CYBERNOMiCS

CISO-APPROVED MANAGED AI SERVICE PROVIDER

HOW AGENTIC USAGE COSTS ARE CALCULATED

EXACTLY — METER BY METER

A unit-economics report for executives, AI leaders, security teams, finance, and solution providers.

Pricing snapshot: August 2, 2026

US public list prices; regional and contract rates may differ.

EXECUTIVE BRIEF

The number on the invoice is not “the cost of the agent.”

It is the cost of the agent’s complete execution path—successful and unsuccessful.

THE GOVERNING EQUATION

Total agent cost = model inference + repeated context + tool calls + retrieval and storage + runtime and infrastructure + external services + retries + human oversight.

The central conclusion. A user can submit one prompt while the agent creates five, ten, or fifty separate billable events behind the scenes. Every model request is metered; each request may reprocess prior history, tool schemas, retrieved data, screenshots, and earlier tool outputs. Reasoning tokens can be billed even when the user never sees them.[1,3,4,5]

01

COUNT REQUESTS, NOT PROMPTS

Agent frameworks aggregate usage across every model call, including tool calls and handoffs. One run is often many requests.

02

CONTEXT IS RE-BILLED

Stateful conversation does not make prior context free. Previous input may be fed into later requests and billed again, although cached prefixes can receive discounted rates.

03

TOOLS CREATE TWO COSTS

A tool can have a direct per-call fee and also increase token cost through its schema, search results, screenshots, and returned data.

04

PRICE THE SUCCESS

The financially correct denominator is cost per successful, accepted business outcome—not cost per prompt, token, or run.

Cybernomics recommendation: Require every production agent to emit a run-level cost ledger tied to a measurable outcome, with hard caps on turns, tool calls, reasoning tokens, and retries.

01 — THE COST EQUATION

The exact unit-economics formula

This framework works across direct APIs, managed agent runtimes, and low-code credit systems.

```
C_run = SUM(Model_i) + SUM(Tool_j) + Data + Runtime + External + Human + Waste
```

Calculate each completed run from its actual usage records—not from a nominal "average prompt."

For each model request i:

```
Model_i = (U_i*P_in + C_i*P_cache + W_i*P_write + O_i*P_out) / 1,000,000
```

U = uncached input tokens | C = cached reads | W = cache writes | O = output tokens, including billed reasoning where applicable

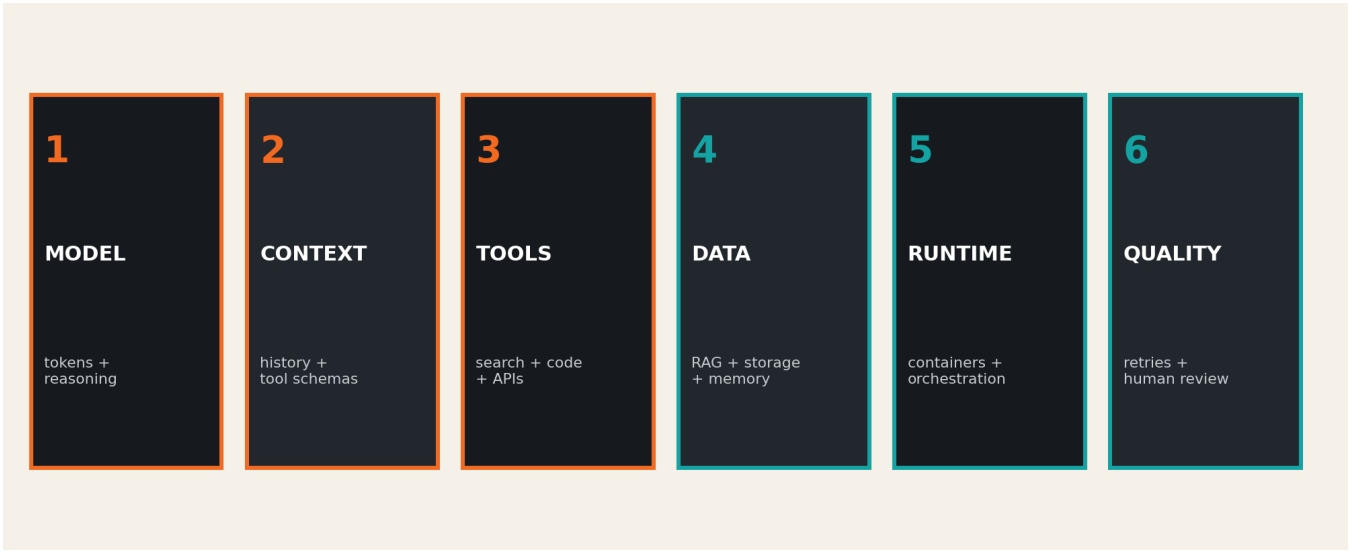
Meter	What enters the meter	Billing basis
Model inference	Every LLM call, subagent call, router, critic, evaluator, and finalizer.	Tokens by category and model rate.
Context expansion	System instructions, conversation history, tool schemas, retrieved passages, screenshots, and tool results.	Usually embedded in input-token totals; often re-billed on later turns.
Tools	Web search, file search, code execution, computer use, voice, document processing, third-party APIs.	Per call, per query, per step, per page, per minute, or tokens.
Data and memory	Embeddings, vector index, managed retrieval, long-term memory, logs, file storage.	GB-day, GB-month, item-month, retrieval call, or tokens.
Runtime and infrastructure	Managed agent sessions, containers, vCPU, RAM, network, queues, databases.	Session-hour, container-minute, vCPU-second, GB-second, request.
Quality and governance	Retries, fallbacks, guardrails, evaluation calls, human review, exceptions, remediation.	Expected attempts, review minutes, and failure rate.

