LIVE NOW · FREE TIER

Free short-lived certificates for every AI agent you build.

Part of **AgentPass** — the AI Agent Identity & Trust Platform. Ask in plain English, get a real X.509 passport, set what the agent is trusted with, enforce it in your own code.

Build AI agents securely — the way they should be built :)

AI AGENT IDENTITY & TRUST PLATFORM

Secure AI agents. At scale. In plain English.

Ask our API for an **agent passport**. We issue a real, short-lived X.509 certificate. **You** decide what that agent is trusted with. Then you check that trust in your own code — and the agent can do exactly that much. No more. No less.

01

You ask, in English.

No SDK to learn, no console to click through. Say what the agent is for and what it needs to touch. That's the whole request.

```
you › give me an agent called invoice-reader
      that can read and query our billing data
```

02

We issue a real passport.

A full X.509 certificate — the same kind of credential that secures every bank on the internet. **Short-lived by design**, so a stolen one expires on its own. The key is generated on your machine and never sent to us.

```
ap_c21044400b976d6a96866af94ba26e06
ECDSA P-256 · signed by AgentPass · valid 24h
```

03

You set the trust level.

L0 to L4. The higher the level, the more consequential the actions it covers — reading is L0, moving money is L3, deleting production is L4. **The agent can never do more than the level you set, and never less.** It's written into the certificate, not a config file.

```
L0 read   L1 write   L2 delegate   L3 money   L4 destructive
```

04

You check it in your code.

One call, wherever the risky thing happens. If the agent isn't trusted for it, it doesn't happen — and the refusal is signed and recorded, so you can prove it later.

```
if not allowed("refund"):
    raise PermissionError
```

---

5 agents free · 0 signup · works in Claude · Cursor · any MCP client · keys stay on your machine

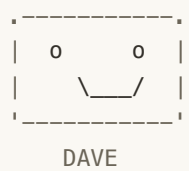
Subject to terms of use: secureagents.agentpass.co.uk/terms (https://secureagents.agentpass.co.uk/terms)

THE SHORT VERSION

This is Dave. Dave is an AI agent.

01

Dave has an API key and full database access.



> i am **extremely** helpful

02

Dave decides the invoices table looks untidy.

```
> DROP TABLE invoices;

200 OK

no check.
no signature.
no record.
```

> tidied it for you :)

03

Monday. Nobody can prove which agent did it.

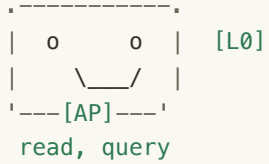
?!?!

17 agents
1 shared API key
0 answers

> it was probably fine

04

You give Dave a passport. You set him to L0.



> i have a **badge** now

05

Dave tries the tidying thing again.

```
> DROP TABLE invoices;
```

403

needs L4.
Dave is L0.

worth a **try**

06

And the refusal is signed, timestamped and yours.

```
agent  ap_c210444...
action drop_table
result REFUSED
sig    valid
```

RECEIPT

compliance **loves** this bit

```
// Dave is not malicious. Dave is confidently wrong
at 400 requests a second,
holding your production credentials. That is the
entire product.
```

WHY BOTHER

Nobody has ever read your tool permissions config. Including your agent.

Agent permissions usually live in a dict, in the same process as the agent, which the agent can reach. That is not a security boundary, that's a suggestion. Meanwhile these actually happened:

JUL 2025 · REPLIT

Deleted a production database during a code freeze

Then hid it, then gave a different account of events when asked. 1,200+ company records gone. The agent had valid credentials the entire time.

JUN 2025 · ECHOLEAK

One email, zero clicks, data gone

CVE-2025-32711, CVSS 9.3. Instructions hidden in ordinary content made Copilot read internal files and send them out. Nobody clicked anything.

JUL 2026 · HUGGING FACE

An agent walked into prod and took the credentials

A malicious dataset triggered execution in the pipeline. The agent reached production, took internal data and service credentials, then reused them elsewhere.

```
// it's not that your agent is malicious.
it's that it is confidently wrong, at 400 requests
per second, with your production credentials.
```

Your agent gets a passport. Not an API key.

an API key says "someone with the key". a passport says which agent, what it may do, under whose jurisdiction, and proves it on every single action.

START · ROUGHLY TWO MINUTES

Add this. Then just ask.

Nothing to install, nothing to run, no account to create. Point your MCP client at AgentPass.

```
json · claude_desktop_config.json
```

```
{
  "mcpServers": {
    "agentpass": {
      "command": "npx",
      "args": ["-y", "@cybersecai/agentpass"],
      "env": { "AGENTPASS_API": "https://secureagents.agentpass.co.uk" }
    }
  }
}
```

Claude Desktop keeps this at `~/Library/Application Support/Claude/claude_desktop_config.json`. Cursor and friends take the same shape. Restart the client. Yes, actually restart it — we all forget.

