MCP DEV SUMMIT · APRIL 3

Path to V2 for MCP SDKs

Max Isbey · Anthropic · MCP Python SDK maintainer

Max Isbey

Python SDK maintainer

FROM	New Zealand
PREVIOUSLY	Rocket Lab — Operations Flight Software Engineer / AI stuff
NOW	MCP Python SDK · Anthropic
FOCUS	Open Source MCP Python SDK Maintainer

Today

01 **Why horizontal scaling is painful**
Transport history and the horizontal scaling problem

02 **MRTR**
Multi-round-trip requests and compatibility layers

03 **What we changed**
FastMCP rename, low-level cleanup, pluggable transports

04 **Migration Eval Framework**
Automated migration guide testing

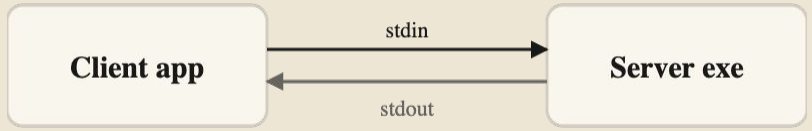
SCALING

The original transports

November 2024

Standard I/O

Local, subprocess



✓ Bidirectional · simple · still most common

HTTP + SSE deprecated

Remote, persistent connection

✗ Connection drops → session lost · 1 conn per client

SCALING

Streamable HTTP

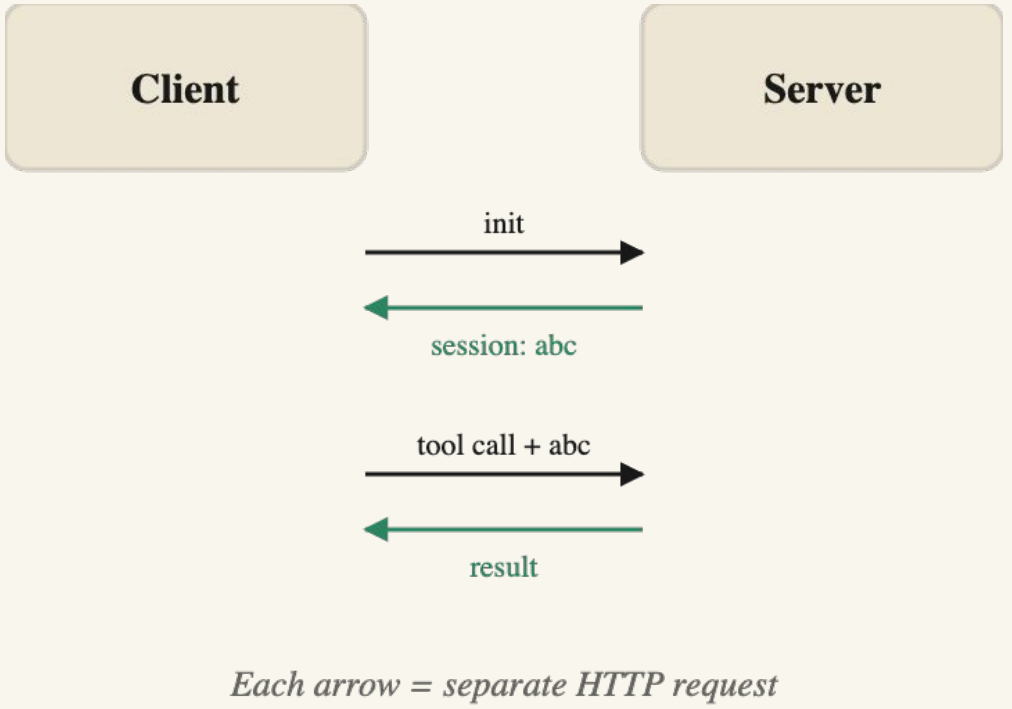
March 2025 — replaces HTTP+SSE

What changed

Requests and responses are individual HTTP requests. No more persistent SSE stream required. Results come back as the response to the same request.

