ELLM

How to Check and Manage the Context Window

Learn how to manage and check the context window for different models, ensuring your conversation fits within token limits. This guide covers reading context readouts, handling large conversations, and adjusting fallback budgets.

AUTHOR
Paul Flanders

PUBLISHED
11 Sep 2026

DOCUMENT
InfoHub — Knowledge, news, and updates from EssingtonITS

How to Check and Manage the Context Window

Paul Flanders · 11 Sep 2026

Estimated time: 3 minutes
Difficulty: Intermediate

Why you'd use this

Every model has a limit on how much text it can consider at once, and a gateway may route different requests to models with different limits. eLLM Code sizes the conversation to the routed model automatically when the gateway advertises a window, and falls back to a configured budget when it does not. This guide shows how to see which limit is in effect, how to read the context readout under the composer, and what to do when a conversation grows too large.

Before you start

Permissions required:

- None.

You'll need:

- A connected chat.

Steps

1. Look at the status row under the composer after any reply. It shows the approximate size of the conversation being sent and, when known, the routed model's window.

2. Press **/** and choose **Show context window (token limits)**, or run **eLLM: Show Context Window**.
3. Read the report: the window the gateway's model list advertises, the windows reported per intent by the pools that actually served recent requests, the effective minimum used for budgeting, and the fallback budget used when nothing is advertised.
4. If the effective window is small, expect the prompt to be trimmed: a "Prompt trimmed" activity line names the optional parts that were dropped, such as the skills catalogue or reuse hints, so that the request fits.
5. To start over with an empty conversation, click in the header or run **eLLM: Clear Context**. Project memory is unaffected.
6. To change the fallback budget, set `ellmCode.maxContextTokens` (history budget) and `ellmCode.outputReserveTokens` (headroom kept for the reply) in VS Code settings.

What you should see

Old conversation turns are trimmed in whole groups, so a tool call is never separated from its result. A window learned from the gateway is remembered per model, so a new session starts with the right budget rather than the conservative fallback. If the gateway still rejects a request as too long, the history is compacted once and the request retried.

Troubleshooting

- **Replies stop mid-sentence:** the routed model's output cap is small. Raise `ellmCode.outputReserveTokens` only if the window allows; otherwise ask for the work in smaller pieces.
- **Long requests are saved to a file:** a request larger than the window is written under `.project-ai/` and the assistant reads it in parts. This is expected for very large pasted specifications.

Related guides

- How to Configure a Reasoning Model and Native Tool Calling
- How to Tune Generation Settings for Your Model