

BROCCWIN: Browser-Hosted Execution of a Legacy Windows MMO Client Through a Synthetic Compatibility Runtime

Publication III — Academic technical white paper

HEAV3NS-BROCCWIN · Version 1.0.1 · 11 October 2026

Commissioned by MAJO, Executive Chair, HEAV3NS

Prepared by Dr. GPT, the commissioned AI researcher

Review status: repository-grounded technical report; no journal acceptance or external peer review is claimed.

1. Abstract

This report examines browser-hosted execution of a supplied legacy Windows MMO client through BROCCWIN, the compatibility environment integrated by HEAV3NS-COMBO. Original x86 application code executes in a WebAssembly interpreter with reconstructed Windows interfaces, browser graphics/input/audio adapters and a local transport relay to original native services. No conventional game-source port was used. In the final retained control pair, an original empty login interface is observed at 440.703 seconds with a fresh browser profile and 438.797 seconds with the reused profile. An audit reports 321 matching original-input entries before and after the runs, 211 checked served/observer identities and four reviewed image files. The cold login observation records 1,073 frame advances over 31.094 seconds; the warm observation records 1,080 over 31.109 seconds. Both reported error lists are empty. These results support a bounded execution milestone. They do not demonstrate complete gameplay compatibility, production readiness, 3× end-to-end acceleration or long-duration stability. The allocator candidate's cold comparison is interrupted by a screenshot-helper failure; later source-only cleanup and incompletely tested observer revisions remain outside the prepared build. The report documents architecture, identity scopes, measurements and limitations for independent review without reopening engineering.

Keywords: software preservation; Windows compatibility; x86 interpretation; WebAssembly; browser execution; experimental integrity; legacy MMO.

2. Introduction and research problem

A legacy executable depends on more than an instruction decoder. It expects an operating-system contract, library lifecycles, resource layout, graphics behavior, input semantics and service state. The research problem is whether a browser-hosted arrangement can supply enough of those conditions to execute an inherited Windows client while retaining its compiled application logic.

BROCCWIN is the internal project identity introduced on 9 October 2026. HEAV3NS-COMBO names the browser integration. Older Theseus paths and observations are retained. The scoped objective was a browser **client** port; native servers and SQL were explicitly excluded from porting on 10 October, while client interoperability remained required. [E10–E16](#)

This paper evaluates a narrower proposition than that full objective: the retained original client reaches its login interface through the recorded browser runtime and continues producing frame progress during a short observation. Historical gameplay evidence is considered separately. The reporting phase introduces no new compatibility test or optimization. Its contributions are an architectural account, an auditable synthesis of retained measurements and a delineation of the source/build/observation boundaries at freeze.

3. Background and related approaches

Execution preservation can pursue several arrangements. Full-system emulation retains an operating-system image and emulated devices; user-mode emulation supplies a smaller process environment. Bellard's QEMU paper describes both modes and a dynamic translation implementation. BROCWIN's documented application path instead uses a Wasm x86 interpreter and reconstructed API contracts; no QEMU-derived implementation or comparative benchmark is established. [Bellard, 2005](#)

Windows compatibility layers offer another reference point. Wine's own description emphasizes Windows API compatibility on other operating systems. Wine-Assembly is the pinned dependency here, with an x86/Wasm execution boundary; the project name must not be taken as evidence that it is the Wine project, that Wine internals were reused, or that Wine compatibility coverage transfers to it. [WineHQ, E16](#)

WebAssembly specifies a portable abstract machine, while its embedding supplies host facilities. The current specification is background, not a declaration that this runtime implements every present feature or conforms to WebAssembly 3.0. Microsoft's PE documentation supplies the original executable-format contracts, and Khronos WebGL 1.0.3 supplies a browser rendering reference. These standards explain boundaries; their existence cannot establish this implementation's correctness. [WebAssembly specification](#), [Microsoft PE format](#), [WebGL 1.0.3](#)

Rothenberg's preservation research agenda emphasizes retaining interpretive software together with explanatory metadata. BROCWIN's manifests, frozen candidates and failure records are pertinent to that requirement. The experiment supplies neither a general preservation proof nor a comparison against all migration/emulation alternatives. [Rothenberg, 1999](#)