MONTHLY FORECAST

C_month = triggers × expected attempts × variable cost per attempt + fixed monthly costs. Then divide by accepted outcomes to obtain the economically meaningful unit cost.

Anatomy of one agent run

The user sees one task. The billing system sees a chain of requests and services.



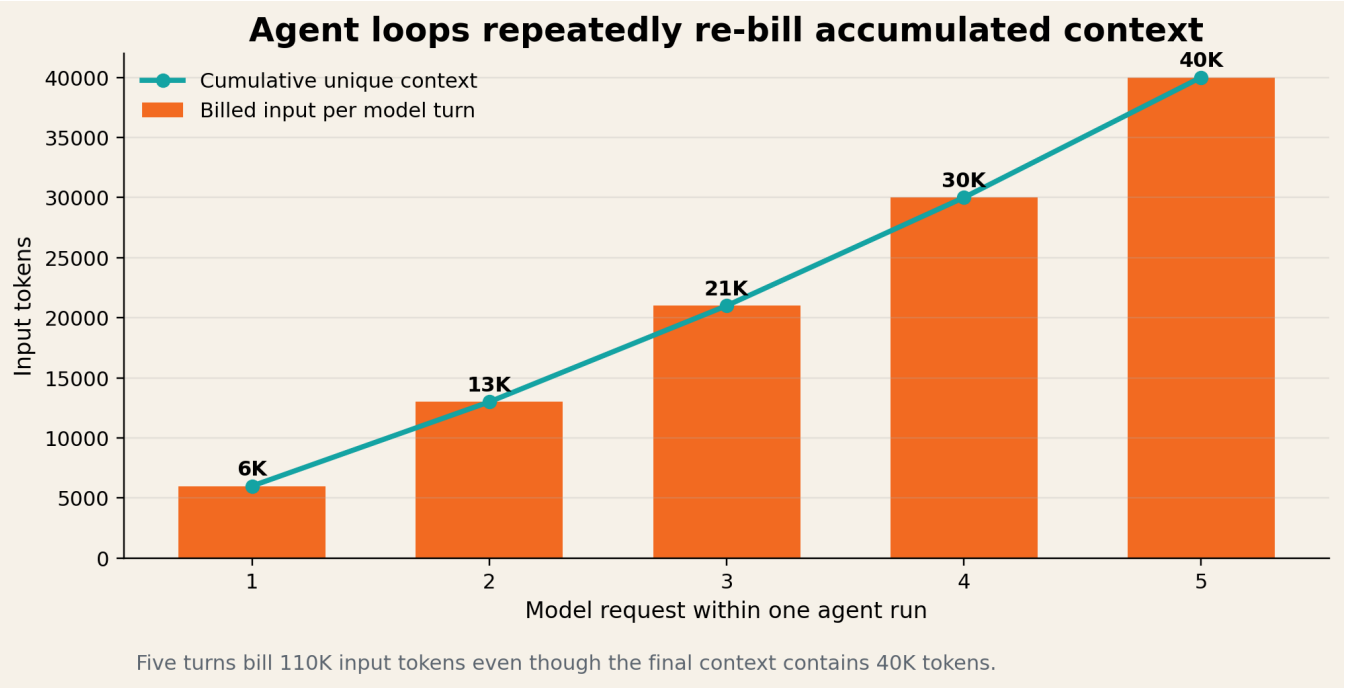
1	**Trigger and intake —** A user message, scheduled event, webhook, or new record starts the run. Intake classification may itself call a model.
2	**Planning —** The planner receives instructions, available tool definitions, history, and the task. Its output and hidden reasoning are metered.
3	**Tool loop —** The model chooses a tool, emits arguments, receives the tool result, and is called again. Each loop can repeat the growing context.
4	**Retrieval and memory —** Search queries, retrieved passages, embeddings, vector storage, memory reads, and write-backs can add separate meters.
5	**Validation and action —** A critic, policy check, or human approval may run before a transaction, email, or system change is executed.
6	**Finalization —** The final model call summarizes results. Logging, tracing, evaluation, and storage continue after the answer appears.

