

Another way to add/insert actions is by typing and selecting that action in 'Search for Action using name or tags' text field which is located on top and bottom of Action Collection. You can type in the action name/tag, select it and add it via pressing enter or button.



### Adding

1. Left click on 'Search for Action using name or tags' text field in Action Collection
2. Begin typing in the name of the action you want to add or tag it belongs to
3. As soon as you begin typing a list will appear showing you the list of possible actions to add
4. Use Up/Down arrow to select action you want and press Enter or Left Click on it
5. Press Enter again or click on green 'add action' button (to the left of text field) to add an action to action collection

6. The text focus will automatically go to first parameter of that action so you can begin typing in values for them

## Inserting

1. Left click on the Action in 'Action Collection' to select it bellow which you want to add an action
2. Left click on 'Search for Action using name or tags' text field in Action Collection
3. Begin typing in the name of the action you want to add or tag it belongs to
4. As soon as you begin typing a list will appear showing you the list of possible actions to add
5. Use Up/Down arrow to select action you want and press Enter or Left Click on it
6. Click on white with green 'insert action' button (to the left of text field) to the left to insert an action bellow action that was selected in first step



Clicking on 'Action Name' link will open that action under Actions tab.

# Copy and Paste, Move, Remove

---

## Copy and Paste

It is possible to Copy and Paste actions with values in action collection inside the same Action/Test Case or even in to different ones.



1. In Action Collection screen select the actions you want to copy by left clicking on it
2. If you want select multiple actions to copy then just Ctrl + Left Click on any other action (like selecting multiple lines in Excel)
3. Click on white 'copy selected actions' button (to the right of ' Search for Action using name or tags' text field) or press Ctrl-C on keyboard
4. Left click on the action in 'Action Collection' screen (to select it) under which you want to paste the copied actions
5. Click on orange and white 'paste actions' button (to the right of ' Search for Action using name or tags' text field) or press Ctrl-V on keyboard
6. You can also Paste these copied actions with values to other Action/Test Case 'Action Collection' screen

## Move

You can move actions up or down in Action Collection screen.



1. In Action Collection screen click on the green Up arrow to the right of the action to move it up
2. In Action Collection screen click on the green Down arrow to the right of the action to move it down

## Removing an Action from Action Collection



1. Click on red button to the right of the action to remove it from Action Collection



# Parameters in Action Collection

---

## Entering Values for Parameters

Values is what get associated with parameters to be used by actions and which are defined in 'Parameter' section of Action screen (see details here: [Parameters](#)). There are 3 types of Parameters: String, Boolean and Array of String and here is how to enter values for them:

### String

1. You can type in any text value using any type and number of characters
2. Pressing Enter key will keep the value you entered in the field and go to next parameter value field
3. `<NULL>` value means the parameter will be returned as 'null' on the code level (not to be confused with empty value which is considered to be a valid one)
4. Clicking on Drop down arrow will display easy-to-enter values: a) Possible Values that can be selected (and that were defined on Action [parameters](#) level) b) Parameters (if it's an [Action Collection on Action level](#)) c) Variables (that are defined on [variables](#) page).
5. You can mix variable value plus text to combine the two: `<text value>${ variable }` (example: I'm using the following browser `${ Browser }`)

### Boolean

1. You can only select value from a drop down either as TRUE or FALSE
2. Pressing Enter key will keep the value you entered in the field and go to next parameter value field
3. `<NULL>` value means the parameter will be returned as 'null' on the code level
4. Boolean parameter will actually be defined as a boolean on a code level (example: `boolean bMyBooleanParameter`)

### Array of String

1. To enter values you will need to type in any string text you want and press enter, if you want to enter more values then set cursor to the right of first value entered, type in another value and press enter
2. Array of String stores multiple String values which are stored in array on a code level and retrieved appropriately (example: `aMyArrayParameter[0]`, `aMyArrayParameter[1]`)
3. This type is never null on code level but if there are no values specified then this Array of String is returned as empty of values

Parameter Name	Parameter Value
My string parameter	This is my string value which also has variable value at the end: \${Browser}
My boolean parameter	TRUE
My array of string parameter	string test 1 × string value 2 × 3rd value ×

## Retrieving Parameter Value in Code

Parameters are passed to actions as HashMaps which can be retrieved in a usual code manner:

```
params.get("Name of my parameter goes here")
```

That's why every action method should require a HashMap to be passed to it and the following example code using the above example values (in the image):

```
public class MyTestActionClass {  
    public void run(HashMap<String, Object> params) {  
        System.out.print(params.get("My string parameter"));  
        System.out.print(params.get("My boolean parameter"));  
        System.out.print(params.get("My array of string parameter")[0]);  
    }  
}
```

will print out:

This is my string value which also has variable value at the end: Firefox (**if firefox was used for execution**)  
true (**actual Boolean value of the parameter**)  
string test 1 (**since it's the first value of array of string**)

# Execution Flow

## Execution Flow

For each action in Action Collection you can define a flow to support standard, multiple point of verification and negative test case flows. This is a powerful feature which allows you to keep actions and their code as is but have them do different things simply by specifying a different flow value. To change to a different value you just need to click on 'Execution Flow' value (by default it's 'Record Error Stop Test Case') and select the one you need from a drop down.

### Record Error Stop Test Case

1. If this current action code does an assertion then fail the test case and DO NOT CONTINUE executing any other actions in this test case
2. This is a default flow that is typical for *positive* test cases and appears every time you add/insert an action in to Action Collection

*Example:*

Action Collection				
Search for Action using name or tags				
	Action Name	Execution Flow	Parameter Name	Parameter Value
1	<a href="#">Open Browser</a>	Record Error Stop Test Case	URL	www.amazon.com
			Browser Type	\$(Browser)
2	<a href="#">Search Amazon</a>	Record Error Stop Test Case	search for	Star Trek Blu Ray
3	<a href="#">Select Item</a>	Record Error Stop Test Case	Item Index	0

In the above example if action 'Search Amazon' fails and produces an assertion (for example because search field didn't appear so text couldn't be entered) then test case will fail and not proceed forward to the next actions (eg: Select Item)

### Ignore Error Continue Test Case

1. If this current action does an assertion then DO NOT FAIL a test case and proceed to next step as if nothing happened
2. This flow is typically used for *negative* test cases where you expect your action to fail while the next one validates if a valid error condition is met

*Example:*

Action Collection				
Search for Action using name or tags				
	Action Name	Execution Flow	Parameter Name	Parameter Value
1	Login to amazon	Ignore Error Continue Test Case		
			Username	bad username
			Password	bad password
2	Verify expected Error Message	Record Error Stop Test Case		
			Error message text	Invalid username and/or password

In the above example a test case expects that Login action should fail and give an assertion because we are putting a bad username and password (so RedwoodHQ will not report this test case as failed) and will go to our next action step 'Verify expected Error Message' which will validate if proper error message has appeared.

### Record Error Continue Test Case

1. If this current action code does an assertion then fail the test case but still continue to the next step
2. This one is typically used if you want to validate multiple things instead of just stopping on first verification error

*Example:*

Action Collection				
Search for Action using name or tags				
	Action Name	Execution Flow	Parameter Name	Parameter Value
1	<a href="#">Open Browser</a>	Record Error Stop Test Case	URL	www.amazon.com
			Browser Type	\${Browser}
2	<a href="#">Search Amazon</a>	Record Error Stop Test Case	search for	Star Trek Blu Ray
3	<a href="#">Verify search result</a>	Record Error Continue Test Case	For returned index	0
			This should be the name of the product	Star Trek the Final Frontier
4	<a href="#">Verify search result</a>	Record Error Continue Test Case	For returned index	1
			This should be the name of the product	Star Trek Wrath of Khan
5	<a href="#">Verify search result</a>	Record Error Stop Test Case	For returned index	2
			This should be the name of the product	Star Trek The Voyage Home

In the above example even if the step #3 does an assertion because it's not able to find 'Star Trek the Final Frontier' as the first returned index it will fail the test case but will still go to the next step to verify 'Star Trek Wrath of Khan' as second index and then to the next step afterwards. So this way it will be assured that all 3 verification's are done and show failures for specific actions when assertion happens (since it's possible that not all of them will fail).

# Return Value

## Return Value

If an action code returns a value (for example: `return "hello world";`) then it can be stored as a variable in Action Collection and be used by any other action in there. By specifying any variable name you want in 'Return Value' column (just click on 'Return Value' column cell and type in any value you want with NO spaces/special characters) for the action that returns it, you can then pass that value in to another action parameter field by specifying variable name like this (assuming variable name is named 'myVar'): `${myVar}`

### Example

Action Collection					
Search for Action using name or tags					
	Action Name	Execution Flow	Parameter Name	Parameter Value	Return Value
1	Generate random number	Record Error Stop Test Case			randomValue
			Number from	1	
			Number to	20	
2	Print out parameter value	Record Error Stop Test Case			
			Value to print	This is a random value that was generated: \${randomValue}	

In the above example action code 'Generate random number' returns a value and a 'Print out parameter value' will just do a print statement for whatever value has been specified in 'Value to print'. We have type in our own variable name 'randomValue' which stored the value from 'Generate random number' action and then we passed that variable to 'Value to print' parameter of 'Print out parameter value' action. So if we assume that value that was generated is '19' then we'll see the following statement printed out by 'Print out parameter value' action:

This is a random value that was generated: 19



Return Value is a powerful feature that can be used in other ways like: **a)** Using value that was generated by application (eg: some unique ID) to be used by other actions **b)** Transferring CLI process between one action to another to continue from where the previous one left off

# Machine Role and Machine Variables

## Machine Role

This feature allows you to have action logic to be executed across multiple agent machines all in ONE test case. When you add a new action to the Action Collection list it'll automatically have a 'Default' role and what that means that if under execution you select a machine which also has 'Default' role specified for it (more information on specifying machine roles can be found [here](#)) then this action will execute on that agent box (more information on machine role execution can be found [here](#)). If you point an action to another 'Machine Role' by clicking on the Role column value that is there and changing to another one then it will get executed against that box.