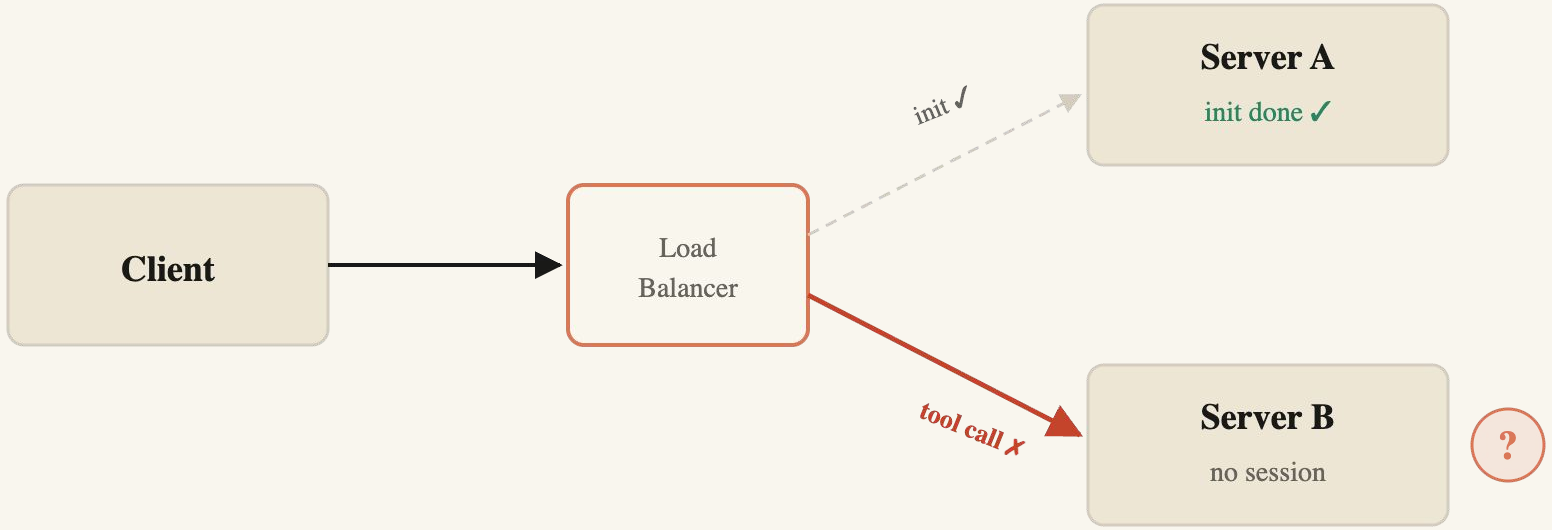
Session IDs

Now an official spec concept (previously just SDK convention). Server assigns on init, client includes in all subsequent requests.



