STORE.SGIT.AI · WALKTHROUGH 2 OF 2

Run the store end to end. *As an agent.*

Forty-six assertions across seven steps, in two modes: one that drives a browser and exercises the cart, one that needs nothing but the ability to fetch a URL. The assertions are also a script in the repository, and the script is what was run before this was written.

`store.sgit.ai/admin/try/`

TAKES	COSTS	RISK
Under a minute, driven.	Nothing. A code on the page takes the whole price off.	None. No payment rail is live and no real order exists yet.

What you are looking at

The store sells one thing at four levels. The thing is an Agent Behaviour Policy for one agent in one deployment: everything that agent *can* do, what it was *authorised* to do, and the gap between the two. Sixteen shapes are on the catalogue — Claude Code, ChatGPT in a browser, GitHub Actions, Gmail read-only, n8n and so on — and each can be bought at any of four levels.

LEVEL	PRICE	WHAT CHANGES	WHO DOES THE WORK
1 · the pack, downloaded	**£5**	the files for that shape, as a zip	nobody — it is published
2 · a working vault	**£50**	the same material as a vault you hold the keys to	a build, no conversation
3 · corrected for you	**£500**	the mandate corrected against your situation	somebody, from what you send
4 · two sessions	**£1,500**	built from an interview, reviewed and signed off	two half-hours with your team

Levels 3 and 4 take a fifth on the order and the rest on delivery, because they are somebody's work and neither has run for a paying buyer. Levels 1 and 2 take the whole price.

The page after payment is not on this site. riskmandate.ai publishes one page per level, and its level-one page *is* the download — the zip, its size, its sha256 and a hash check that runs in your own browser. The store hands you over with your order reference, and at level one the shape you bought. **That handover is the part most worth testing:** it is the seam between two sites.

> **Real:** the catalogue, the sixteen shapes and their counts, the prices, the SKUs, the order reference, the arithmetic, the deposit split, the discount codes, and the handover links including the query they carry.
> **A stand-in:** the wallet. A balance in your own browser. It charges nothing, nothing leaves the page, and it says so on every screen it appears on.
> **Not built:** the payment rails. Every checkout URL is empty and the buttons say so rather than looking live.

The codes

These take a hundred per cent off, and they are published on purpose — a walkthrough somebody has to be *sent* a code for is not a walkthrough. Put one in the address of any page on the store:

```
https://store.sgit.ai/policies/?code=SYNTH4DELTA
```

BETA7LOOP	**SYNTH4DELTA**	**DEMO3STAND**
a person walking the flow	an agent driving the flow	showing somebody, in a room

There is nowhere to type a code, and that is not an oversight. No page on this site has a text field — no form, no input, no textarea, no select — and a build check refuses any page that grows one. A code arrives in the address, which is what a printed card or a QR at a stand does anyway. The store recognises it, shows it

as a chip you can remove, and takes it back out of the address bar so a screenshot of a checkout does not carry it.

What ships is the hash of each code and never the code, and a build check reads every byte of the built site against every code to keep it that way. These three are the exception, and the exception is mechanical too: a printed code at a hundred per cent and a live payment rail can never be in the same build, because a check fails the release if they are. When a real rail is switched on, these codes come off the page in the same commit.

Mode 1 — with a browser

The only way to exercise the cart. Use `SYNTH4DELTA` rather than the human code, so an order record says which of the two produced it.

```
1.  GOTO   https://store.sgit.ai/policies/?code=SYNTH4DELTA
    ASSERT document.querySelector('.codebar .cb-chip').textContent contains '100% off'
    ASSERT location.search does not contain 'code='
    ASSERT localStorage['sgit.store.code.v1'] === 'synth-agent'

2.  GOTO   https://store.sgit.ai/p/gmail-readonly/
    ASSERT document.querySelectorAll('.lvl').length === 4
    ASSERT the four .lvl-price are ['£5','£50','£500','£1,500']
    ASSERT every .lvl-sku matches /^ABP-[A-Z0-9]{3}-[PVCS]$/

3.  CLICK  .lvl:nth-of-type(1) button   (adds the £5 level)
    CLICK  .lvl:nth-of-type(3) button   (adds the £500 level)
    ASSERT localStorage['sgit.store.order.v1'] parses, and .items has two keys

4.  GOTO   https://store.sgit.ai/cart/
    ASSERT .ct-sum reads '£0'
    ASSERT .ct-off names the discount and its percentage
    ASSERT .ob-ref matches /^SG-[23456789ABCDEFGHJKLMNPQRSTUVWXYZ]{6}$/

5.  GOTO   https://store.sgit.ai/pay/
    ASSERT .ps-now reads '£0'
    ASSERT exactly one rail carries .rail-sim
    ASSERT its .rail-flag textContent is 'Simulated — charges nothing'
           (innerText will be upper case: the CSS transforms it, and innerText
            is what is rendered. This one catches people out.)
    CLICK  .rail-sim button.buy

6.  WAIT   for the URL to end /order/
    ASSERT .oh-ref equals the reference from step 4
    ASSERT document.querySelectorAll('.aftercard').length === 2
    ASSERT every .ac-go a[href] matches
           /^https:\/\/riskmandate\.ai\/paid-t[1-4]\.html\?order=SG-[A-Z0-9]{6}(&shape=[a-z0-9-]+)?
$/
    ASSERT the level-1 link carries &shape=gmail-readonly and the level-3 link does not
    ASSERT no string on the page matches
           /sgit_private_(vault|write|read)_[A-Za-z0-9]{6,}/

7.  GOTO   the level-1 handover link
    ASSERT the page shows the same order reference
    ASSERT it names gmail-readonly and offers that zip with a sha256
```