### Example

Action Collection						
	Action Name	Execution Flow	Parameter Name	Parameter Value	Return Value	Machine Role
1	<a href="#">Generate Database test data</a>	Record Error Stop Test Case	Type of data	Accounting		Server
2	<a href="#">Login to application</a>	Record Error Stop Test Case	Username	test		Default
			Password	test		
3	<a href="#">Open 'Accounting' report</a>	Record Error Stop Test Case				Default

In the above example the 'Generate Database test data' will be executed on the agent machine which has a role of 'Server' and the next action 'Login to application' will now get executed on a machine which has 'Default' role assigned to it. So this way in one test case interacts with both server side box as well as the client side one. In order to execute this test case you would need to check the check boxes for 2 machines under 'Select Machines' option in execution screen (since both of them will be used), like this:

Select Machines					
Search:					
<input checked="" type="checkbox"/>	Host Name/IP	Threads	Max Threads	Port	Roles
<input checked="" type="checkbox"/>	<a href="#">192.168.197.136</a>	1	1	5009	Server
<input checked="" type="checkbox"/>	<a href="#">192.3.44.12</a>	1	1	5009	Default

## Machine Variables

Each machine can have a set of variable values which are unique to it and be retrieved by actions. This can be very useful in order to provide a value that is unique to this box and so actions will be able to use it in test case execution (to set variable value for a machine read [here](#)). The variable format for this is: `${ Machine.<Role>.<Variable name> }` (for example: `${ Machine.Default.HostName }`)

### Example

Action Collection						
Search for Action using name or tags						
	Action Name	Execution Flow	Parameter Name	Parameter Value	Return Value	Machine Role
1	<a href="#">Generate Database test data</a>	Record Error Stop Test Case	Type of data	Accounting		Server
2	<a href="#">Open Browser</a>	Record Error Stop Test Case	URL	http://\${Machine.Server.IPAddress}:2000/login.html		Default
			Browser Type	Firefox		

In the above example our Server role box is the one that is going to be used by our client Default role box so we need to use its IP address. Since we defined it's IP address as a variable that can be retrieved during execution (and was defined on Machine Variables level shown in screenshot bellow) we can now use it by 'Open browser' action to navigate to the URL of our Server from Default (aka client) side.

**Machine Variables**

Add Variable

Name	Value
IPAddress	192.168.197.136

OK

Cancel



# Scripts

---

## Scripts Page

This is where code development for test automation is done. An embedded IDE with Git Source control allows users to fully develop and commit the code they want all under a browser interface.

## Topics

- [Automation Script](#)
- [Git Source Control](#)
- [Search/Replace](#)
- [Upload/Delete File/Folder](#)
- [Adding/Deleting Jar\(s\) and Binaries](#)
- [Importing TestNG/JUnit](#)
- [Execution from Script](#)
- [Ivy](#)

# Automation Script

---

## Create Script

Redwood HQ provides you with workspace for creating Java, Groovy, C# and Python automation scripts. There is no limitation on what code you can create as long as it follows the coding standards of language of choice. You can have multiple languages be in a single project and a single test case able to use actions which point to these different languages.

1. On the Scripts tab, in the Scripts pane, in the folder tree, click the src folder.
2. (optional) To place the new script in a subfolder inside the src folder:
3. Create a new folder – on the toolbar, click New, click Folder, and then, in the New Folder dialog box, enter the name for the folder and click OK. Click an existing subfolder.
4. On the toolbar, click New, click File, and then, in the New Folder dialog box, enter the name for the file and click OK.
5. In the script editor, type the script and, on the toolbar, click Save All.



The script should meet certain requirements. In case the action has parameters, they must be put in a hash map in the script.

The following snippets automate an action which has the string parameter 'print this'.

The first snippet is created in Java. Note that the name of the file should be the same as the class name, JavaPrintlnExample.java.

```
import java.util.*;
public class JavaPrintlnExample {
    public static void run(HashMap<String, Object> params) {
        System.out.println(params.get("print this"));
    }
}
```

The second snippet is a Groovy script:

```
public class GroovyPrintlnExample {
    public static void run(def params) {
        println(params."print this")
    }
}
```

## Compile/Build Scripts

Before execution, you must compile your scripts. RedwoodHQ includes the Apache Ant builder (build.xml).

1. On the Scripts tab, on the toolbar, click Build Scripts.
2. The stack trace appears in the Output pane on the bottom. Error messages include a link to the corresponding line in the script.

All your scripts are aggregated into a JAR file.

# Adding/Deleting Jar(s) and Binaries

---

## Adding Jar(s)

User can add a new Jar to library to be used by the scripts.

1. Click on Scripts tab
2. Right click on External Libraries in left pane
3. Find the jar you want to upload and click on it
4. Click on 'Open' button and this jar will now be uploaded and used by the scripts
5. Click on 'Build' button to validate new jar gets compiled
6. Click on 'Git Push' button if you want to push this new jar to other users

## Deleting Jar(s)

Here are the steps to delete the jar or to replace it by deleting first and then adding the new one you want to use.

1. Click on Scripts tab
2. Expand External Libraries in left pane
3. Right click on the jar file you want to remove
4. Click on Delete and click on Yes on confirmation
5. Click on 'Git Push' button if you want to push this new deletion change to other users

## Adding Binaries

User can also add any binaries that will be used by automation. The two binaries that already come with sample project are the Chromedriver and IE drivers for selenium but you can also add any other one to be used for your automation purposes. This is also where Geckodriver for Firefox or any other type of binaries used by your test automation tool.

1. Click on Scripts tab
2. Right click on Bin in left pane
3. Find the binary file you want to upload and click on it
4. Click on 'Open' button and this jar will now be uploaded and used by the scripts
5. Click on 'Git Push' button if you want to push this new file to other users

## Deleting Binaries

Here are the steps to delete the binaries or to replace it by deleting first and then adding the new one you want to use.

1. Click on Scripts tab
2. Expand Bin in left pane
3. Right click on the jar file you want to remove
4. Click on Delete and click on Yes on confirmation
5. Click on 'Git Push' button if you want to push this new deletion change to other users

# Git Source Control



Each user has their own code they work with that is regulated by internal Git repository, so if one user test does any type of modification to the code and doesn't 'push it' then other users will not see his changes. This is done so when a user works on their code, they are not concerned that they will interfere with other users code and executions and only push when they feel confident to do so. User also has an option when to Pull the code from main branch that was pushed by other users. at statement in the log when this action code is used.

## Pushing/Pulling code

- After user modifies the code they would need to **Push** it in order for other users to get it. After they click on Push button an option will appear where user can select the files they wish to push and provide commit notes to specify the changes they have made to trunk.
- If user wants to receive latest code changes from trunk they need to click on **Pull** button which will fetch all latest changes.

## Version Control

By expanding lower portion of Scripts screen you'll be able to select Version Control option and see all the previous commits that have been done to the currently opened file. Clicking on the icon on the last column to the right will open the comparison diff between your current file and the one for that version.

Version	Date ▾	Author	Commit Message	Exists in Master	
0c06cbb113740c814d58c58a4bccbf0...	Sun Jan 03 2016 12:49:46 GMT-0800 ...	John Doe	Update to work with latest sleep	<span style="color: green;">●</span>	
8062bb942ae9877b70c2a13aca07e2...	Tue Dec 29 2015 16:08:05 GMT-0800...	John Doe	new project	<span style="color: green;">●</span>	

Output
Version Control
Find

## Accessing Git directly

You can access Git interface directly by going to: \vendor\Git and running bash git utility from there. Now you can access any projects and users git repository in RedwoodHQ and be able to run any additional commands or resolve any potential issues. Location example of admin user git repository in sample project: \public\automationscripts\Sample\admin

## External Repository

You can setup the external repository (eg: Github) so that all the code will be committed there rather than RedwoodHQ internal one. Here is how to set it up:

1. Make sure all code is checked in (pushed) and pulled by all users in RedwoodHQ
2. Go to project settings, click on "Use External Repo" check box and type in full URL of the remote repo, click Submit
3. Click on user name in right top corner and select "SSH Key" menu item
4. Copy that SSH key and add it in your remote repo
5. Click on "Enable Remote Repo" and wait while your code is being pushed to remote repo
6. Your remote repo will now have a branch that corresponds to your user name. Use that branch to create master branch on remote repo (eg: Github).
7. Now all of your users will need to go through process of adding their SSH keys to the remote repo.
8. All code that is pushed will be pushed to your personal branch all code that is pulled will be pulled from master on remote repo, it is your responsibility to create

# Search/Replace

---

To find certain places in the script, you can use the search capability. You can search by:

- Text – on the toolbar above the script editor, in the find text box, enter text. If the text is case-sensitive, select the Case check box.
- Regular expression – on the toolbar above the script editor, in the find text box, enter a Perl-compatible regular expression and select the Regex check box.

Move up and down the search results using green arrows.

To make replacements in the script, besides specifying the search pattern, in the replace text box, enter the text to replace found passages. To replace one by one, click Replace, to replace all at once, click Replace All.

