Playwright MCP + Cline

Windows Integration and Verification — Technical Overview

Updated 18 October 2026 • Verified local rehearsal; client deployment pending

1. Latest verified position

The independently possible local preparation is complete. Two fresh browser workflows ran through the Windows Cline graphical interface, with all five browser assertions passing in each run. The rehearsal also exercised actual editor restarts, disabling the Playwright server, exact restoration of the original working configuration, re-enabling the server, and repeating the browser workflow after the final configuration change.

The final evidence image shows two test tasks: “Practice browser test” is completed, “Capture test evidence” remains active, and the counter reads “1 item left.” The screenshot was checked both as a real file in the intended workspace and by visually inspecting its contents. It is 36,804 bytes and measures 929 × 917 pixels.

The final change adds an explicit working directory for the local MCP process. Without that setting, a relative screenshot path inherited the editor’s application directory. The corrected configuration was tested after another real editor close and reopen. The last run had no rejected or failed browser tool calls. A separate public-site favicon HTTP 404 remains recorded; it did not invalidate the five task assertions.

Local completion does not establish client deployment or contractual acceptance. The live walkthrough, the client-operated installation, account-specific provider and meeting checks, independent client repetition, funded milestones, and acceptance remain pending. This overview describes verified rehearsal behavior and an installation procedure to adapt after inspection of the target computer.

The earlier guides, original recordings, configuration backups, and test evidence remain retained separately. This document adds a readable technical overview; it does not replace the private recovery archive or silently turn historical rehearsal results into client results.

Reading order: sections 2–3 explain the purpose and architecture; sections 4–8 contain the inspection, setup, browser, restart, and recovery procedures; sections 9–10 explain evidence and troubleshooting; sections 11–12 cover handover and future use. Begin with section 4 when guiding a new installation. Use section 7 when repeating a working setup without changing it.

2. Purpose, scope, and project history

The integration extends an existing Windows Cline installation with Microsoft’s official Playwright MCP server. The practical objective is to let Cline request browser actions, show those requests for human review, interact with an agreed public test website, and save verifiable evidence. A successful connection indicator is only a setup check. Acceptance requires the complete browser workflow through Cline’s graphical interface.

The design preserves the existing editor, Cline installation, configured provider, unrelated MCP servers, and browser profiles. Work is limited to the components and configuration needed for this integration. No private application source, production server, customer database, account login, hosted MCP endpoint, tunnel, or remote-control tool is needed for the test run.

The preparation progressed from an earlier recorded browser walkthrough to a more complete repeatability rehearsal. The earlier video retained selector-error recovery rather than presenting an uninterrupted success. Later runs added explicit restart checks, a practical rollback test, file-location verification, and a full browser retest of the final configuration. That progression explains why an older video and the latest written evidence describe different levels of verification.

On 17 October 2026, two fresh Windows Cline GUI workflows passed their browser assertions. During that preparation, the screenshot working-directory issue was identified and corrected. On 18 October, this overview was prepared from the retained project record, with the latest verified state placed first. No later client installation or acceptance is established by those dates.

There are two contractual delivery stages. The first is a live walkthrough on the implementer’s Windows computer. The second is a guided installation and verification on the client’s Windows computer, with written instructions and independent repetition. A recording supports the first stage but does not replace its live walkthrough. A successful local rehearsal supports preparation for the second stage but does not satisfy its client-computer requirements.

The implementation estimate is four to six total active working hours for both stages, subject to the inspected environment and required approvals. This is an estimate rather than a two-hour or same-day completion guarantee. Session timing and any continuation must be mutually agreed through the project’s established communication channel.

3. Architecture, configuration, and information flow

The execution chain is Windows VS Code → Cline → a local Node process running Playwright MCP over standard input/output → an isolated installed Chrome browser → the agreed public test website. Cline discovers the server's browser tools, requests a tool call, and presents its arguments for review. Approved browser operations execute on the same Windows computer.

The MCP server is a local process. It does not require an externally hosted project server or a public listening endpoint. The rehearsed configuration pins the official package version instead of allowing an unnoticed package change between runs. The browser uses isolated mode to avoid deliberately attaching to the user's personal Chrome profile or existing signed-in tabs.

Local browser execution and AI-provider processing are different parts of the system. If Cline's selected AI provider is cloud-based, prompts, tool descriptions, approved browser results, and request metadata may leave the computer. An isolated browser does not change that provider data flow. Review the actual account's training, retention, and related settings before the client test, explain the intended test payload, and obtain the required explicit approval. Do not promise zero retention without account-specific evidence.