OpenAI's Agents SDK, for example, reports aggregate usage across all model calls in a run—including tool calls and handoffs—and exposes per-request usage entries for precise costing.[7]

03 — THE HIDDEN MULTIPLIER

Context accumulation is where “cheap” agents become expensive

Agent loops frequently pay to reread their own work.



Five turns bill 110K input tokens even though the final context contains 40K tokens.

The 2.75× context multiplier. In the illustrated run, the final model request contains 40,000 input tokens, but the five requests together bill 110,000 input tokens. The agent paid repeatedly for instructions, history, tool calls, and prior results as the context grew.

STATE IS NOT FREE

Using a conversation ID or previous response ID simplifies state management, but it does not eliminate billing for prior context. OpenAI explicitly states that previous input tokens in a response chain are still billed as input tokens.

What typically enters input tokens:

- System prompt, policies, role instructions, output schemas, and examples.
- All user messages and selected prior assistant outputs retained in the working context.
- Tool definitions and JSON schemas. OpenAI notes that callable function definitions are injected into the system message and billed as input tokens.
- Retrieved document chunks, web-search content, database records, screenshots, OCR text, and tool error messages.
- Encrypted or persisted reasoning items when the architecture carries them into future turns.

Caching changes the rate—not the need to measure. Exact-prefix cache hits can be dramatically cheaper, but cache writes may carry a premium and a changing prefix can destroy the hit. GPT-5.6 cache writes are billed at 1.25× the uncached input rate and cache reads at the discounted cached-input rate.[2]

04 — MODEL-TOKEN ACCOUNTING

How token charges are calculated

Convert every reported token category into dollars at the rate for the exact model, service tier, context tier, and region used.

Category	Exact billing treatment	What creates it
Uncached input	New prompt/context tokens processed at the standard input rate.	Instructions, user data, history, tool schemas, retrieved content.
Cached input	Tokens served from an eligible matching prompt prefix at the cached-input rate.	Still appear in usage; do not subtract them from total input and then ignore them.
Cache writes	Eligible prefix tokens written to a cache. Some providers charge a premium.	OpenAI GPT-5.6: 1.25× uncached input. Anthropic: 1.25× for 5-minute or 2× for 1-hour writes.
Output	All generated tokens charged at the output rate.	Visible answer, function-call arguments, formatting, and provider-defined generated tokens.
Reasoning	Internal inference tokens used to plan or "think."	OpenAI reasoning tokens are included in output usage and billed as output tokens.
Long-context uplift	A higher rate activated above a model-specific threshold.	For GPT-5.6 Sol, requests above 272K input tokens are priced at 2× input and 1.5× output for the full request.

```
Cost per token = listed price per 1M tokens / 1,000,000
```

Example: $12 per 1M output tokens = $0.000012 per output token = $0.012 per 1,000 output tokens.

Important: A run can incur output-token cost without producing a useful visible answer. OpenAI notes that reasoning may exhaust the output budget before visible text appears.[3]

Tools are not free just because the model called them

Most agent deployments combine direct tool charges with the token cost of tool definitions and returned data.

Capability	Meter	Current public example
Web search	Per search/query + search-result tokens	OpenAI: $10/1K calls plus search content tokens. Anthropic: $10/1K searches plus token costs. Google Gemini 3.x: after monthly free allowance, $14/1K individual search queries; one prompt may trigger more than one query.
File search / RAG	Per retrieval call + vector storage + tokens	OpenAI Responses file search: $2.50/1K tool calls; vector storage $0.10/GB-day after 1 GB free. Retrieved content then increases model input.
Code / shell	Session, container, or runtime	OpenAI 1 GB container: $0.03 per 20-minute session, with eligible sessions billed by minute and a 5-minute minimum. Anthropic Managed Agents uses $0.08 per running session-hour.
Computer use	Per step or tokenized screenshots	Copilot Studio standard computer-use step: 5 credits; premium: 15 credits. Anthropic computer use adds tool/system tokens and screenshot vision tokens.
MCP / functions	Tokenized definitions, calls, and results; sometimes external API fees	OpenAI MCP has no additional platform fee per tool call, but imported definitions and call/result content consume tokens. Your MCP server or SaaS API may charge separately.
Managed knowledge base	Storage + standard or agentic retrieval	AWS Bedrock Managed Knowledge Base: $5/GB-month raw data, $1/1K Retrieve calls, or $4/1K Agentic Retrieve calls plus $1/1K underlying Retrieve calls.

DO NOT DOUBLE COUNT

If search-result or tool-output tokens are already included in the provider's input-token usage, charge the token category once—then add only the separate per-call fee. Maintain one authoritative ledger for every meter.

Why invoices from different agent platforms look incompatible

The economics are comparable only after every platform is normalized into cost per successful outcome.

