Terminal – Plugin Specification Document

Overview

Terminal is a Premiere Pro and After Effects plugin designed to give editors a command-line style workflow directly inside their editing software. Instead of navigating through menus or manually adjusting settings, users can summon a search bar with a hotkey and type natural-language commands to instantly apply effects, adjustments, and transformations to their timeline.

This makes complex editing tasks faster, more intuitive, and highly customizable.

Core Functionality

1. Search Bar Activation

- Trigger Hotkey: Command + 1 (Mac) / Ctrl + 1 (Windows).
- When triggered, a search bar overlay appears on screen (floating UI element).
- The search bar reads and processes user keystrokes in real time.

---

2. Command Interpretation

The plugin parses text input and maps it to available actions within Premiere Pro or After Effects:

A. Applying Effects

- Users type in the effect name.
- Terminal searches installed effects and applies the requested one to the selected clip(s).
- Example:

- Typing Gaussian Blur instantly applies the Gaussian Blur effect.

B. Applying Effects with Parameters

- Users can include parameter values directly in the command.
- Example:
  - Typing Exposure 4 applies Lumetri Color with Exposure set to +4.
  - Typing Blur 50 applies Gaussian Blur with Blurriness set to 50.

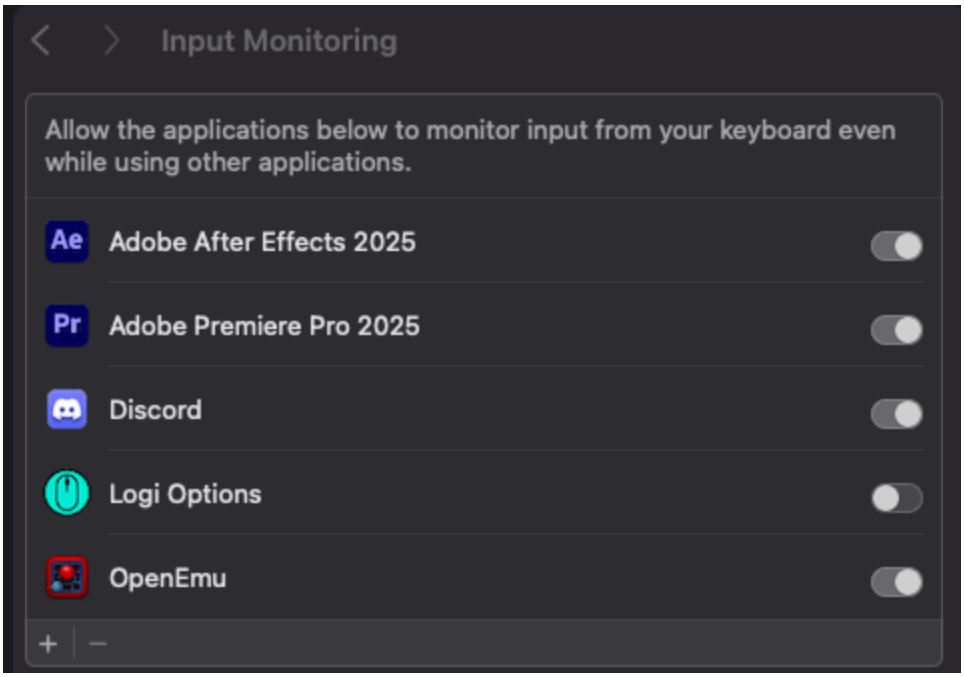
C. Adjusting Clip Properties

- Users can adjust transform and property values without opening effect controls.
- Example:
  - Typing Scale 150 sets the clip's scale to 150%.
  - Typing Opacity 75 sets opacity to 75%.