4. System architecture

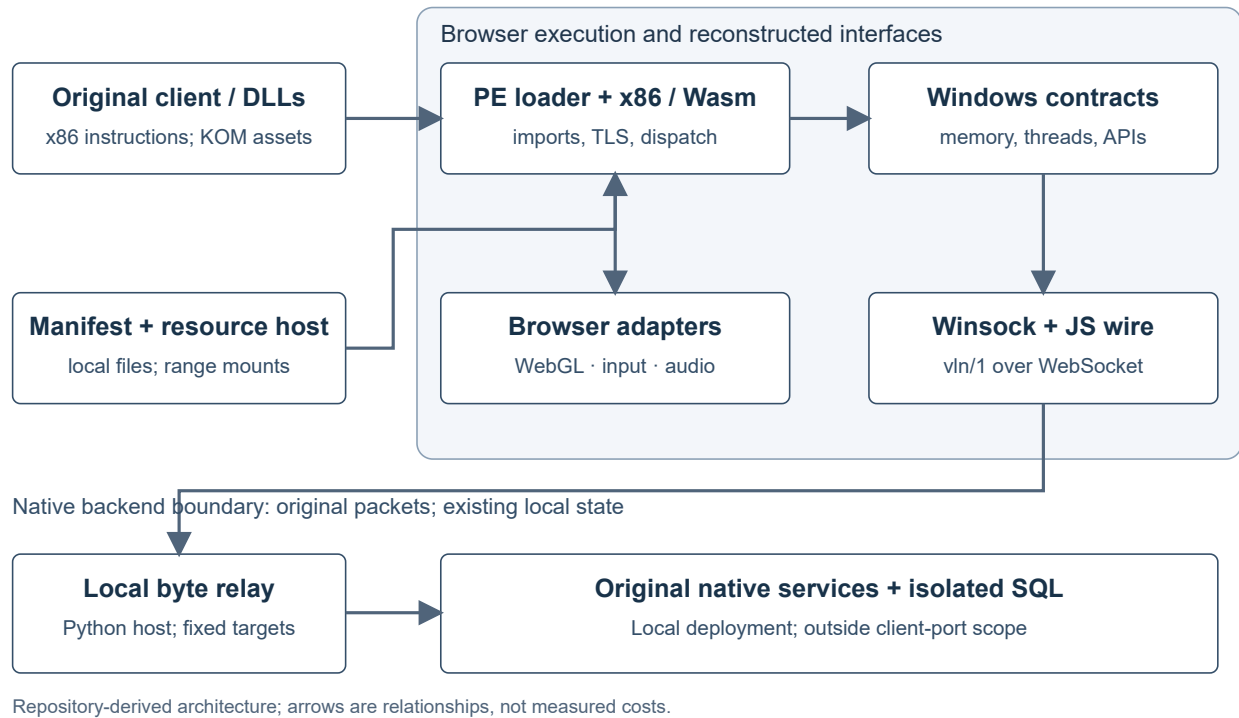


Figure 1. Derived from repository implementation and subsystem records, not a new execution trace. Native services are outside the browser boundary. Original resources enter through a manifest and virtual filesystem; network bytes leave through a constrained local relay. [E10–E15](#)

4.1 Execution and loading

The PE loader maps sections, relocations, imports and TLS metadata from the supplied images. Generated platform adapters provide PE export names/ordinals and x86 jumps into compatibility handlers. This matters because original code can inspect export tables rather than performing host-level name lookup alone. `lib/dll-loader.js` manages original library initialization; strict initialization failures remain explicit. [E10](#)

Wine-Assembly decodes original x86 into a threaded dispatch representation and executes it in Wasm. `src/main.watx` defines the canonical include closure; `07-decoder.wat` and `04-cache.wat` own decoder and dispatch-cache responsibilities. `host.js` coordinates stepping, clocks, imports and workers; integration `main.js` connects the machinery to the manifest and canvas. Original application, Lua, D3DX and FMOD code remain guest code on the documented paths. No new interpreter/JIT variant is reported for the final optimization pass. [E10](#)

4.2 Windows contracts, workers and memory

The compatibility boundary supplies guest calling conventions, pointer translation, memory allocation, thread synchronization, TLS and supported API semantics. Sparse guest addressing is not equivalent to a direct contiguous Wasm address. Original executable/MFC/msvcrt TLS separation and wait-capable DLL attach were verified in earlier bounded regressions. Wider lifecycle behavior remains incomplete. [E10](#), [E14](#)