You can also do a search across multiple files:

- Right click on the folder you want to search and click on Find in Path
- Type in the value you want to search and click OK
- On the bottom panel you can see the list of links which will open the file and go to the line where the value was found



# Upload/Delete File/Folder

---

## Upload

You can upload any files and folders necessary for test execution (for example, a data file), script, or an external Java or Groovy library.

1. On the Scripts tab, in the Scripts pane, in the folder tree, depending on the kind of file you want to upload, right click and select New – Upload for a single file or Upload Directories for all files and folders under specified location:

- The bin folder – for any file necessary for executing tests (eg: chromedriver.exe or any other file)
- The src folder – for a script (eg: .java or .groovy)
- The External Library folder – for a Java or Groovy external library (eg: jars)

2. (optional) To place the file in a subfolder:

- Create a new folder – on the toolbar, click New, click Folder, and then, in the New Folder dialog box, enter the name for the folder and click OK.
- Click an existing subfolder.

3. On the toolbar, click New, click Upload, and then, in the File Upload dialog box, select the file you want to upload and click Open

## Delete

1. On the Scripts tab, in the Scripts pane, in the folder tree, click the file or folder you want to delete.
2. On the toolbar, click Delete.
3. In the Delete Confirmation dialog box, click Yes.

# Importing TestNG/JUnit

---

## Import TestNG/JUnit

You can import your existing TestNG/JUnit tests and then use RedwoodHQ for execution and further test automation development. Importing test cases is very simple:

1. Go to Scripts page and upload your TestNG/JUnit folders under 'src' folder (for more information how to upload directories go [here](#)).
2. Upload all the projects library jar files if necessary (for more information how to upload jars go [here](#)).
3. You can click on a Build button to validate that code is compiling correctly and then do a Git Push.
4. On top click on 'Import Test Cases' button and wait for RedwoodHQ to process all your files and determine all test cases in there.
5. A new popup will appear afterwards that will show you all test cases that were identified for import and their type (TESTNG or JUNIT).
6. Check all the check boxes for the test cases you want to import and click on OK.
7. Afterwards a success message will appear saying how many test cases were imported.
8. Click on Test Cases tab and you will now see all the test cases that were imported with names generated based on this pattern: <Package>.<Class Name>.<Method> (you can modify test case names anytime).
9. Test cases are now ready to be executed



If you see that some test cases were not imported or you manually want to add more, you can always create a test case and point at it yourself (for more information how to do that go [here](#)).

# Execution from Script

---

## Execution from Script page

TestNG and JUnit tests can also be executed right in the Scripts IDE page that will run the script right on current client machine (make sure you have Agent installed on it). This can be very useful for creating/doing modifications to a script and then running a quick validation of it without running through full Execution.

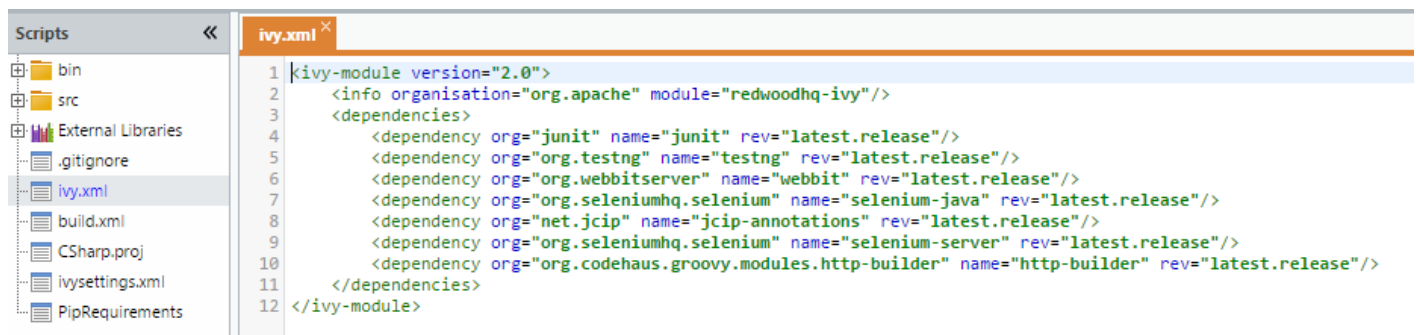
1. Click on Scripts tab and open a script file which contains TestNG/JUnit code.
2. Click on 'Run TestNG/JUnit Test Case in opened script' button (green play arrow).
3. Check the check-box for a test you want to execute right now (only one can be selected at a time).
4. Click on 'OK' button.
5. Test case runs through and results appears on the bottom 'Output' pane.

# Ivy

Apache Ivy is a dependency management tool which highly integrated with Apache Ant (used in RedwoodHQ to do builds).

## Setting Up Dependency

RedwoodHQ already comes with ivy.xml which can be modified in order to set dependency on whatever libraries your project needs to get everytime your user does a build. Here is an example of a pre-setup ivy.xml that gets created when setting up a sample project with Selenium and other dependencies:



Here are the steps to modify dependency that your project requires:

1. Go to Scripts
2. On the left pane double click on ivy.xml
3. Add a dependency to the xml and click on Save
4. Click on build and your user will now fetch a new dependency
5. Do a Git Push in order for other users to get latest ivy.xml with updated dependency

You can also modify actual ivy settings by opening ivysettings.xml

# Execution

---

## Execution

This is where you create an execution, run it and then analyze and compare results. [All Executions] lists all the possible executions that were created and can be managed while clicking on Execution name links or New Execution button brings 'Execution' screen to run the tests.

The *recommended* way of creating Execution is 1 per execution cycle and then lock it for historical results. A good example of that would be creating execution for Smoke Test against a specific build, run it and then lock it. After the new build comes in, you create a new Smoke Test execution for that new build, run that and lock it as well. That way you'll have two historical executions which you can then use 'Aggregate' report button to see their historical pass/fail rate for all test cases inside them.

## Topics

- [Details, Email, Settings, Variables, Totals and Export Results](#)
- [Select Machines](#)
- [Test Cases in Execution](#)
- [Execution Test Details](#)
- [Historical Comparison Results](#)
- [Data Driven Test Cases in Execution](#)

# Details, Email, Settings, Variables, Totals and Export Results

---

## Details

▲ Details

**Name:**

Amazon Shopping - 7.0.1 - Build #27

**Tags:**

regression ×


**Test Set:**

Amazon Shopping

**State:**

Ready To Run

Lock



### Name

**(required)** A given name to the execution. It is recommended to keep the name predictable and descriptive what it is testing (for example what version and build number of application this execution is testing).

### Tags

User can specify multiple tags by typing in a tag value and pressing enter or select them from drop down list. This will identify to which tag type the execution belongs to (for example that this is 'regression' execution).

### Test Set

**(required)** When creating an Execution you have to point at a Test Set which points at a set of test cases (see more details about Test Set creation [here](#)). Once you save a new Execution you can never switch to a different set.

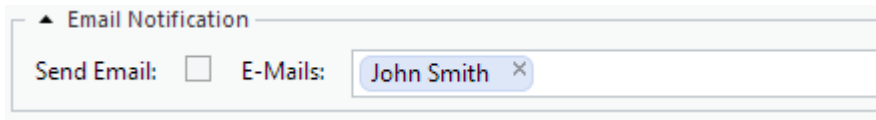
### State

There are two states: 'Ready To Run' (execution is ready to be executed) and 'Running' (execution is currently executing).

### Lock

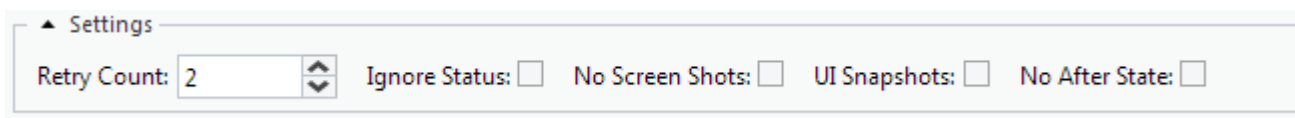
An un-locked execution (default state) means that Execution can be run but if user clicks on Lock icon and it means that running it is disabled. This is typically done for executions which are done and no longer need to be executed but we want to keep its results for historical purposes (since lock will also keep all of Test Cases frozen for this execution even if test set and test cases were modified down the line).

## Email Notification

A screenshot of the 'Email Notification' settings panel. It features a title bar with a triangle icon and the text 'Email Notification'. Below the title bar, there is a 'Send Email:' label followed by an unchecked checkbox. To the right of this is an 'E-Mails:' label followed by a text input field containing 'John Smith' and a small 'x' icon to its right.

You can select multiple users from Email Notification drop down who will receive an email after every time Execution finishes running if 'Send Email' check box is checked (for more information on setting up Email settings go [here](#)).

## Settings

A screenshot of the 'Settings' panel. It has a title bar with a triangle icon and the text 'Settings'. Below the title bar, there are four settings: 'Retry Count:' with a value of '2' and a spinner control; 'Ignore Status:' with an unchecked checkbox; 'No Screen Shots:' with an unchecked checkbox; 'UI Snapshots:' with an unchecked checkbox; and 'No After State:' with an unchecked checkbox.

### Retry Count

If a test case fails how many times should it be tried again. Specifying value of 2 means it will try test case up two more times before reporting failure, if it does Pass within these first tries then it will not try again and report passing result.

### Ignore Status

If a Test Case (or an Action that is being used by the Test Case) has Status as 'To be Automated'/'Needs Maintenance' then the execution will not run it but if you check 'Ignore Status' check box then it will run this test case no matter what its Status is.

