The Bricks Have to Fit

A Builder Notes log on Tool Builder, stable blocks, token economics and why Codex should not build before the building blocks exist.

DENNIS HEDEGREEN BUILD V1.0 HTTPS://HEDEGREENRESEARCH.COM/ARTICLES/THE-BRICKS-HAVE-TO-FIT/

I have not stopped building tools.

I have been building the thing that should make tools safer to build.

That distinction matters.

The public surface of Hedegreen Research can make the work look quieter than it is. If there is no new tool on the front page, it can look as if the tool work paused.

But some work does not appear as a new page.

Some work appears as a better floor.

A private Runtime Viewport proof for 24 Doors Real. The important line is the boundary: no public route, upload, publication or TID action was performed.

The important line in that screenshot is not the title of the tool.

It is the boundary.

No public route.
No upload.
No publication.
No TID action.

That is the work right now: building a private tool-making layer without confusing a local proof with a public release.

I am trying to build the internal Tool Builder before asking Codex to keep building more public tools on top of unstable pieces.

Codex should not build before the building blocks exist.

That sounds slower than vibe coding.

In the beginning, it is.

Vibe coding can be useful. I do not want to pretend otherwise. A lot of the early Hedegreen Research surface exists because I was willing to move quickly, follow the shape of an idea, and let AI help turn half-formed questions into working objects.

But if every tool is made from a new set of crooked blocks, the cost does not disappear.

It moves.

It moves into repair.
Into context.
Into lost decisions.
Into UI drift.
Into broken assumptions.
Into the next session where the machine has to rediscover what the previous machine improvised.

Fast is not always cheap.

Sometimes fast just moves the cost into maintenance.

Hedegreen Research cannot keep being built that way.

It has to become more stable and safer than anything I have built before in my life.

Not because I suddenly became less willing to experiment.

Because the experiments now have to survive.

Play Well

I am Danish, so eventually every infrastructure problem becomes a question of whether the bricks fit.

The LEGO Group describes its name as coming from the Danish words `LEG GODT`.

Play well.

That is usually treated as the charming part of the story. But the part I care about is not only the play.

It is the mould.

Play is only possible because the bricks are precise.

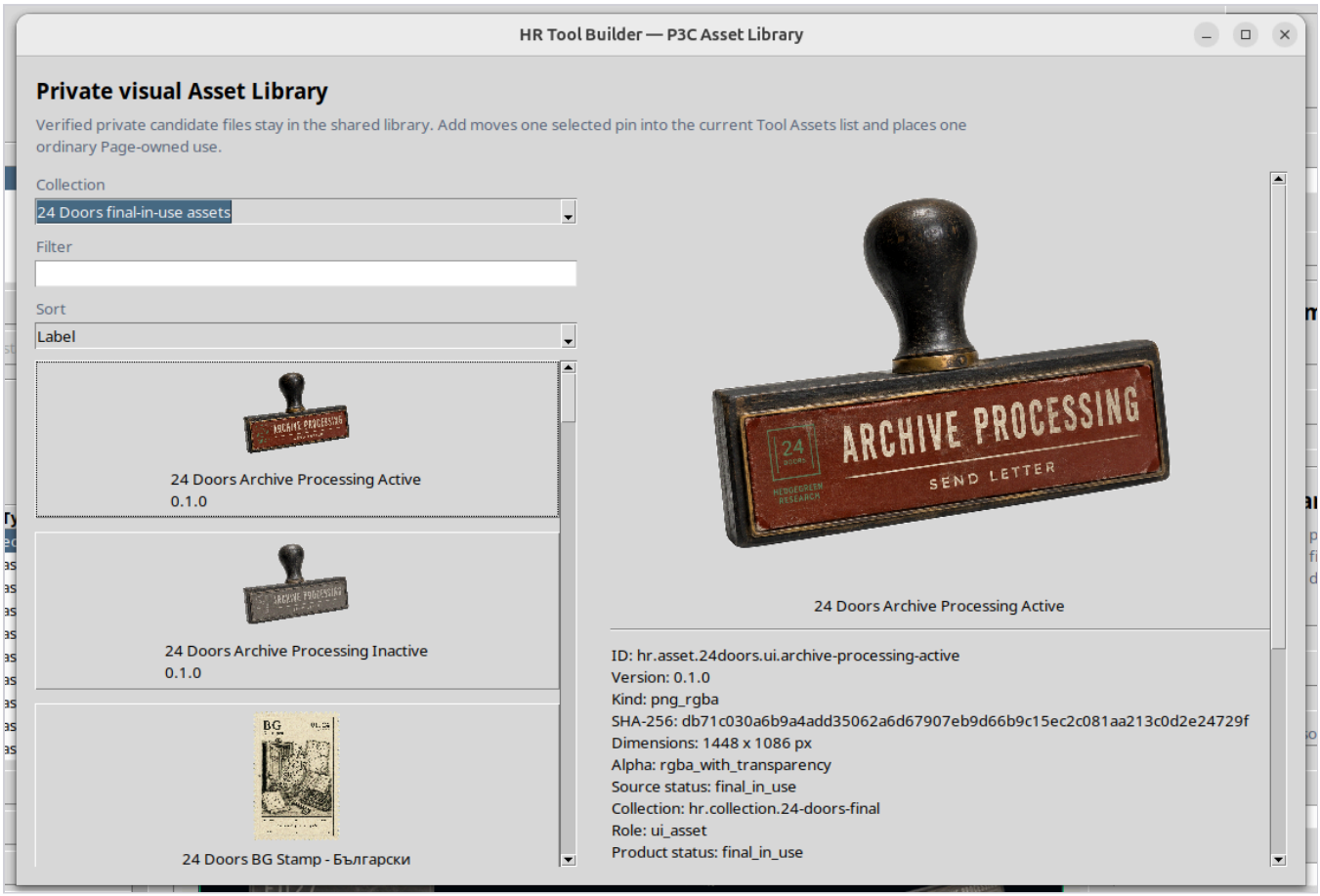
The piece has to fit. Not just once. Not just in the first box. Not just in the hands of the person who made the original set.