SCALING

The horizontal scaling problem



Streamable HTTP

Elicitation

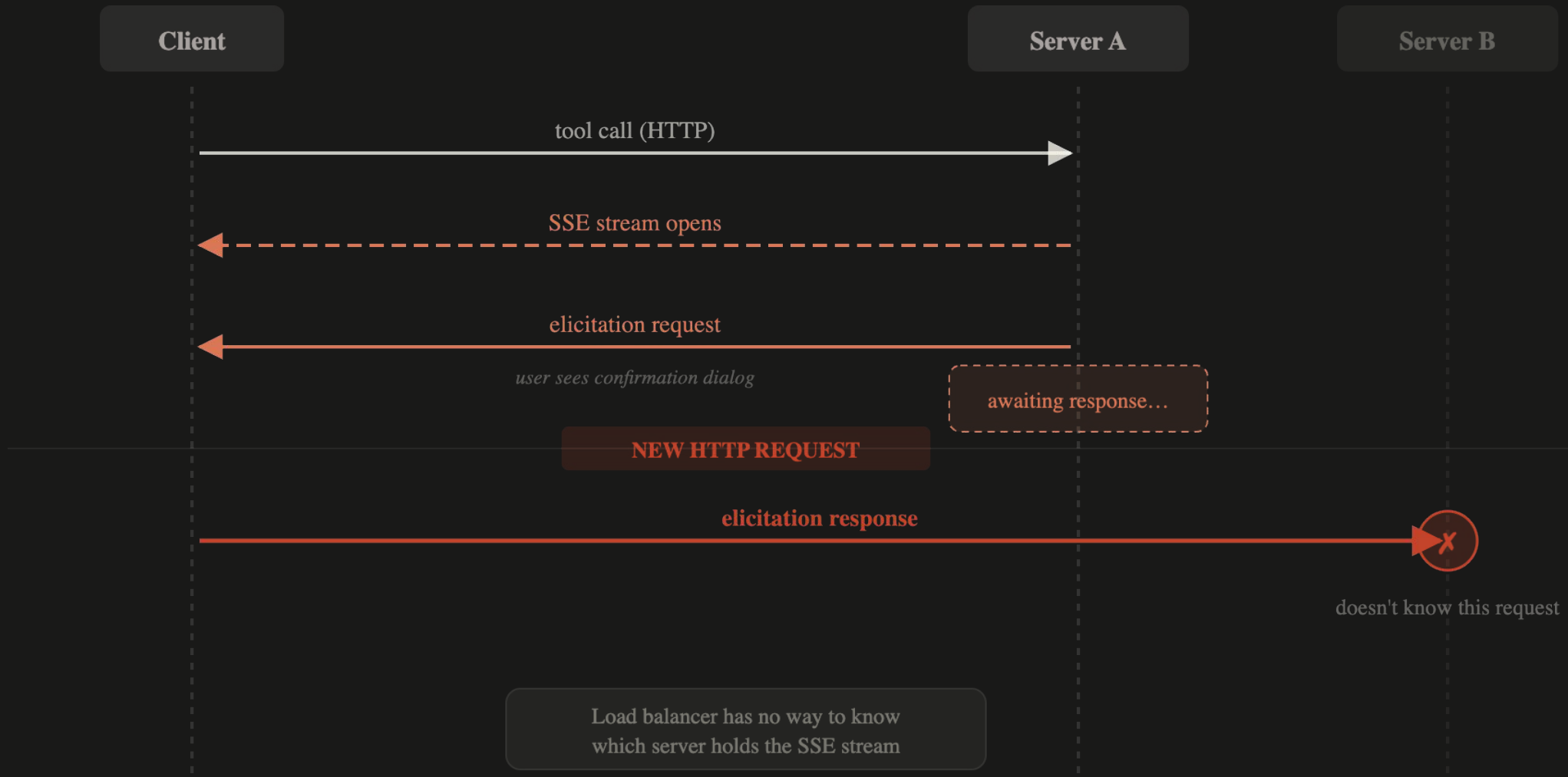
```python
@mcp.tool()
async def delete_folder(path: str, ctx: Context) -> str:
    files = list_files(path)

    if files:
        result = await ctx.elicit("Folder has files. Delete anyway?")
        if not result.confirmed:
            return "Cancelled"

    delete(path)
    return f"Deleted {path}"
```

SCALING

How elicitation works

SCALING

Elicitation breaks at scale



SCALING

The unofficial fix

Unofficial stateless mode (SDK convention)

- Initialization is optional — skip the handshake
- No session IDs assigned
- Tool calls work without prior init
- Unlocks simple actions on scaled deployments

