ELLM

How to Configure a Reasoning Model and Native Tool Calling

Learn to configure reasoning models and native tool calling to optimise your gateway's performance. This guide is ideal for improving planning stages and ensuring compatibility with different model types.

AUTHOR
Paul Flanders

PUBLISHED
10 Sep 2026

DOCUMENT
InfoHub — Knowledge, news, and updates from EssingtonITS

How to Configure a Reasoning Model and Native Tool Calling

Paul Flanders · 10 Sep 2026

Estimated time: 3 minutes
Difficulty: Intermediate

Why you'd use this

Many gateways serve more than one model: a fast code model for everyday edits and a larger model that is better at planning, architecture and review. eLLM Code can send the reasoning-heavy turns to that larger model automatically. Separately, some routed models expect OpenAI-style function calling rather than the plain-text action format; native tool calling lets both kinds of model work with the same pipeline. Use this guide when replies from planning stages are weak, or when your gateway's model ignores the text action format.

Before you start

Permissions required:

- None beyond a working endpoint.

You'll need:

- The identifier of the stronger model on your gateway, if you have one.
- To know whether your gateway routes by the `intent` header. If it does, you can leave the reasoning model blank and let the router decide.

Steps

1. Open the chat panel and click ⚙.
2. Scroll to the advanced fields. In **Reasoning model**, enter the stronger model's identifier, for example `qwen-72b`. Leave it blank to rely on router routing.
3. Tick **Native function-calling** to advertise the assistant's actions as tools as well as text. This is on by default and is safe with any routed model.
4. Optionally tick **Compact prompt** if you are using a small or quantised local model that struggles with a long system prompt.
5. Click **Save**.
6. Run **eLLM: Plan & Build (Plan Mode)** and describe a small feature. Planning turns now go to the reasoning model.

What you should see

Turn on **Debug logging** in the same panel and run **eLLM: Show Debug Trace**. Each request line records the model and intent used, so you can see planning and review turns naming the reasoning model while code edits name the worker model.

Troubleshooting

- **The endpoint rejects tool-role messages:** the extension detects this and falls back to the text protocol for the rest of the session. An activity line reads "Tool fallback". No action needed.
- **The endpoint rejects unknown body fields:** turn off `ellmCode.sendIntentInBody` in VS Code settings. The intent header is still sent.
- **Planning replies truncate:** the reasoning model may have a small context window. Run **eLLM: Show Context Window** to see what each model reports.

Related guides

- How to Check and Manage the Context Window
- How to Tune Generation Settings for Your Model
- How to Plan and Build a Feature with Plan Mode