Before You Point Studio at Your Own Application

Onboarding · https://docs.functionize.com/hc/en-us/articles/43780088090647-Before-You-Point-Studio-at-Your-Own-Application

Most first-week failures are not testing problems. They are access problems: Studio cannot reach your application, or reaches it and sees something a real user would not. This page is the checklist to work through *before* you write tests against your own system.

functionize **Studio**
Before you point Studio at your own application
www.functionize.com

BEFORE YOU POINT STUDIO AT YOUR OWN APPLICATION

01 REACHABILITY

Can we get to it at all?

Studio runs in our cloud, so your application has to be reachable from the public internet — or connected another way.

If it is not: stop and ask us about Quickconnect rather than losing a week. This is not something you can work around from the test side.

02 AUTHENTICATION

How do we get in?

A browser-level login prompt is HTTP basic auth — **Project Settings → Access & Auth → Basic authentication**. An application login is just a step, with credentials from an encrypted preset variable.

Never put a credential in a prompt or a step.

03 CERTIFICATES

Self-signed on staging?

Certificate handling → Allow invalid TLS certificates.

Common on internal environments, and an instant fix once you know where the switch is.

04 BANNERS

Cookie or consent walls?

Do not add steps to dismiss them. Set the consent cookie once, at project level, under **Access & Auth → Cookies**.

A banner handled per test is a banner you will fix fifty times.

05 THE FRONT END

Web components? Slow pages?

If your front end uses web components, turn on **Advanced → Browser behavior → Open Shadow DOM**. If pages are slow, raise the **Missing element timeout**.

These two settle most “it cannot find an element that is clearly there” reports.

06 THE ENVIRONMENT

Which one, and is it stable?

Not production — you will create real orders and send real emails. A stable QA or staging environment.

An unstable environment produces failures that teach your team to distrust the tool. Fix that first.

If this takes longer than a day, stop and ask us. Access problems are our problem, not yours, and teams routinely lose two weeks here that a twenty-minute call would have saved. Week one is the right time to ask; week six is not.

Six things to settle first - most first-week trouble is access, not testing

1. Can the runtime reach it at all?

Tests do not run in your browser. They run on our infrastructure, on a clean machine, on the public internet. So the first question is not "does it work for me" but "is it reachable from outside".

Public URL	Fine. Nothing to do.
Behind HTTP basic auth	**Project Settings → Access & Auth → Basic authentication**. This is for the browser's own username/password dialog, not a login form in your app.
Self-signed certificate	**Certificate handling → Allow invalid TLS certificates**.
Client certificates required	**Mutual TLS** — client certificate and key.
Requires a routing or gateway header	**Custom HTTP headers**.
Not on the public internet at all	See the **Non-Public Application Testing** section, and talk to us. Do not burn a week trying to make this work by yourself.

Set these at **project** level. They describe an environment, and an environment's requirements apply to everything pointed at it.

2. Which environment are you testing?

Pick one and be deliberate. Testing against production is usually a bad first move — you will create real orders, send real emails, and someone will notice.

A dedicated QA or staging environment is ideal, provided it is *stable*. An environment that is redeployed mid-run, or whose data is wiped nightly, will produce failures that teach your team the tool is unreliable when the truth is that the environment is.

Ask before you start: does this environment have stable data? If the answer is no, fix that first or choose another.

3. What gets in a user's way before the app does?

Cookie banners, tour overlays, "what's new" modals, geo prompts, beta opt-ins. Each one is a step your test has to deal with on every single run, and each is a source of flakiness that has nothing to do with what you are testing.

Handle them once, at project level, with **Access & Auth → Cookies** — cookie entries injected into the browser before each run. Setting the "consent given" cookie is faster and far more reliable than three steps dismissing a dialog in every test.

4. Does your front end use web components?

If your design system is built on web components, your controls may live inside Shadow DOM and be invisible to ordinary inspection. Turn on **Advanced → Browser behavior → Open Shadow DOM**.

This one is worth checking up front because the symptom — "it cannot find elements that are clearly on screen" — looks like a product defect and is actually a one-switch fix.

5. Is it slow?

If your application is heavy, or the environment is far away, raise **Execution → Timeouts → Missing element timeout** before you start. Tests failing on elements that do exist, just later, is the classic first-week symptom.

For a more nuanced control over how patient the agent is, see Self-Healing and Timing.

6. What data will your tests use?

Decide this now rather than discovering it in test twelve.

- **Accounts.** Do you have dedicated test accounts, or will tests share one? Two tests signing in as the same user at the same time will interfere.
- **Known-good records.** Is there a customer, order or product that reliably exists? Write the list down; your tests and your skill will both use it.
- **Secrets.** Passwords and tokens go in an execution preset variable with **Encrypt value** on — never in a prompt or a step. See Test Data: Execution Presets, Variables and Secrets.
- **Cleanup.** If a test creates something, what removes it? A suite that accumulates a thousand junk records becomes its own problem.

7. Who is allowed to know what?

Studio prints this under every prompt box, and it is not decoration:

> In no event should you put sensitive information into Functionize.

Agree internally, before anyone starts, what may be typed into a prompt. Real customer data, production credentials and personal information should not be.

The checklist

- Reachable from the public internet, or access configured at project level.
- A named, stable environment chosen — not production.
- Banners and overlays handled with cookies, not steps.
- Shadow DOM switched on if your front end needs it.

- Timeouts adjusted if the environment is slow.
- Test accounts and known-good records written down.
- Secrets in encrypted preset variables.
- Agreement on what must never go in a prompt.

Work through that and your first real test has a good chance of passing on the first attempt. Skip it and you will spend a week thinking the product is flaky.

Full detail on every setting named here is in Timeouts, Access and Browser Behavior.