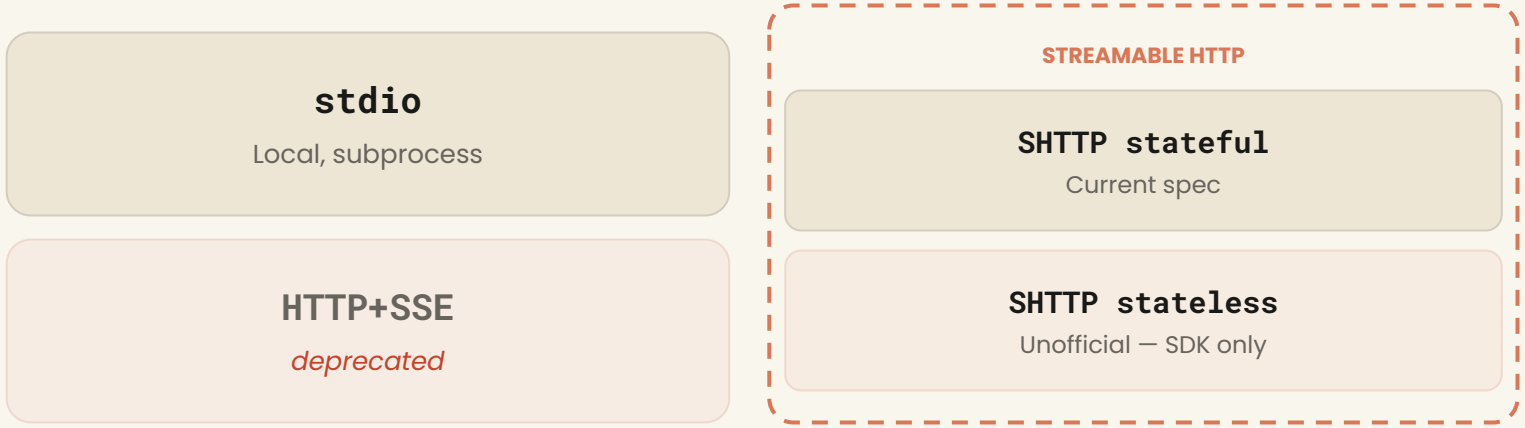
But...

- No session ID → can't correlate elicitation responses
- **Elicitation and sampling won't work**
- Only solves half the problem
- Not in the official spec (*technically not spec compliant*)

That's what the June 2026 spec is trying to solve →

SCALING

Recap: 3–4 transports



SCALING

Recap: 3–5 transports

stdio

Local, subprocess

HTTP+SSE

deprecated

STREAMABLE HTTP

SHTTP stateful

Current spec

SHTTP stateless

Unofficial — SDK only

SHTTP stateless

Official — **June 2026 spec**

NEW

All three Streamable HTTP variants technically share the same name

MRTR

Multi Round-Trip Requests

BEFORE — SSE stream

- Client → tool call
- Server opens SSE stream
- stream held open ∞
- Server → "confirm?"
- **Client → new HTTP request**

✗ Wrong server. Stream on another box.

AFTER — MRTR

ROUND 1

- Client → tool call
- **IncompleteResult + "confirm?"**

✓ Request ends cleanly

ROUND 2

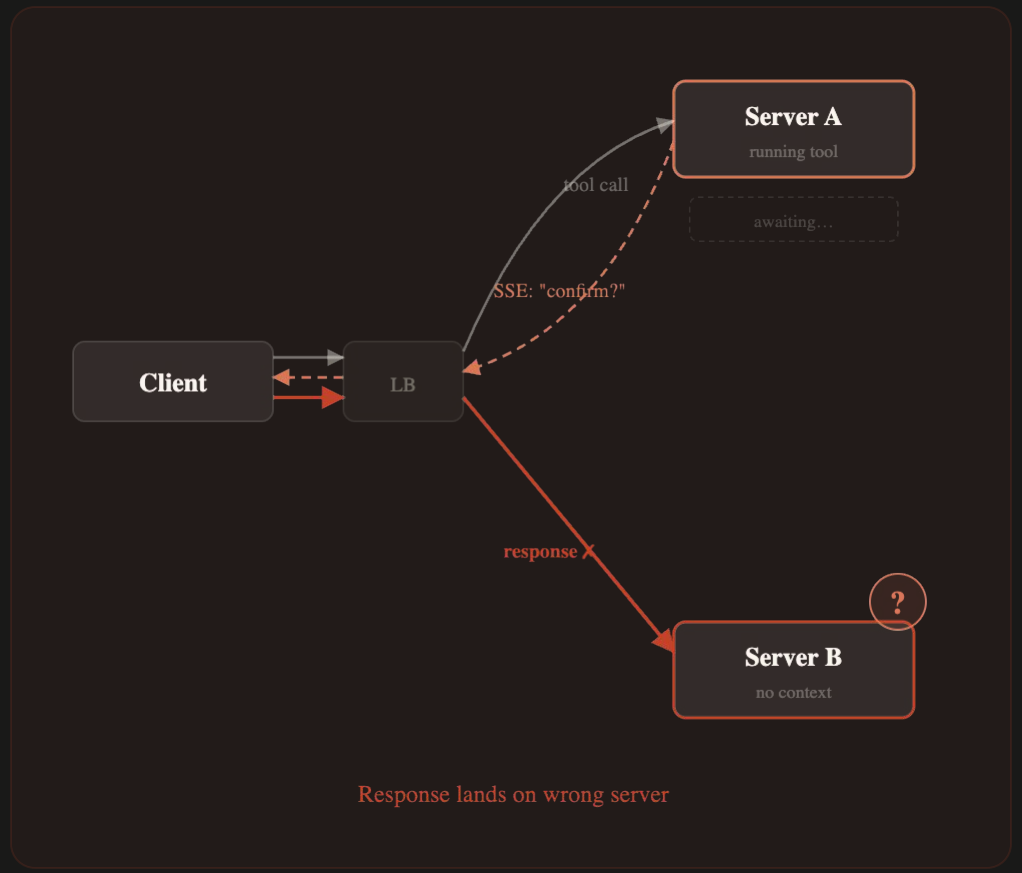
- Client → same call + answer
- **Server → final result**