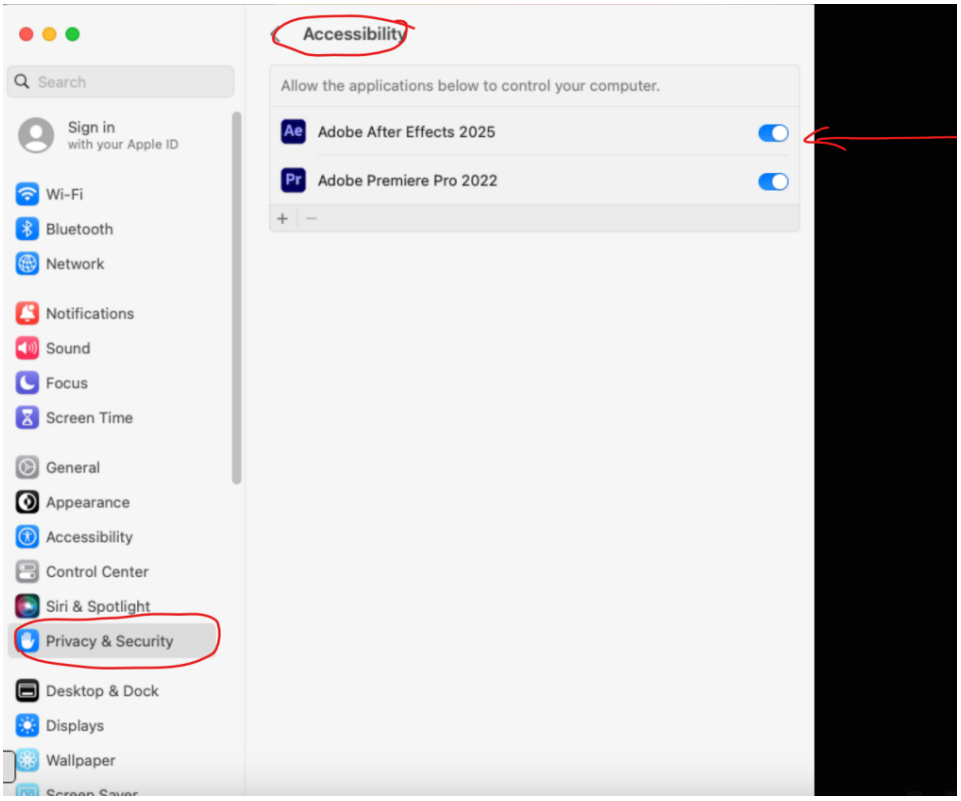
Technical Requirements

Keystroke Access

- Mac:
  - Requires special backend setup since macOS restricts apps from monitoring keystrokes without user permission.
  - The user must enable input monitoring in System Preferences > Security & Privacy > Privacy > Input Monitoring.



- Also the user must enable Accessibility System Preferences > Security & Privacy > Accessibility.



    - Terminal must clearly notify users about these required permissions during installation/setup.

- Windows:
    - Keystroke access is more straightforward and typically requires fewer user permissions.
    - Should be implemented using native Windows APIs for keyboard event capture.

Backend Logic

1. Command Parsing
    - Parse user text input into structured commands (e.g., [action] + [value]).
    - Match against installed effects and available clip properties.
2. Effect Application Engine
    - For effects:
        - Search internal Adobe effect database.
        - Apply effect to target layer/clip.
        - If parameters are included in the command, set them automatically.
    - For transforms:
        - Directly modify clip's transform properties (scale, position, rotation, opacity, etc.).
3. Error Handling
    - If a command doesn't match anything, display a "No matches found" message.
    - Fuzzy matching should be supported (e.g., "blur" finds "Gaussian Blur").

User Setup & Onboarding

Installation

- Plugin installed as standard Adobe extension (Premiere Pro + After Effects).
- Windows and Mac builds required.

First Launch

- Prompt users with setup steps:
    - Mac Users: Guide them to enable Input Monitoring and Accessibility in System Preferences > Security & Privacy
    - Windows Users: Simple confirmation message (no extra steps in most cases).

Usage Tutorial

- On first launch, provide a quick overlay tutorial:
    1. Press Command/Ctrl + 1 to open search bar.
    2. Type effect names or property adjustments.
    3. Hit Enter to apply instantly.

Example User Scenarios

1. Quick Transform Change
    - User types Scale 120 → Selected clip instantly scales to 120%.
2. Instant Effect with Custom Settings
    - User types Exposure -2 → Lumetri Color applied, Exposure set to -2.

3. Fast Effect Search
    - User types CRT → Searches all installed effects, applies matching “CRT Effect” if available.

Development Notes

- Ensure cross-platform consistency (Premiere Pro + After Effects).
- Prioritize low-latency command execution (instant application to keep workflow smooth).
- Implement logging/debugging for failed or unrecognized commands.
- Consider future expansion:
    - Batch commands (e.g., Scale 120 + Exposure 2).
    - Preset saving (store custom commands for repeated use).
    - AI-assisted fuzzy matching (spell correction, synonyms).