CODE EXECUTION WITH MCP READOUT

Code Execution with MCP

Load fewer tools, keep bulky data out of the model’s context

A visual readout of presenting MCP servers as code APIs: on-demand tool loading, filtering and control flow in code, privacy, state, and skills.

FIRST-PARTY POST AND DOCS / INDEPENDENT LUMINITY SYNTHESIS



Code execution with MCP

As tools multiply, direct calling can raise cost and latency. The pattern: present MCP servers as code APIs.

01

Why direct calling gets expensive

Most clients load every tool definition upfront	CONTEXT
Tool descriptions raise response time and cost	COST, LATENCY
Every intermediate result passes through the model	TOKENS
A 2-hour meeting could mean 50,000 more tokens	ARTICLE'S EXAMPLE
Large documents can exceed context limits	BREAKS THE FLOW
More copying mistakes with large or complex data	ACCURACY

Example: download a transcript from Google Drive and attach it to a Salesforce lead.

02

Present servers as code

EACH TOOL IS A FILE

```
servers
├── google-drive
│   ├── getDocument.ts
│   ├── ... (other tools)
│   └── index.ts
├── salesforce
│   ├── updateRecord.ts
│   ├── ... (other tools)
│   └── index.ts
└── ... (other servers)
```

- **List ./servers/** — to find servers like google-drive and salesforce — STEP 1
- **Read the tool files it needs** — like getDocument.ts and updateRecord.ts — STEP 2
- **Load only what the task needs** — the definitions for the current task — STEP 3

The article's implementation uses TypeScript and presents this as one way to do it, not the only way.

03

The same task, with and without code

Direct tool calls — 2 PASSES

The full transcript is returned into context, then written out again. It flows through the model twice.

With code — IN CODE

The transcript moves between tools inside the execution environment. The agent sees what the code logs or returns.

Stated result — ARTICLE

150,000 → 2,000 tokens, a 98.7% time and cost saving, stated where the article describes loading only the definitions a task needs. It does not say how this was measured.

ALSO CITED

- Cloudflare's "Code Mode" published similar findings
- "LLMs are adept at writing code"

Sources: Anthropic Engineering, "Code execution with MCP: Building more efficient agents," anthropic.com/research/code-execution-with-mcp (published Nov. 4, 2025; retrieved Nov. 5, 2026). Sections: introduction, "Excessive token consumption from tools makes agents less efficient," and "Code execution with MCP improves context efficiency." 150,000 to 2,000 tokens is the article's stated figure; it does not say which scenario it measures or how.

Attribution: Anthropic, Claude and Claude Code are trademarks of Anthropic PBC. Independent analysis of public documentation; not sponsored or endorsed by Anthropic. © 2026 Luminity Digital, Inc.

Load less, return less

Models are good at navigating filesystems, so definitions can be read when needed and results filtered in code.

01

Progressive disclosure

Navigate the filesystem — OPTION 1

Tools as code on a filesystem let the model read definitions on demand, not all up front.

Add a search_tools tool — OPTION 2

The agent searches, for example “salesforce,” and loads only the tools it needs.

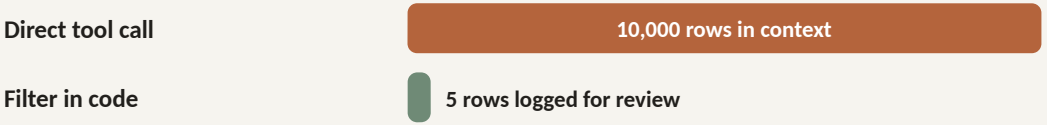
A DETAIL-LEVEL PARAMETER LETS THE AGENT PICK HOW MUCH TO LOAD

- Name only
- Name and description
- Full definition with schemas

02

Filter before results reach the model

FETCHING A 10,000-ROW SPREADSHEET




```
const allRows = await gdrive.getSheet({ sheetId: 'abc123' });
const pendingOrders = allRows.filter(row =>
  row["Status"] === 'pending'
);
console.log(`Found ${pendingOrders.length} pending orders`);
console.log(pendingOrders.slice(0, 5)); // Only log first 5 for review
```