✓ Request ends cleanly

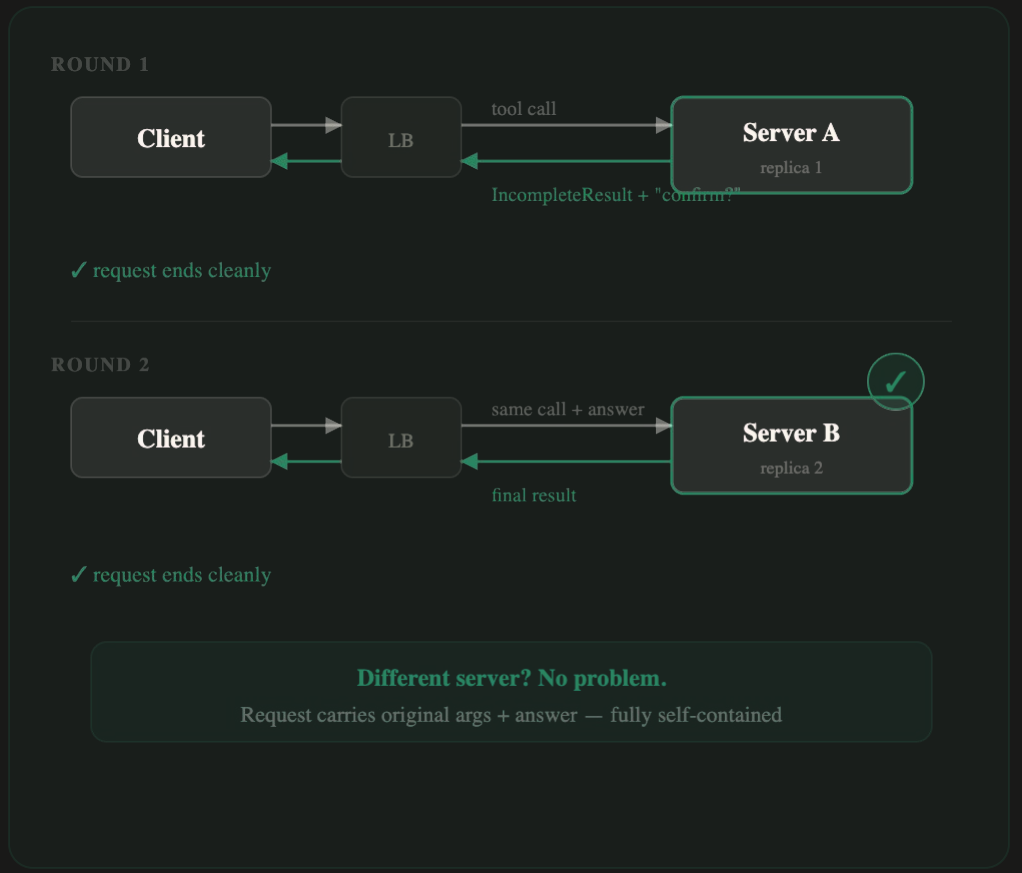
MRTR

MRTR solves horizontal scaling

BEFORE — SSE elicitation



AFTER — MRTR



MRTR

Elicitation today

```python
@mcp.tool()
async def delete_folder(path: str, ctx: Context) -> str:
    files = list_files(path)

    if files:
        result = await ctx.elicit("Folder has files. Delete anyway?")
        if not result.confirmed:
            return "Cancelled"

    delete(path)
    return f"Deleted {path}"
```

Sequential. The await blocks. If you know Python, you can read this.

MRTR

With MRTR

```python
@mcp.tool()
def delete_folder(path: str, params) -> Result | IncompleteResult:
    files = list_files(path)              # ← runs every round

    if files:
        if "confirm" not in params.responses:
            return IncompleteResult(
                ask="Folder has files. Delete anyway?"
            )

        if not params.responses["confirm"]:
            return "Cancelled"

    delete(path)
    return f"Deleted {path}"
```

Function re-runs from the top each round. Side effects fire every time.

MRTR

Your code doesn't change - Compatibility layer

unchanged

```python
@mcp.tool()
async def delete_folder(path: str, ctx: Context) -> str:
    files = list_files(path)

    if files:
        result = await ctx.elicit("Folder has files. Delete anyway?")
        if not result.confirmed:
            return "Cancelled"

    delete(path)
    return f"Deleted {path}"
```

SDK parks the coroutine. Complexity stays in the dispatch layer.

MRTR

SDK compatibility layer

WHAT YOU WRITE

```python
@mcp.tool()
async def delete_folder(path: str, ctx: Context) -> str:
    files = list_files(path)

    if files:
        result = await ctx.elicit("Folder has files. Delete anyway?")