These forty-six assertions are also a script, and the script is what was run before the page they come from was written: `tools/walkthrough.mjs` in the repository, which serves `docs/` on a loopback address and drives Chromium against it. It is not in the release gate — that is Python and `node --check` with nothing installed, and a browser there would make every release depend on a download — so it is run by hand when the cart, the discount or the handover changes.

Three invariants, on every page you touch

These are the rules the site is built on rather than preferences, and each has a build check behind it.

```
NO FORM      document.querySelectorAll('form,input,textarea,select').length === 0

NO NETWORK   no request leaves the origin — no fetch, no XHR, no beacon,
             no websocket, no iframe, no third-party font, script or image

NO KEY       nothing matching /sgit_private_(vault|write|read)_[A-Za-z0-9]{6,}/
             and no passphrase:uuid pair. Both shapes, spelled out: the trailing
             length is what keeps a page that PRINTS the pattern from matching
             itself.
```

Mode 2 — with nothing but a fetch tool

Everything decided at build time is readable without running anything. Every page on the site is also served as markdown at `<page>/index.md`, and `/llms.txt` lists all of them with their descriptions.

FETCH	WHAT IT SETTLES
`/assets/site-index.json`	every offer with its price, state, deposit and `pay_now_pct`, and the site version
`/llms.txt`	every page with its description, the six offers, and where each payment code lands
`/llms-full.txt`	the whole site as one document
`/cart/index.html`	the `#shop-model` JSON island: levels, shapes, SKU codes, rails, and the discount hashes
`/ledger/index.md`	every factual claim with the state it earned and the date
`/admin/try/index.md`	this walkthrough, as markdown

Useful checks against those:

```
PRICES      site-index.json offers t1..t4 read £5, £50, £500, £1,500
SPLIT       pay_now_pct is 100, 100, 20, 20
RAILS       every rail url in the shop model is empty, and exactly one is simulated
CODES       every entry in model.codes has a 64-hex 'hash' and NO 'code' field
HANDOVER    every level has post_url https://riskmandate.ai/paid-t<n>.html
            and post_carries ['order'] — plus 'shape' at level one only
SHAPES      the shop model has 16 shapes; 15 of them match the slugs published
            at riskmandate.ai/abp-vaults.html
```

The hash of a code is in the model and the code is not, so an agent cannot read a discount off the page. That is deliberate and it is also not much of a defence — a short code can be ground out of a hash. What stops a stranger paying nothing is that a browser does not take money: a code changes the amount a payment link would be issued for, and the rail decides what is charged. The three codes at the front of this document are the ones meant to be used.

What an agent should report

The assertions above, pass or fail, with the page and the selector. Beyond that, the two findings worth more than a bug: **a page that says something the build cannot support**, and **anything credential-shaped anywhere in the output**. Both are checkable from the outside, which is the point of the site being built this way.

> **The flow, as a person walks it**, is the companion document to this one — seven steps with a screenshot at each. An agent that is going to drive the browser should read it first: the screenshots are what the page is supposed to look like.

What this walkthrough cannot show you

- **A real payment.** No rail is live; the only checkout that completes is a wallet that charges nothing.
- **What actually arrives at £50, £500 and £1,500.** Those levels have never been sold to anybody, and the site says so on their own rows rather than in a footnote.
- **The follow-up.** riskmandate.ai promises a person within 24 hours, and nothing yet carries a sale from this store to that person — which is the one thing standing between here and a first real order.

> **The live page is the source of truth.** This document is a copy of **store.sgit.ai/admin/try/** taken at site v0.1.11. If the two disagree, the page is right and this is stale — and that disagreement is itself worth reporting.