### No Screen Shots

Every time a test case fails it will generate a screen shot for the action that has failed (which can be viewed in 'Test Details' tab), but if this check box is set then it will never generate screenshots (useful if running test cases which are not UI based like REST/SOAP, API, CLI, etc.).

### UI Snapshots

Checking this option will take a screen shot for every single action regardless if it has failed or not (which can be viewed in 'Test Details' tab). This is a useful feature to be able to analyze all the steps visually that led up to the failure but is *not recommended* to be used for all large executions since they can fill up RedwoodHQ server (but deleting execution will delete all screenshots on back-end as well).

### No After State

If set then do not execute After State if it's specified in a test case (for example After State closes the

browser but I want to keep it open for current test case execution after test is done to analyze potential issue).

## Set Variables

▲ Set Variables		
Name ▲	Value	Tags
Browser	Firefox	
TestEnvironmentURL	http://www.amazon.1.com	

If a variable was set to be an 'Execution Var' in Variables screen, then it will appear in this list here and execution will use the value that was set in this execution (for more information on setting up Variables go [here](#)). This is useful to setup values specific to this execution run like which Browser to use, URL of test application to use, etc.

### Name

Name of the variable as defined on Variables page.

### Value

The value that will be assigned to the variable for this execution. If Possible Values are specified for this variable then they can be selected via drop down.

### Tags

Variables can also be Taged for ease of search and selection.

## Execution Totals and Chart

▲ Execution Totals							
Total:	28	Passed:	13	Failed:	15	Not Run:	0
Run Time Sum:						0h:1m:6s	

Execution Totals and Execution Chart will show a current snapshot of the execution results and how it stands right now.

### Total

Total amount of test cases in this execution as defined in Test Set.



**Passed**

How many test cases have Passed the execution run.

**Failed**

How many test cases have Failed the execution run.

**Not Run**

How many test cases have not been executed yet OR are in 'Need Maintenance/To be Automated' status.

**Run Time Sum**

The total sum time of all test case 'Elapsed Time' column. This shows how much it would take if 1 machine with 1 thread it would've taken to execute so far (so can potentially be used as a multiplier to determine how much time it would take if more machines/threads are assigned for execution).

**Export Results**

User can easily export current execution results in to PDF file by clicking on 'View Results as PDF' button (located on top). After clicking on the button a PDF file will be downloaded that will list execution name, total's overview and details of each test case whether it passed or failed and with what error.

# Select Machines

## Select Machines

▲ Select Machines

Search:

<input type="checkbox"/>	Host Name/IP ▲	Threads	Max Threads	Port	Roles	State	Description	Base State Result	Machine Base State	Tags
<input type="checkbox"/>	<a href="#">192.168.197.1</a>	1	5	5009	Default		host name is:Server-1		Populate Database with Test Data	Development
<input type="checkbox"/>	<a href="#">192.168.197.128</a>	1	5	5009	Server,Default	Running 1 of 5	host name is:WIN-LIAGIMCPQOO			Regression
<input type="checkbox"/>	<a href="#">192.3.44.12</a>	1	1	5009	Default		host name is:Server-2			Regression

Here you can select against what machines you want your test cases to be executed against and how many threads should be used for that execution.

### Host Name/IP

An actual IP address or a valid Host name of the machine where RedwoodHQ agent resides which will be used for execution. Clicking on this link will open a VNC tab of this machine.

### Threads/Max Threads

You can specify multiple threads to be used by execution by putting a number which does not exceed the 'Max Threads' value which has been specified in Machines page for this particular box. Multiple threads is very useful where you want to run tests in parallel on a single box rather than just using 1 box per test case. Some of good example where multi-threaded execution can come in useful is API testing where there is no UI interface and typically machine can handle multiple parallel processes. Selenium also support multi-threaded execution on a single box and can be used as well, just make sure that you have valid After State selected to close the browser after it's done with it.

### Port

RedwoodHQ Agent port as has been defined during the installation (default port is 5009).

### Roles

Actions in action collection might have logic to be able to execute against multiple machines (eg: Server configuration on one while browser execution on another) and here you'll see to what role machine belongs to as defined on Machines page. So if a test case requires two roles (eg: Server and Default) then you would need to select 2 machines for its execution otherwise you'll get an error for these test cases saying that no machine with valid role was selected for it (to see how to define a Role for a Machine click [here](#)).

### State

How many threads out of how many available are currently being used right now by any execution. This is to show if this machine and amount of threads is currently available for execution.

**Description**

Description text as defined on Machines page for this machine.

**Machine Base State**

This field will contains the list of all Actions (you can type in there to do a search by name/tag) and once selected it will execute this Action only ones and then proceed to execute all selected test cases afterwards once it Passes (if it Fails then execution will stop and not execute test cases). This can be a useful option for things like preparing an environment for execution (eg: Installing/deploying specific build to environment which the test cases will use).

**Base State Result**

If Machine Base State action is selected then during execution run you will see the following links appear (clicking on the links will open Test Details page to see status and results of possible failure):

*Running* – Means that Machine Base State action is currently being executed

*Passed* – Means that Machine Base State action is done executed and passed

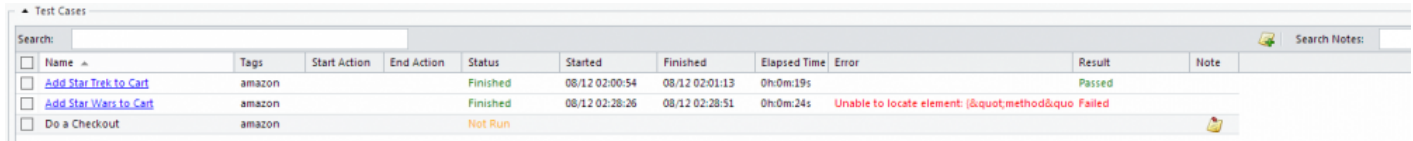
*Failed* – Means that Machine Base State action is done executed and failed (recommended to click on the link to see what cause a failure)

**Tags**

Tags which are assigned to the machine in Machine Page for ease of search.

# Test Cases in Execution

## Test Cases



<input type="checkbox"/>	Name	Tags	Start Action	End Action	Status	Started	Finished	Elapsed Time	Error	Result	Note
<input type="checkbox"/>	<a href="#">Add Star Trek to Cart</a>	amazon			Finished	08/12 02:00:54	08/12 02:01:13	0h:0m:19s		Passed	
<input type="checkbox"/>	<a href="#">Add Star Wars to Cart</a>	amazon			Finished	08/12 02:28:26	08/12 02:28:51	0h:0m:24s	Unable to locate element: (\"method\" Failed	Failed	
<input type="checkbox"/>	<a href="#">Do a Checkout</a>	amazon			Not Run						

This part of the page deals with two things: Selecting test cases for execution and seeing their current results. Checking the check box next to test case tells the execution that user wants to run this test case after they click on Run button (once the Run button is clicked the test cases will automatically get unchecked but they will be executed).

### Search

Entering text here will filter and show test cases which contain the Name or Tag value that is being searched for.

### Name

The name of a test case which will become a link to view Test Details to monitor it's current state as it's being run or analyze it's failure (for more information on Test Details page analysis please go [here](#)).

### Tags

Tag values which are defined for this test case.

### Start/End Action

This feature allows you to specify from and to which action step to run, allowing user to execute only specific steps in a test case rather than a whole thing. So if you specify 'Start Action' value to 5 and 'End Action' value to 12 it will then begin executing from step #5 and stops executing after it is done with step #12. Can be very useful for analyzing test case failures from specific step rather than wait for entire execution again (especially for big test cases).

### Status

Shows current state of a test case:

*Not Run* – This test case was not yet executed.

*Running* – The test case is currently being executed (clicking on this link will open VNC tab to the box where execution is happening)

*Finished* – Test Case finished it's execution.

**Started/Finished**

The Started time of a test case execution and it's Finished time.

**Elapsed Time**

How long did it take for this test case to execute.

**Error**

The error text that was outputted by coded assert statement.

**Result**

Shows current Result of a test case:

*Passed* – Means test case has passed the execution and no assertions were raised.

*Failed* – Means that test case has failed the execution and assertion was raised.

*Blank* – Test case was not executed yet.

**Note**

Clicking on this field will bring up a Note text field where you can specify a text that you want. For example it could be a bug # of a failed test case or an explanation of why test case was not run for this execution.

Doing a mouse over the note icon for a test case will show the text (or clicking on it will).

**Search Notes**

Entering text here will filter and show test cases which contain the text that is being searched for in its Note section (eg: Searching for word 'bug' or specific bug #).

# Execution Test Details

## Test Details for Action Collection Test Case

<b>Details</b> <b>Name:</b> <a href="#">Add Star Wars to Cart</a> <b>Status:</b> <span style="color: green;">Finished</span> <b>Result:</b> <span style="color: red;">Failed</span>							
Screen Shots							
Results							
Order	Action Name	Parameters		Status	Result	Error	Screen Shot
1	<a href="#">Open Browser</a>	URL	www.amazon.com	Finished	Passed		
		Browser Type	Firefox				
2	<a href="#">Search Amazon</a>	search for	Star Wars Blu Ray	Finished	Passed		
	<a href="#">Send Keys to Element</a>	ID	twotabsearchtextbox	Finished	Passed		
		ID Type	ID				
		Text	Star Wars Blu Ray				
	<a href="#">Click on Element</a>	ID	nav-submit-input	Finished	Passed		
		ID Type	Class Name				
3	<a href="#">Select Item</a>	Item Index	100	Finished	Failed		
	<a href="#">Click on Element</a>	ID	//*[@id="result_100"]/h3/a	Finished	Failed	Unable to locate element: ("method":"xpath","selector":"//*[@id="result_100"]/h3/a") Command duration or timeout: 10.21 seconds For documentation on this error, please visit: <a href="http://seleniumhq.org/exceptions/no_such_element.html">http://seleniumhq.org/exceptions/no_such_element.html</a> Build info:	<a href="#">View</a> <a href="#">actions.selenium.utils.Elements.findElement(q:org.openqa.selenium.By\$XPath\$Locator@281) Full Trace...</a>
		ID Type	XPath				

This page shows the details of a test case (which uses Action Collection) that was executed and it's current step results. This page can be viewed even during execution where you'll be able to see the current state of steps and Logs section update automatically without any reload.