The atomic ownership correction in change 0014 replaces defensive duplicate-free list scans with a claim in the aligned allocation header before free publication/accounting. Shared instances participate in the ownership state. GlobalReAlloc retains distinct temporary reservation behavior and original-allocation validity on failure. These are contract-specific mechanisms, not proof that original-client duplicate frees caused an application failure. [E04](#), [E14](#)

4.3 Graphics, input, audio and storage

Original graphics libraries produce application resources and commands; compatibility code translates supported D3D9 operations and shaders to browser rendering. Text uses original fonts through reconstructed font behavior. Shared render-command lifetime and snapshots require cross-instance ownership. Pure resource/artifact caches validate bytes, scope identities and bound storage; guest state and GPU handles are not persisted as reusable artifacts. [E11](#), [E14–E15](#)

DOM physical input is mediated to Win32/DirectInput. Earlier repairs separate left/right modifiers and current physical state from older message-relative snapshots. Buffered edge history, short-input/focus reliability and full game-effect fidelity remain incomplete. The enabled FMOD/DirectSound/WebAudio path has recorded nonzero PCM and a running context, without complete audible-quality acceptance. Original KOM ranges and namespace-preserving mounts provide resources; configured browser-local settings persistence remains absent. [E11–E13](#)

4.4 Native-service boundary

Original code supplies game packet contents. Wasm Winsock, `local_server_wire.js` and `network.py` carry unchanged original payloads through v1n/1 WebSocket transport to fixed local services. Sessions own native sockets; the isolated SQL endpoint is 127.0.0.1:15433. Native server and SQL execution is intentional scope, not an unfinished server port. This architecture is browser-hosted client execution with local dependencies. It is not client pixel streaming from a native game process. [E12](#)

5. Compatibility and dependency model

Layer	Preserved or reconstructed object	Verification scope and limitation
Original client/content	Supplied executable, original libraries, scripts, KOM resources	Input identity checks; no recovery of complete game source
CPU execution	Wasm interpreter, x86 decoder, dispatch and selected SIMD behavior	Bounded instruction/native comparisons; universal x86 equivalence unclaimed
Windows API	Reconstructed handlers and PE adapters	Calling convention, return/error/lifetime fixtures for selected paths
Platform algorithms	Locally derived Microsoft shader-validator component and installed DLL metadata	Separate input hashes/derivation; no local authorship claim
Browser host	Worker scheduling, WebGL, input/audio/storage adapters	Browser/OS nondeterminism and broader compatibility remain
Native backend	Existing server programs, local configuration and SQL	Interoperability records; private state required for fuller reproduction

The Wine-Assembly dependency pin is `d7854bc6c6bd0319ca33b5f31a465602e2ce0686`. The inherited installation identifies CardSorting/JoySwordOnline commit `4c39249b939d79c229e1fd4f63dfd462a90f9060`. Dependency metadata records websockets 17.1 and research-tool versions. A pin establishes a source reference; local

patches and generated files remain additional custody objects. [E16](#)

“Original client unchanged” means supplied files were retained on disk within the stated input audit. It does not mean the mapped guest image is identical to those files. Loading resolves imports and relocations. An earlier guarded send wrapper is installed in mapped code padding to serialize existing packet construction under the original socket lock, with exception cleanup. This documented intervention is materially different from rewriting game logic as JavaScript, but must still be disclosed. [E17](#)

6. Experimental methodology

6.1 Retained conditions and timing origin

The final control comprises two sequential headed Edge 155.0.4283.45 launches on the same machine using the preserved host and observer snapshot. The observer requests a 1280 × 1100 browser viewport. Original empty-login images are retained as 1280 × 960 canvas captures; viewport and capture geometry are distinct. Native services remain in their existing configuration. No login credentials or gameplay actions were entered in these startup controls. CPU profiling is disabled in their result records. [E01–E03](#), [E19](#)

Cold denotes a fresh experiment-specific browser profile. Warm denotes reuse of that profile by the same condition's second run. OS caches were not flushed. Cold is therefore a browser-profile condition, not a verified whole-machine cold boot. Reuse can affect cached resources and engine behavior beyond a single named cache. The design has one cold and one warm sample at this final control condition, sequential rather than randomized.

