CELLULE-PRO DOCUMENTATION

Tools Mode

Function-calling · OpenAI-compat tool-use

cellule.ai · pro.cellule.ai
2026-05-18

Contents

0.1 Tools-mode — universal routing for tool-calling clients

Cellule-PRO automatically routes tool-calling requests (`POST /v1/chat/completions` with `tools=[...]`) to the **tool-fluent** workers in the fleet, that is, those that can chain several OpenAI-conformant `tool_calls` JSON objects per the schema.

This page documents the mechanism, the recommended models, monitoring and the admin override.

1. Why a dedicated mechanism

Not all LLMs can do **reliable** tool-calling. On a simple 1-step prompt (`get_weather('Paris')`), a Qwen 3.5 9B emits a valid `tool_call`. But on a 2+ step prompt (`list_dir → write_file`, typical of a coding agent like opencode/Cursor/Continue), the 9B **derails at step 2**: it prefers to generate code as markdown in `content` rather than calling the tool.

Doctrine validated by David 2026-04-29: tools-mode requests must be routed to **runtime-validated tool-fluent models**, not to a coarse `params_b  7B` threshold.

2. Runtime detection — no per-model hardcode

Cellule-PRO **does not hardcode** the list of tool-capable models. The shipped image is universal: a customer can connect a worker with any GGUF (Llama, DeepSeek, Mistral, Phi, Gemma, Qwen, etc.). The pool **probes** each worker on handshake via a standardized V2 tool-bench.

Multi-step V2 bench

At boot, the pool sends the worker a 2-turn conversation:

1. **Step 1**: prompt `"Use list_dir to see what's in /tmp, then use write_file to save /tmp/bench.txt with content 'hello world'."`
   - 2 tools `list_dir` and `write_file`. The worker must emit `tool_calls=[list_dir(/tmp)]`.
2. **Step 2**: the pool injects a fake `list_dir` result (`bench_existing.txt\nworkfile.json\nlogs/`) and asks the worker to continue. The worker must emit `tool_calls=[write_file(/tmp/bench.txt, ...)]`.

Result	Verdict	
Step 1 OK + Step 2 OK	`tools_capable=TRUE`	
Step 1 OK + Step 2 markdown	`tools_capable=FALSE`	(typical 9B)
Step 1 without tool_call	`tools_capable=FALSE`	(typical 4B)
Step 1 timeout/error	`tools_capable=FALSE`	

Cost: ~50-100 prompt tokens × 2 + ~50 completion tokens × 2 = ~300 tokens per worker, ~5-10 sec depending on the worker. Run in the background post-handshake, **non-blocking** for boot.

DB persistence

Migration `032_tools_capable.sql` adds:

Column	Type	Meaning
`tools_capable`	`BOOLEAN`	NULL=not tested, TRUE=V2 bench OK, FALSE=fail
`tools_bench_at`	`TIMESTAMP`	Last bench (manual/auto re-bench)
`tools_bench_score`	`REAL`	0.0 or 1.0 (V1: binary)

3. Routing filter

When a request arrives with `tools=[...]` (or a conv already sticky-tagged tools-mode, cf. rc57):

```
# Pseudocode pool.get_available_worker
if require_tools_capable:
    candidates = [w for w in workers if w.info["tools_capable"] is True]
    # Strict: FALSE and NULL excluded
```

The `params_b ≥ 7B` stays as a **safety guard** (a 4B even falsely tagged OK will not serve a heavy payload).

Fail-open at bootstrap

If **no** worker has yet passed its bench (post-pool restart, series of reconnects), the `tools_capable=True` filter is temporarily relaxed: the pool falls back to the `params_b` floor only. This avoids systematic 503 during the bench window (~5-30 sec). The filter becomes strict again as soon as at least 1 worker has PASSED.

Env toggle `ROUTING_TOOLS_REQUIRE_CAPABLE=on|off` (default `on`).

4. Recommended models

List valid per the 2026-04-29 doctrine (will be updated with future tests). V2 bench PASSES on:

Model	tools_capable V2 bench
Qwen3-Coder-30B-A3B-Instruct (MoE)	(recommended)
Qwen 3.6 35B-A3B (MoE)	(recommended)
Qwen 2.5 32B-Instruct	(to validate)
Qwen 2.5 14B-Instruct	(to validate)
Qwen 2.5 Coder 7B-Instruct	(to validate — Coder but small)
Qwen 3.5 9B	tool-chaining derails at step 2
Qwen 2.5 7B-Instruct	(to bench)
Qwen 3.5 4B / 2B	too small

IT-admin deploy recommendation: to enable opencode/Cursor/Continue reliably, plan at least **one 30B-A3B+ worker** in the fleet. The pool automatically routes tool-calls to it; the other workers (9B/4B) serve normal chat requests without degradation.