DIRECT API	You pay model tokens and explicit tool/data/runtime meters. **Example:** OpenAI Responses, Anthropic Messages, Gemini API. **Implication:** Most transparent; requires complete telemetry.
MANAGED RUNTIME	Tokens plus a managed session or container meter. **Example:** Claude Managed Agents, hosted agents in Microsoft Foundry. **Implication:** Adds runtime convenience and an additional execution meter.
CREDIT / ACTION	The vendor abstracts token complexity into credits per response, action, step, or token tier. **Example:** Microsoft Copilot Studio. **Implication:** Easier forecasting; can be expensive for highly agentic loops.
SEAT / INCLUDED	Certain employee scenarios are zero-rated inside a user license, subject to scope and fair use. **Example:** Microsoft 365 Copilot licensed-user agent scenarios. **Implication:** Not equivalent to unlimited external or autonomous usage.

Microsoft Foundry example. Foundry-native agents created with prompts and workflows have no separate agent-creation or agent-running fee; customers pay underlying models and tools. Hosted agents, by contrast, are billed for dedicated container compute consumed per hour.[12]

Microsoft 365 Copilot inclusion. Classic answers, generative answers, agent actions, tenant graph grounding, and eligible agent-triggered flows can be zero-rated for authenticated Microsoft 365 Copilot licensed users. Computer-using agents are specifically excluded from that inclusion.[9,10,11]

07 — CURRENT PUBLIC RATE-CARD SNAPSHOT

Selected prices relevant to agentic workloads

Snapshot dated August 2, 2026. Use the exact deployment region, service tier, and contract price when building a production forecast.

Provider / meter	Public US list-price mechanism
OpenAI GPT-5.6 Terra — standard, short context	$2 input / $0.20 cached read / $2.50 cache write / $12 output per 1M tokens
OpenAI tools	$10/1K web searches; $2.50/1K file-search calls; $0.10/GB-day vector storage after 1 GB free; 1 GB container $0.03 per 20-minute session
Anthropic Claude Sonnet 5 — introductory through Aug. 31, 2026	$2 input / $2.50 5-minute cache write / $4 1-hour cache write / $0.20 cache hit / $10 output per 1M tokens
Anthropic Claude Managed Agents	$0.08 per running session-hour, plus tokens and web search; idle and approval-wait time is not metered as runtime
Google Gemini 3.5 Flash	$1.50 input / $0.15 cached input / $9 output per 1M tokens; cache storage $1 per 1M tokens per hour
Google grounding with Search	5,000 free search requests per month shared across Gemini 3.x, then $14/1K individual search queries
Microsoft Copilot Studio PAYG	$0.01 per Copilot Credit; generative answer 2 credits; agent action 5; tenant graph grounding 10; premium reasoning 10 credits per 1K tokens plus feature rate
AWS Bedrock Managed Knowledge Base	$5/GB-month raw data; $1/1K standard retrieval calls; agentic retrieval $4/1K calls plus $1/1K underlying retrieval calls

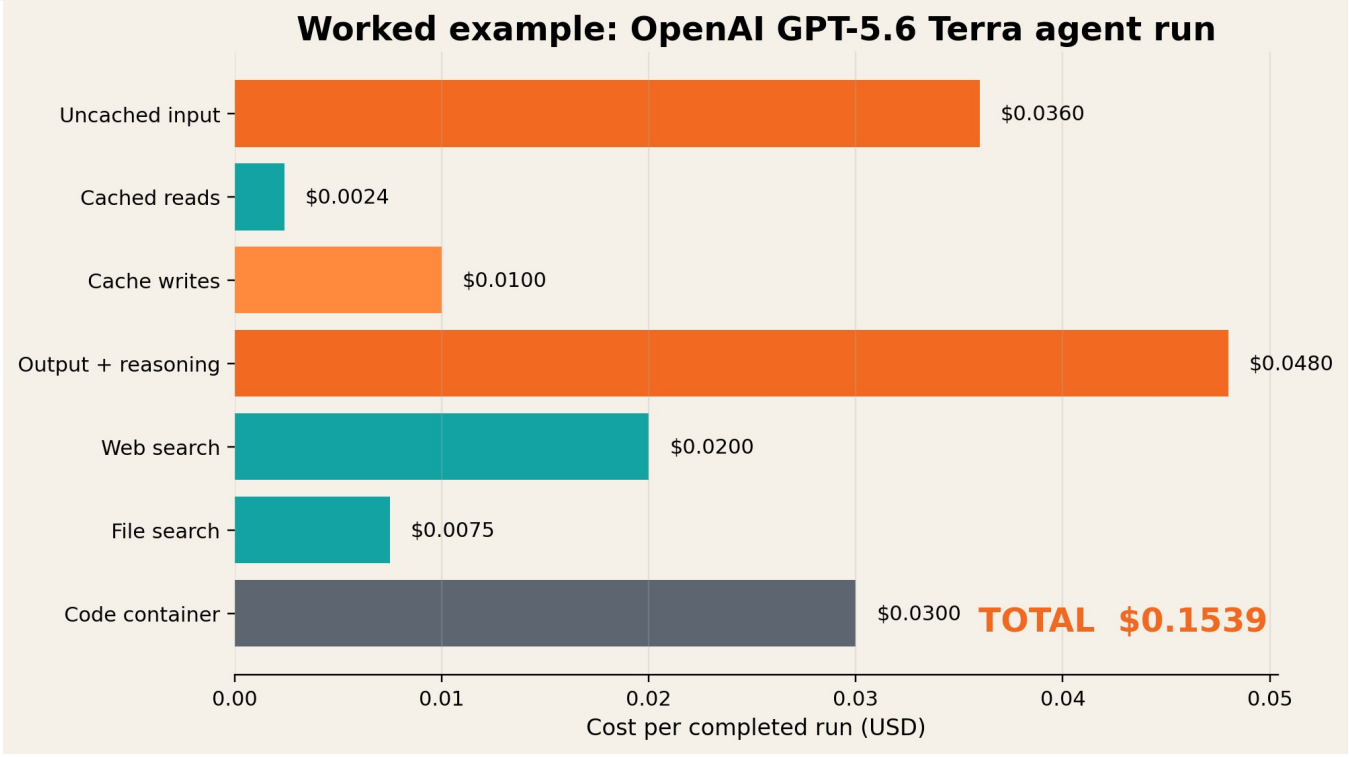
Rate-card trap: A low input-token rate does not guarantee a low agent cost. Output/reasoning, search, computer use, and repeated tool loops can dominate the bill.

