Gemini API Tool Retrieval: how defer_loading keeps large agent tool libraries cheap

21 November 2026 · 7 min read · blogs.iictp.org/gemini-api-tool-retrieval-defer-loading

Short answer: Tool Retrieval is a preview feature of the Gemini API's Interactions API, announced by Google on 19 November 2026, that lets an agent keep a large tool library on standby and load each tool's parameter schema only when it needs it. You add a tool_search tool and set defer_loading to true on the tools to hold back; the model still sees every deferred tool's name and description, searches for the one it needs, and the API loads the full schema and lets the model call it inside the same request.

TL;DR The short version

- Tool Retrieval, a Gemini API preview announced on 19 November 2026, lets an agent load each tool's full schema only when it needs it.
- You add a tool_search tool and set defer_loading on tools; the model still sees every name and description and searches for the one it needs.
- Because the model chooses from names and descriptions alone, write them like search documents and keep frequently used tools loaded up front.

Read online, listen to it, or change the text size: https://blogs.iictp.org/gemini-api-tool-retrieval-defer-loading

What is Gemini API Tool Retrieval?

Function calling normally means sending the full JSON schema of every tool with every request. With dozens or hundreds of tools, Google says this inflates input tokens, adds latency and lowers tool-selection accuracy as the candidate list grows. Tool Retrieval keeps the long tail of schemas out of the prompt until they are used.

How it works

- **Server-side retrieval (default).** Add {"type": "tool_search"} to the tools list and mark tools with defer_loading: true. When the model needs one, it emits a tool_search_call naming the function, the API returns a tool_search_result with the schema, and the model emits a normal function_call. All three steps appear in interaction.steps, which helps when debugging tool names and descriptions. There is no index for you to build or host.
- **Client-side retrieval.** If you already maintain a tool registry, an embedding index or per-user permission filtering, declare tool_search with "execution": "client" and your own search parameters. The model calls your search function, you return matching function schemas in a function_result, and the conversation continues. This adds

one round trip per search, and tools do not have to be declared up front, so new integrations can appear mid-conversation.

- **Remote MCP servers.** Setting defer_loading: true on an mcp_server tool loads every tool that server exposes on demand, so a large server can be connected without listing its tools.

What is new compared with earlier approaches

Teams have worked around large tool sets by hand-building routers, splitting agents by domain, or running their own embedding search before each call. Tool Retrieval moves that step into the model's own loop: the model decides which tool it needs from names and descriptions, and the schema arrives in the same request. Google also notes an interaction with prefix caching: deferred schemas stay out of the start of the prompt and loaded schemas are appended later, so the cached prefix stays stable from turn to turn.

Pricing and supported models, as stated by Google

- No separate charge. Tokens are billed at the standard rate of the model you call.
- Enabling tool_search adds a small, fixed block of input tokens, and each schema the model loads is billed as input for the rest of the conversation, like a tool declared up front.
- You save tokens when the schemas you never load outweigh the ones you do. Google advises measuring this against your own traffic.
- Works with Gemini 3.6 and later models, including all Gemini 4 models, in the Interactions API. Google's example uses the model ID gemini-4-hydrogen.

Limitations and open questions

- **Descriptions carry the load.** The model chooses from names and descriptions alone, so vague or overlapping descriptions will now cause misses rather than just noise. This is our reasoning from the design, not a measured result.
- **Not every tool should be deferred.** Google recommends keeping frequently used tools loaded up front and deferring the long tail.
- **Preview.** The feature is in preview in the Interactions API, so field names and behaviour may change.
- **Access control is still yours.** Server-side retrieval searches what you declared. If users must only see tools they are allowed to use, filter before declaring, or use client-side retrieval with your permission check.

How to use Tool Retrieval

1. Use a supported model (Gemini 3.6 or later) through the Interactions API with the google-genai SDK.
2. Add {"type": "tool_search"} to your tools list.
3. Set "defer_loading": true on each function or mcp_server tool you want to load on demand; leave your most-used tools undeferred.
4. Inspect interaction.steps to see which tools the model searched for and loaded.
5. Compare input tokens and tool-selection accuracy with and without deferral on a sample of real requests.

What it means for engineers and technical teams

- Write tool names and descriptions as if they were search documents: specific verbs, distinct scopes, no near-duplicates.
- Log tool_search steps. They are a cheap signal for which tools are discoverable and which are never found.
- Re-check prompt-injection exposure when connecting large MCP servers, since more tools become reachable. See what is prompt injection.

Frequently asked questions

- **What does defer_loading do in the Gemini API?** It keeps a tool's parameter schema out of the prompt until the model asks for that tool through tool_search.
- **Does Gemini API Tool Retrieval cost extra?** Google says there is no separate charge; you pay normal token rates, including a small fixed block for tool_search and the schemas that get loaded.
- **Can I use my own tool search?** Yes. Declare tool_search with "execution": "client" and return matching schemas from your own registry.
- **Does it work with MCP?** Yes, with remote MCP servers in the Gemini API, by setting defer_loading on the mcp_server tool.

Related reading

- What is Model Context Protocol?
- AI agents explained
- What is an LLM context window?
- Applied AI in the TPBOK

Sources were read on 21 November 2026. Preview APIs, model IDs, prices and access rules change often, so check the linked documentation before you build on them.

> Related credential: AI Practitioner. Explore the exam facts and syllabus.

Sources and further reading

1. Google (Loher, L., Kassner, N.). Introducing Tool Retrieval in the Gemini API. 2026-11-19. blog.google/innovation-and-ai/technology/developers-tools/gemini-api-tool-retrieval

Published by IICTP, an independent technical credential body, on 21 November 2026. The current version of this post is at https://blogs.iictp.org/gemini-api-tool-retrieval-defer-loading. More guides: blogs.iictp.org.