## Details

Details for a given test case details.

### Name

A name of a test case that this details page references. Clicking on this link will go to Test Cases tab and open this test case.

### Status

Current status of this test case:

*Running* – Test Case is currently being executed.

*Finished* – Test Case finished execution.

**Result**

*Passed* – Means test case has passed the execution and no assertions were raised.

*Failed* – Means that test case has failed the execution and assertion was raised.

**Screen Shots**

Expanding this section will show the screenshot that was taken on failure OR the screenshot of a first step if 'UI Snapshots' check box was set under Settings for Execution. Clicking on the image will open a new tab to show image in full detail. Clicking on Next or Back will show screenshot of a next or previous step that was taken if 'UI Snapshots' option was set and will also highlight the action step it belongs to for Results section.

**Results**

This section shows all the Action steps which were executed and where the failure (if any) happened.

**Order**

Step number of an action that was executed.

**Action Name**

The name of the action that was executed for any given step. Clicking on this link will open Actions tab and this particular action. If the icon is a folder then it means this action is an Action Collection which can be further expanded to see what steps are executed inside it.

**Parameters**

Shows Parameters and the values that were used for this test case. If a value was a variable then it will show an actual variable value that was used at that moment rather than a variable name.

**Status**

*Finished* – This step finished its execution.

*Not Run* – This step was not yet run/not yet finished executing.

**Result**

*Passed* – This step passed the execution and no assertions were raised.

*Failed* – This step has failed the execution and assertion was raised.

*Blank* – This step was not yet executed/still running.

**Error**

The error text that was outputted by coded assert statement for this step.

**Screen Shot**

This screenshot link gets generated for a failed step to see a screenshot after action reports a failure (or for every step if 'UI Snapshots' option was checked) and opens a new tab to show image in full detail.

**Trace**

If test case has failed then it will generate a link (or multiple links) of trace where code assertion happened. Clicking on the link will open Scripts tab and the script and line number. If you want to see an entire trace of a failure then clicking on 'Full Trace' link will open a popup with that information.

**Logs**

Logs section provides an output of all print statements that have happened during this test case execution. User can do a search in there to quickly parse the logs or download them.

**Action Name**

Name of an Action (or test case if it was scripted) that generated the print statement.

**Date**

The date and time of when the print statement was generated.

**Message**

The actual print statement itself.





*Running* – Test Case is currently being executed.

*Finished* – Test Case finished execution.

**Result**

*Passed* – Means test case has passed the execution and no assertions were raised.

*Failed* – Means that test case has failed the execution and assertion was raised.

**Error**

The error text that was outputted by coded assert statement.

**Trace**

Full trace of a scripted failure and trace links clicking on which will open Scripts tab and the script and line number where failure has happened.

# Historical Comparison Results (Aggregate Report)

---

## Looking through Historical Results

Historical results in RedwoodHQ are checked when doing an aggregate report between multiple executions. Since each execution represents individual run (eg: Smoke Test Build #23) this way you can easily check how test cases behaved between these executions (eg: 'Smoke Test Build #23', 'Smoke Test Build #24' and 'Smoke Test Build #25').

In order to run Historical Results, check the check boxes next to executions you want to compare and click on 'Aggregate Report' button. The new screen will appear which will show how the executions compare to each other and below that will have individual test case Status as it did between various executions.

# Data Driven Test Cases in Execution

## Data Driven Test Cases in Execution

Test Cases								
Search:								
<input type="checkbox"/>	Name ^	Tags	Status	Started	Finished	Elapsed Time	Error	Result
<input type="checkbox"/>	<a href="#">Add Star Trek to Cart 1</a>	amazon	Finished	04/29 08:59:20	04/29 08:59:43	0h:0m:23s		Passed
<input type="checkbox"/>	<a href="#">Add Star Trek to Cart 2</a>	amazon	Finished	04/29 08:59:43	04/29 09:00:02	0h:0m:19s	Unable to locate element: ("method":"xpath","selector	Failed
<input type="checkbox"/>	<a href="#">Add Star Trek to Cart 3</a>	amazon	Finished	04/29 08:59:43	04/29 09:00:06	0h:0m:22s		Passed

If the test case becomes Data Driven (see documentation on that [here](#)) then after clicking on Data Refresh button the test case will change from a single one in to multiple ones (depending on how many rows there are). Each test case data row will be treated as a separate test case in execution and you'll be able to analyze results accordingly.

### Data Refresh button

Every time test case does a change for Test Case Data (whether it's *modification of rows/columns* and/or *modification of data in these rows*) user needs to click on Refresh button (next to Search text field) in order for execution to be able to run test case with latest updates. If data has changed for a specific row which was already executed then it will be refreshed as Not-run (since this data iteration of a test case is different now).

### Test Case Data in Test Details

[All Executions]

[Execution] Amazon Shopping

[Test Details] Add Star Trek to Cart\_1

▲ Details

Name:

Add Star Trek to Cart 1

Status:

Finished

Result:

Passed

Test Case Data:

Movie

Star trek blu

When accessing the Test Details of an executed test case which is data driven it will show user 'Test Case Data' (in Details section) to inform what data driven values were used for this instance.

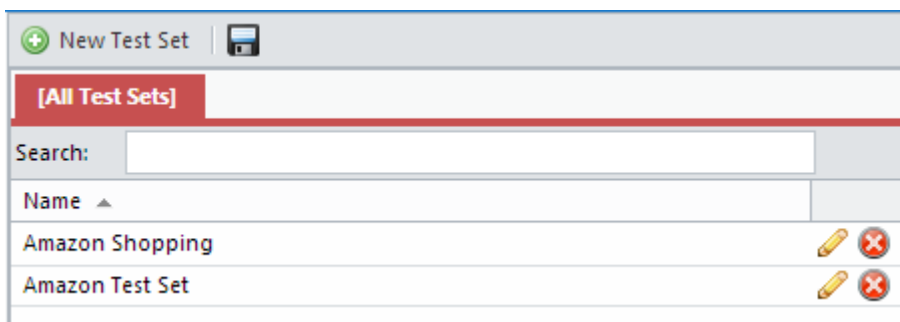
# Test Sets

---

## Test Set

Test Set is simply a collection of test cases which will be used by execution. It allows for creation of specific set rather than constantly showing all the test cases and can be used for things like defining 'Smoke Test' or 'Regression' sets.

## All Test Sets



### New Test Set

Opens new tab to create new test set.

### Save

Save currently open Test Set.

### Search

Search test set by name value.

### Name

Name of the test set.

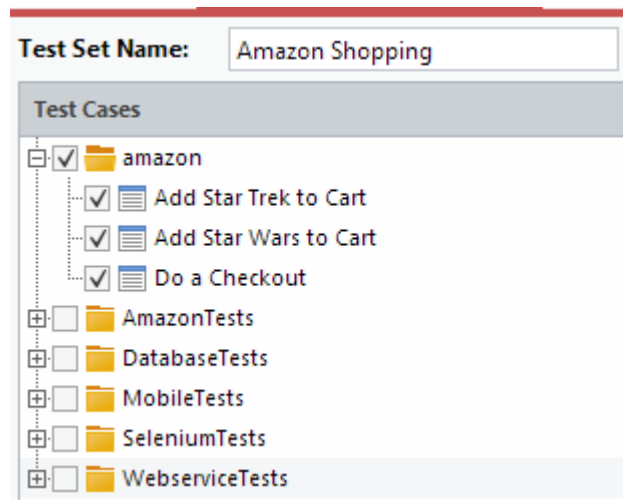
### Edit Button

Will open a new tab for that test set to modify it.

### Delete

Will delete test set.

## Test Set



**Test Set Name:**

**Test Cases**

- ☒ ☒ amazon
  - ☒ ☒ Add Star Trek to Cart
  - ☒ ☒ Add Star Wars to Cart
  - ☒ ☒ Do a Checkout
- ☐ ☐ AmazonTests
- ☐ ☐ DatabaseTests
- ☐ ☐ MobileTests
- ☐ ☐ SeleniumTests
- ☐ ☐ WebserviceTests

A test set definition page which shows all test cases filtered by tags (like Test Case Tree in Left Pane). Here you can check entire tag that will automatically check all the test cases under it or you can check specific test cases you want. If a new test case was created with a tag that was already checked, it will not be automatically included in a test set and would require user to update test set and check that test case in order for it to appear under execution.

### Test Set Name








Text field where you can specify any name for this test set.

### Test Cases

List of test cases sorted by tags which can be checked which would mean they are included as part of this test set.

# Variables

## Variables

 Add Variable Search: <input type="text"/>					
Execution Var	Tags	Name ▲	Value	Possible Values	
<input checked="" type="checkbox"/>	General	Browser	<NULL>	Internet Explorer,Firefox,Chrome	 
<input type="checkbox"/>	Domain	CurrentVersion	2.0		 
<input checked="" type="checkbox"/>	General	TestEnvironmentURL	<NULL>		 

On this page you create variables that will be used by automation. Variable values can be accessed either by Action Collections or by scripted code and values for it can be specified either inside Execution or the Variables page itself.