08 — WORKED EXAMPLE

A complete OpenAI agent-run calculation

Controlled assumptions make every dollar traceable.

Assumption	Value
Model	GPT-5.6 Terra, standard, short-context tier
Model requests	4 total requests across planning, tools, and finalization
Uncached input	18,000 tokens
Cached reads	12,000 tokens
Cache writes	4,000 tokens
Output	4,000 tokens, including billed reasoning
Built-in tools	2 web searches + 3 file-search calls + one 1 GB code container session
Search content	Already included in the input-token totals above
Vector storage	0.8 GB total account storage; below the 1 GB free allowance



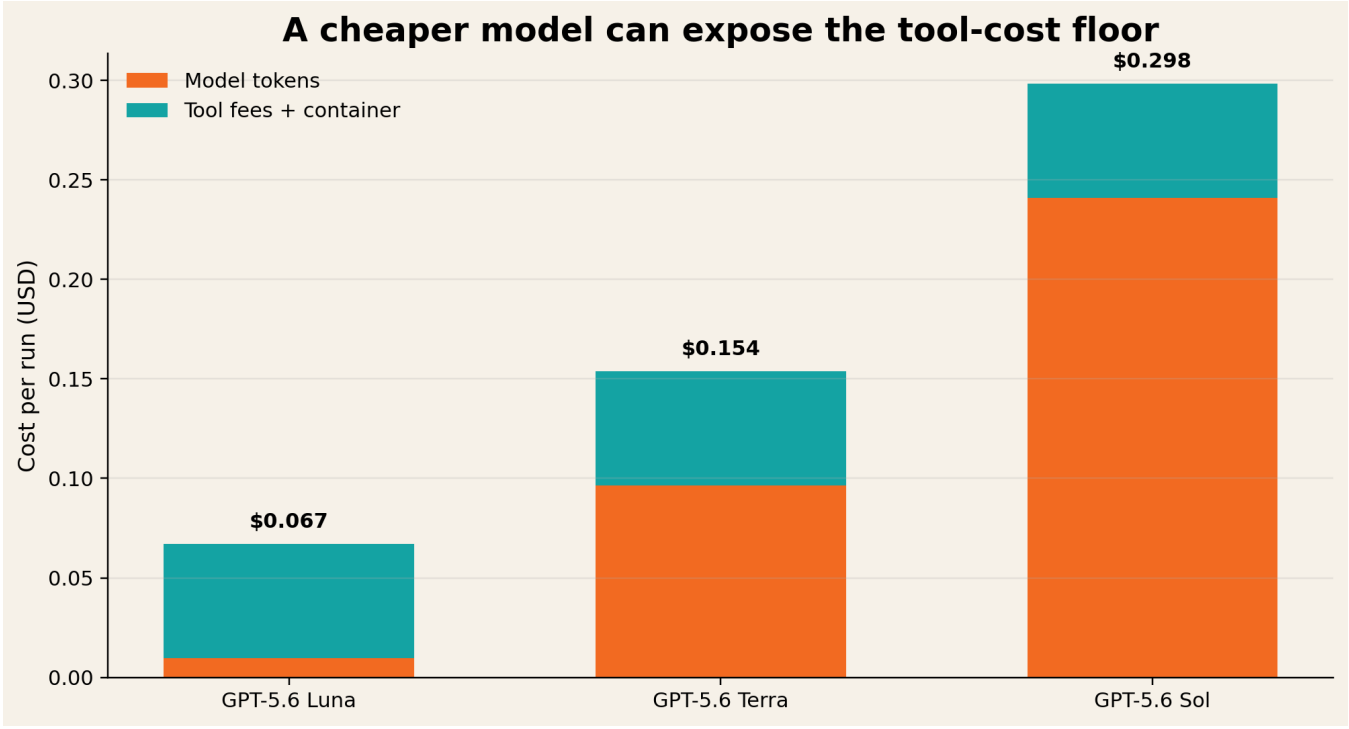
$0.0360 + $0.0024 + $0.0100 + $0.0480 + $0.0200 + $0.0075 + $0.0300 = $0.1539