5. Monitoring the status

Via the admin dashboard

"Workers" tab → **Tools** column:

Badge	Meaning
green	`tools_capable=TRUE` (passes V2 bench)
red	`tools_capable=FALSE`
gray	NULL — bench not yet launched or in progress

(UI to be delivered soon — the DB value is exposed as of rc58.)

Via SQL

```
SELECT worker_id, model_path, tools_capable, tools_bench_at, tools_bench_score
FROM workers
WHERE is_online = TRUE
ORDER BY tools_capable DESC NULLS LAST, worker_id;
```

Via logs

```
cellule.tools_checker  INFO  tools_bench Cellule-Coder-test: PASS (ok, 5.2s)
cellule.tools_checker  INFO  tools_bench Cellule-9B-Alpha-test: FAIL (step2 derail markdown ..., 8.1
```

6. Manual re-bench

Manually via SQL (resets the value, the pool re-benches on next handshake):

```
UPDATE workers SET tools_capable = NULL, tools_bench_at = NULL, tools_bench_score = NULL
WHERE worker_id = 'Cellule-Coder-test';
```

Then kick the worker: `docker restart cellule-worker-XXX` or let the pool re-bench at the next heartbeat.

("Re-bench" UI button to be delivered soon.)

7. Admin override

If the admin knows a worker is tool-fluent but the V2 bench fails (false negative) or the other way around:

```
-- Force TRUE (bypass bench)
UPDATE workers SET tools_capable = TRUE WHERE worker_id = 'My-Worker';

-- Force FALSE (excluded from tools-mode)
UPDATE workers SET tools_capable = FALSE WHERE worker_id = 'Buggy-Worker';
```

The override is **persistent**: as long as the worker is not manually re-benched (cf. §6), the value stays.

8. Validated opencode client configuration

```
{
  "$schema": "https://opencode.ai/config.json",
  "model": "cellule/CELLULE",
  "provider": {
    "cellule": {
      "npm": "@ai-sdk/openai-compatible",
      "name": "Cellule-PRO Pool",
      "options": {
        "baseURL": "http://<POOL_VIP>:18095/v1",
        "apiKey": "sk-cellule-XXXXXXXXXXXXXXXXXXXXXXXX"
      },
      "models": {
        "CELLULE": {
          "name": "Cellule-PRO Smart Pool",
          "limit": { "context": 131072, "output": 4096 }
        }
      }
    }
  }
}
```

Notes:

- `<POOL_VIP>` = virtual IP address of the HA virtual pool (rc62 workstream), entered in the first-boot wizard at the *Network → HA virtual pool* step. All pools in the cluster share this VIP — if one falls, another takes over in less than 3 seconds via VRRP. Cf. ADMIN_GUIDE § "HA virtual pool". If you did not enable the VIP in the wizard (single-pool deployment without HA), use the concrete pool IP instead.
- **No** `"tools": true` in the model block (the `@ai-sdk/openai-compatible` provider auto-detects native OpenAI tools). Adding it can interfere with detection.
- `limit.context` should reflect the max ctx of your widest tool-fluent worker (e.g. `131072` if you have a Coder 30B-A3B at `n_ctx=131072`).
- The pool automatically routes to the tool-fluent worker — no need to specify a particular model_id on the client side.

9. Known limitations

- **V2 bench = 2-turn harness**: does not test nested tools (nested objects), arrays, enums, or 5+-turn loops. A model can PASS V2 and fail on a very complex opencode case. For those, the admin override (§7) to FALSE evicts the false positive.

- **Re-bench is not automatic** on model change today. If a worker changes model (SELF-HEAL, update_model), the old `tools_capable` stays in RAM until the next pool restart or manual re-bench. (TODO: auto invalidation.)
- **No cross-language bench**: the V2 bench prompt is in English. A model that PASSES in English may fail in French. (TODO: bilingual prompt.)

10. Doctrine summary

1. Cellule-PRO **detects** tool capability at runtime, **not** by hardcoded model-name.
2. Tools-mode routing is **strict**: only a worker proven to PASS the V2 bench receives tools traffic.
3. The customer IT admin has **nothing to configure** on the pool side — they deploy a fleet, the pool discovers which workers are tool-fluent.
4. To enable a reliable coding agent (opencode/Cursor/Continue) you **must** have a 30B-A3B+ worker in the fleet.
5. If no tool-fluent worker is available, the pool **fail-opens** (rc59 mode): it still serves but with possible degradation; observable via the `tools_capable` DB column.