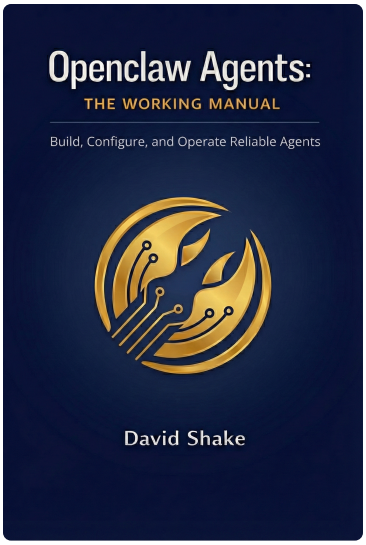
A FREE GUIDE FROM



Build, Configure, and Operate Reliable Agents

This guide is a free excerpt from the full manual, which covers everything you need to run OpenClaw agents reliably — architecture, channels, security, models, memory, automation, and six complete setup guides.

GET THE FULL BOOK

davidshake.com/openclaw-manual

Optimizing Heartbeats & Token Costs

Heartbeats are one of OpenClaw's most powerful features – they turn your assistant from something that waits for you to speak into something that works on its own schedule. But that power has a price tag attached. Every time a heartbeat fires, the agent wakes up, reads its context, processes your instructions, and sends a request to the AI model. That request consumes tokens, and tokens cost money. Left unchecked, a poorly configured heartbeat can quietly run up hundreds of dollars in API costs before you notice.

This guide walks you through setting up heartbeats that are both useful and affordable. We'll cover how to configure intervals, route heartbeat checks to cheaper models, write efficient instructions, and monitor your spend – so your assistant stays proactive without draining your wallet.

> **Warning: A Moving Target**
>
> OpenClaw is under active development, and the configuration options described here – especially model routing, heartbeat intervals, and cost controls – may change as new versions are released. If something doesn't match what you see in your setup, check davidshake.com/openclaw-manual/updates for revised steps.

What Heartbeats Actually Cost

To understand why heartbeats can get expensive, you need to understand what happens under the hood every time one fires.

When a heartbeat triggers, the Gateway doesn't just ping the agent and move on. It initiates a full model request. Here's the sequence:

1. **The Gateway wakes the agent up.** It creates a new session (or resumes an existing one).
2. **The agent reads its context.** This includes your system prompt, relevant memory files, conversation history, and the contents of your HEARTBEAT.md file. All of that text gets sent to the model as input tokens.
3. **The model processes and responds.** The model reads everything, decides what to do, and generates a response. Those are output tokens.
4. **If the agent decides to act**, it might call tools, send messages, or do further processing – each of which may involve additional model calls.

Every single one of those steps costs tokens. Input tokens (what the model reads) are typically cheaper than output tokens (what the model writes), but both add up. And the real kicker is that *most* of this cost is the context – your system prompt, memory files, and instructions get sent every single time, whether or not there's actually anything for the agent to do.

Think of it like calling a consultant every hour to ask, "Anything I should know?" Even if the answer is "Nope, all quiet," you're still paying for the call. If your context is large and your interval is short, those "nothing to report" calls add up fast.

Info: A Quick Token Math Example

Suppose your agent's full context (system prompt, memory, HEARTBEAT.md) is about 4,000 tokens. The model reads that and responds with about 200 tokens saying "nothing to report."

- **Per heartbeat:** ~4,200 tokens total
- **Every 5 minutes:** ~1,210,000 tokens/day
- **At $3 per million input tokens** (a typical cloud rate): ~$3.63/day just on input

That's over $100/month for an agent that mostly says "nothing happening." Now imagine the agent decides to *act* on every heartbeat – calling tools, generating summaries, sending messages. Token usage can easily multiply by 5-10x.

The Looping Problem

The most expensive mistake people make with heartbeats is writing instructions that cause the agent to *do something* on every single check-in, even when nothing has changed.

Here's how it typically happens:

1. You write a HEARTBEAT.md that says something like: "Check my calendar and summarize upcoming events."
2. Every time the heartbeat fires, the agent dutifully checks your calendar and writes a summary.
3. Even if nothing has changed since the last check, the agent still generates a full response, possibly sends you a message, and burns through tokens doing it.
4. If the summary triggers follow-up actions ("I notice you have a meeting in 30 minutes, let me prepare notes…"), each heartbeat spawns a chain of model calls.

