

chatui: a L^AT_EX package for typesetting AI chat transcripts

Nilesh Verma
me@nileshverma.com

v1.0.0, 2026/08/21

Abstract

`chatui` typesets AI chat transcripts: user/assistant messages, reasoning (“thinking”) traces, tool-call traces, and file/image attachments, styled as a **terminal**, **web**, or **desktop** chat UI, in light or dark theme. It is meant for papers, reports, and documentation that need to show a worked conversation with an AI assistant, including the reasoning and tool calls behind a response, without hand-rolling `tcolorbox` code for every figure.

Contents

1	Installation	1
2	Quick start	2
3	Package options	2
4	Environments and commands	3
4.1	<code>chatconversation</code>	3
4.2	<code>chatuser</code> / <code>chatassistant</code>	3
4.3	<code>chatthinking</code>	3
4.4	<code>chattool</code>	4
4.5	Attachments: <code>chatimage</code> and <code>chatfile</code>	4
5	Customizing colors	5
6	Customizing fonts	5
7	Long conversations and page breaks	5
8	Known limitations	6
9	Version history	6
10	License	6

1 Installation

Drop `chatui.sty` next to your `.tex` file, or into your local `texmf` tree. Requires `tcolorbox` ([most] libraries), `tikz`, `etoolbox`, `xparse`, `kvoptions`, `xcolor`, `pifont`, and `graphicx`, all standard and present on any full $\text{\TeX}$ Live or $\text{\MiKTeX}$ install.

2 Quick start

```
\usepackage[style=web]{chatui} % style = terminal | web | desktop

\begin{chatconversation}[chat.example.com]

\begin{chatuser}[John Doe]
How do I show tool calls in a LaTeX paper?
\end{chatuser}

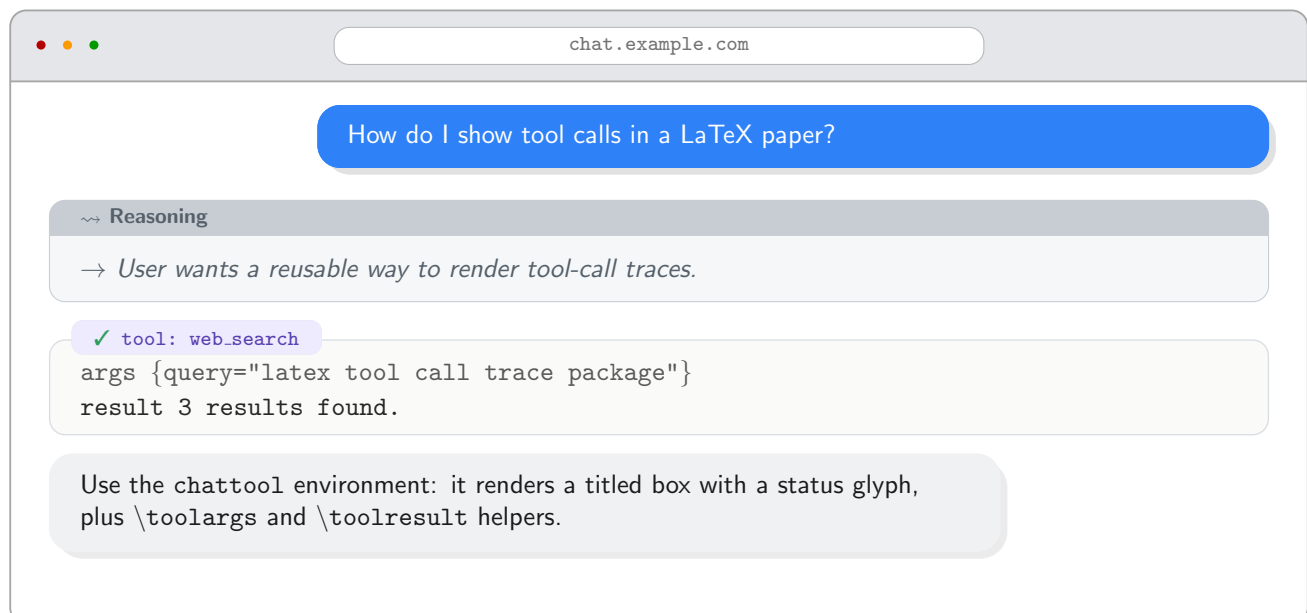
\begin{chatthinking}
\tstep{User wants a reusable way to render tool-call traces.}
\end{chatthinking}

\begin{chattool}{web_search}
\toolargs{query="latex tool call trace package"}
\toolresult{3 results found.}
\end{chattool}

\begin{chatassistant}
Use the chattool environment: it renders a titled box with a
status glyph, plus \toolargs and \toolresult helpers.
\end{chatassistant}

\end{chatconversation}
```

This renders as:



3 Package options

`style terminal` (monospace, shell-prompt look), `web` (rounded chat bubbles + browser chrome, *default*), or `desktop` (macOS-style window with traffic-light dots).

`theme auto` (*default*: `terminal` is dark, `web/desktop` are light), `light`, or `dark`. Dark theme uses the same palette across all three styles, so switching `style` under `theme=dark` doesn't change the color scheme, only the chrome.

4 Environments and commands

4.1 chatconversation

The outer window chrome (terminal title bar, browser bar, or macOS window). Takes one optional argument, the title text shown in the bar:

```
\begin{chatconversation}[chat.example.com]
...
\end{chatconversation}
```

It is **breakable**: a long conversation flows across pages, and the chrome is drawn once, on the first page only. See Section 7.