        if not result.confirmed:
            return "Cancelled"

    delete(path)
    return f"Deleted {path}"
```

under the hood

Familiar sequential code — no re-entrancy

WHAT THE SDK DOES

1. **Function runs**
   `list_files()` executes normally
2. **Hits `await ctx.elicit()`** ⏸ frozen
   SDK in memory **parks coroutine**
3. **Sends IncompleteResult**
   HTTP request ends cleanly → client gets elicitation
4. **Client re-calls with answer**
   `same tool call + elicitation response`
5. **SDK coroutine resumes** ▸ resumed
   Function continues from the line with the result `await`

For non-horizontally-scaled servers · coroutine held in memory between rounds

`exact impl TBD · task augmentation also under consideration`

ADDITIONAL CHANGES

Other things we have changed

Renamed to avoid confusion with external FastMCP

`FastMCP (name collision)`

✓ **`MCPServer`**

Low-level: decorators → explicit constructor args

`@server.call_tool()`

✓ **`Server(call_tool=fn)`**

Types cleanup across low-level interface

`Inconsistent/untyped`

✓ **`Fully typed handlers`**

ADDITIONAL CHANGES

Other things we have changed

Renamed to avoid confusion with external FastMCP

`FastMCP (name collision)`

✅ **`MCPServer`** ✨

Low-level: decorators → explicit constructor args

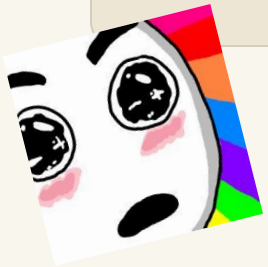
`@server.call_tool()`

✅ **`Server(call_tool=fn)`**

Types cleanup across low-level interface

`Inconsistent/untyped`

✅ **`Fully typed handlers`**



CHANGES

Pluggable transports

"Pluggable transports" usually means swapping both bottom layers.

gRPC with protobuf over JSON-RPC makes no sense — you'd smuggle JSONRPC as a string and lose protobuf's whole value proposition.