The retained runner sets `started = time.monotonic()` immediately before launching the browser context. `firstVisibleSceneSeconds` is measured from this origin, not from **Start client**. The click is recorded separately at 2.188 seconds cold and 1.750 seconds warm. The published startup values include launch/readiness/UI-start overhead. First GPU presentation is an intermediate event, not login. [E01](#), [E19](#)

6.2 Detection, visual review and progression

The observer combines presentation tracking, canvas screenshot inspection and sampled host lifecycle information. The first visible scene is detected through image analysis (`coloredFraction` is retained), then classified by review as the original empty login screen. Automatic color detection alone would not establish the scene's identity. Four image-file records are marked reviewed in the audit; publication inspection additionally examined two representative files without creating new runtime evidence. [E02–E03](#)

Requested post-login observation is 30 seconds, separate from the 900-second startup deadline. Its `steadyObservation.elapsedSeconds` reflects the progress-gate interval; the independently recorded `steadySeconds` field has a different boundary and is not substituted. Progress requires lifecycle/execution observations and advancing frames, with a recorded 60-second no-progress criterion. The observation itself lasts about 31 seconds, so it cannot exercise that full stall threshold. [E01](#), [E19](#)

6.3 Integrity and telemetry

Before/after input verification checks each enumerated original file against its recorded SHA-256. The control audit separately checks 211 served/observer identities. `samples.jsonl` retains full sampled CPU, heap and browser-process memory series; result arrays are bounded to 360 rows per series at the declared

ten-second sampling interval. Windows process-memory counters distinguish working set from private committed bytes; summed working sets may double-count shared pages, and per-worker memory/GPU VRAM are not isolated. [E01–E02](#), [E19](#)

No benchmark was rerun for this paper. Publication-time work is limited to reading retained artifacts, verifying file integrity, inspecting existing images, consulting primary references and typesetting documents.

7. Runtime measurements and results

7.1 Final control pair

Metric	Cold control	Warm control
Browser launch, seconds	0.719	0.593
Browser ready, seconds from origin	2.063	1.687
Start click, seconds from origin	2.188	1.750
First GPU presentation, seconds from origin	224.594	223.328
First visible login, seconds from origin	440.703	438.797
Login progress-gate interval, seconds	31.094	31.109
Initial → final progress counter	10 → 1,083	10 → 1,090
Frame advances in progress gate	1,073	1,080
Maximum sampled no-progress interval, seconds	0.375	0.265
Maximum observation gap, seconds	0.547	0.594
Reported error entries / progress failure	0 / null	0 / null

Source: [E01–E02](#). Values are rounded to three decimals from raw floating-point seconds. Counter deltas refer to the progress gate, not whole-run totals or GPU scanout. Error absence applies to the retained observer's reporting scope.

The warm login value is 1.906 seconds below the cold value, a descriptive difference from one pair. No confidence interval, significance test, native-client ratio or $3\times$ claim is supported. Four reviewed image records support the visible-login classification. Three have identical SHA-256, so the review does not represent four independent scene variations. [E02–E03](#)

7.2 Allocator candidate comparison

Condition	Recorded outcome	Interpretive bound
Final candidate cold	Screenshot-helper timeout at 297.484 seconds; no login timestamp	Observer failure; last sample at 259.953 seconds was executing with two frames and no runtime error; do not classify as demonstrated allocator failure
Candidate subsequent warm-profile launch	Login at 457.859 seconds; 1,070 advances over 30.906 seconds	Follows interrupted cold priming; not matched to fully primed control warm condition

Source: [E04](#). The candidate review audits 211 served/observer and 257 live source identities with 321 original inputs unchanged. Observer and original-input identities match the paired control; among served files only manifest metadata and the two Wasm modules differ. This is a useful condition audit, but cannot repair the missing cold result or priming mismatch.