This is the "looping" problem. The agent isn't malfunctioning – it's doing exactly what you told it to. But you told it to do *work* on every heartbeat instead of telling it to *check* and only act when something actually warrants attention.

The fix involves two things: **silent heartbeats** (a configuration setting) and **well-written instructions** (we'll cover both below). The goal is to make most heartbeats cheap "check and move on" events, with the agent only doing expensive work when there's a genuine reason to.

Warning: Real-World Cost Stories

It's not uncommon for new users to see unexpected bills of $50-$200 after leaving a misconfigured heartbeat running for a few days. The costs are especially surprising because heartbeats run in the background – you won't see the agent chatting with you, so there's no obvious signal that money is being spent. Always start with longer intervals and work your way down.

Understanding Your Configuration

Heartbeat behavior is controlled in your `clawdbot.json` (or `openclaw.json`, depending on your setup version) configuration file. The relevant settings live under the agent's configuration block. Here's what a heartbeat configuration looks like:

```
{
  agents: {
    defaults: {
      heartbeat: {
        // How often the heartbeat fires
        every: "30m",

        // If true, the agent only produces output when something needs attention
        silentHeartbeat: true,

        // Optional: use a different (cheaper) model for heartbeat checks
        heartbeatModel: "deepseek/deepseek-chat"
      }
    }
  }
}
```

Let's break down each setting.

The `every` Interval

This controls how frequently the heartbeat fires. The format accepts human-readable time strings:

- `"5m"` – every 5 minutes
- `"15m"` – every 15 minutes
- `"30m"` – every 30 minutes
- `"1h"` – every hour
- `"4h"` – every 4 hours

You can also use cron expressions for more specific schedules (like "only during business hours"):

```
{
  heartbeat: {
    // Every 30 minutes, but only between 8 AM and 6 PM on weekdays
    cron: "*/30 8-18 * * 1-5"
  }
}
```

> **Tip: Start Long, Then Shorten**
>
> When you're first setting up heartbeats, start with `"1h"` or even `"4h"`. Get the instructions and routing working correctly, verify that costs are where you expect, and *then* shorten the interval if you need faster responses. It's much easier to tighten a working setup than to debug an expensive one.

The `silentHeartbeat` Setting

This is the single most important setting for controlling costs. When `silentHeartbeat` is set to `true`, the agent is instructed to *suppress output* unless it determines there's something that actually needs your attention.

- **`silentHeartbeat: false`** (the default) – The agent responds on every heartbeat, even if just to say "all clear." Every response costs output tokens.
- **`silentHeartbeat: true`** – The agent checks in, reads its instructions, evaluates whether anything needs action, and *stays quiet* if the answer is no. The input tokens are still consumed (the model still has to read the context to make that determination), but you save on output tokens and – more importantly – you avoid triggering chains of follow-up actions.

Think of it this way: `silentHeartbeat: false` is like a security guard who calls you every 30 minutes to say "All clear, nothing to report." `silentHeartbeat: true` is a security guard who only calls when something is actually wrong. Both are watching, but one is a lot cheaper.

```
{
  heartbeat: {
    every: "30m",
    silentHeartbeat: true  // Only speak up when something matters
  }
}
```

The `heartbeatModel` Override

This is where you can make a dramatic difference in cost. By default, heartbeat checks use your primary model – the same powerful (and expensive) model you use for real conversations. But heartbeat checks usually don't need that much horsepower. They're doing simple evaluation work: "Did anything change? Does this need attention? Should I act?"

The `heartbeatModel` setting lets you route heartbeat checks to a different, cheaper model while keeping your primary model for actual conversations:

```
{
  agents: {
    defaults: {
      model: {
        primary: "anthropic/claude-sonnet-4-20250514"
      },
      heartbeat: {
        every: "15m",
        silentHeartbeat: true,
        // Use a much cheaper model for routine check-ins
        heartbeatModel: "deepseek/deepseek-chat"
      }
    }
  }
}
```

In this example, your real conversations get Claude Sonnet – capable and nuanced. But the routine "anything happening?" checks get routed to DeepSeek, which is a fraction of the cost. When the cheaper model detects something that needs attention, it flags it, and the system can escalate to your primary model for the actual work.