Keep configuration in one inspected MCP entry. Its command identifies the Windows Node executable; arguments identify the existing npx script, pinned official package, browser, and isolation option. The working directory identifies the approved clean test workspace. The enabled state and empty server auto-approval list are explicit. Never paste another computer's absolute executable or workspace paths into a client setup.

The process working directory matters because tools may return relative artifact paths. The final rehearsal sets the supported cwd field to the clean workspace and verifies the resulting screenshot on disk. A model statement about where it saved a file is insufficient evidence. The actual returned path and file contents determine the result.

Credentials stay in the existing approved provider configuration. They are not part of the MCP example, browser prompt, screenshot, shared document, or handover archive. Existing telemetry and provider settings are preserved during this integration; preserving a setting is not proof that its privacy behavior satisfies a particular account requirement. Record what was actually inspected and leave unresolved settings clearly marked.

4. Verified environment and read-only inspection

The rehearsal used Windows 11 Pro, version 10.0.26200; VS Code 1.142.0; Cline 4.1.25; Windows Node 24.20.0; npm/npx 12.0.2; Chrome 155.0.8059.80; and the official @playwright/mcp package pinned at 0.0.83. These are recorded observations, not minimum versions or an instruction to upgrade a different computer. The official package documentation requires Node 18 or newer; compatibility still needs to be checked for the approved package and installed environment.

1. Start with a clean Windows VS Code window and the existing Cline extension. Close confidential projects and unrelated documents before sharing the application window. The client retains keyboard and mouse control. Do not install, reset, replace, or update Cline during the opening inspection.
2. Record the Windows, VS Code, Cline, Node, npm/npx, and browser versions using the installed interfaces or explained read-only commands. Record whether each component already exists and works. Do not treat a terminal running in a different operating environment as proof of the Windows extension’s runtime.
3. Open Cline’s MCP management interface. In the rehearsed version, the path was Customize, then MCP, then Installed. Labels may vary. Inspect the existing servers, enabled states, tools, and actual configuration location through the installed Edit Configuration control. Hide sensitive values rather than copying complete settings files into messages.
4. Inspect the configured provider and approval controls without exposing credentials. Verify the applicable account policy separately. Confirm whether browser tools already exist; adding a second duplicate server is unnecessary. Identify the clean workspace that will receive test artifacts and keep personal browser profiles outside the test.
5. Explain the findings before proposing changes: what already works, what is missing, the exact minimum addition, why it is needed, how it will be verified, and how it can be reversed. Obtain the required confirmation before moving to the backup and change phase.

Expected result: an accurate inventory and a narrowly scoped change plan, with the existing installation preserved. Read-only inspection is an actual first phase, not a label applied after downloads or settings changes have already started. If inspection reveals an incompatible or unexpected setup, revise the plan before executing a command.

5. Backup and the minimum MCP configuration

Before editing, explain the actual MCP configuration file and whether it contains sensitive entries. Agree a backup location on the same computer, copy the original there, and verify that the copy exists and matches the source. Matching SHA256 hashes provide stronger evidence than a filename alone. Keep client backups local; do not request that the client upload their complete configuration.

Preserve the existing mcpServers object and every unrelated entry. The following structural example shows the rehearsed approach. Angle-bracket values are deliberately incomplete: replace them only with Windows paths discovered and approved on that computer. It is not a ready-to-paste configuration.

```
{
  "mcpServers": {
    "playwright": {
      "type": "stdio",
      "command": "<verified Windows Node executable>",
      "args": [
        "<verified existing npx-cli.js>", "-y",
        "@playwright/mcp@0.0.83",
        "--browser", "chrome", "--isolated"
      ],
      "disabled": false,
      "autoApprove": [],
      "cwd": "<approved clean Windows test workspace>"
    }
  }
}
```

The pinned version records what passed the rehearsal. Review compatibility and package provenance before using it on another installation; do not silently replace it with latest. The npx command may download a package on first use even when Node already exists. Explain that network access and the official source before the client approves execution.

Show the exact non-sensitive diff before saving. Check JSON syntax and verify that unrelated entries remain unchanged. Return to MCP management, confirm the entry is enabled and connected, and inspect the tools. The rehearsal exposed 25 tools; a different approved version may expose a different set, so record the actual result.

The full backup restoration procedure must be understood before the change. Preserve a separate copy of the current configuration before restoring the original. Never overwrite settings that have changed unexpectedly since the comparison; inspect that drift first. Backup, disable/re-enable, and full restoration are distinct operations and should be explained separately.