Direct provider cost per completed run before external SaaS charges, application infrastructure, retries, and human review.

09 — MODEL ROUTING ECONOMICS

The same workflow can cost 4.4× more by model tier

But tool costs create a floor that model switching cannot eliminate.



Tier	Cost / attempt	10,000 attempts	Cost / success at 88%
Luna	$0.06714	$671.40	$0.07630
Terra	$0.15390	$1,539.00	$0.17489
Sol	$0.29850	$2,985.00	$0.33920

ROUTING RULE

Use a low-cost model for deterministic extraction, classification, routing, and summarization. Escalate only ambiguous or high-risk cases to a more capable model. The routing model and escalation attempt must both be included in the run ledger.

How credit-based agent usage converts to dollars

Copilot Studio replaces most raw token accounting with feature credits, except token-tiered AI tools and reasoning.

Feature	Rate	PAYG equivalent at $0.01/credit
Classic answer	1 credit	$0.01
Generative answer	2 credits	$0.02
Agent action	5 credits	$0.05
Tenant graph grounding	10 credits	$0.10
Agent flow actions	13 credits per 100 actions	$0.00013 per action
Premium reasoning	10 credits per 1K tokens	$0.10 per 1K tokens, plus the feature rate
Computer use — standard	5 credits per step	$0.05 per step
Computer use — premium	15 credits per step	$0.15 per step

Worked credit example. One run performs one generative answer (2 credits), four agent actions (20), one tenant graph grounding event (10), and uses 6,000 premium reasoning tokens (60). Total = 92 credits = $0.92 on pay-as-you-go. At 10,000 runs, the variable charge is $9,200.

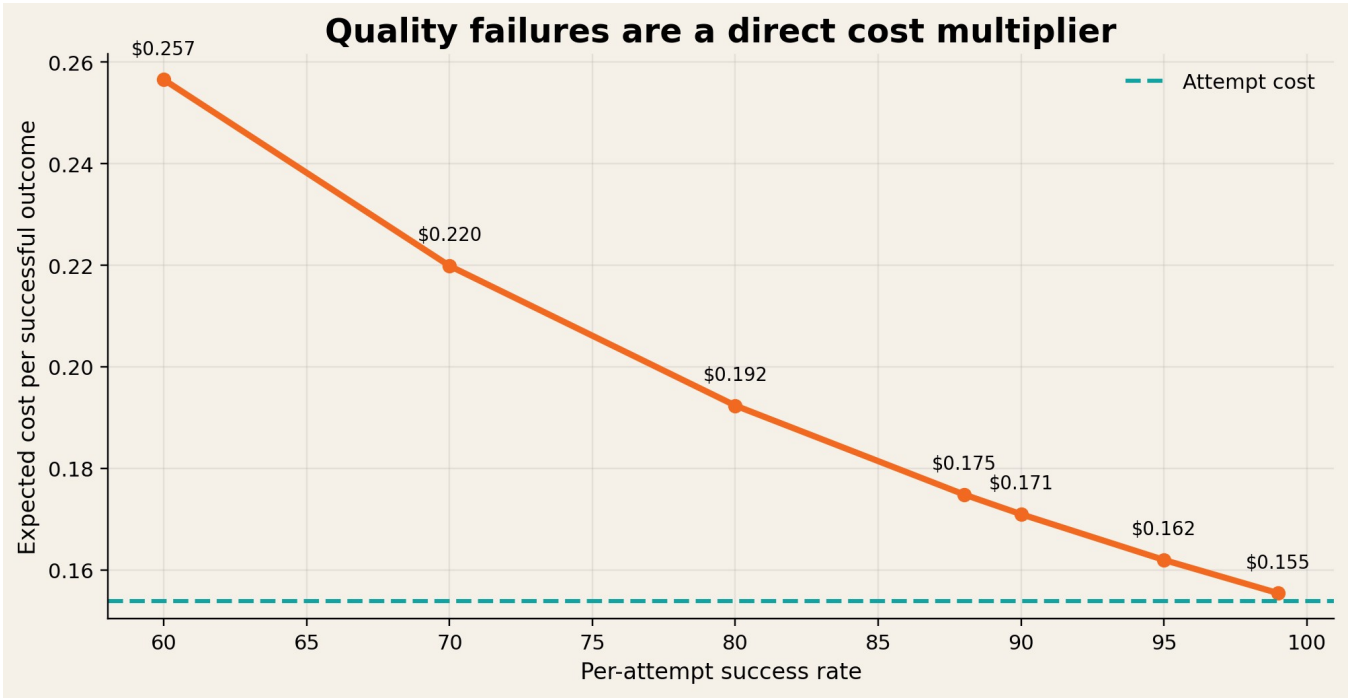
```
Copilot Studio cost = SUM(feature credits) × $0.01
```

