Web Functional Testing - Overview

This topic provides an overview of SOAtest's web functional (cross-browser) testing capabilities. In this section:

Web Testing Introduction

Web interface testing can be difficult to automate. Teams often abandon automated testing in favor of manual testing because automated tests produce many false positives—or because so much time and effort is required to maintain the test suites.

SOAtest is designed to reduce these obstacles. It isolates and tests individual application components for correct functionality across multiple browsers—without requiring scripts. Dynamic data can be stubbed out with constant data to reduce test case noise. Validations can be performed at the page object level as well as the HTTP message level. SOAtest also verifies the client-side JavaScript engine under expected and unexpected conditions through asynchronous HTTP message stubbing.

About the WebDriver Engine

You can use the Selenium WebDriver for playing back web scenarios or the Parasoft Native Driver (legacy) engine. For details, see About the Selenium WebDriver Engine.

Browser Recording and Cross-Browser Execution

The first step in web testing is browser recording, which is described in Browser Recording and Playback.

Once created, tests can be executed as described in Executing Functional Tests.

Extending and Reusing Web Scenarios

You can rapidly extend your recorded web scenarios to meet your goals. The following examples are common practices for extending and reusing web scenarios:

- Configuring cross-browser testing; see Configuring Browser Playback Options
- Configuring user actions; see Configuring User Actions (Navigation, Delays, etc.)
- Configuring validations; see Validating or Storing Values
- Configuring wait conditions; see Configuring Wait Conditions
- Configuring actions that occur before and after test execution; see Adding Set-Up and Tear-Down Tests.
- Configuring execution options, such as test sequence, test relationship, and test flow logic; see Configuring Test Suite Properties - Test Flow Logic, Variables, etc..
- End-to-end testing of scenarios that extend through web interfaces, backend services, ESBs, databases, and everything in between; End-to-End Test Scenarios
- Load testing; see Load Testing
- Penetration testing; see Penetration Testing
- Runtime error detection; see Performing Runtime Error Detection
- Data-driven testing; see Parameterizing Tests with Data Sources, Variables, or Values from Other Tests.
- Using stubs and environments to configure a predictable and accessible test bed; see Configuring Testing in Different Environments.

Browser Support

The following browsers are supported for playback with the Selenium WebDriver engine:

- Firefox 47.0.1 and earlier
- Google Chrome
- Internet Explorer 8+
- Safari 7-9.x on Mac OSX
- Microsoft Edge Windows Anniversary Edition (version 38.14393)+

The following browsers are supported for recording and playback with the Parasoft Native Driver (legacy) engine:

- Internet Explorer 8+
- Google Chrome 9+
- Firefox 3.6-54 (playback only)

You can also perform mobile interface testing through a desktop browser as described in Mobile Interface Testing.

Firefox Notes

- Playing back inputs that allow you to browse to a file to upload it is not supported in Firefox 3 and later.
- If Firefox is in the middle of an automatic update (the update was downloaded but not yet installed), web scenarios might not play back correctly (Virtualize will launch Firefox, and Firefox will open a dialog to check add-on compatibility; this causes playback to fail). If this occurs, launch Firefox outside of Virtualize to complete the update installation. Once the update is completed, web scenarios can be played back as usual.

Internet Explorer Notes

- Before using Internet Explorer, see Configuring Internet Explorer Settings.
- With Internet Explorer 9 or 10, we recommend using addEventListener() when registering event listeners. However, if you are using attachEvent(), which is an IE-specific event model, you will need to manually update `<Virtualize Directory>\eclipse\plugins\com.parasoft.xtest.libs.web_<version>\root\browsers\ie\HTMLUtil.js` as follows to force Virtualize to emulate JavaScript events using the IE-specific model. You need to make the following changes in this file:

  SOAtest

  `"ext.HTMLUtil.preferLegacyEventModel = false;"`
  to
  `"ext.HTMLUtil.preferLegacyEventModel = true;"`

  Virtualize

  `"_wk_HTMLUtil.preferLegacyEventModel = false;"`
  to
  `"_wk_HTMLUtil.preferLegacyEventModel = true;"`

- The first time that Internet Explorer 9 is launched, it will open a warning dialog that states "The Recorder Registrar is now ready to use." You need to click **Enable** to ensure that your web scenarios record and play back correctly. For best results, you should close and re-open the browser after clicking **Enable**.
- File inputs inside modal dialogsare not supported.
- Record and playback on file inputs (inputs that allow you to browse to a file to upload it) is not supported in Internet Explorer 8 and later.