03

Control flow runs as code

WAITING FOR A DEPLOYMENT NOTIFICATION IN SLACK

```
let found = false;
while (!found) {
  const messages = await slack.getChannelHistory(
    { channel: 'C123456' });
  found = messages.some(m =>
    m.text.includes('deployment complete'));
  if (!found) await new Promise(r => setTimeout(r, 5000));
}
console.log('Deployment notification received');
```

- ✓ Loops, conditionals, and error handling use familiar code — NOT CHAINED CALLS
- ✓ More efficient than alternating tool calls and sleep commands — LOOP IN CODE
- ✓ Lower “time to first token”: the environment runs the if — LATENCY

Similar patterns work for aggregations, joins across sources, or extracting fields, without bloating the context window.

Sources: Anthropic Engineering, “Code execution with MCP: Building more efficient agents,” anthropic.com/research/code-execution-with-mcp (published Nov. 4, 2025; retrieved Nov. 5, 2026). Sections: “Progressive disclosure,” “Context efficient tool results,” and “More powerful and context-efficient control flow.” The bars are scaled by Luminity; the code blocks are the article’s, with some lines wrapped for width.

Attribution: Anthropic, Claude and Claude Code are trademarks of Anthropic PBC. Independent analysis of public documentation; not sponsored or endorsed by Anthropic. © 2026 Luminity Digital, Inc.

Privacy and state

Intermediate results stay in the execution environment by default. Files and saved code carry work across runs.

01

Privacy-preserving operations

1 **Google Sheets** — rows with real emails, phones, names

2 **MCP client** — tokenizes PII; keeps the lookup

3 **Salesforce** — receives the real values

WHAT THE AGENT SEES IF IT LOGGED sheet.rows

```
{ salesforceId: '00Q...', email: '[EMAIL_1]',
  phone: '[PHONE_1]', name: '[NAME_1]' },
```

BY DEFAULT

Intermediate results stay in the execution environment, so data you don't want to share never enters context

SENSITIVE WORKLOADS

The harness can tokenize PII automatically: the MCP client intercepts it before it reaches the model

DETERMINISTIC RULES

Define security rules for where data can flow to and from

02

State persistence and skills

1 **Write to files** — intermediate results, so work can resume

2 **Save as a function** — working code kept in ./skills/

3 **Add a SKILL.md** — a structured skill models can reference

Toolbox — higher-level capabilities

Skills are folders of reusable instructions, scripts, and resources for models to improve performance on specialized tasks.

LUMINITY NOTE

- The article describes tokenization as something the harness or MCP client can do

NOT STATED

- Any specific product that ships it
- Confirm what your own harness implements

Sources: Anthropic Engineering, "Code execution with MCP: Building more efficient agents," anthropic.com/research/code-execution-with-mcp (published Nov. 4, 2025; retrieved Nov. 5, 2026). Sections: "Privacy-preserving operations" and "State persistence and skills." The flow graphic is Luminity's rendering; the tokenized output is the article's, wrapped for width. The note strip is Luminity's. Informational, not legal or compliance advice.

Attribution: Anthropic, Claude and Claude Code are trademarks of Anthropic PBC. Independent analysis of public documentation; not sponsored or endorsed by Anthropic. © 2026 Luminity Digital, Inc.

Weigh the cost

The article's own caveat and summary, then a Luminity decision aid that maps its patterns to common situations.

01

The article's caveat

- ! Code execution introduces its own complexity — ARTICLE
- ! Agent-generated code needs a secure execution environment — ARTICLE
- ! Sandboxing, resource limits, and monitoring — ARTICLE
- ! Operational overhead that direct tool calls avoid — ARTICLE

Weigh against · reduced token costs, lower latency, improved tool composition

02

Your situation → the matching pattern

- ✓ Many servers or tools connected → PROGRESSIVE DISCLOSURE
- ✓ Large documents or datasets between tools → FILTER IN CODE
- ✓ Polling, retries, or branching → CONTROL FLOW IN CODE
- ✓ Sensitive fields in tool data → TOKENIZE AT CLIENT
- ✓ Tasks the agent repeats → SAVE AS SKILLS

Luminity's reading, not tested guidance. Check first: sandbox, resource limits, monitoring.