Solution: Dispatcher pattern — swap both layers

CHANGES

The Dispatcher pattern

BEFORE

BaseSession

MCP semantics

initialize(), call_tool(), list_tools()… progress tokens, cancellation, validation

19 methods

Wire protocol tangled

JSON-RPC wrap, ID correlation, receive loop, stream management

Want gRPC? Reimplement the whole session.

extract

AFTER

python-sdk PR #2320

BaseSession

MCP semantics only

initialize(), call_tool(), list_tools()…

19 methods — unchanged

Dispatcher 5 methods

JSON-RPC (default) | gRPC | Protobuf | …

send_request · send_notification · send_response set_handlers · run

Implement 5 methods → all 19 MCP methods work for free

Draft PR · design settled

VALIDATION

Migration documentation eval framework

VALIDATION

The eval fleet

MIGRATION.md
864 lines
the thing under test

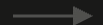
v1 code
8 scenarios

v2 wheel
installed in venv

24 isolated workers · parallel-8

01-r1	01-r2	01-r3
02-r1	02-r2	02-r3
03-r1	03-r2	03-r3
04-r1	04-r2	04-r3
05-r1	05-r2	05-r3
06-r1	06-r2	06-r3
07-r1	07-r2	07-r3
08-r1	08-r2	08-r3

peek-logger hook allows the read, just logs it → peeks.log ×24



result.json ×24

139

gap reports
clustered → ranked

12

De-duplicated action items

VALIDATION

The 24 runs

20/24 passed **5**/24 peeked **139** individual reports

	r1	r2	r3
01 - examples-fastmcp	⚠	⚠	⚠
02 - examples-snippets	✓	✓	✓
03 - examples-servers	✓	✓	✓
04 - examples-clients	✓	✓	✓
05 - lowlevel-heavy	✓	✓	✓
06 - fastmcp-power	✗	✗	✗
07 - client-heavy	✓	✓	✓
08 - type-edges	✓	⚠	✓

passed, no peek · passed, peeked · verify failed

Row 01 — all 3 peeked

Consistent doc hole: two undocumented removals

Row 06 — all 3 failed verify

Migrations were correct — verify.sh used a v1-only attribute. All 3 workers diagnosed it independently.

“DOC GAP: migration guide does NOT call out the `camelCase → snake_case` field naming change. I only figured it out from the runtime error's suggestion.”

— Claude worker 07-r2, notes.md

21/24 runs · 7 of 8 scenarios

CLOSE

What’s next - Python SDK

BETA — Q2 2026

ROADMAP

Dispatcher extraction	*Draft PR*
Auth rework	*In design*
Stateless by default	*Pending June spec*
MRTR	*Pending June spec*
Docs	*Ongoing*

`github.com/modelcontextprotocol/`**`python-sdk`**

Maintainers

`@maxisbey` `@Kludex` `@felixweinberger` `@pcarleton`

Full stable release planned for the next major MCP spec release

Feedback? Please file an issue on GitHub

CLOSE

TypeScript SDK

v2 ALPHA — OUT NOW

"Any runtime. Any validator. Less boilerplate."

Runs everywhere

Node, Bun, Deno, Cloudflare Workers — runtime auto-detected, zero config

Bring your own stack

Zod, Valibot, Arktype, or raw JSON Schema. Hono, Next.js, SvelteKit natively.

Less boilerplate

Flat imports, fully typed handlers, ctx helpers — no deep paths or schema imports

Enterprise-ready

SSO built in, redesigned errors, protocol extensions, aligned with next spec

Migration on autopilot

Point your coding agent at the bundled migration skill — upgrading is a prompt

`github.com/modelcontextprotocol/typescript-sdk`

Maintainers

`@felixweinberger` `@mattzcarey` `@kkonstantinov` `@pcarleton`

Stable release planned for the next major MCP spec release

Feedback? Please file an issue on GitHub or upvote existing ones

Thanks

github.com/modelcontextprotocol/python-sdk