For reasoning-capable models: feature rate + 10 credits per 1,000 premium tokens. Verify license-based zero-rating before applying PAYG.

Computer-use example. Microsoft's four-step time-sheet example consumes 20 credits with a standard model or 60 credits with a premium model—$0.20 or $0.60 PAYG—before any other agent features in the overall workflow.[11]

The cost that rate cards do not show

A model can be inexpensive per attempt and expensive per accepted result.




```
Expected cost per success = cost per attempt / per-attempt success rate
```

When attempts have equal cost and failures are independent. Example: $0.1539 / 0.88 = $0.17489 per successful outcome.

For a maximum of R attempts and failure probability f:

```
E[attempts] = (1 - f^R) / (1 - f)     |     P(success within R) = 1 - f^R
```

When attempt costs escalate, use the general form: Expected run cost = Σ P(reaching attempt k) × cost_k. Divide by the probability that the workflow produces an accepted result.

Waste categories to measure separately:

- Automatic retries after timeouts, malformed JSON, failed tool calls, and transient API errors.
- Planner loops that revisit the same tool or repeatedly search without converging.
- Fallback model calls after a low-cost model fails confidence or validation thresholds.
- Guardrail and evaluation calls—often separate model requests—that reject or rewrite outputs.
- Human review minutes, rework, escalations, and downstream errors caused by incorrect actions.

12 — BUILD THE MONTHLY FORECAST

From technical telemetry to a finance-ready budget

Forecast volume, workflow shape, success, and fixed costs independently.

Forecast block	What to estimate	Control
1. Demand	Monthly triggers by workflow, channel, customer, or department.	Do not use total employees as a proxy unless every employee has the same usage pattern.
2. Workflow shape	Average and percentile model requests, tool calls, retrieved tokens, runtime, and steps.	Model p50, p90, and p99; averages hide runaway loops.
3. Model mix	Share routed to each model, service tier, and context tier.	Include router, critic, evaluator, and fallback models.
4. Success and retry	First-pass acceptance rate, retry probability, escalation rate, and maximum attempts.	Budget accepted outcomes, not merely initiated runs.
5. Fixed data costs	Vector storage, memory, logs, databases, observability, minimum commitments.	Allocate shared costs by run, department, or value driver.
6. Human operations	Review sampling, exceptions, support, prompt maintenance, evaluation, and governance.	Use fully loaded labor rates.

```
C_month = N_triggers × E[C_attempts per trigger] + C_storage + C_runtime + C_infra + C_human
```

Report alongside: cost per accepted outcome, cost by workflow, p95 cost per run, and gross margin if the agent is resold.

13 — COST-CONTROL ARCHITECTURE

The controls that reliably reduce agent spend

Optimization should preserve outcome quality and security—not merely reduce tokens.

01 CAP THE LOOP

Set maximum model turns, searches, tool calls, computer-use steps, runtime, and spend per run.

02 ROUTE BY DIFFICULTY

Default to a low-cost model and escalate only when confidence, risk, or complexity warrants it.

03 CACHE STABLE PREFIXES

Place static instructions and tool definitions before variable content; use explicit cache breakpoints where supported.

04 REDUCE TOOL SURFACE

Load only relevant functions. Tool schemas are context and therefore cost money.

05 TRIM TOOL RESULTS

Return compact structured fields, not entire pages, logs, or records. Limit web-fetch and retrieval content.

06 COMPACT HISTORY

Summarize or reset context at natural boundaries. Do not carry irrelevant prior turns indefinitely.

07 DETERMINISTIC FIRST

Use code or flows for known sequences; reserve model judgment for ambiguous decisions.

08 BOUND REASONING

Set output/reasoning budgets by task class. A high reasoning setting should be an explicit business decision.

09 DESIGN FOR FAILURE

Validate before high-cost actions, reuse successful intermediate results, and avoid restarting the entire workflow.

10 METER THE OUTCOME

Alert on p95 cost, failed-tool loops, low cache hit rate, repeated retrieval, and cost per accepted result.

CISO / FINANCE GATE

No agent should enter production without an owner, allowed tools, data classification, transaction limits, rollback method, cost budget, quality threshold, and a complete run ledger.