For a **Script** to retrieve variable value it has to do this:

```
redwood.launcher.Launcher.variables.<Variable Name>
```

### Example

```
class TestCode{
    public void run(HashMap<String, Object> params){
        String getBrowserVariableValue = redwood.launcher.Launcher.variables.Browser;
    }
}
```

To see how variables are used in **Action Collection** click [here](#).

### Add Variable

This button will generate a new variable line to enter information for.

### Search

Searches variables by name and tag.

### Execution Var

If this check box is checked then it means that this variable will appear in Execution where the value for it will be specified. If this is unchecked then whatever Value is specified on Variables page for this variable will be used by ALL executions.

**Tags**

Tag values for this variable.

**Name**

The name of the variable has to follow the code syntax rules: No space and no special characters.

**Value**

The value for this variable to be used by executions. If 'Execution Var' is checked for this variable then this variable will appear on Execution but it can be overridden there to whatever value execution user wants.

**Possible Value**

Used for 'Execution Var' variables, values entered here (type in value and press enter) will be part of drop down selection in Execution.

**Edit**

Will enable editing of this variable.











**Delete**

Will delete the variable.



# Machines

## Machines

 Add Machine		Search: <input type="text"/>						
Host Name/IP ▲	Port	VNC Port	Max Threads	Tags	Roles	Description	State	
<a href="#">192.168.197.1</a>	5009	3006	5	Development	Default	host name is:Server-1		  
<a href="#">192.168.197.128</a>	5009	3006	5	Regression	Server, Default	host name is:WIN-L1AGIMCPQOO		  
<a href="#">192.3.44.12</a>	5009	3006	1	Regression	Default	host name is:Server-2		  

This list will display all machines where RedwoodHQ Agent is installed on (which will get auto registered on this page after installation). It shows the current list of boxes which can be used for test execution and you can define thread and variables value for it.

### Host Name/IP

An actual IP address or a valid Host name of the machine where RedwoodHQ agent resides. Clicking on this link will open a VNC tab of this machine.

### Port

RedwoodHQ Agent port as has been defined during the installation (default port is 5009).

### VNC Port

RedwoodHQ Agent VNC port as has been defined during the installation (default port is 3006).

### Max Threads

You can specify multiple threads that machine can handle to be used by execution by putting a number. Multiple threads is very useful where you want to run tests in parallel on a single box rather than just using 1 box per test case. Some of good example where multi-threaded execution can come in useful is API testing where there is no UI interface and typically machine can handle multiple parallel processes. Selenium also support multi-threaded execution on a single box and can be used as well, just make sure that you have valid After State selected to close the browser after it's done with it.

### Tags

Tags which are assigned to the machine ease of search.

### Roles

Actions in action collection might have logic to be able to execute against multiple machines (eg: Server configuration on one while browser execution on another) and here you'll define to what role machine

belongs to by entering a value and pressing enter (or selecting multiple values from drop down). For more information of Role usage in Action Collection can be found [here](#).

### Description

Description text to describe this machine.

### State

How many threads out of how many available are currently being used right now by any given execution.

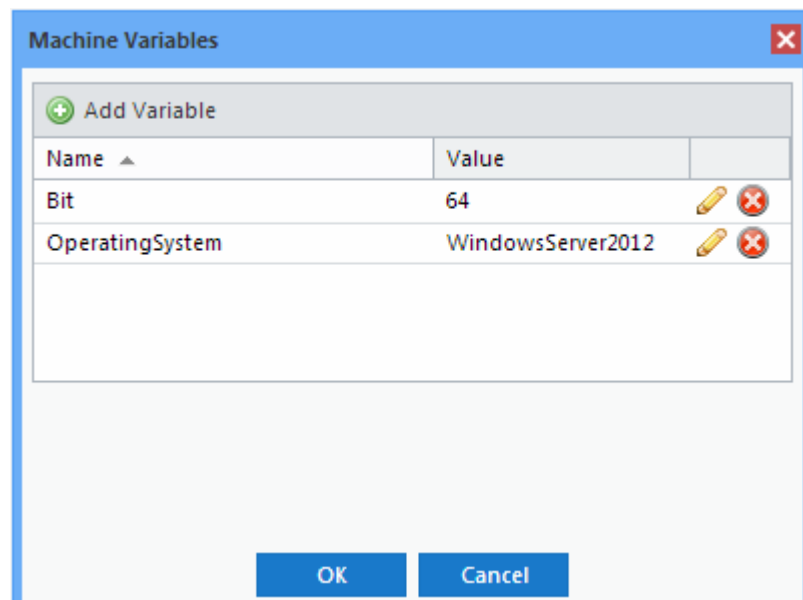
### Edit

Will enable editing of this machine.

### Delete

Will remove this machine (but if RedwoodHQ Agent is not shutdown this machine will appear again, so make sure to shut it down first before doing any deletion).

## Machine Variables



The image shows a 'Machine Variables' dialog box with a blue title bar and a red close button. Inside, there is a table with two columns: 'Name' and 'Value'. The table contains two rows: 'Bit' with value '64' and 'OperatingSystem' with value 'WindowsServer2012'. Each row has a yellow pencil icon and a red 'X' icon to its right. Above the table is a button with a green plus icon and the text 'Add Variable'. Below the table are 'OK' and 'Cancel' buttons.

Name	Value
Bit	64
OperatingSystem	WindowsServer2012

Clicking on Machine Variables button for any given machine will bring up Machine Variables screen. All variables which are defined here belong to this specific machine and are retrieved by Action Collection parameters like this: `${ Machine.<Role>.<Variable Name> }`. This feature is useful if you need to retrieve information during execution by action collection specifically for this machine rather than execution or global variables. So if the above machine example belonged to 'Default' role and I would want to retrieve 'OperatingSystem' by a parameter I would type this in action collection: `${`

`Machine.Default.OperatingSystem }` (for more information on pointing to machine variable in Action Collection click [here](#)).

# Continuous Integration

## Continuous Integration

Using tools like Jenkins and Team City you should be able to setup a Continuous Integration where it will create and then kick off the execution and report results.

### Usage

1. Install Continuous Integration agent on RedwoodHQ server machine.
2. Copy node.exe file from \RedwoodHQ\vendor\nodejs folder to \RedwoodHQ\cli (makes it easier for execution).
3. Setup Continuous Integration to run a RedwoodHQ CLI command under \RedwoodHQ\cli folder:  
`node CIExecution.js <parameters>` (details of parameters are described bellow).
4. CI will kick off CLI command which will create a new Execution in RedwoodHQ and start running all test cases inside it.
5. When Execution finishes it will:
  - Send results to Continuous Integration tool
  - Will also keep results in RedwoodHQ
  - Lock execution in RedwoodHQ (because we are officially done with it)



If you want to see all Continuous Integration executions in RedwoodHQ that have been done so far then make sure to set check box under [All Executions] to see Locked and sort by Last Run column to see executions in chronological order. You can also do a search to filter out to specific tag/name.

### CLI Command Parameters

Parameter Name	Parameter Description	Example
<code>--name [name]</code>	Name of the execution that will get created	<code>--name "Amazon Test"</code>
<code>--user [username]</code>	User ID that will be used for this execution	<code>--user admin</code>

<code>--testset [testset name]</code>	The name of the Test Set for this execution	<code>--testset "Smoke Test"</code>
<code>--machines</code> <code>[hostname1:threads:baseState,hostname2:threads:baseState]</code>	Machine name/ip, amount of threads to use and Machine Base State (optional)	<code>--machines</code> <code>192.5.5.33:2,192.5.6.33:2</code> machines, 2 threads for each state) or <code>--machines</code> <code>"192.5.5.33:1:Install"</code> (use one machine, 1 thread Base State on it first)
<code>--pullLatest [true,false]</code>	Should it do a Git Pull to get the latest for that User	<code>--pullLatest true</code>
<code>--retryCount [1]</code>	Set Retry Count in execution	<code>--retryCount 2</code>
<code>--variables [name=value,name2=value2]</code>	Specify execution variable value	<code>--variables</code> <code>Browser=Firefox,TestB</code>
<code>--tags [tag1,tag2]</code>	Assign tag(s) to execution	<code>--tags SmokeTest,Func</code>
<code>--emails [email1,email2]</code>	Email addresses of the users of RedwoodHQ to whom report email will be sent after execution finishes	<code>--emails john@acme.com,b</code>
<code>--project [projectname]</code>	Name of the project where execution will happen	<code>--project Sample</code>
<code>--ignoreScreenshots [true,false]</code>	Set 'No Screen	<code>--ignoreScreenshots t</code> Screen Shots' to true)

	Shots' option on execution	
--	----------------------------------	--

**Full CLI Command example:**

```
node CIExecution.js --name "Amazon Test" --user admin --testset "Smoke Test"
--machines "192.5.5.33:1:Install New Build" --pullLatest true --retryCount 2
--variables Browser=Firefox,TestBuildNumber=17 --tags SmokeTest,Functional
--project Sample --ignoreScreenshots false
```

# Elasticsearch with Kibana

---

## RedwoodHQ + Elasticsearch with Kibana

User can setup Elasticsearch with Kibana to be able to create much more detailed results and reports. You are able to graph various points of results and output them in presentable charts. Here are the steps to set it up:

1. Install and start Elasticsearch and Kibana from: <https://www.elastic.co/>
2. Open < RedwoodHQ install directory >\properties.conf file
3. Change following entries:  
ELKServer=  
ELKServerPort=9200  
ELKServer is localhost if elasticsearch is installed on same machine as RedwoodHQ
4. Restart RedwoodHQ and you are now ready to collect data and visualize it using Kibana

# Settings

---

## Settings

Settings deals with user management, projects that can be added or deleted as well as Email setup (which is used by Execution).