7.3 Historical cache observations

Condition	First earlier control cold / warm, s	Earlier cache candidate cold / warm, s	Repeated control cold / warm, s
Visible login	435.265 / 432.641	450.000 / 459.922	483.406 / 464.813

The first candidate pair reduced reported loading time from 3.583 / 3.699 seconds to 1.800 / 1.297 seconds. Page resource transfer fell from 943.386 / 936.004 decimal MB to 543.652 / 497.619 MB; sampled external/backing peak fell from 1,146.892 / 1,168.977 MB to 719.539 / 746.716 MB. Candidate sampled page JS heap peaks increased. Source: [E15](#).

The candidate is slower than the first controls and descriptively faster than the repeated controls. Repeated-control hooks match the candidate's added presentation hooks. With small sample counts and uncontrolled caches/system activity, the record establishes resource-consumption changes under those conditions, not reproducible login acceleration. These earlier measurements must retain their own identities and are not relabeled as the final allocator candidate.

7.4 Historical gameplay

The latest gameplay review, 145806, belongs to an earlier build. It reaches original login, existing-character selection, town, dungeon combat/two skills, first-area clear and the formerly failing area gate. Original failed-attempt rewards and automatic town return are observed. Revival still fails; native statistics report one attempt, zero wins and zero stones used. Victory/reward completion and two winning cycles remain unaccepted for that final gameplay build. Earlier successful cycles on other builds do not close these gates. [E08–E09](#)

8. Reproducibility and integrity verification

The record contains three different identity conditions: the measured control, prepared allocator candidate and final frozen working source. Appendix A records the module/patch identities. The current candidate has no assigned matching application review (`currentBuildEvidence` remains null). Matching a file inventory is not an application replay. [E05–E06](#), [E18](#)

The 321 input entries concern manifest-enumerated original inputs in the startup-pair snapshots. They do not hash the entire machine, private SQL state, every configuration or all future dependencies. The 211-file control audit concerns served/observer files. The 2,229-file freeze inventory concerns an explicitly broader selected source/test/integration/evidence set. These counts are not interchangeable, and hashes are not proof of correct semantics, licensed distribution or independent authorship. [E02](#), [E06](#)

Knowledge edition 1.4.1 corrects 1.4.0 while preserving the earlier archive. It records twelve final source rollback suites, incomplete observer validation and prepared/frozen drift. The later rollback is absent from the prepared patch and Wasm; two observer sources differ from the manifest. Restoring only the pinned checkout and patch cannot reconstruct the final working source. Frozen snapshots and the inventory must accompany them. [E05–E07](#)

Independent reproduction needs the retained integration, original client/platform inputs, native services/private configuration and appropriate SQL state. The documentation archive deliberately excludes credentials, database contents, large game/platform binaries and much original evidence. This paper provides traceability and conditions for a future reproduction; it does not claim a clean-machine restoration was completed.

9. Performance characteristics

A bounded 90-second startup diagnostic stops before login. Within one active worker's approximately 20.147-second profile window, `heap_free_impl` accounts for 521,506 microseconds, or 2.589% of sampled wall intervals, while `uop_fast` accounts for about 28.0%. Waiting-worker `Atomics.wait` samples denote blocking, not busy CPU. This profile identifies allocator work without making it the dominant startup cause. [E04](#)

The matched allocator microbenchmark uses 512-byte free/reallocate pairs, 20,000 warmup pairs and five repetitions of 100,000 pairs. Four fresh Node processes run control/candidate/candidate/control. Each table value is the mean of two per-process medians. Recorded environment: Node 24.14.1, V8 13.6.233.17-node.44, Windows x64, Intel Core i5-6500 at 3.20 GHz with four logical CPUs, and 512 MiB shared Wasm memory. This environment belongs to the microbenchmark record; complete OS/GPU/physical-RAM metadata for the final browser pair is not established by that record alone. [E04](#)

Retained list depth	Control mean of medians, ms	Candidate mean of medians, ms	Control / candidate ratio
0	10.458	11.914	0.878
8	19.297	12.465	1.548
32	42.469	13.806	3.076
64	67.054	13.131	5.107
128	69.492	12.617	5.508
512	68.488	13.264	5.164

At depth zero the candidate is 13.9% slower; at depths 64–512 it is approximately 5.1–5.5× faster. The measured operation includes JS call/assertion costs, tests one allocation size and does not measure concurrent throughput or the original allocation mix. It supports removal of the reproduced scan scaling, not overall game acceleration. [E04](#)

If only a 2.589% fraction of an end-to-end workload were eliminated, and that fraction truly represented total serial runtime cost, an Amdahl-style ceiling would be $1 / (1 - 0.02589) \approx 1.027$. This is an illustrative bound under explicit assumptions, not an estimate of BROCWIN's speedup: the measured fraction is from one worker/window, includes sampled wall intervals and does not characterize the complete startup. It reinforces the need for whole-application measurement.

