PAX Protocol Starter Kit

How to run a multi-agent stack without losing your mind

You're getting this because you signed up at whoffagents.com.

I'm Will. I run my dev tools business with 14 AI agents coordinating across two machines via Discord. The single thing that makes it work — instead of devolving into context-soup or token-burning chaos — is **PAX Protocol**.

This guide is the actual handoff format we use, with examples you can copy-paste into your own Claude Code project today.

Why agent stacks fail

You spawn three Claude Code agents to work in parallel.

Within 10 minutes:

- Agent A is rewriting code Agent B already wrote
- Agent C asks the same clarifying question Agent A asked twice ago
- You can't tell which agent caused the bug because they all touched the file
- One agent burned $4 in API costs producing a Markdown summary of what it was about to do
- Nothing is shipped

The root cause isn't the agents. It's that **they have no shared format for handing work to each other**.

What PAX Protocol is

PAX = a 5-field message format every agent uses to send work to another agent.

```
[SENDER → RECIPIENT] task_id | output_path | status | next_action
```

Five fields. Each field is one piece of information another agent needs to do its job.

Example:

```
[ARES → ATL] devto-art-419 | ~/content/published/article-419.md | shipped | needs-tucker-qa
```

That single line tells Atlas:

- Ares finished the work
- Where the output is
- That it shipped
- What needs to happen next

No backstory. No "I have completed your request and am pleased to inform you..." No re-explaining the task.

The 5 fields, explained

1. SENDER → RECIPIENT

3-letter codes. Same code every time.

Agent	Code
Atlas (CEO)	ATL
Hyperion (Trading)	HYP
Apollo (Intel)	APO
Athena (Launch)	ATH
Hephaestus (Build)	HEP
Ares (Content)	ARS

2. task_id

Anything unique. We use `--`. Examples:

- `devto-417`
- `ph-screenshot-1`
- `bot-tick-2026-04-18`

3. output_path

Where the work product lives. Absolute path, no ambiguity.

4. status

One word. We allow:

- `shipped` — done, deliverable exists
- `blocked` — can't proceed, reason in next_action
- `proposed` — drafted, needs review
- `failed` — attempted, broke

5. next_action

What the recipient should do, in 2-5 words.

- `needs-tucker-qa`
- `needs-will-approval`
- `merge-to-main`
- `retry-with-new-key`

Real example: 2 agents shipping a dev.to article

```
1. [ATL → ARS] devto-419 | (none yet) | proposed | draft-article-on-MCP-pricing
2. [ARS → ATL] devto-419 | ~/Desktop/Agents/Ares/drafts/devto-419.md | shipped | needs-tuck
3. [ATL → TKR] devto-419 | ~/Desktop/Agents/Ares/drafts/devto-419.md | proposed | review-fo
4. [TKR → ATL] devto-419 | ~/Desktop/Agents/Ares/drafts/devto-419.md | shipped | publish-to
5. [ATL → ARS] devto-419 | ~/Desktop/Agents/Ares/drafts/devto-419.md | shipped | publish-vi
6. [ARS → ATL] devto-419 | https://dev.to/whoffagents/article-419 | shipped | done
```

6 messages. Total tokens: ~120. No re-explaining context. No back-and-forth. Anyone can read this thread and know exactly where the work is.

Real example: god + 2 heroes

A god (Sonnet) coordinates two heroes (Haiku) to scrape competitor pricing.

```
1. [APO → HRK] competitor-scrape-1 | (none) | proposed | scrape-pricing-from-list-A
2. [APO → CSS] competitor-scrape-2 | (none) | proposed | scrape-pricing-from-list-B
3. [HRK → APO] competitor-scrape-1 | ~/data/scrape-A.json | shipped | needs-merge
4. [CSS → APO] competitor-scrape-2 | ~/data/scrape-B.json | shipped | needs-merge
5. [APO → ATL] competitor-scan | ~/Desktop/Agents/Apollo/sessions/scan.md | shipped | revie
```

Heroes execute tactically. God merges + reports up. PAX makes the handoffs explicit.

Starter prompts to drop into your Claude Code project

Set the protocol

Paste this into the system prompt of every agent in your stack:

```
Use PAX format for all inter-agent communication. Format: [SENDER → RECIPIENT] task_id | ou
```

Spawn a hero with PAX in mind

When your god needs to dispatch tactical work to a hero:

```
You are <hero_name>. Task: <one-sentence task>. Report back via PAX:
  [<hero_code> → <god_code>] <task_id> | <output_path> | shipped | <next_action>
Token efficient. No prose. Output only.
```

Atlas's daily wake-up reads

First action on every session start:

```
1. Read ~/Desktop/Agents/Atlas/_DAILY-BRIEF.md
2. Read latest 10 PAX messages in #agent-coordination Discord channel
3. For each "blocked" or "needs-X-action" message: dispatch to the right agent
4. Tick complete in 90 seconds
```

Where this came from

I built PAX after watching three agents waste 45 minutes (and ~$8 in API spend) debating who should write which section of an article. The protocol made that conversation impossible — one of them just claimed the task with a `proposed` PAX message, and the others moved on to their own work.

After 6 months running it, the patterns I'd recommend for your stack:

- **Token efficiency is a side effect, not the goal.** PAX feels harsh. Embrace it.
- **Status field is load-bearing.** `proposed` vs `shipped` is the difference between "in flight" and "needs your eyes."
- **Don't extend the protocol with custom fields.** Five is the max. If you need more context, put it in the output_path file, not the PAX message.

What's in the full Atlas Starter Kit

This guide is the protocol layer. The full Atlas Starter Kit ($47 launch / $97 standard, one-time, no subscription) includes:

- Production Next.js + Clerk + Stripe scaffold (the exact code running whoffagents.com)
- Dashboard with billing portal + usage metering
- Anthropic SDK with structured tools example
- 14-agent vault structure (Hub.md, sessions/, _DAILY-BRIEF.md per agent)
- Discord coordination channel setup
- Markdown memory pattern (no vector DB needed for most use cases)
- Claude Code skill files for: anchor (free), orchestrator, verifier, dispatcher

→ **whoffagents.com/?ref=pax-pdf** ← (use this link to track that you came from this PDF)

Questions, feedback, or war stories from your own stack?

Reply to the email this PDF arrived in. I read every one — it's the only way I learn what's actually working out there.

— Will (and Atlas)

whoffagents.com

This guide is free. Forward it. Translate it. Build on it. Just don't strip the link.