Chrome Notes

- When specifying the path to the Chrome executable on Linux, choose `google-chrome` (e.g. `/opt/google/chrome/google-chrome`)—not `chrome`. On Windows, the path is typically `C:\Users\usernane\AppData\Local\Google\Chrome\Application\chrome.exe`. On Mac, it's typically `/Applications/Google Chrome.app`.
- If you just recorded a web scenarios and the browser contents for the last step was not captured, you can play the scenario to capture this content. Note that this happens when you stop recording by closing the browser window. The recommended best practice is to stop recording by clicking the **Stop Recording** button (instead of by closing the browser window).

- On Chrome, Auto Generate Asynchronous Request Tests captures only requests from an `XMLHttpRequest` object. It does not detect asynchronous requests from hidden `IFrame` calls. If needed, you can add such asynchronous request tests manually.

- When a user acts on an element, Chrome (unlike Internet Explorer and Firefox) does not check other frames to ensure that the locator is unique across frames. As a result, a recorded action might be played back in a frame other than the one it was recorded in. If you experience this unexpected behavior, determine a unique locator and change the locator being used in the Browser Playback tool.

- In some cases, login is required before asynchronous requests can occur. Firefox and IE will detect this, and test suites will automatically create a setup test that performs this login. The asynchronous test will then use the cookies created by logging in. Chrome does not support this. For Chrome, SOAtest can generate asynchronous request tests, but the login step needs to be set up manually—either as a setup test, or a test suite tool that is run before the asynchronous tests.

- Record and playback on file inputs (inputs that allow you to browse to a file to upload it) is not supported in Chrome.
- When recording or playing back with Chrome on Mac using the Parasoft native engine, closing Chrome opens a dialog that says "Google Chrome wants to use your confidential information stored in 'Chrome Safe Storage' in your keychain. Do you want to allow access to this item?" Even if you choose "Always Allow", this dialog opens every time that Chrome is started and then closed. To prevent this dialog from opening every time:
  - Open **Applications> Utilities> Keychain Access**.
  - Control-click **Chrome Safe Storage**, then choose **Get Info** from the shortcut menu.
  - Switch to **Access Control**, select **Allowall applications to access this item**, and click **Save Changes**. You will need to enter an administrator password to continue.

Safari Notes

- Safari support is available for playback using the Selenium WebDriver engine.
- For a list of associated limitations, see Safari-Specific Issues.

Microsoft Edge Notes

You must have the correct MicrosoftWebDriver executable for your version of Edge. If you receive an "Unable to launch Microsoft Edge" error message, you may need to download the driver. Visit https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver for additional information.

Troubleshooting Playback in Microsoft Edge

SOAtest/Virtualize cannot launch Edge when the SOAtest/Virtualize process is running with privileged access. Take the following actions to resolve this issue:

1. Use a non-administrator account or an administrator account with User Account Control notifications turned on to run SOAtest/Virtualize as a non-privileged process.
2. Enable user Account control notifications for administrator users:

   Non-Windows Home edition:

   a. Open the Local Security Policy (secpol.msc) and enable **User Account Control Admin Approval Mode for the Built-in Administrator account** under Local Policies/Security options.

   Windows Home edition:

   a. Use regedit.exe to navigate to HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
   b. If it does not exist, create the DWORD value "FilterAdministratorToken"
   c. Set the value of "FilterAdministratorToken" to 1

3. If launching SOAtest/Virtualize from a Scheduled Task during startup, ensure that the **Run only when user is logged on** option is enabled.