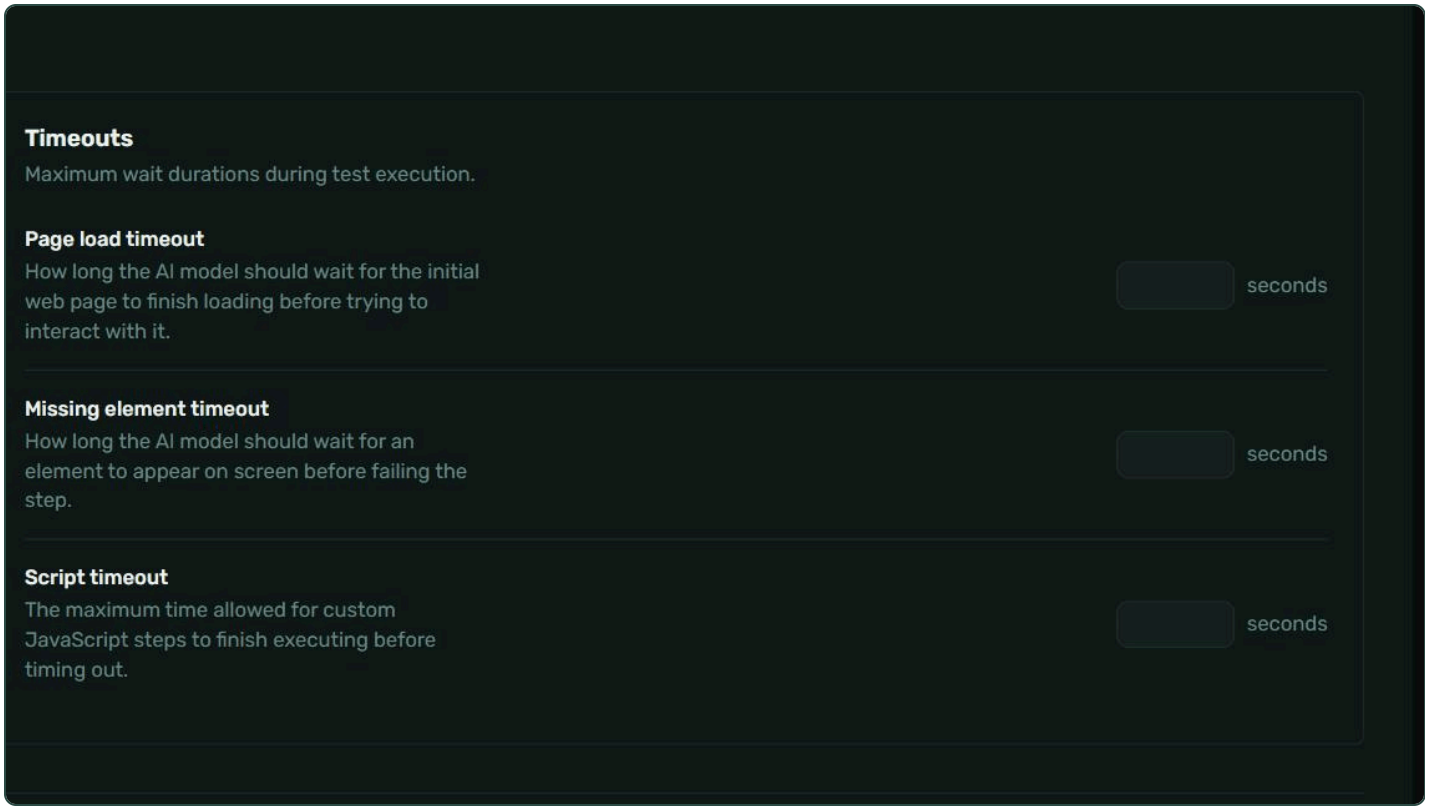
Timeouts, Access and Browser Behavior: Getting Studio Into Your Application

Settings · https://docs.functionize.com/hc/en-us/articles/43769958538647-Timeouts-Access-and-Browser-Behavior-Getting-Studio-Into-Your-Application

Most applications worth testing are not a public website with no front door. They sit behind authentication, use client certificates, need particular cookies, or are simply slow. This article covers the settings that get Studio into your application and keep it there: Timeouts, Access & Auth, and Browser behavior.