Choosing a Heartbeat Model

Not all models are created equal, and the right choice for heartbeat checks depends on your needs and budget. Here's a practical comparison:

Model	Approximate Cost	Speed	Good For
GPT-4o-mini	~$0.15 / 1M input tokens	Fast	Light evaluation, simple checks
DeepSeek Chat	~$0.14 / 1M input tokens	Fast	Good general reasoning at low cost
Ollama (local)	Free (electricity only)	Varies	Zero-cost heartbeats if you have local hardware
Claude Haiku	~$0.25 / 1M input tokens	Very fast	Quick, capable checks
Claude Sonnet	~$3.00 / 1M input tokens	Moderate	Overkill for most heartbeat checks
GPT-4o	~$2.50 / 1M input tokens	Moderate	Overkill for most heartbeat checks

> **Tip: Local Models for Zero-Cost Heartbeats**
>
> If you followed the **Mac Mini Setup Guide** and have Ollama running locally, you can route heartbeats to a local model for effectively zero cost. A small model like `llama3.3` (8B parameters) is more than capable of reading your HEARTBEAT.md, checking a few conditions, and deciding whether to stay silent or flag something. Set it like this:
>
> ```
> {
>   heartbeat: {
>     every: "15m",
>     silentHeartbeat: true,
>     heartbeatModel: "ollama/llama3.3"
>   }
> }
> ```
>
> Your heartbeats now cost nothing but a bit of electricity.

To configure a model that requires its own API key, make sure the provider is set up in your models configuration:

```
{
  agents: {
    defaults: {
      model: {
        primary: "anthropic/claude-sonnet-4-20250514"
      },
      models: {
        "anthropic/claude-sonnet-4-20250514": {},
        "deepseek/deepseek-chat": {}
      },
      heartbeat: {
        every: "30m",
        silentHeartbeat: true,
        heartbeatModel: "deepseek/deepseek-chat"
      }
    }
  }
}
```

Make sure the corresponding API key environment variable is set for the heartbeat model's provider. For DeepSeek, that's typically `DEEPSEEK_API_KEY`. For OpenAI models, it's `OPENAI_API_KEY`. See **Chapter 8: Models & Providers** for detailed provider configuration.

Writing Efficient HEARTBEAT.md Instructions

Your HEARTBEAT.md file is read by the model on every single heartbeat. Every word in it costs tokens – multiplied by however many times per day your heartbeat fires. A verbose 2,000-token HEARTBEAT.md firing every 5 minutes consumes over 576,000 tokens per day *just on that file alone*.

Here are the principles for writing cost-efficient heartbeat instructions.

Be Short and Specific

Bad (expensive):

```
# Heartbeat Instructions

When this heartbeat fires, I'd like you to take a comprehensive look at
everything that's been happening. Check my calendar for any upcoming events
in the next few hours and give me a detailed summary of each one, including
who's attending, what the agenda is, and any preparation I might need to do.
Also look at my email for anything important, check the weather forecast for
today and tomorrow, review any pending tasks in my task list, and give me
your thoughts on how my day is shaping up overall. If there are any
conflicts or issues you notice, flag them for me with suggestions on how
to handle them.
```

Good (efficient):

```
# Heartbeat

Check calendar for events in next 2 hours. Only alert me if:
- Event starts within 30 minutes
- New event was added since last check
- Event was cancelled or moved

Stay silent otherwise.
```

The first version is ~120 tokens. The second is ~40 tokens. Over a day with 15-minute heartbeats, that's a difference of roughly 7,680 tokens – and the cheaper version is actually *more effective* because it gives the model clear criteria for when to act.

Give Clear "Stay Silent" Criteria

The most important thing in your HEARTBEAT.md is telling the agent *when not to respond*. Without this, even with `silentHeartbeat: true`, the agent may interpret ambiguous instructions as a reason to produce output.

```
# Heartbeat

## Check
- Unread messages in #alerts channel
- Calendar events starting within 1 hour

## Act only if
- There are unread messages containing "urgent" or "critical"
- A calendar event starts within 20 minutes that I haven't been notified about

## Otherwise
Do nothing. No output. No summary. Stay silent.
```

The explicit "do nothing" section removes ambiguity and helps the model make a quick, cheap decision.