6. Controlled installation and individual approvals

The client performs and approves each action during the guided installation. Screen sharing remains view-only. Explain one numbered step, its purpose, the expected result, and its reversal; wait for confirmation before advancing. Delegated approval during the private rehearsal does not become permission to approve actions on the client's behalf during the live session.

1. Reuse compatible existing components. Propose a download only when inspection shows a missing requirement. Verify the official publisher and package identity, explain the exact command and network destination, identify affected files, and obtain approval. Administrator access needs a separate concrete justification; it is not a routine prerequisite for this local configuration.
2. Make the verified local backup, then add only the agreed Playwright entry. Do not replace the whole configuration with the example in this document. Use executable paths established by inspection and the approved clean workspace as cwd. Reopen the management view and confirm connection and available browser tools.
3. Open Cline's Auto-approve controls. In the rehearsed interface, Read files, Edit files, Execute commands, Fetch web content, and Use MCP servers were all unchecked. The summary showed Auto-approve: None. Check any version-specific broader permission mode as well; the empty MCP autoApprove list alone does not prove all application-wide approvals are disabled.
4. Before every browser request, inspect the tool name, destination, target, text, screenshot path, and expected effects. Approve only the current explained operation. Reject an unexpected file read, unrelated website, unsupported selector, or broader action and ask for a corrected request. Repeated approval prompts are expected under this operating model.
5. Stop on a client instruction to stop, private-data exposure, unexpected administrator requests, unrelated changes, billing errors, unsafe rollback, or scope expansion. Explain the observed condition and propose the smallest justified next step. Do not solve an approval inconvenience by enabling automatic approvals.

Expected result: the official local server is available, the existing installation remains intact, and every operation is individually reviewable. This stage establishes controlled readiness; it does not establish browser test success. Continue to the actual Cline GUI workflow only after the public website, test data, provider data flow, and test itself have received the required approval.

7. Complete browser workflow through Cline

Use the agreed public TodoMVC test page at https://demo.playwright.dev/todomvc/. Enter only the test labels below. Do not use real accounts, customer information, messages, purchases, or private forms. Start the task in Cline's graphical interface and review every tool request; a separate terminal browser script does not substitute for this integration test.

1. Start a fresh test task and ask Cline to use only the local Playwright MCP browser tools for the agreed page. Specify that automatic approvals remain off and that it must wait for each request to be approved. Do not authorize workspace file reading, command execution, or unrelated navigation as part of this browser test.
2. Approve navigation to the public page and inspect its current snapshot. Confirm the expected TodoMVC interface is visible. If an earlier task list is present, agree how to obtain a clean test state without affecting unrelated browser data.
3. Add "Practice browser test" and "Capture test evidence." Review the text and target in each request. Mark only "Practice browser test" complete. Use the latest snapshot references; do not assume a reference from an earlier run still identifies the same element.
4. Check five assertions: the first exact label exists; the second exact label exists; the first item is completed; the second remains active; and the counter reads "1 item left." Record each result separately so the summary cannot conceal a missing state check.
5. Approve screenshot capture into the clean workspace. Read the returned path, locate the actual non-empty image, and open it. Confirm that the screenshot supports the asserted state. A successful tool response and a model's description do not replace checking the saved file.
6. Record rejected proposals, executed errors, recoveries, and relevant console messages. A recovered run can be useful evidence, but it must not be described as uninterrupted. Close only test-created browser tabs or windows when the agreed workflow is finished.

Reusable task wording: "Using local Playwright MCP, open the agreed TodoMVC test, add the two test labels, complete only the first, verify all five stated assertions, and save a screenshot in this approved workspace. Wait for every tool approval. Report actual errors and the actual screenshot path. Do not read private files or visit other sites." The installed tools and current page snapshot remain authoritative.

8. Restart, disable/re-enable, and full recovery

Restart verification establishes that configuration survives an actual process lifecycle. Finish or stop the current Cline task safely, then close and reopen the relevant editor window as agreed. Confirm the same Playwright entry reconnects, its tools appear, automatic approvals remain disabled, and unrelated settings are intact. Repeat the full section 7 workflow, including both labels, completion state, counter, and screenshot.

Do not confuse MCP configuration persistence with website task persistence. An isolated browser can start with an empty list after restart. That is compatible with a working MCP configuration; recreate the two test items and verify the behavior again. Merely refreshing the page or reopening a management panel is not equivalent to closing and reopening the editor process.

For the safe rollback test, first verify the original backup remains available. Explain the proposed disable operation and obtain approval. Disable only the Playwright entry through the supported interface or configuration and confirm its disabled state. Preserve the entry and leave shared Node, Chrome, Cline, and unrelated servers installed.