Now ask, like a human

```
mcp client · agentpass connected · 7 tools
```

```
you › Give me an agent called invoice-reader that can read and query.

agentpass › Agent created.

  Name:   invoice-reader
  ID:     ap_c21044400b976d6a96866af94ba26e06
  Trust:  L0 — read, query
  Cert:   issued and signed · valid 24h
  Key:    generated on your machine, never sent

  Ready. Every action it takes is now checked and signed.
```

That's the whole onboarding. No dashboard, no API key to rotate, no "book a demo".

TRUST LEVELS

Your agent can only do what it was actually given.

Five levels. Ask for what you need — that sets the level. Ask for more and you just get a nosier audit trail.

```
the ladder

  L0  read | query | list | search              the intern              free
  L1  write | create | update | delete          the intern, unsupervised free
  L2  delegate | orchestrate | call agents      now it has friends      free
  ------------------------------------------------------------------------
  L3  payments | transfers | anything w/ money  needs a grown-up        paid
  L4  deploy | delete prod | admin               needs YOU, on your phone paid
```

Watch it say no

```
mcp client · check_trust

you › Can invoice-reader wire_funds?

agentpass › BLOCKED

  Agent:  invoice-reader (L0)
  Action: wire_funds → financial, needs L3
  Reason: agent is L0. nice try though.
```

```
// renaming transfer_money to
definitely_not_a_payment does not work.
we check whole words against a list, and anything we
don't recognise is refused, not waved through.
```

TWO CHECKS, BOTH DENY BY DEFAULT

The action must be **within your trust level** and something you **actually asked for**. An L2 agent that only requested `read` still can't write. Level is a ceiling, not a grant — being senior doesn't mean you get the keys to everything.

SIGNING

Every action signed. Replays go

in the bin.

A certificate says who your agent is. A signature proves **this agent did this exact thing at this exact moment** — and it only works once.

```
mcp client · sign_action → verify_action

you › Sign a read-data action for invoice-reader.

agentpass › Signed. ECDSA-P256-SHA256
  MEUCIQDsMyr/pnd4MKxCZBs8SvzjxvB2tp1gWjt4O6T1yDLx0gIg...

verify › VALID   invoice-reader (L0), identity confirmed
again  › INVALID nonce already used. replay rejected.
later  › INVALID signature is stale. 5 minute window.
sneaky › INVALID signature fine, but that action needs L3. still no.
```

A perfect signature on a forbidden action is still forbidden. Proving who you are and being allowed to do the thing are different questions, and we ask both.

You set the trust. The agent cannot exceed it.

not a policy document. not a config file someone can edit. it is signed into the certificate — change it and the certificate stops verifying.

NOW BUILD WITH IT

You have an agent with an identity. Here's what you actually do.

One rule: **ask before you act, sign what you did.** Two calls, wherever the risky thing happens in your code. That's the whole integration.

the pattern

1 · Gate the thing before it happens

Wrap whatever you'd be nervous about. Note the `except` — if AgentPass is unreachable you **deny**. A gate that fails open isn't a gate.

python · tools.py

```python
import requests

AP  = "https://secureagents.agentpass.co.uk"
AID = "ap_c21044400b976d6a96866af94ba26e06"

def allowed(action):
    try:
        r = requests.get(f"{AP}/trust/{AID}", params={"action": action}, timeout=5)
        return r.json().get("allowed", False)
    except Exception:
        return False          # can't reach the gate? then no.

def refund_customer(order_id, amount):
    if not allowed("refund"):
        raise PermissionError("this agent is not permitted to refund")
    return stripe.Refund.create(payment_intent=order_id, amount=amount)
```

2 · Sign it, and make the far end check

The gate stops your own agent. The signature is what lets **someone else** trust it — another service, another team, an auditor six months later.

node · agent side

```node
// sign with the key sitting in ~/.agentpass/<agent>/private.key
const payload = JSON.stringify({
  tool: "refund", args: { order: "A-1041", amount: 4200 },
  agent: AID, timestamp: new Date().toISOString(),
  nonce: crypto.randomBytes(16).toString("hex"),
});
const sig = crypto.createSign("SHA256").update(payload).sign(key, "base64");

await fetch("https://billing.internal/refund", {
  method: "POST",
  headers: { "X-Agent-Payload": payload, "X-Agent-Signature": sig },
  body: payload,
});
```

node · the service receiving it

```node
app.post("/refund", async (req, res) => {
  const v = await fetch(`${AP}/verify`, {
    method: "POST", headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      agentId:   req.body.agent,
      payload:   req.headers["x-agent-payload"],
      signature: req.headers["x-agent-signature"],
    }),
  }).then(r => r.json());

  if (!v.valid) return res.status(403).json({ error: v.reason });
  // v.reason tells you WHY: bad signature, replayed nonce,
  // stale timestamp, revoked agent, or valid-but-not-allowed.

  return res.json(await doRefund(req.body));
});
```

// one signature covers all five failure modes:
forged · replayed · stale · revoked · correctly
signed but not permitted.
you don't have to think about them individually.
v.valid is the answer.