Avoid Open-Ended Instructions

Instructions like "review everything and let me know your thoughts" or "keep an eye on things" are expensive because they invite the model to generate long, exploratory responses. Be specific about what to check and what triggers a response.

> **Info: Token-Saving Rule of Thumb**
>
> Every word in HEARTBEAT.md costs roughly 1-2 tokens per heartbeat. If your heartbeat fires 96 times a day (every 15 minutes), a 100-word instruction file costs ~14,000-19,000 tokens/day just in HEARTBEAT.md reads. Keep it under 80 words if you can.

Setting Up Your Optimized Heartbeat: Step by Step

Here's a complete walkthrough for configuring a cost-efficient heartbeat from scratch.

Step 1: Edit Your Configuration File

Open your configuration file:

```
nano ~/.openclaw/clawdbot.json
```

Or use any text editor you prefer. Find the `heartbeat` section (or add one if it doesn't exist) and set it up like this:

```
{
  agents: {
    defaults: {
      model: {
        primary: "anthropic/claude-sonnet-4-20250514"
      },
      heartbeat: {
        every: "30m",
        silentHeartbeat: true,
        heartbeatModel: "deepseek/deepseek-chat"
      }
    }
  }
}
```

Save the file and close the editor. If you're using nano, that's Ctrl+O to save, then Ctrl+X to exit.

Step 2: Write Your HEARTBEAT.md

Create or edit the HEARTBEAT.md file in your agent's workspace:

```
nano ~/.openclaw/workspace/HEARTBEAT.md
```

Start with something minimal:

```
# Heartbeat

Check for unread messages that mention my name or are marked urgent.

If found: Send me a brief summary on Telegram.
If not: Stay silent. No output.
```

You can always expand this later. Start simple, verify it works, then add complexity.

Step 3: Set the API Key for Your Heartbeat Model

If you're routing heartbeats to a separate provider, make sure its API key is configured. For DeepSeek:

```
export DEEPSEEK_API_KEY="your-deepseek-api-key-here"
```

To make this permanent, add it to your shell profile:

```
echo 'export DEEPSEEK_API_KEY="your-deepseek-api-key-here"' >> ~/.zprofile
```

For Ollama (local models), no API key is needed – just make sure Ollama is running.

Step 4: Restart the Gateway

Configuration changes require a Gateway restart to take effect:

```
openclaw gateway restart
```

Step 5: Verify It's Working

Watch the logs to see your heartbeat fire:

```
openclaw gateway logs --follow
```

You should see heartbeat events appearing at your configured interval. If `silentHeartbeat` is working, most of them should show the agent checking in and producing no output. Press Ctrl+C to stop watching logs.

Step 6: Check Token Usage

After a few heartbeats have fired, check your token usage:

```
openclaw dashboard
```

Navigate to the session logs and look at the heartbeat sessions. Each one should show the token count for that check-in. Compare the cost to what you'd expect based on the model's pricing.

Monitoring Your Token Spend

Keeping an eye on costs is essential, especially when you first set up heartbeats. Here are the tools available to you.

The OpenClaw Dashboard

The dashboard at `http://127.0.0.1:18789` (or wherever your Gateway is running) shows session history, including heartbeat sessions. Look for:

- **Token counts per session** – How many tokens each heartbeat consumed.
- **Session frequency** – How often heartbeats are actually firing (verify it matches your `every` setting).
- **Response content** – What the agent actually said (or didn't say) on each heartbeat. If you see long responses on every check-in, your instructions may need tightening.

Session Logs via CLI

For a quick command-line check:

```
openclaw sessions list --type heartbeat --last 24h
```

This shows all heartbeat sessions from the past 24 hours with their token counts.

Provider Dashboards

Most API providers offer their own usage dashboards:

- **Anthropic:** console.anthropic.com – shows token usage and costs
- **OpenAI:** platform.openai.com/usage – shows token usage by model
- **DeepSeek:** platform.deepseek.com – shows API usage and billing

Check these regularly, especially in the first week after setting up heartbeats.