14 — REQUIRED TELEMETRY

The minimum viable agent cost ledger

Capture raw usage at request level, then aggregate by run, workflow, customer, and accepted outcome.

Ledger block	Fields
Identity	run_id, workflow_id, tenant/customer, trigger_id, user/channel, timestamp
Model request	request_id, model, service_tier, region, context_tier, latency, status
Token usage	uncached_input, cached_input, cache_write, output, reasoning, image/audio tokens
Tool usage	tool_name, call_count, query_count, step_count, page_count, bytes/tokens returned, tool error
Data/runtime	vector_GB_days, retrieval_calls, memory_items, session_seconds, container_size, vCPU/GB-seconds
External cost	SaaS/API transaction cost, database/network cost, payment or messaging fees
Quality	attempt_number, validation result, fallback model, human review minutes, accepted/rejected outcome
Financial output	provider_cost, internal_infra_cost, labor_cost, total_cost, value/revenue, gross_margin

Executive dashboard metrics

- Cost per successful outcome and cost per $1 of business value.
- p50 / p90 / p95 / p99 run cost and model-request count.
- First-pass success, retry rate, fallback rate, and human-escalation rate.
- Cache-read rate, cache-write-to-read ratio, average context multiplier, and retrieved tokens per run.
- Tool cost share, output/reasoning cost share, and top runaway workflows.
- Gross margin by customer or service package for a managed AI provider.

CYBERNOMICS CONCLUSION

The right question is not “How much does an agent cost?”

The right question is: “What does this specific agent cost per controlled, accepted business result?”

THE ANSWER IN ONE LINE

Agentic usage cost is the sum of every metered model request, token category, tool operation, data/runtime service, retry, and human control required to produce one successful outcome.

The practical implication. Agent economics must be designed at the workflow level. A vendor’s token price is only one input. Architecture determines how many times the model is called, how much context is repeated, how much tool output is returned, how often the run fails, and whether a human must intervene.

For buyers: Demand a cost model that includes p95 run behavior, retries, tools, storage, infrastructure, and oversight. Reject proposals that quote only tokens or only subscription seats.

For builders: Instrument before scaling. Add hard run budgets, use model routing, design compact tools, and optimize cost per accepted result rather than raw token consumption.

For managed AI service providers: Price customer outcomes above fully loaded variable cost and operational risk. Protect margin with usage bands, scope limits, overage pricing, and workflow-specific service levels.

CYBERNOMICS®

CISO-approved Managed AI Service Provider (MAISP). We help organizations design, govern, deploy, and economically operate AI agents—with security, performance, adoption, and cost controls built in.

SOURCES AND METHODOLOGY

Official pricing and technical documentation

All prices are public US list-price examples accessed August 2, 2026. Rates change; verify current regional and contract pricing before procurement.

1. OpenAI. API Pricing. https://developers.openai.com/api/docs/pricing
2. OpenAI. Prompt caching. https://developers.openai.com/api/docs/guides/prompt-caching
3. OpenAI. Reasoning models. https://developers.openai.com/api/docs/guides/reasoning
4. OpenAI. Function calling — token usage. https://developers.openai.com/api/docs/guides/function-calling
5. OpenAI. Responses API migration — prior context billing. https://developers.openai.com/api/docs/guides/migrate-to-responses
6. OpenAI. MCP and Connectors pricing behavior. https://developers.openai.com/api/docs/guides/tools-connectors-mcp
7. OpenAI. Agents SDK usage tracking. https://openai.github.io/openai-agents-python/usage/
8. Anthropic. Claude Platform pricing. https://platform.claude.com/docs/en/about-claude/pricing
9. Microsoft. Copilot Studio billing rates and management. https://learn.microsoft.com/en-us/microsoft-copilot-studio/requirements-messages-management
10. Microsoft. Copilot Studio licensing. https://learn.microsoft.com/en-us/microsoft-copilot-studio/billing-licensing
11. Microsoft. Copilot Studio computer use licensing. https://learn.microsoft.com/en-us/microsoft-copilot-studio/computer-use
12. Microsoft Azure. Foundry Agent Service pricing. https://azure.microsoft.com/en-us/pricing/details/foundry-agent-service/
13. Google. Gemini Developer API pricing. https://ai.google.dev/gemini-api/docs/pricing
14. AWS. Amazon Bedrock pricing. https://aws.amazon.com/bedrock/pricing/
15. Microsoft Azure. Copilot Studio pay-as-you-go pricing. https://azure.microsoft.com/en-us/pricing/details/copilot-studio/

Methodology note. The worked examples are analytical scenarios created by Cybernomics from the listed public rate cards. They are not vendor quotes. Search-result tokens are assumed included in the input-token totals where explicitly stated, preventing double counting. Human labor, external SaaS transactions, taxes, enterprise discounts, regional premiums, and cloud infrastructure are excluded unless specified.