Timeouts

Under **Execution** → **Timeouts**, three limits, all in seconds:



Page load timeout	How long to wait for the initial page to finish loading before trying to interact with it.
Missing element timeout	How long to wait for an element to appear before failing the step.
Script timeout	The maximum time a custom JavaScript step may take.

Missing element timeout is the one that bites. If tests fail on elements that you can clearly see do exist, this is usually why — the element arrived after Studio stopped waiting. Raise it before you start rewriting steps.

Timeouts are a blunt instrument, though: a longer timeout slows down every genuine failure too, because a step that will never succeed now takes the full duration to admit it. For "how patient should the agent be" in a more nuanced sense, the **Timing model** slider is the better control — see Self-Healing and Timing.

Access & Auth

Five sections, for applications that are not simply open.

Cookies

"*Cookie entries injected into the browser before each run.*" Two lists — standard **Cookies** and **HTTP cookies** — each with an Add button.

This is how you skip things that are not what the test is about: a consent banner, a feature flag, a locale preference, a "you have seen this tour" marker. Setting the cookie is faster and far less brittle than adding three steps to dismiss a dialog at the start of every test.

Custom HTTP headers

"*Headers sent with every request during test execution.*" Useful for API gateways, environment routing headers, or a header your staging environment requires to let traffic through.

Basic authentication

For environments behind HTTP basic auth — the browser-level username and password prompt, not a login form in your application. If your staging site throws a native browser dialog before you ever see the page, this is the section you need.

Put these on the project, not on individual tests, and treat them as environment configuration rather than test data.

Certificate handling

Two switches, both of which loosen security, so read them carefully:

- **Allow invalid TLS certificates** — continue despite certificate warnings. Reasonable on an internal staging environment with a self-signed certificate.
- **Disable browser web security** — "*loosens cross-domain restrictions and browser security mechanisms.*"

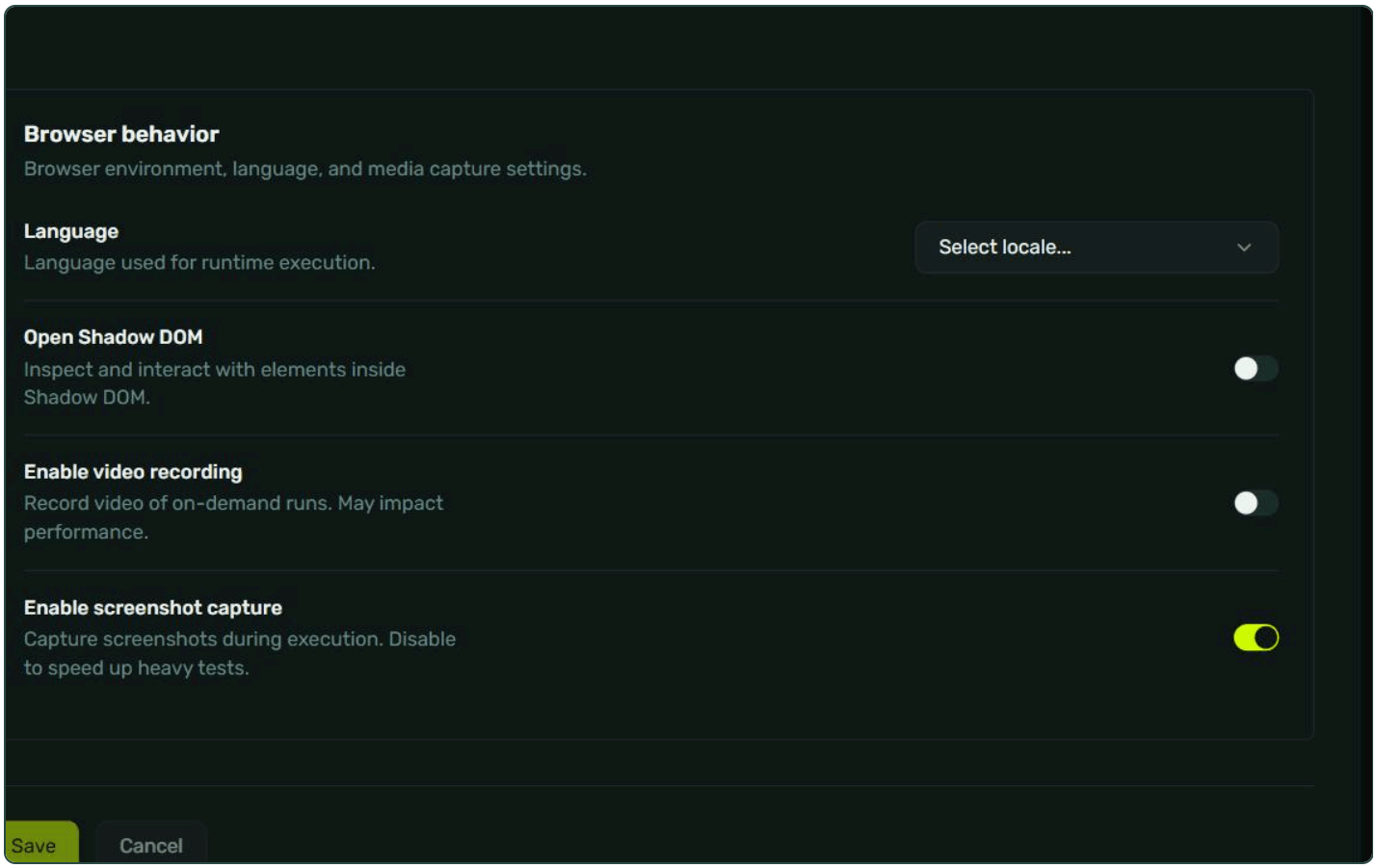
Be deliberate about the second one. Turning off web security changes how the browser behaves, so your test is no longer exercising the application the way a real user's browser would. It can mask a genuine cross-origin bug. Use it to get unblocked, not as a default.