Explain re-enabling the same entry and obtain approval. Confirm reconnection, then run a short Cline-initiated navigation, page-verification, and screenshot check. Record both the unavailable state while disabled and the working state after re-enabling. If the operation cannot be shown safely, stop and agree an alternative rather than removing shared dependencies.

For full configuration restoration, preserve the current file as a separate recovery copy. Compare the expected current state against the actual file before writing. If unexpected edits exist, inspect them first. Restore the verified original backup, compare its hash with the restored file, and inspect Cline's resulting MCP state. The original configuration may already contain a working Playwright entry; restoration means returning to that actual baseline, not assuming it removes Playwright.

The rehearsal verified disabling, exact restoration of the original working backup, and re-enabling. It then added only the explicit working directory, performed another actual editor reopen, and repeated the complete browser test successfully. The final active bytes matched the reviewed proposal. These observations establish the local sequence, while the client must still perform the corresponding approved sequence on their own machine.

Expected result: configuration persistence, reversible control of the local integration, and a tested recovery path, supported by actual state checks rather than a written rollback promise alone.

9. Results, evidence, and verification limits

The private local acceptance record reports 12 of 12 independently applicable preparation criteria met. Two fresh Windows Cline GUI browser runs each passed all five assertions. Two actual editor close/reopen cycles were exercised. The final run followed the working-directory correction and saved its screenshot in the intended workspace. These are local rehearsal results; client-machine and contractual acceptance criteria remain separate.

The first new run included one rejected invalid target proposal before execution. A corrected exact snapshot reference was then approved. There were no executed browser tool failures in that run. The final run had no rejected proposals or executed browser tool failures. The public page produced a favicon HTTP 404, which is retained in the record rather than omitted from a claim of a completely clean console.

Configuration checks found the final MCP change was limited to the explicit working directory. Provider and global settings hashes, provider/model fields, and approval fields matched the preserved state. The backup restoration produced an exact hash match. The screenshot was verified on disk and visually. These checks are evidence for the recorded scope, not proof about every setting or possible behavior of the application.

Three local PowerShell helper scripts passed native parser checks with zero syntax errors. JSON and configuration consistency checks passed. Selected authored scripts, guides, handover material, and evidence passed a secret scan. A fresh local model review of two helpers returned PASS and its input hashes matched the final source. That review was static; the reviewer did not independently operate the graphical browser workflow.

PSScriptAnalyzer was unavailable. There was no applicable application build or type-check suite for this configuration-and-documentation task, and Python-specific Bandit analysis did not apply to the authored PowerShell helpers. Missing or inapplicable checks are stated explicitly. These results do not justify a blanket production-readiness or security guarantee.

The older walkthrough video is retained as historical evidence: an edited 51-second H.264 clip, 1440 × 1080, 765 frames, with no audio. The original recording remains separate. The edited file was fully decoded and visually reviewed, including its recovery disclosure. It is supplementary to the required live walkthrough.

The shared handover contains the numbered Windows guide, test prompt, local verification summary, and reviewed public-site screenshot. Full transcripts, private configuration, backups, account details, and correspondence remain outside this document and outside the shareable package.

10. Troubleshooting and lessons from rehearsal

If Cline does not appear, inspect extension activation and the installed version's view command. Do not immediately reinstall or reset a working extension. If Playwright is absent, inspect the actual saved configuration location, JSON nesting, enabled state, and server-management view. A correctly written example in a separate document does not prove that Cline loaded it.

If the Node or npx process fails, inspect the exact Windows executable path and the actual error. The extension host may have a different environment from a shell. The rehearsal used verified absolute executable paths to remove that ambiguity. Keep the paths in configuration; do not scatter machine-specific values through instructions or helper logic.

If browser interaction fails, inspect the latest page snapshot. In the rehearsed package, a target can be an exact current reference such as e8 or an appropriate unique selector. Descriptive text such as "textbox [ref=e8]" is not the same reference. Reject an invalid proposal before execution where possible, approve the explained correction, and record any actual failure honestly.

If a screenshot is missing from the workspace, inspect its returned path and the server's process working directory. The rehearsal found evidence in the editor application directory because cwd was absent. The fix was an explicit approved workspace cwd, followed by a real restart and full GUI retest. Check the file itself before accepting the model's explanation of its location.

If the editor is not the verified foreground window, stop before typing. The local rehearsal helper refused input when it could not verify focus. That refusal is a useful safeguard, not an error to bypass by sending keys to whichever window is active. The live client controls input directly; local desktop helpers are not a substitute for that arrangement.