It has to keep fitting later.

That is the philosophy I want to borrow for Hedegreen Research.

Not the toy.

The compatibility contract.



The private asset library treats visual pieces as versioned, inspectable blocks, not loose decorations.

A component should still make sense when another tool uses it.
A package should still describe what it contains after the session is gone.
A receipt should still explain what happened when I no longer remember the conversation that produced it.
A version should not silently erase the old one.
A public tool should not depend on a private improvisation nobody can inspect.

If the blocks do not fit, Codex can still build something that looks tall.

For a while.

Then the tower starts asking for tokens just to remain upright.

Tokens Are Part of the Architecture

This is not only an engineering preference.

It is an economic one.

I have always known that Hedegreen Research is in a race against money.

Rent is part of that.
Tokens are part of that.
Compute is part of that.
The wider economy of AI access is part of that.

I do not know how long manual AI work will remain affordable.

I do not know whether the current level of access will become cheaper, more expensive, more limited, more controlled, or simply more uneven.

So I am building motors.

Every stable block is a future token I do not have to spend twice.

If a button can be a known object, Codex does not need to invent a button.
If a runtime action has a contract, Codex does not need to guess the state model.
If a tool package can be built deterministically, Codex does not need to remember the whole tool from chat history.
If the UI can be corrected by me inside the Tool Builder, I do not need to ask AI to repaint the whole interface every time something is slightly wrong.

That is the practical reason this matters.

The Tool Builder is not only an internal developer convenience.

It is an access strategy.

It is a way to make the expensive part happen less often.

Why the Tools Went Quiet

The next public tool I want is not just another static page.

I want a small test tool that can eventually live inside TID and be updated often.

But TID also has to grow up for that.

It cannot only be a place where a tool appears as a loose HTML page.

If Tool Builder is going to matter publicly, TID has to understand tool identity.

Tool IDs.
Packages.
Versions.
Receipts.
Update history.

It has to know that a tool is not just the current surface.

It is a line of controlled changes.

That is a future infrastructure task, not something I am starting inside this article.

But it has to be named here, because it explains why I am not rushing the next public tool out as if the old surface is enough.

The old way worked until it did not.

It let me move.
It let me publish.
It let me prove that an idea could become an instrument.

But the next phase cannot depend on one-off construction forever.

The Builder Before the Tool

The current Tool Builder split is simple.

Prototype Tool is the testbench.

24 Doors Real is the production candidate.

That distinction exists because I do not want the real tool to become the messy place where every motor experiment happens.

First the blocks have to work in the testbench.

Then they can be used deliberately in the product.

That is slower than letting Codex keep patching the visible thing.

It is also the only way I can see this becoming durable.

Hedegreen Research is not trying to become a collection of impressive one-off demos.

It is trying to become a public working system.

That means the internal work matters even when it is not visible yet.

The receipts matter.
The gates matter.
The boring package boundaries matter.
The refusal to publish from a half-held state matters.

The public work should look alive.

But underneath it, the bricks have to fit.

Build Log

This article is being written inside the same economy it describes.

The Tool Builder work is being done inside the same economy too.

The usage indicator is part of the workshop.

It was 44% remaining.

Now it is 42%.

Enough to keep going for a while.

Not enough to pretend the meter is not there.

That is not a tragedy. It is not even a complaint by itself. It is just the condition of the work showing up in the work.

The previous reset article treated the usage meter like a weather report for the work.

This one treats it as a build-stop condition.

I have spent a lot of access in less than an hour.

That matters because this is not only an article about Tool Builder.

It is also a live build log from the edge of the Tool Builder process.

If I keep building because there is still a little access left, I risk turning the article against its own point.

The point is not to build until the tower bends.

The point is to build the blocks well enough that I can stop, return, and keep going without losing the shape.

So this is where the first build log stops.

The right move is not to force another Tool Builder step.

The right move is to leave the build in a state I can return to.

I cannot build right now.

Source note. The LEGO name and brick-history references are based on LEGO Group history pages about the name, the 1958 brick and the stud-and-tube principle. The Tool Builder screenshots are private local workshop evidence, not public release evidence for 24 Doors Real.

RELATION MEMORY

BUILDS ON Reset Me, Baby, One More Time · Turns the usage meter from a reset/workflow condition into a build-stop condition.