OFFICE HOURS QUICK REFERENCE · V02 · MAY 2026

GitHub Copilot Prompt & Token Optimization

Why this matters.
Under usage-based billing, input, output, and cached tokens are metered against your AI-credit pool. Output tokens cost roughly 5x more than input. Most input is not what you type: it is instructions, open tabs, tool schemas, and chat history replayed across turns.

---

1. The 12 highest-ROI practices

1 **Constrain output**
Add `Code only, no explanation. Bullets over paragraphs.` to repo instructions. Effect: 40-70% fewer output tokens on code tasks. Setup: 1 min.

2 **Shrink always-on context**
Compress `copilot-instructions.md` and `AGENTS.md` to landmines only. Delete generated boilerplate the agent can rediscover. Effect: 40-60% fewer instruction tokens every call. Setup: 15 min.

3 **Scope context with `applyTo:`**
Split one large instructions file into smaller scoped files that load only for matching paths or file types. Effect: 50-80% less scoped load when the file does not apply. Setup: 15 min.

4 **Ask Mode for simple questions**
Reserve Agent Mode for true multi-step work. Pick Ask Mode for one-shot Q&A. Effect: 60-90% lower token use vs agent loops. Setup: 0 min.

5 **Use Auto model selection**
Default to Auto. Pin a higher-tier model only when the task warrants it. Effect: lower billed rate on eligible usage. Setup: 0 min.

6 **Audit MCP servers**
Disable MCPs you do not actively use. Each tool schema can cost tokens per agent step. Effect: 5K-190K fewer tokens per agent task. Setup: 5 min.

7 **Prefer Skills over always-on MCPs**
Skills load title and description until invoked. MCP tool schemas are loaded into agent context. Effect: drastically lower steady-state context. Setup: 10 min.

8 **Use CLI tools for deterministic work**
For repeatable browser, file, or command flows, pipe CLI commands instead of multi-turn agent reasoning. Effect: fewer LLM round-trips and less tool-schema replay.

9 **Be precise in prompts**
`Add a null check to getUser() and return 404` beats `can you maybe handle errors here?` Effect: better first pass, fewer follow-up turns. Setup: 0 min.

10 **Retune prompts per model**
Vendor prompting guides differ by model and version. Ask Copilot to adapt your instructions for the model you use. Effect: cuts rework from poor first-pass output. Setup: 10 min/model.

11 **Run `/chronicle improve`**
Run weekly in Copilot CLI or VS Code to scan recent sessions for recurring misunderstandings and generate instruction patches. Effect: 10K-30K tokens saved per repeated pattern. Setup: 2 min.

12 **Start fresh sessions often**
History is replayed every turn. When the topic changes, open a new chat. Effect: removes carry-over input cost. Setup: 0 min.

Figures are scoped to the mechanism named in each row and are not additive.

---

2. Context is everything: five context surfaces

SURFACE	FILE	WHEN LOADED	BEST FOR
Repo instructions	`copilot-instructions.md`, `AGENTS.md`	Always, every request	Project-wide AI README
Custom instructions	`*.instructions.md`	By file pattern with `applyTo:`	File-specific rules
Prompts	`*.prompt.md`	When invoked	Task templates
Agents	`*.agent.md`	When mentioned	Specialist personas
Skills	`SKILL.md`	Automatically when relevant	Specialized expertise

Cost rule of thumb.
The further right the context surface is, the more demand-loaded it is and the lower the steady-state token cost. Push as much as possible from always-on repo instructions into on-demand Skills and prompts.

---

3. MCP vs Skills vs CLI

TOOL	BEST FOR	LOADING	TRADEOFF
MCP	Remote APIs, SaaS, databases, cross-platform integrations	Tool metadata loaded into agent context	Strong auth and governance, but every tool call can become another LLM turn
Skills	Repeatable workflows, org knowledge, orchestration	Progressive loading: title and description first, full content on invoke	Human-readable and lower steady-state context
CLI	Local dev, speed, low cost, composable deterministic operations	Zero schema overhead	Best for shell pipes and batch commands; govern carefully at scale

The roundtrip rule.
Every MCP or tool call can be a separate LLM turn with system prompt, chat history, and loaded tool schemas sent again. A CLI command can chain many operations in one line and return one filtered result. For deterministic work, prefer CLI. Reach for MCP when you need typed protocol, auth, sandboxing, or remote APIs.

4. Where tokens actually go

SOURCE	ALWAYS SENT?	LEVER
`.github/copilot-instructions.md`	Every turn	Compress
`AGENTS.md`	Every agent step	Compress
Open editor tabs and selection	Often	Close unused tabs
Chat history	Grows over time	Start fresh
MCP tool schemas	Per agent step	Audit or convert to Skill or CLI
Skill metadata	Name and description only	Prefer Skills where appropriate
CLI command output	Only what you return	Filter before returning with `| head` or `| jq`
Model output	Billed at higher rates than input	Constrain output

5. Mode selection

TASK	BEST MODE	WHY
Explain this function	Ask	One-shot, no tools needed

TASK	BEST MODE	WHY
Refactor file X	Edit / Inline	Bounded scope, fewer tools
Implement feature across three files and run tests	Agent	Genuinely multi-step
Reproducible browser or file pipeline	CLI tool	Deterministic, no LLM round-trip per step
Recurring confusion in past sessions	`/chronicle improve`	Patches instructions permanently

6. Anti-patterns

! **Bloated instructions**
Cost: paid every turn. Fix: strip filler and keep landmines.

! **Every MCP enabled just in case**
Cost: large upfront and per-task overhead. Fix: disable until needed or convert to Skill/CLI.

! **MCP for work one shell pipe can do**
Cost: one LLM turn per step. Fix: use a chained CLI command.

! **Agent for simple explanation**
Cost: 5-10x overhead. Fix: use Ask.

! **One mega-thread for days**
Cost: linear history replay. Fix: new chat per topic.

! **Verbose prompts and pinned premium models**
Cost: more input, more output, and higher billed rate. Fix: direct imperatives and Auto by default.

7. Five-minute setup checklist

1. Add `Code only, no explanation. Bullets over paragraphs. Be terse.` to repo instructions.
2. Delete generated boilerplate from `copilot-instructions.md` and `AGENTS.md`.
3. Open the MCP panel and disable unused servers.
4. Move repeated workflows from MCP or instructions into a Skill.
5. Set model selection to Auto.
6. Bookmark `/chronicle improve` for weekly use.
7. Close editor tabs you are not actively using.

Further reading

- GitHub Docs: Copilot
- GitHub Docs: About billing for Copilot
- GitHub Docs: Customizing Copilot Chat responses
- Anthropic public pricing, for input/output pricing asymmetry examples.

Practitioner resource; for official guidance, see GitHub Copilot documentation and GitHub billing documentation.