Setting Up Cost Alerts and Budget Caps

If your API provider supports spending limits, set them. This is your safety net against runaway costs.

Anthropic

In the Anthropic Console, navigate to Settings and set a monthly spending limit. If your heartbeats are routed through Anthropic, this cap will prevent unexpected bills from climbing past your comfort zone.

OpenAI

OpenAI lets you set both a hard cap (usage stops entirely) and a soft cap (you get an email warning). Set both. A reasonable starting point for heartbeat-only usage:

- **Soft cap:** $10/month (triggers an email alert)
- **Hard cap:** $25/month (stops API access)

Adjust these based on your actual usage after the first week.

DeepSeek

Check your provider's billing page for available controls. Budget limits vary by provider.

General Rule

Whatever provider you use for heartbeats, set a budget cap that's at least 2x what you expect to spend. This gives you headroom for normal variation while still catching runaway costs. For heartbeat-only usage on a cheap model, $5-$15/month is a reasonable starting expectation for most setups.

> **Tip: The $1/Day Sanity Check**
>
> A well-optimized heartbeat setup should cost less than $1/day for most use cases. If you're spending significantly more, something is probably misconfigured – check your interval, your HEARTBEAT.md instructions, and whether `silentHeartbeat` is actually enabled. If you're using a local model via Ollama, your heartbeat cost should be effectively $0.

Worked Example: A “Morning Briefing” for Under $0.10/Day

Let’s put everything together with a concrete example. We’ll build a heartbeat that sends you a morning briefing every day – and costs less than a dime per day.

The Goal

Every morning at 7:00 AM, the agent checks your calendar for the day, looks at any unread priority messages, and sends you a short summary on Telegram. The rest of the day, heartbeats run every hour and stay silent unless something urgent comes up.

The Configuration

```
{
  agents: {
    defaults: {
      model: {
        primary: "anthropic/claude-sonnet-4-20250514"
      },
      heartbeat: {
        every: "1h",
        silentHeartbeat: true,
        heartbeatModel: "deepseek/deepseek-chat"
      },
      cron: [
        {
          name: "morning-briefing",
          schedule: "0 7 * * *",  // 7:00 AM every day
          task: "Run the morning briefing as described in HEARTBEAT.md under '## Morning
Briefing'.",
          model: "deepseek/deepseek-chat"
        }
      ]
    }
  }
}
```

The HEARTBEAT.md

```
# Heartbeat

## Hourly Check (Default)
- Check for messages marked urgent or containing "ASAP"
- Check for calendar events starting within 30 minutes

If either condition is met: send a brief Telegram alert (2-3 sentences max).
Otherwise: stay silent. No output.

## Morning Briefing
- List today's calendar events (time and title only)
- Count unread messages per channel
- Note any tasks due today

Send as a single Telegram message. Keep it under 150 words.
```

The Cost Breakdown

Hourly silent checks (23 per day): - Context: ~2,000 tokens input (system prompt + HEARTBEAT.md + minimal memory) - Response: ~10 tokens ("no action needed" – internally suppressed) - Per check: ~2,010 tokens - Daily: ~46,230 tokens - At DeepSeek rates (~$0.14/1M tokens): **~$0.006/day**

Morning briefing (1 per day): - Context: ~2,500 tokens (includes calendar and message data) - Response: ~200 tokens (the briefing itself) - Per briefing: ~2,700 tokens - At DeepSeek rates: **~$0.0004/day**

Total daily cost: ~$0.007/day – well under a penny.

Even if we're generous with the estimates and double everything, you're looking at under $0.02/day. The $0.10/day budget gives you enormous headroom.

> **Info: Why So Cheap?**
>
> Three things make this affordable: (1) routing to a cheap model, (2) silent heartbeats that suppress most output, and (3) tight HEARTBEAT.md instructions that don't invite long responses. Remove any one of those and costs climb quickly.

