GUIDE 2

Platform validation

Seven checks that prove the platform is sound before you trust it.

H3, the Hackathon as a Service Harness
303 Overwatch A.S.B.L

Generated 29 August 2026 from `docs/guides/02-validation.md` in the H3 repository. The markdown is the source; edit that, not this document.

Contents

Proving the platform is sound before you trust it with real work.

Seven checks, each with a stated pass condition. Allow about thirty minutes. Run them again after any upgrade.

Validation is not the same as setup. Setup gets it working; validation establishes that what it produces can be defended to somebody who was not in the room. Skip it and your first real exercise doubles as your first test.

Why validate a tool at all

Because the output is evidence. An exercise record is offered to an auditor, a regulator or an employer, and any of them may ask how you know the tool did what it claims. These seven checks are the answer, and they are cheap to re-run.

Record the date, the version and the result. `python3 -m h3.cli --version` gives you the version.

Run the suite with warnings promoted to errors. A warning your interpreter happens not to print is still a defect, and the version on your machine decides which ones you see. Two leaked HTTP responses were invisible on Python 3.10 and reported on 3.14.

```
python3 -W error::ResourceWarning -m unittest discover -s tests -t tests
```

Verify. The last line reads `OK`, with no failures and no errors. A few tests report as skipped, which is expected and is explained by the skip reason each one prints. The test count rises as H3 grows, so the count is not the pass condition: `OK` is.

Check 1. The test suite

```
export PYTHONPATH=.
python3 -m unittest discover -s tests -t .
```

Pass: every test passes, in under a second, with no network.

What it proves. The identifier grammar, the validators, the adapters and the data boundary all behave as specified on this machine, not just on ours.

Check 2. Determinism, with no model

```
rm -rf /tmp/val && python3 -m h3.cli init --project /tmp/val \
  --platform "Demo Orbital" --statement "An operator."
python3 -m h3.cli f01 --project /tmp/val --adapter echo \
  --decided-by "Your Name"
python3 -m h3.cli show --project /tmp/val
```

Pass: ten elements, and the counts read PCE 2, SEG 3, SVC 2, AST 3. Run it twice and get the same answer both times.

What it proves. The pipeline is deterministic where it claims to be. Any variation you see later comes from the model, which is the only place variation belongs.

Check 3. The validators actually refuse

```
python3 -m h3.cli validate --project /tmp/val
```

Pass: `PASS no violations`.

Then prove the opposite. Open the CONOPS file, change one element's tag to `ZZ`, and validate again.

Pass: it fails, names the element, and cites `V002` with the list of tags that would have been valid. You will also see two `V009` violations naming that element's children: retagging it orphaned them, and the validators noticing that is the correct second-order result rather than noise.

Now change the tag back to what it was, and validate again until it reports `PASS`. Checks 5 and 6 read this same record, and a record left deliberately broken makes them report on damage you introduced on purpose. Do not skip this line: it is the one readers skip.

What it proves. The validators are load bearing rather than decorative. A checker that never refuses anything has never been tested.

Check 4. The data boundary refuses egress

```
H3_ANTHROPIC_KEY=not-a-real-key python3 -m h3.cli f01 \
  --project /tmp/val --adapter anthropic --layer PCE --decided-by "Your Name"
```

Pass: it refuses, names the data class as ORG, and says `Nothing was sent`.

Note that a key is present in that command. The refusal is not caused by a missing key; it is caused by the policy.

What it proves. Your platform description cannot reach a third party by accident, only by an explicit per-command decision.

Check 5. The analyst data has no dangling edges

```
python3 -m h3.cli maltego --project /tmp/val
```

Pass: the entity count equals the element count, and every `Source` and `Target` in `links.csv` appears as an `ETEN` in `entities.csv`.

What it proves. The graph an analyst opens describes the platform you enumerated, with nothing pointing at something that does not exist.

Check 6. The package is checkable

```
python3 -m h3.cli package assemble --project /tmp/val --decided-by "Your Name"
python3 -m h3.cli package verify --project /tmp/val
```

Pass: `verify` reports `PASS`, every SHA-256 matching.

Then prove it refuses. Change one byte in a packaged file and verify again:

```
echo "edited" >> /tmp/val/package/tooling/record/decisions.md
python3 -m h3.cli package verify --project /tmp/val
```

Pass: it reports `FAIL`, names that file as `CHANGED`, and prints both the hash in the manifest and the hash on disk. Exit status 1.

Re-assemble to leave the project clean:

```
rm -rf /tmp/val/package
python3 -m h3.cli package assemble --project /tmp/val --decided-by "Your Name"
```

What it proves. A handover nobody can check is a handover on trust. This is the check that turns the manifest from a list into evidence.

Read `package/README.md` while you are here. On a record built with `--adapter echo` it says the package is SAMPLE DATA, which is the qualification travelling with the package rather than being printed once to a terminal nobody kept.

Check 7. Offline

Disconnect the host from the network. Re-run checks 1, 2, 3, 5, 6 and 7.

Pass: all of them behave identically.

What it proves. The claim that this runs in an isolated environment is tested rather than asserted. This is the check people skip and then discover on the morning of an exercise.

Recording the result

Write the date, the H3 version, the checks run and the outcome into your own records. If you are running H3 as part of a regulated programme, that record is the tool-qualification evidence a reviewer will ask for, and it is far easier to produce now than to reconstruct later.

Next

Guide 3, Operations, runs your first exercise end to end, and closes a record that survives being read.