4.2 chatuser / chatassistant

Message bubbles. Both take one optional argument, the display name shown above the bubble (styles `web` and `desktop`) or as the shell prompt (style `terminal`):

```
\begin{chatuser}[John Doe]
Where does the config file live?
\end{chatuser}

\begin{chatassistant}[Assistant]
It's at ~/.config/app/settings.toml.
\end{chatassistant}
```

4.3 chatthinking

A reasoning/thinking trace box. Takes two optional arguments: the title (default **Reasoning**) and the mode:

`steps` (*default*): arrow-prefixed bullets, one per `\tstep{...}`.

`paragraph` : plain upright prose, for longer chains of thought that read better as continuous text than as bullets.

```

\begin{chatthinking}
\tstep{Need to grep the repo for "auth" and "middleware".}
\tstep{Two candidate files found; the second is a test fixture.}
\end{chatthinking}

\begin{chatthinking}[Reasoning][paragraph]
This reads as ordinary prose instead of stepped bullets.
\end{chatthinking}

```

↪ Reasoning

→ *Need to grep the repo for "auth" and "middleware".*
→ *Two candidate files found; the second is a test fixture.*

↪ Reasoning

This reads as ordinary prose instead of stepped bullets, which suits a longer chain of thought better than a list of short arrow-prefixed lines.

4.4 chattool

A tool-call trace box. Takes two optional arguments (status, kind) and one mandatory argument (the tool name):

`status` *ok* (*default*, ✓) | *error* (×) | *running* (⋯).

`kind` *api* (*default*, “tool: name” chip) | *cli* (shell-style “\$ name”) | *code* (function-call style “</> name”).

Inside, use `\toolargs{...}`, `\toolresult{...}`, or `\tooljson{...}` for the payload. JSON needs manual `\\` line breaks and `\quad` for indentation, since $\text{\LaTeX}$ does not preserve literal whitespace.

```

\begin{chattool}[ok][cli]{find . -name "*.log" | head -1}
\toolresult{./data/latest.log}
\end{chattool}

\begin{chattool}[ok][code]{get_weather(city="Austin")}
\toolargs{city="Austin"}
\tooljson{\{\}
\quad "temp_c": 34.2\}
\}}
\end{chattool}

```

✓ \$ find . -name "*.log" | head -1

result ./data/latest.log

✓ </> get_weather(city="Austin")

args {city="Austin"}

```

{
  "temp_c": 34.2
}

```

4.5 Attachments: chatimage and chatfile

Use inside `chatuser`/`chatassistant` (or standalone) to show an uploaded image or file. `chatimage` clips the image to the bubble's own corner radius; `chatfile` renders a small bordered chip with a document icon, filename, and size:

```
\begin{chatassistant}
Here is the generated report:
\chatfile{crash-report.pdf}{212 KB}
\end{chatassistant}
```

Here is the generated report:



crash-report.pdf (212 KB)

5 Customizing colors

Every color is a plain `xcolor` name. Redefine any of them after loading the package, with a named color, a mix, or a raw hex code:

```
\usepackage[style=web]{chatui}
\colorlet{chatuiUserBg}{teal}           % named xcolor
\colorlet{chatuiAssistantBg}{gray!10}    % mixed
\definecolor{chatuiToolAccent}{HTML}{7C5CFC} % raw hex
```

Available color names:

- `chatuiUserBg`, `chatuiUserFg`
- `chatuiAssistantBg`, `chatuiAssistantFg`, `chatuiAssistantBorder`
- `chatuiThinkBg`, `chatuiThinkBorder`, `chatuiThinkFg`
- `chatuiToolBg`, `chatuiToolBorder`, `chatuiToolAccent`
- `chatuiOk`, `chatuiErr`, `chatuiWarn`: status glyphs and chattool accents
- `chatuiAppBg`, `chatuiAppText`, `chatuiAppChrome`, `chatuiAppChromeText`: web/desktop window chrome
- `chatuiAppPillBg`, `chatuiAppPillBorder`, `chatuiAppPillText`: web browser-bar URL pill
- the `chatuiTerm*` family: terminal-style chrome and boxes

6 Customizing fonts

Two hooks control every font family in the package. Redefine them after `\usepackage` (and after loading any font package you want to use):

```
\usepackage{fourier}
\renewcommand{\chatuiFontSans}{\rmfamily} % web/desktop bubbles, prose
\renewcommand{\chatuiFontMono}{\ttfamily} % terminal + tool/code text
```

Only sizing is fixed internally, tuned to each box’s proportions; the family is entirely yours to swap.

7 Long conversations and page breaks

The outer `chatconversation` window is breakable and splits cleanly across pages: the chrome (title bar, dots) only draws once, on the first page, and the border/background continue seamlessly on later pages. Individual messages, reasoning boxes, and tool calls are each a single atomic unit (they don’t break mid-box), so a page break always falls *between* two messages, never through the middle of one, matching how a real chat UI never splits a single message bubble. A single message longer than a full page is an unsupported edge case.

8 Known limitations

- A single message, reasoning box, or tool call that is itself taller than one full page will not render correctly (see Section 7).
- `chatimage` loads the whole image once via `\includegraphics`; very large source images will bloat the PDF as with any L^AT_EX document.
- No syntax highlighting for code inside tool results: payloads are plain monospace text.

9 Version history

v1.0.0 (2026/08/21) First public release: three styles (terminal, web, desktop), light/dark/auto themes, reasoning traces (steps or paragraph), tool-call traces (api, cli, or code kinds, JSON payloads), image and file attachments, page-break-safe long conversations, customizable colors and fonts.

10 License

Released under the L^AT_EX Project Public License (LPPL), version 1.3c or later. See the `LICENSE` file.