### Topics

- [Users](#)
- [Projects](#)
- [Email](#)



# Users

---

## Create User

By default, RedwoodHQ offers one pre-created user account with administrative privileges. A user with administrative privileges can create accounts for other users. The list of existing user account is displayed on the Settings tab when Users is selected in the pane on the left.

The default account credentials are:

- Username: admin
- Password: admin

1. On the Settings tab, in the pane on the left, click Users.
2. On the toolbar above the list of existing user accounts, click Add User.
3. In the User Properties dialog box, in the User ID box, define the ID for the user.
4. The ID of a user is this user's login to Redwood HQ.
5. In the First/Last Name box, specify the user's name.
6. Email of the user
7. Role
  - Developer – Has access to all functionality of RedwoodHQ
  - Test Designer – Has access only to Test Cases and Executions
8. (optional) In the Tags box, define tags for the user.
  - Tags are used for searching.
  - After typing each tag, press Enter.
9. In the Password box, define a password for the user and, in the Repeat Password box, type it again.
10. Click Submit.
11. The newly added user account appears in the list on the Settings tab.

## Modify User Password

1. On the Settings tab, in the pane on the left, click Users.
2. In the list of user accounts, click the account of the user whose password you want to change.
3. To find the necessary user account, you can use the search capability – on the toolbar above the list of user accounts, in the Search box, type the user's ID, name, tag, or role. To reset the search, click x.
4. Double-click the selected user account, or click Edit for it.

5. In the User Properties dialog box, in the Password box, type the new password and, in the Repeat Password box, type it again.
6. Click Submit.

## Delete User

1. On the Settings tab, in the pane on the left, click Users.
2. In the list of user accounts, for the account you want to delete, click Delete.

# Projects

---

## Create a Project

A project gathers scripts, actions, test cases, variables, test sets, and executions that belong to a certain area of testing. Projects are separated from each other and cannot share content.

The list of existing projects is displayed on the Settings tab when Projects is selected in the pane on the left.

1. On the Settings tab, in the pane on the left, click Projects.
2. On the toolbar above the list of existing user accounts, click Add Project.
3. In the Project Properties dialog box, in the Project Name box, define the name for the project.
4. In the Project Template list, select the template.
5. A project template is pre-populated with scripts, actions, and test cases that, in a new project, can be used as samples: **Java Based Selenium**
6. Click Submit.

## Select a Project

1. In the upper right corner, in the Choose Project list, click the name of the project you want to work with.
2. In the confirmation dialog box, click Yes.

## Delete a Project

1. On the Settings tab, in the pane on the left, click Projects.
2. In the list of projects, for the project you want to delete, click Delete.

## Clone a Project

1. On the Settings tab, in the pane on the left, click Projects.
2. In the list of projects, for the project you want to clone, click Clone Project.
3. Specify new project name for the clone and press Submit.
4. The clone will copy everything but the executions.

# Email

---

## Email

In execution, under Email Notification you can add users and set the Send Email check box to true. What will happen is that after execution is done, it will sent out an email to all these users which has a link to the execution as well as Execution Totals number (passed, failed, etc.).

In order to setup the Email settings, go to Settings -> Email and specify the values for it.

1. Server Host is where you specify RedwoodHQ server name/ip address.
2. SMTP Host is the email server which is used by your company to send out the email.
3. Protocol can be either SMTP or SMTPS.
4. Optional parameters (use only if needed): SMTP Port; User Name; Password

# Install Guide

---

## Overview

### What is RedwoodHQ Server?

For a team to work in RedwoodHQ to develop test automation (API or GUI) they just need **1** copy of RedwoodHQ installed on the machine which everyone can reach via ip address and it will be a central repository for all scripts, test cases and executions.

**Nothing needs to be installed on the users end to work with RedwoodHQ**, they just need to open a browser (IE, Chrome, Firefox or Safari) and navigate to ip address of RedwoodHQ Server machine + application port number (typically 3000) that was specified during install (eg: <http://192.168.44.554:3000>) and then login with user credentials (admin/admin is first user, all others can be added under [Users](#) page). The only exception is that if user wants to use 'Looking Glass' feature, then they would need to install RedwoodHQ Agent on their OS.

It is common for people to first experiment with RedwoodHQ on temporary location (personal laptop/PC) and later on move to more permanent one so that it can be accessed at any time. In that case you just simply install RedwoodHQ on that other machine and begin using it, but if you also want to migrate the scripts/test cases you have already created then it is easy to do so by following [Migration](#) instructions.

### [RedwoodHQ Server Install Guide](#)

### What is RedwoodHQ Agent?

RedwoodHQ Agent is primarily used for execution of test automation and it is compatible with any Windows, Linux and Mac OS X (Linux and OS X requires special setup). Once the agent is installed on the machine it will self register on RedwoodHQ server and users will be able to select it under 'Machines' option for execution.

If user wants to use optional 'Looking Glass' feature then they would need to install Agent on their OS as well.

### [RedwoodHQ Agent Install Guide](#)

# RedwoodHQ Server

---

## Install and Configure

For **Windows** install you can follow [Quick Start 1 guide](#).

For **Linux** or **Mac** installation please [go here](#).

Configuring Server/Agent requires to modify properties.conf file, [go here for more information](#).

## Recommended System Requirements for RedwoodHQ Server install location

### OS

- Windows 7 – 64 Bit
- Windows 8 – 64 Bit
- Windows Server 2008 R2
- Windows Server 2012

### Hardware

- 4 Gig Ram or more
- Quad CPU or higher
- 1.2 Gig Hard Drive Space for Installation
- 300 Gig Hard Drive Space for all future files

## Download and Install

[Update RedwoodHQ Server](#)



It is recommended doing regular backups of RedwoodHQ Server machine like for any other critical environment.

# Update Windows RedwoodHQ Server

---

## Update

Updating RedwoodHQ server is a very simple procedure but will require a windows restart in order to be completed:

1. Download the RedwoodHQ installer you want from [RedwoodHQ Downloads](#) page
2. Double click on Installer file
3. When Installer window appears click on Next
4. You will be prompted on whether you want to update RedwoodHQ, click on Yes
5. The older version will now be uninstalled (all database and code files will be kept, they never get removed)
6. After uninstall completes a message will appear that your windows will be restarted, clicking on OK will restart the computer
7. Once installation finishes you will have a newly updated version of RedwoodHQ which will also begin to **automatically update RedwoodHQ agents** if necessary
8. If you are **updating from version 2.0 or 2.1** and you would like to use new **Python/C# Language as well as Ivy** then you would need to **create a new project** after update completes



In order to minimize the risk it is always recommended on doing a backup of a machine where RedwoodHQ Server is installed on before doing an Update.

# MongoDB: v2 to v3 migration

---

1. Make sure RedwoodHQ is running
2. Open command line interface and navigate to: \RedwoodHQ Install Location\vendor\MongoDB\bin
3. Type “mongodump” and press Enter
4. Wait until process is complete (might take some time depending on how much data you have in RedwoodHQ)
5. Close command line interface
6. Shut down RedwoodHQ: Start -> RedwoodHQ -> **Stop RedwoodHQ**
7. Rename folder: \RedwoodHQ Install Location\data **to** \RedwoodHQ Install Location\data\_old
8. Create new folder: \RedwoodHQ Install Location\data
9. Create new folder: \RedwoodHQ Install Location\data\db
10. Rename folder: \RedwoodHQ Install Location\vendor\MongoDB **to** \RedwoodHQ Install Location\vendor\MongoDBv2
11. Rename folder: \RedwoodHQ Install Location\vendor\MongoDBv3 **to** \RedwoodHQ Install Location\vendor\MongoDB
12. Move folder: \RedwoodHQ Install Location\vendor\MongoDBv2\bin\dump **to** \RedwoodHQ Install Location\vendor\MongoDB\bin
13. Open command line interface and navigate: \RedwoodHQ Install Location\vendor\MongoDB\bin
14. Type “mongod —dbpath ..\..\data\db” and press Enter
15. Open **another** command line interface and navigate: \RedwoodHQ Install Location\vendor\MongoDB\bin
16. Type “mongorestore” and press Enter
17. Wait until it's done restoring
18. Kill mongod process in cmd window (eg: Ctrl-C) and close all command line windows
19. Delete folder: \RedwoodHQ Install Location\vendor\MongoDB\bin\dump
20. Start RedwoodHQ: Start -> RedwoodHQ -> **Start RedwoodHQ**

Everything should work as is or better now. It is recommended to keep data\_old folder (which now contains MongoDBv2 old db) just to make sure you can roll back to it, but if after few weeks things are working fine, then you can delete it if you want to free up some space.



# Linux or Mac Server Install

---

## The steps to install and run Linux/Mac Server:

1. Download Linux/Mac Server from here: [RedwoodHQ Download Page](#)
2. Extract the Server to the directory that you want
3. Open terminal and switch to a sudo user for full privileges (eg: sudo su)
4. Give full privileges access to the files in RedwoodHQ directory (eg: chmod -R 777 [directory where RedwoodHQ is extracted])
5. Start Server:  
Go to: [RedwoodHQ directory]  
Run this command: start.sh
6. Start Agent:  
Go to: [RedwoodHQ directory]/agent  
Run this command: start.sh
7. Server is now running as ip address:port which can be accessed through Web Browser (eg: http://127.0.0.1:3000)

Go to [Quick Start Guide 1](#) and from Execute section begin learning to use RedwoodHQ, have fun!