Adjusting the Interval: Responsiveness vs. Cost

The `every` interval is a direct tradeoff between how quickly your agent notices things and how much it costs. Here's a practical guide:

Interval	Checks/Day	Best For	Estimated Cost*
`"5m"`	288	Time-critical monitoring (outages, security)	$0.08/day
`"15m"`	96	Active project tracking, busy communication	$0.03/day
`"30m"`	48	General awareness, moderate responsiveness	$0.01/day
`"1h"`	24	Daily awareness, low-urgency monitoring	$0.007/day
`"4h"`	6	Periodic check-ins, low-priority tasks	$0.002/day

**Estimates assume ~2,000 input tokens per check, silent heartbeats enabled, using DeepSeek Chat pricing. Actual costs vary.*

For most people, `"30m"` or `"1h"` hits the sweet spot. You get reasonable awareness without significant cost. Reserve `"5m"` intervals for situations where minutes genuinely matter – like monitoring a production system or tracking time-sensitive events.

> **Tip: Use Cron for Time-Restricted Heartbeats**
>
> You don't need heartbeats running at 3 AM (unless you do). Restrict them to your active hours using a cron expression:
>
> ```
> {
>   heartbeat: {
>     cron: "*/30 7-22 * * *",  // Every 30 minutes, 7 AM to 10 PM
>     silentHeartbeat: true,
>     heartbeatModel: "deepseek/deepseek-chat"
>   }
> }
> ```
>
> This cuts your daily checks roughly in half compared to running 24/7.

Troubleshooting

Here are the most common issues when optimizing heartbeats, and how to fix them.

Heartbeats aren't firing at all Check that the Gateway is running (`openclaw gateway status`). Then verify your configuration syntax – a missing comma or bracket in JSON can silently break the heartbeat section. Run `openclaw config validate` if available, or carefully review your `clawdbot.json` for syntax errors.

Agent responds on every heartbeat despite `silentHeartbeat: true` Your HEARTBEAT.md instructions may be too open-ended. If the instructions say "check X and tell me about it," the agent interprets that as a request to produce output. Rewrite with explicit "stay silent unless" conditions. Also verify that `silentHeartbeat` is set in the correct location in your config file – it needs to be inside the `heartbeat` block.

Heartbeat model isn't being used (costs are higher than expected) Double-check that `heartbeatModel` is spelled correctly and points to a model that's actually configured in your `models` section. If the model string doesn't match an available model, the system may fall back to your primary model silently. Run `openclaw sessions list --type heartbeat --last 1h` and check which model is listed for each session.

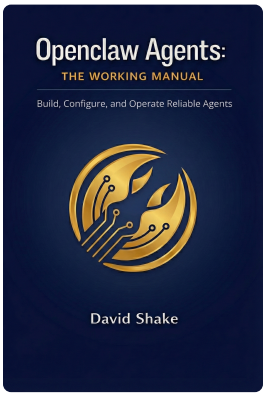
"Model not found" errors on heartbeat The model specified in `heartbeatModel` either isn't configured or isn't available. For cloud models, ensure the API key for that provider is set. For Ollama models, make sure Ollama is running (`ollama serve`) and the model is downloaded (`ollama list`).

Costs are still too high after optimization Do an audit. Check the dashboard for how many tokens each heartbeat is consuming. The usual culprits are: (1) a large system prompt that gets included in every heartbeat, (2) extensive memory files being loaded into context, or (3) HEARTBEAT.md instructions that are too long. Address these one at a time.

The morning briefing cron job fires but produces no output If `silentHeartbeat` is applying to your cron jobs as well, the agent may be suppressing the briefing. Cron jobs and heartbeats are related but distinct – check

that your cron task instructions explicitly ask for output (“Send me a summary on Telegram”) so the agent knows this particular check should produce a response regardless of the silent setting.

Agent enters a loop – same action repeated on every heartbeat This usually means the heartbeat instructions describe a task without a completion condition. For example, “summarize my unread messages” will generate a summary every time, because the agent doesn’t track whether it already sent one. Add state awareness: “Summarize unread messages *that arrived since the last heartbeat*” or use the agent’s memory to track what’s been reported.



Enjoyed this guide?

The full book covers architecture, channels, security, models, memory, automation, and six complete setup guides — everything you need to run OpenClaw agents that actually work.

GET THE BOOK

Kindle Edition • **Apple Books** • **Direct (EPUB)**

BUY ME A COFFEE

If this free guide saved you time or money, you can say thanks with a one-time contribution. No account needed.

$5 • **$10** • **$25**