If a provider returns a billing, quota, or rate error, stop and explain it. Do not switch providers, raise spending caps, purchase credits, or retry indefinitely. If restoration fails, retain both backup and current state, inspect the exact error and diff, and propose one justified correction with its own verification and reversal.

The general procedure is consistent: state the observed symptom, separate a suspected cause from a verified cause, identify one minimum correction, explain affected files and expected results, obtain the required approval, and test the corrected behavior. Preserve errors in the evidence and reconcile them with the final summary. A model-generated "success" paragraph is not a substitute for tool responses and visible state.

11. Live guidance and milestone handover

The live session begins only under the applicable funded-work and commencement conditions. Milestone 1 requires a live Windows Cline walkthrough and its acceptance. Milestone 2 begins only after the first stage is accepted, the second stage is funded, and the client expressly authorizes commencement. Accepting instructions or scheduling a call does not itself authorize installation or testing on the client's computer.

Before a meeting, agree the date, start time, time zone, scope, and expected duration through the established project channel. Confirm the actual meeting service and its recording, transcription, AI-summary, storage, and retention controls. Do not assume that a generic platform description proves the settings of a particular meeting room. Screen sharing remains view-only and should show the clean application window where possible.

During guided installation, the client performs every action and provides each required approval. The first activity on their computer is strictly read-only inspection. Explain numbered steps continuously: what to open, what to inspect, what the expected result means, and when to stop. Pause after each instruction instead of reading ahead while the client is still completing an earlier action.

Optional AI assistance during the session needs its own clear explanation and the client's comfort and required approval. Identify the tool, the proposed information flow, and the public/test data involved. Do not expose private files, credentials, or confidential screen content. Do not treat an implementer's existing paid API access as permission to send client information to that provider.

The first milestone's completion record should identify the live walkthrough, browser assertions, screenshot, errors and approved recovery, and any outstanding issue. The second should record the client's actual versions, approved commands and sources, non-sensitive configuration change, backup location known to the client, browser results, restart verification, rollback results, and independent repetition.

Keep milestone submissions, important approvals, communication, and payments in the agreed Upwork workflow. This overview is technical preparation and documentation; it does not submit a milestone, schedule a call, establish funding, or record client acceptance. Replace pending items only when the corresponding client activity is actually performed and evidenced.

The handover is complete only when the client can follow the written procedure independently and reproduce the agreed behavior. Retain the existing setup and narrow rollback options rather than making successful operation depend on the implementer's remote access or an external project server.

12. Independent use, maintenance, and references

For routine use, open the approved clean workspace, launch the existing Cline installation, confirm the local Playwright server is connected, and verify automatic approvals remain disabled. Start the section 7 test workflow, review each tool request, check the five assertions, and open the actual saved screenshot. If no configuration change is needed, do not repeat installation steps merely to use the integration.

When changing a package, browser, editor, or Node version, record the current working versions and back up the relevant configuration first. Explain and approve the minimum change. Repeat the full GUI browser test, actual restart test, artifact-location check, and appropriate rollback verification. A new version's successful connection does not establish that browser tools and evidence capture still behave as before.

Maintain one stable Google Doc for this overview. Keep its identifier beside the local technical-overview source and PDF. Add newer verified results before older context where appropriate, preserve useful historical material, and label superseded observations with their dates. Do not replace history with an unexplained success claim or rewrite a pending client activity as completed.

Before sharing documentation, review it for credentials, personal account details, private filesystem paths, confidential correspondence, and client-linked incident information. Share only the reviewed handover or overview. Keep recovery backups, raw transcripts, private memory, and account configuration local. A clean test screenshot can support the explanation without exposing unrelated desktop content.

Completion checklist: Cline opens; existing settings are preserved; local Playwright tools appear; every browser assertion passes; the screenshot exists and matches the result; the full workflow passes after restart; disable/re-enable and recovery are understood and tested as agreed; automatic approvals remain off; the client independently repeats the procedure; and written results distinguish actual observations from pending checks.

Official reference: Microsoft Playwright MCP repository — https://github.com/microsoft/playwright-mcp. Use it for the official package identity, supported options, and compatibility. The tested version remains the pinned rehearsal version; documentation can change after the recorded test.

Official reference: Cline MCP overview — https://docs.cline.bot/mcp/mcp-overview. Use the installed interface and current Cline documentation to locate server management and understand local MCP behavior. Approval guidance — https://docs.cline.bot/features/auto-approve. These references support product behavior; the retained rehearsal evidence supports this project's actual results.

This document records the verified local state as of 18 October 2026. Client installation, meeting-specific privacy checks, provider-account permission checks, and formal milestone acceptance require their own dated completion records.