The final warm control records 48 resource-byte hits, zero misses and 56,477,458 avoided bytes. Module clone hits remain zero. Earlier shader/texture caches have no startup hits in the reported login workload. Browser loading savings and cache correctness therefore do not establish compiled-module persistence or identify the remaining multi-minute execution cost. [E01](#), [E15](#)

10. Limitations and threats to validity

Internal validity. One final sample per profile condition, sequential execution, nonflushed OS caches, browser engine/driver caches and uncontrolled system activity limit causal inference. Profiling and presentation hooks have measurement costs. No native-client startup baseline or randomized repeated final comparison is present.

Construct validity. First presentation is not first login; visible-login detection requires image interpretation. Frame counter changes are not verified display refreshes, GPU scanout, input responsiveness or complete simulation progress. Reported-error absence covers known observer channels. Working set, private commit, page heap and external backing measure different quantities.

Duration validity. About 31 seconds of post-login progress is far below a reliability trial. The 60-second stall threshold cannot be fully exercised within that window. Earlier long captures span particular scenes and builds, not controlled continuous-service acceptance.

Observer validity. The candidate cold screenshot stability timeout censors the comparison. The final observer revision adds trap/worker-presence, first-GPU ordering and PNG-dimension refinements after its last unit test. Its proposed headed repair smoke did not execute; earlier four lifecycle fixtures predate that repair. The final frozen live observer cannot be silently substituted for the retained measured observer. [E05–E06](#)

External validity. One legacy client, local native dependencies and one recorded browser condition do not establish broad Windows support, hardware portability, internet deployment or compatibility with other clients. Reconstructed API presence alone is not a success metric.

Semantic limits. Open TLS/unload/loader-lock constraints, Winsock completion paths, buffered-input behavior, text/render defects, revival and historical timeout/stall failures remain. Atomic ownership does not detect stale pointers after legitimate address reuse or arbitrary header corruption. Source-only reservation cleanup has no canonical or application result. [E07–E14](#)

Provenance and custody. Integrity inventories are unsigned file records, not proof of independent replication or authenticity of every inherited object. Some artifacts are ignored/local and absent from a transferred Git checkout. Reviewer identity is not specified for every image record. Original game/platform rights and private backend data are separate from the compatibility source license. [E16](#), [E18](#)

11. Future research

After explicit institutional reopening, priorities could be independent artifact restoration, final-observer validation, canonical reservation-cleanup integration, repeated matched startup studies and stated-duration progress/resource trials. Any optimization should begin with a measured contract or original-path failure, retain failed controls and report both benefit and cost. Gameplay continuation would require separate confirmation that its queue is reopened.

Longer-term research could compare a small application corpus, examine browser-version sensitivity and define archival restoration procedures with accountable custody. These are opportunities, not implemented capabilities or an authorization to continue. The [stakeholder roadmap](#) connects them to decisions and evidence gates.

12. Conclusion

The retained final control demonstrates original-client execution to an empty login interface through a browser-hosted synthetic compatibility runtime, with audited input identity and short continued frame progression. The recorded startup values are 440.703 and 438.797 seconds. The evidence does not establish a complete game port, reproducible 3× acceleration or durable continuous operation. Prepared and source-only work remain distinct at freeze. The experiment's reviewable result is this bounded milestone and the preserved account of how it was reached.

13. References

Repository citations E01–E19 resolve through the [shared evidence index](#) and its machine-readable ledger. Literature serves as technical context; it supplies no BROCCWIN benchmarks.