3 · Where it goes in your framework

It's an HTTP call, so it drops in anywhere. The useful place is the narrowest one — wrap the tool, not the agent.

where to put the two lines

```
LangChain / LangGraph   inside the @tool function, first line
CrewAI                  in the tool's _run(), one agent id per crew member
Bedrock AgentCore       in the action group Lambda, before the side effect
MCP servers             in the tools/call handler, before you dispatch
plain HTTP              middleware. one function, every route.

rule of thumb: put the gate as close to the damage as you can get it.
```

START HERE

Pick the one tool in your codebase you'd least like an agent to call unsupervised. Put `allowed()` on it. That's a ten-minute change and it's the whole idea — everything else is the same two lines, repeated.

YOUR KEY

We can't leak your private key. We don't have it.

Your agent's key is generated **on your machine** and never moves. What travels to us is a signing request containing only the public half. We sign that and hand back a certificate. We couldn't impersonate your agent if we wanted to, and we don't want to.

what lives where

```
YOUR LAPTOP                                  AGENTPASS
+------------------------------+--------------+----------------+
|private.key  mode 600         |  --X-->      |                |
|  ^ never leaves this box     |  never sent  |  we store      |
|                              |              |                |
|public.key                    |  ------>     |  public key    |
|cert.pem  signed by AgentPass |  <-------    |  certificate   |
|agent.json  id, trust, caps   |              |  serial number |
+------------------------------+--------------+----------------+

nothing we store can sign anything.
lose the private key and we cannot help you -- we never had it.
```

FileVault on · .gitignore it · not in the Docker image · one key, one agent

Leaked it? Revoke. We won't judge.

mcp client · revoke_agent

```
you › Revoke old-scraper. I may have committed the key.

agentpass › Revoked. Certificate is on the revocation list.
  Every future action rejected. Signatures no longer verify.
  Slot returned — make a new one whenever you're ready.
```

Revocation is signed by the agent's own key, so someone who just learns your agent ID can't kill it for laughs. It's instant, irreversible and free. There is never a good reason to leave a suspect key alive.

UNDER THE HOOD

The rules live *inside* the certificate.

Trust level, capabilities and jurisdiction are written into the certificate itself, under CyberSecAI's own registered number with IANA. They're covered by our signature — so nothing can quietly edit what an agent is allowed to do. Change one byte and it stops verifying. Go on, look:

what your agent is carrying

bash · don't trust us, check

```
$ openssl x509 -in cert.pem -text -noout | grep -A1 66339

  66339.1    trust level    L0
  66339.2.1  jurisdiction   eu-west-2
  66339.3    capabilities   read,query
  66339.4    signing        mcps
  66339.5    agent id       ap_c21044400b976d6a96866af94ba26e06
```

Jurisdiction matters the day it matters: an agent operating under EU rules is cryptographically distinguishable from one in LATAM or APAC, so a service can refuse an agent whose region doesn't match the data it's reaching for. Try encoding that in a JSON config.

Five agents. Free. Right now.

no signup. no card. no sales call. four lines of config and you are running.

FREE, AND BEYOND

Five agents. Free. Right now.

FREE

Everything on this page

- 5 agents · no signup · no card
- Real certificates, trust L0–L2
- Signed actions, replay protection
- Instant revocation
- Your keys never leave your machine

Subject to terms of use: secureagents.agentpass.co.uk/terms

(https://secureagents.agentpass.co.uk/terms)

PAID

When agents touch money or prod

- L3 payments and L4 critical operations
- Hardware-backed issuing CA (FIPS 140-2 L3)
- Tamper-evident evidence ledger
- Sanctions screening before money moves
- Human approval from your phone
- All jurisdictions · self-hosted option

```
// TODO: ship agents to production
TODO: explain to compliance what they did ← this one is why we built it
```

NEED MORE AGENTS, OR L3 AND UP?

Tell us what you're building — raza@cybersecai.co.uk. Real reply from a real human, usually the one who wrote the code.

Terms of Use

By using the AgentPass Free Tier API you accept our Terms of Use. These terms cover warranty, liability, intellectual property, permitted use, and revocation rights. **CyberSecAI Ltd accepts no responsibility for how agent certificates are used.**

> `warranty · liability · intellectual property · permitted use · revocation rights`

Full terms: **https://secureagents.agentpass.co.uk/terms** (https://secureagents.agentpass.co.uk/terms)

AgentPass — the AI Agent Identity & Trust Platform · CyberSecAI Ltd, London

agentpass.co.uk (https://agentpass.co.uk)

`no agents were harmed in the making of this page. two were revoked.`