Mutual TLS

An **MTLS client certificate** and **client key**, for endpoints that require client-certificate authentication. Common in financial services, healthcare and internal service-to-service estates.

Browser behavior

Under **Advanced**, and more useful than its name suggests — this is not a section to skip.



Language	The locale used for runtime execution.
Open Shadow DOM	Inspect and interact with elements inside Shadow DOM.
Enable video recording	Record video of on-demand runs. May impact performance.
Enable screenshot capture	Capture screenshots during execution. Disable to speed up heavy tests.

Shadow DOM

If your application is built with web components — many design systems are — its controls may live inside Shadow DOM, where they are invisible to ordinary inspection. If Studio cannot seem to find elements that are plainly on screen, and your front end uses web components, turn this on before assuming anything else is wrong.

Video and screenshots

Screenshot capture is on by default, and that is what makes a run explainable afterward — the per-step screenshots described in Understanding a Test Run and Its Results come from it. Turning it off speeds up heavy tests, at the cost of the evidence you will want the day one fails.

That is a real trade, so make it consciously: disable screenshots on long data-heavy tests whose failures you diagnose another way, and leave it on everywhere else.

Video recording is off by default and covers on-demand runs. Turn it on temporarily when a failure only makes sense as motion — a flicker, a race, something that appears and vanishes. Turn it back off afterward; the performance warning is there for a reason.

Language

Set the locale when your application changes behavior by language — date formats, currency, right-to-left layouts, or translated text your assertions depend on. A test asserting on English text will fail against a browser that asked for German.

Execution presets

At project level only, under **Execution** → **Execution presets**:

> Execution presets are reusable variable sets that configure how tests run in this project. Select a preset when running tests or starting sessions to apply its variables automatically.

A preset is a named bundle of **variables** you can apply at run time — so the same tests can be pointed at different data or environments without editing them. Create one with **New Preset**; presets are listed with their name and creation date.

This is the mechanism to reach for instead of duplicating a test to run it against staging as well as QA.

A working order

1. **Cannot reach the application at all?** Basic authentication, certificates, mutual TLS, or a required header.
2. **Reaches it but a banner blocks everything?** Cookies.
3. **Finds the page but not the elements?** Shadow DOM first, then Missing element timeout.
4. **Everything works but slowly and flakily?** Page load timeout, then Timing model.
5. **Different data or environment per run?** Execution presets.

And as everywhere in Studio, you can ask instead of hunting:

```
This environment uses a self-signed certificate - allow invalid TLS certificates for this project.
```

```
Our app uses web components. Turn on Open Shadow DOM.
```