03

What the source is, and is not

The summary — ARTICLE

The problems have known solutions from software engineering. Code execution applies those patterns to agents.

Not product documentation — LUMINITY

An engineering post dated Nov. 4, 2025. It states no eligibility, availability, or pricing conditions.

Not named — LUMINITY

No specific sandbox or tokenization product, and no method for the 150,000 to 2,000 token figure.

THE AUTHORS

- Invite readers who implement this to share findings with the MCP community
- Anthropic's docs now describe related features (next slide)

Sources: Anthropic Engineering, "Code execution with MCP: Building more efficient agents," anthropic.com/research/code-execution-with-mcp (published Nov. 4, 2025; retrieved Nov. 5, 2026). Sections: the caveat paragraph that closes "State persistence and skills," and "Summary." The situation-to-pattern mapping and the "not product documentation" and "not named" cards are Luminity's observations.

Attribution: Anthropic, Claude and Claude Code are trademarks of Anthropic PBC. Independent analysis of public documentation; not sponsored or endorsed by Anthropic. © 2026 Luminity Digital, Inc.

In Anthropic's documentation

Documented features that sit next to the article's pattern, with each page's data-handling line (retrieved Nov. 5, 2026).

01

Article idea → documented feature

- ✓ **Tool search tool** — Progressive disclosure. Claude searches your tool catalog and loads only the tools it needs. DOCS
- ✓ **Programmatic tool calling** — Filter and loop in code. Claude writes code in the code execution container that calls your tools. DOCS
- ✓ **Agent Skills** — Reusable skills. Metadata always loaded, instructions on trigger, resources as needed. Use only trusted Skills. DOCS
- ✓ **Code execution tool · Claude Code sandbox** — Sandboxing. Hosted container: 1 CPU, 5 GiB RAM, 5 GiB disk, no internet. Claude Code's sandbox is off by default. DOCS
- ✓ **MCP connector** — Reaching MCP servers. Remote servers over HTTP; only tool calls; no local stdio. Per the programmatic tool calling page, MCP connector tools cannot be called programmatically. DOCS

02

Data-handling lines

CLAUDE API DOCS PAGES	ZDR	HIPAA
Tool search tool	ELIGIBLE*	NOT ELIGIBLE
Programmatic tool calling	NOT ELIGIBLE	NOT ELIGIBLE
Code execution tool	NOT ELIGIBLE	NOT ELIGIBLE
MCP connector	NOT ELIGIBLE	NOT ELIGIBLE
Agent Skills	NOT ELIGIBLE	NOT ELIGIBLE

* ZDR = zero data retention. The tool search page says "Excludes Covered Models."

Chips copy the Claude API pages. Claude Code pages carry no such line; there, in a HIPAA-configuration session, tool search is off.

LUMINITY NOTE

- The hosted container has no internet access, so it does not import MCP servers the way the article's code does

ALSO IN THE DOCS

- Container data kept up to 30 days | MCP connector is Beta | Programmatic calling: not on Haiku 4.5 | API Skills need code execution

Sources: Anthropic Claude Platform Docs, platform.claude.com/docs/en/agents-and-tools/: tool-use/tool-search-tool, tool-use/programmatic-tool-calling, tool-use/code-execution-tool, mcp-connector, agent-skills/overview. Claude Code Docs, code.claude.com/docs/en/: mcp, sandboxing. Retrieved Nov. 5, 2026. Eligibility, availability, and pricing change; confirm on the live pages. Informational, not legal or compliance advice. The mapping and the first strip row are Luminity's; the second row is from the docs pages.

Attribution: Anthropic, Claude and Claude Code are trademarks of Anthropic PBC. Independent analysis of public documentation; not sponsored or endorsed by Anthropic. © 2026 Luminity Digital, Inc.