## Stopping Server/Agent:

1. Stop Server:  
Go to: [RedwoodHQ directory]  
Run this command: stop.sh
2. Stop Agent:  
Go to: [RedwoodHQ directory]/agent  
Run this command: stop.sh

# RedwoodHQ Agent (Windows, Linux, Mac)

---

## Install Windows

1. Open browser and navigate to the ip address + port number of machine where RedwoodHQ is installed on (it's Login page)
2. Click on 'Download Agent' button and save the agent setup file
3. Double click on the setup file and click on Next
4. On the installer step where it's asking you for 'RedwoodHQ Server IP/Host Name', enter the IP address of where the RedwoodHQ server is installed on and click on Next

## [Update RedwoodHQ Agent](#)

## Install Linux (eg: Ubuntu)

1. Download Linux Agent from here: [RedwoodHQ Download Page](#)
2. Extract the Agent to the directory that you want
3. Give full privileges access to the files (eg: `chmod -R 777 [directory where RedwoodHQAgent is extracted]`)
4. Modify properties.conf file (located: `[RedwoodHQAgent directory]/properties.conf`):  
AgentPort = (port the agent should use, set to default 5009)  
AppServerIPHost = (ip address/host name of the RedwoodHQ server)  
AppServerPort = (Server port, set to default 3000)
5. To start agent,  
Go to: `[RedwoodHQAgent directory]/agent`  
Run this command: `start.sh`  
(if start.sh doesn't exist then run this command: `../vendor/nodejs/node app.js`)
6. To stop agent,  
Go to: `[RedwoodHQAgent directory]/agent`  
Run this command: `stop.sh`  
(if stop.sh doesn't exist then terminate node app.js command)

## Install Mac (OS X)

1. Download Mac Agent from here: [RedwoodHQ Download Page](#)
2. Extract the Agent to the directory that you want
3. Give full privileges access to the files (eg: `chmod -R 777 [directory where RedwoodHQAgent is extracted]`)

4. Modify properties.conf file (located: [RedwoodHQAgent directory]/properties.conf):  
AgentPort = (port the agent should use, set to default 5009)  
AppServerIPHost = (ip address/host name of the RedwoodHQ server)  
AppServerPort = (Server port, set to default 3000)
5. To start agent,  
Go to: [RedwoodHQAgent directory]/agent  
Run this command: start.sh  
(if start.sh doesn't exist then run this command: ../vendor/nodejs/node app.js)
6. To stop agent,  
Go to: [RedwoodHQAgent directory]/agent  
Run this command: stop.sh  
(if stop.sh doesn't exist then terminate node app.js command)

# Update RedwoodHQ Agent

---

## Update

Once you install another version of RedwoodHQ Server all agent machines which are pointing to it **will update automatically** (if needed).

If you still want to update RedwoodHQ Agent manually (or if automatically failed for some reason) then follow these steps:

1. Go to the machine where you want to install the agent
2. Open browser and navigate to the ip address + port number of machine where RedwoodHQ is installed on (it's Login page)
3. Click on 'Download Agent' button and save the agent setup file
4. Double click on the setup file and click on Next
5. You will be prompted on whether you want to update RedwoodHQ Agent, click on Yes
6. Once installation finishes you will have a newly updated version of RedwoodHQ Agent ready to go

# Configuration (properties.conf)

---

## Properties.conf

To configure server/agent you need to go to directory where RedwoodHQ is installed and open properties.conf file (make sure to stop server/agent before modification). In there you should be able to modify the following options:

### [Agent]

AgentVersion= [Version of the agent currently deployed] (eg: 2.50.00)

AgentPort= [Port that agent uses] (eg: 5009)

AgentVNCPort= [VNC Port that agent uses] (eg: 3006)

Update= [If update was done it would have Success or Fail value here]

OS= [OS deployed on] (eg: Windows)

### [Server]

ServerVersion= [Version of the Server currently deployed] (eg: 2.50.00)

DBPort= [Port that MongoDB uses] (eg: 27017)

AppServerPort= [What port should AppServer use] (eg: 3000)

AppServerIPHost= [IP Address of the Server, if agent is on same box as server then point at localhost, otherwise put actual ip address] (eg: 127.0.0.1)

ELKServer= [If using Elastic, specify IP address for it here] (eg: localhost)

ELKServerPort= [Port for Elastic server] (eg: 9200)

### [Windows]

VNC= [Used by Windows installer] (eg: No)

AppShortcutFolderName= [Used by Windows installer] (eg: RedwoodHQ)

# Migration to another RedwoodHQ instance

## Migration of Database and Scripts

If you need to move your data from one RedwoodHQ server to another one (eg: to a more permanent location for multi-user access) then follow these steps bellow.

! If you are migrating from **RedwoodHQ 2.40 (or earlier) to 2.50 and later**, please make sure to do [migration of MongoDB from v2 to v3](#).

**FROM** = Where RedwoodHQ Server is currently installed on and data is being moved from

**TO** = Where another fresh version of RedwoodHQ server is installed on and where you are moving the data to

**\RedwoodHQ Install Location\** = Folder location where RedwoodHQ is installed (eg: C:\Program Files\RedwoodHQ\)

! **FROM** will completely overwrite ALL the data and scripts on the **TO** RedwoodHQ server instance!

1. Shut down RedwoodHQ on FROM and TO computers by running 'Stop RedwoodHQ' shortcuts/vbs scripts
2. Delete the following folder on TO computer: \RedwoodHQ Install Location\data
3. Copy the following folder at FROM computer: \RedwoodHQ Install Location\data
4. Paste the folder on TO computer: \RedwoodHQ Install Location\
5. Delete the following folder on TO computer: \RedwoodHQ Install Location\public\automationscripts
6. Copy the following folder at FROM computer: \RedwoodHQ Install Location\public\automationscripts
7. Paste the folder on TO computer: \RedwoodHQ Install Location\public\
8. Start up RedwoodHQ Server by running 'Start RedwoodHQ' shortcuts/vbs scripts
9. Login on TO computer and verify that all data is there

# Looking Glass

---

A cross-browser open-source tool to record, develop, debug Selenium scripts as well as identify and validate HTML objects by XPath, CSS, etc.

Video Overview: [Youtube link](#)

Discuss and ask any questions: [Forum Link](#)

Compatible:

OS: Windows, Linux, Mac OS X

Browsers: Chrome, Internet Explorer, Safari, Firefox

*Please note that this is a Selenium based solution.*

*This means as your browser versions change you need to get latest Selenium jars. See FAQ for more info.*

[Quick Start](#)

[Install Stand Alone](#)

[Looking Glass F.A.Q.](#)

# Looking Glass Quick Start

---

## Inspector

### Inspector

Inspector will identify HTML elements, suggest, verify xpath, css selectors and where you can even test whether Selenium will interact with the object or not.

1. Select browser type you want to work with and click on Open button.
2. Navigate to needed URL.
3. Click on the looking glass icon and click on element you want to inspect.
4. Select "Click" operation from drop down next to "Validate" button.
5. Click on the "Click" button to see if Selenium can click on the selected HTML element.
6. Click on 'XPath' and select any other property you want to see for this object (eg: CSS Selector).
7. Modify the text value (eg: XPath property value) to another one and click Validate to see if it works or not.

### Code

This is where you can record or specify your own code and do a playback against any browser. Support any Java/Groovy code only.

1. Click on Code tab.
2. From menu select Browsers -> and select the type you want.
3. Click on Record (red) button to begin recording and navigate to URL you will use.
4. Do various clicking and typing actions you want and then click on Record (red) button again to stop.
5. Click on Green arrow to playback recorded script.
6. You can modify the code (ANY Java/Groovy code with asserts, loops, etc.) and do a playback against any browser as well.
7. User can also copy/paste their own code in to Code section for quick validation/modification. Use 'Driver variable name' field to make sure it matches the one in your own code.



# Install Stand Alone Looking Glass

---

If you want to use 'Looking Glass' by itself without installing/using RedwoodHQ, then follow these steps bellow:

1. Install Java 1.7.
2. Download zip file from: <https://bitbucket.org/primetestinc/looking-glass/downloads/LookingGlass.zip> (select 'Keep' when prompted by browser).
3. Unzip contents in to your folder (eg: c:\Looking Glass).
4. Double click on 'StartLookingGlass.vbs' to start on windows  
- StartLookingGlass.sh for Linux/Mac
5. If 'Looking Glass' doesn't start, it could be that Java path is not set on your computer, please follow these instructions to set it: <https://forums.bukkit.org/threads/how-to-fix-java-is-not-recognized-as-an-internal-command-or-external-command.139263/>

# **Looking Glass F.A.Q.**

---

## **This FAQ is for Stand Alone version of Looking Glass**

### **How is future support of the browsers going to work? How can I add later/earlier version of Selenium?**

You can replace any Selenium jar that Looking Glass comes with to whatever latest one or the one you want under: \Looking Glass Installed Location\lib

### **Can I use my own jar files and have that code be available in code portion of the Looking Glass?**

Yes you can.

1. Place your jars into \Looking Glass Installed Location\lib directory.
2. Restart Looking Glass and navigate to "Code" tab.
3. Click on "imports" icon.
4. Add any imports you like either from your jar or from selenium. Now your your jar code is available to be used in Code tab.

### **Opening Internet Explorer through Looking Glass gives an error!**

Selenium can be a bit problematic with IE sometimes but we noticed that restarting your computer typically makes this error go away. Also, make sure your IE is not higher than version 10, selenium currently doesn't support 11.