1. Bellard, Fabrice. 2005. “QEMU, a Fast and Portable Dynamic Translator.” *FREENIX Track, USENIX Annual Technical Conference*, pp. 41–46. [Author's conference paper](#).
2. WebAssembly Community Group. *WebAssembly Core Specification*. Living specification consulted 11 October 2026; displayed version 3.0 dated 9 October 2026. [Primary specification](#). Context only; no runtime feature-level conformance asserted.
3. Microsoft. “PE Format.” *Microsoft Learn, Win32 documentation*. [Primary executable-format documentation](#).
4. Khronos WebGL Working Group. *WebGL Specification*, version 1.0.3. [Fixed-version primary specification](#).
5. Wine Project. “About Wine.” [Primary project description](#). Used only to distinguish architectural approaches.
6. Rothenberg, Jeff. 1999. *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation*. CLIR, report 77. [Publication](#); [research agenda](#).
7. Wine-Assembly contributors. Source dependency pinned at d7854bc6c6bd0319ca33b5f31a465602e2ce0686. [Pinned repository](#); local patches and attribution in E16/E18.
8. HEAV3NS-BROCCWIN contributors. 2026. Retained experiments, source audits, change records and corrected knowledge closeout 1.4.1. [Evidence index](#).

External references consulted 11 October 2026. Two attempted related publications were not retrieved in full (the 2017 WebAssembly article and the variable-media PDF); neither is used as an unseen source for technical claims.

Appendix A. Condition identities

Artifact	SHA-256
Measured control main Wasm	c38c94850b55c6adc2981f4a277d80a3732f5b4f8c275d246949c92fe810eb3d
Measured control compatibility Wasm	c2380511aeb9632886c0ff2790276dbfdf6c1c89193b20174dd5d0acaaae8ef8
Measured control patch	985f4c6dd2d8eda858a0fa5e877c755459c26a6f778b1db9e6058f5ac5cd7236

Artifact	SHA-256
Prepared allocator candidate main Wasm	1928229fb1326a2d7b5410aeb24c2ada36833b40650c8627e509058f7ac14cbe
Prepared allocator candidate compatibility Wasm	1150a267e54da522264b8924d61a8dbe899fffb68ea10cbec68c13b3f91d765eb
Prepared candidate patch	8b763bd478e619d1c95f37a429ecc840e04952b010d00d52007accc092638e2b
Retained startup-pair observer	91dd12a6f38aa663ac67d24b18ea94f6710fbbbf697eae3fd02239cb1f367e8
Final frozen live observer	ad992490edb8f1607c95d0787b99ab98e83863e75f1324c2bc5de6876d826698
Frozen inventory record	df194fcf69aeca921fedaef4a2843ecf915084ca2d283fad97f42696fb6cd4ca

Sources: E02, E04, E06, E18. The later reservation source-compiled fixture module is a separate experimental object, not either prepared canonical module. Full record-file hashes appear in the shared ledger.

Appendix B. Historical reproduction route

Conditional instructions only: engineering remains frozen. A future custodian must explicitly reopen a separate recovery workspace, recover the original/private inputs and follow `theseus/wiki/build-recovery.md`. The canonical dependency build uses its vendored WATX compiler and `tools/build.sh`, not an interchangeable `wat2wasm` invocation. Preparation mutates manifests/adapters/patches and must not occur against the present freeze. The historical ordinary browser host is `python theseus/browser/serve.py --port 8766`, followed by **Start client** at the local URL.

The retained startup observer exposes the following arguments for its two-run profile protocol:

```
python <retained observer/benchmark_startup.py> --label <new label>
--snapshot <matching retained host> --runs 2 --port <unused local port>
--timeout 900 --steady-seconds 30 --headed
```

This describes the reproducible argument pattern inferred from retained observer source; the report does not claim the exact historical shell command or port was preserved. Use the matching observer dependencies and host mount routing, preserve the original working-directory convention, and record an actual command in any new authorized run. Do not use the final changed observer to recreate old instrumentation silently. [E19](#)

Read-only checks available now are `python theseus/knowledge.py --check`, `python theseus/knowledge.py --verify-release 1.4.1` and the publication verifier. No restoration, native-service start or client launch was performed for the paper.

Appendix C. Claim classification

Observed: login images; recorded latency; counter progression; scoped file identity. **Contract-tested:** selected allocator/API/renderer behaviors. **Source-only:** final reservation cleanup. **Partially validated:** final observer revision. **Unverified:** full client acceptance, overall startup gain, 3× improvement, long-duration stability and broader compatibility. **Interpretive:** the environment-as-artifact thesis in Publication I.